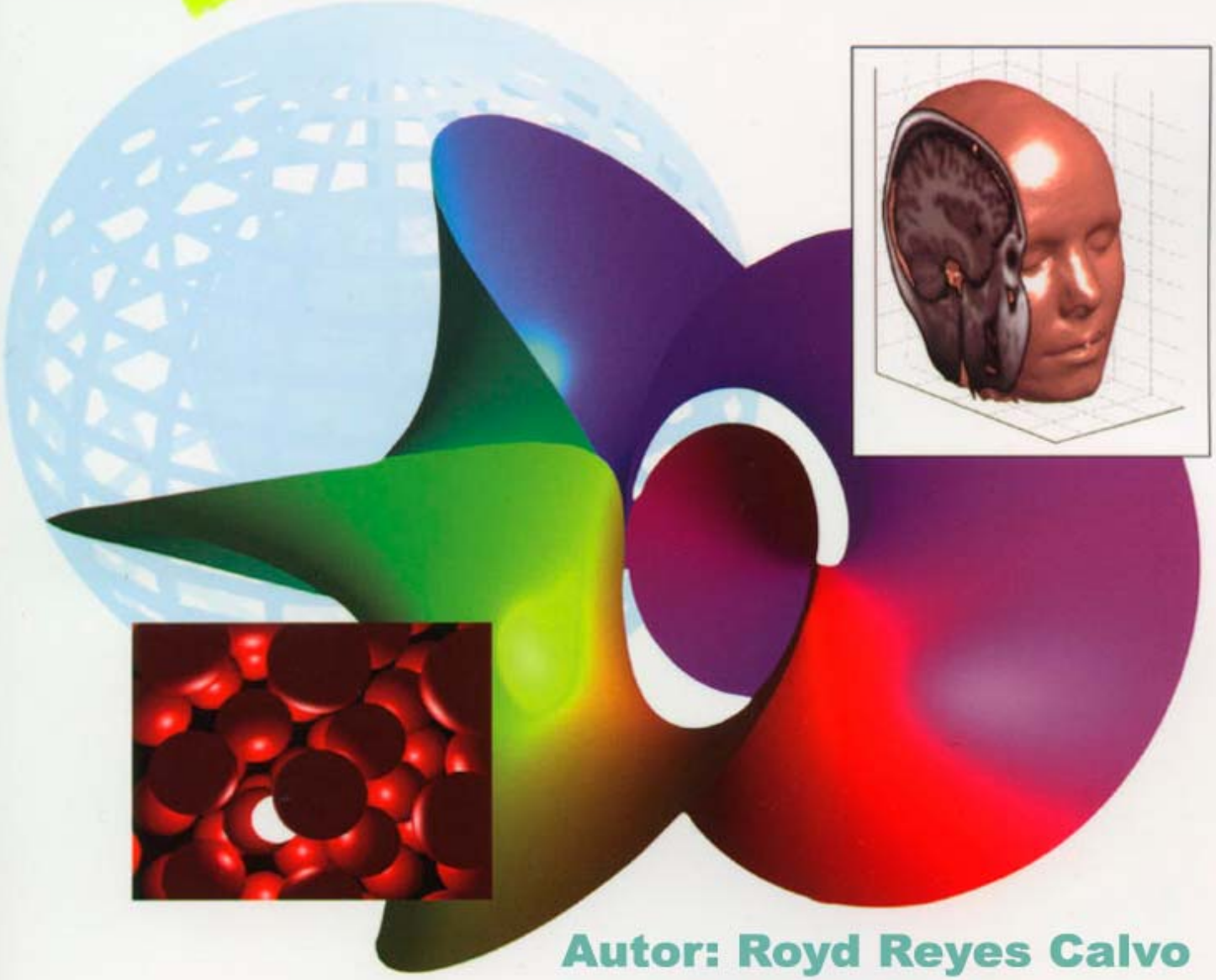
Universidad Central ¨Marta Abreu ¨ de Las Villas
Facultad de Ingeniería Eléctrica, Departamento de Automática

Funciones en
MATLAB
para el análisis y diseño en el campo de la frecuencia



Autor: Royd Reyes Calvo
Tutor: Maria del C. Hdez. Carús

Santa Clara 2004

Universidad Central “Marta Abreu” de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática

TRABAJO DE DIPLOMA

Funciones en Matlab para el análisis y diseño en el campo de la frecuencia

Autor: Royd Reyes Calvo

Tutor: Ing. Maria del C. Hernández Carús

Ing. Profesor Asistente, Facultad de Ing. Eléctrica UCLV.

E-mail: carmen@fie.uclv.edu.cu

Santa Clara

2004

"Año del 45 aniversario del triunfo de la revolución"

Hago constar que el presente trabajo de diploma fue realizado en la Universidad Central "Marta Abreu" de Las Villas como parte de la culminación de estudios de la especialidad de Ingeniería {*Automática, Eléctrica o Telecomunicaciones*} autorizando a que el mismo sea utilizado por la Institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos, ni publicados sin autorización de la Universidad.

_____

Firma del Autor

Los abajo firmantes certificamos que el presente trabajo ha sido realizado según acuerdo de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del Tutor

Firma del Jefe de Departamento donde se defiende el trabajo

_____

Firma del Responsable de
Información Científico-Técnica

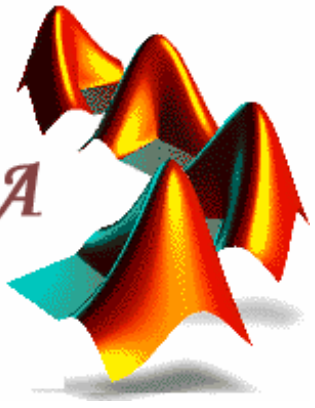
PENSAMIENTO

Saber no es suficiente, debemos aplicar.

Desear no es suficiente, debemos hacer.

- Johann W. Von Goethe

DEDICATORIA



A mis queridos hermanos por brindarme el apoyo necesario para alcanzar mis sueños.

De manera especial a mis padres por lograr de mí el hombre que soy.

A mi novia por ser tan especial y amiga

Agradecimientos

A mi excelente tutora y amiga María del Carmen por su experiencia, consejos y ayuda intelectual.

Al claustro de profesores que me ayudaron a adquirir conocimientos y valores indispensables para la vida.

A mis compañeros por su comprensión y apoyo diario.

A Yaima, Moisés, Rubí, Lili, en fin a todas aquellas personas que me ayudaron en la realización y culminación de esta investigación y por el poco espacio no puedo mencionar.

A todos muchas gracias.

Tarea Técnica

TAREA TECNICA

Determinar las limitaciones que presentan las funciones del Matlab que más se usan por los estudiantes de la carrera de Automática en el trabajo con respuesta de frecuencia.

Hacer una revisión bibliográfica en Internet acerca la existencia de trabajos realizados para modificar las funciones del Matlab asociadas a la respuesta de frecuencia de los sistemas lineales.

Hacer un estudio de las diferentes formas en que puede aparecer un sistema (LTI) así como sus respectivos diagramas de Bode y Nyquist.

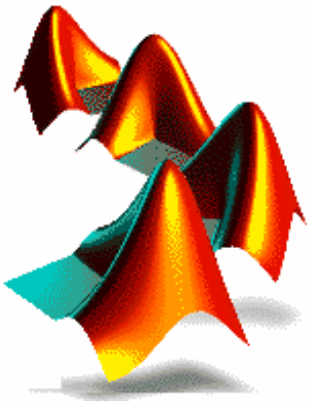
Crear un conjunto de funciones que permitan obtener la respuesta de frecuencia de sistemas SISO más útiles al proceso de enseñanza de esta.

Hacer una función que permita trazar las bandas de Gershgorin para el análisis de respuesta de frecuencia de sistemas MIMO, implementando en ella el algoritmo de la Seudo-Diagonalización

Comprobar la efectividad de las funciones creadas con un conjunto de casos de estudio.

Realizar el informe de tesis.

RESUMEN



RESUMEN

Algunas de las funciones que tiene el Matlab para la obtención de respuesta de frecuencia de sistemas lineales no ofrecen todas las posibilidades para la mejor comprensión de esta materia por los estudiantes, en ocasiones estos se enfrentan a diferencias entre lo que los textos ofrecen y lo que el Matlab devuelve, con este trabajo pretendemos elaborar una herramienta computacional mas afín con la necesidad del aprendizaje, que apoyada en el Matlab brinde a los estudiantes la posibilidad de obtener las representaciones gráficas más adecuadas y analizar lo concerniente a la estabilidad de sistemas lineales a partir de ellas.

Se programan funciones utilizando el propio lenguaje del Matlab pero la programación es transparente para el estudiante, el cual solo tendrá que dar los datos mínimos necesarios de la función de transferencia de la cual desea obtener la respuesta de frecuencia.

Se pretende con este trabajo mejorar el proceso de enseñanza aprendizaje montado sobre un software profesional de alta calidad como es el caso del Matlab pero que no es una herramienta didáctica.

Se analizan una serie de casos donde se pone de manifiesto las diferencias entre los resultados que ofrece el Matlab actualmente y las nuevas posibilidades que se abren con las funciones creadas.

ÍNDICE

INDICE

Introducción

INTRODUCCION

Al enfrentarse al método de la respuesta en frecuencia para el análisis y diseño de sistemas lineales el estudiante puede sentir que este es menos intuitivo que otros métodos que ya ha utilizado en temas previos , sin embargo, tiene ciertas ventajas, especialmente en situaciones reales como modelado de Funciones de Transferencia a partir de datos físicos. Este método es muy usado en varias asignaturas, tanto de la carrera de Automática como en la de Telecomunicaciones.

Sin embargo cuando se utiliza el software Matlab para estudiar la respuesta de frecuencia de los sistemas lineales se presenta una dificultad, los estudiantes se enfrentan a diferencias entre lo que los textos ofrecen y lo que el Matlab devuelve. Además para algunos sistemas las funciones que trae el Matlab para el análisis de respuesta de frecuencia dan resultados erróneos.

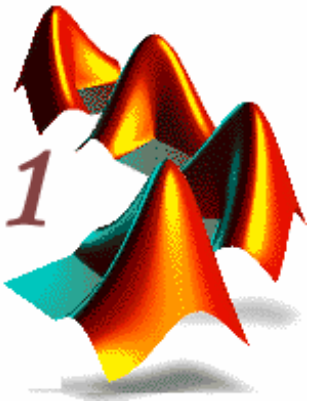
Con este trabajo nos hemos propuesto crear un conjunto de funciones que permitan hallar la respuesta de frecuencia más a fin con el propósito docente, utilizando el código de Matlab, pero de forma tal que la programación sea transparente al usuario y que tenga el mismo entorno grafico que las ya conocidas funciones que este trae por defecto para el análisis y diseño en el campo de la frecuencia. Además se crean dos funciones nuevas que permiten el análisis y diseño de sistemas multivariables usando las bandas de Gershgorin para lo cual no existe nada aún disponible en Matlab.

En el capítulo I se hace una breve descripción del análisis en el campo de la frecuencia y de algunas de las limitaciones que presentan las funciones que trae el Matlab para el análisis y diseño en el dominio de la frecuencia.

En el capítulo II se explica el algoritmo seguido para implementar cada función y los posibles resultados que pueden dar cada una de ellas.

En el capítulo III se analiza la aplicación de las funciones creadas con ejemplos y se establecen comparaciones entre ellas y las que ofrece actualmente el Matlab en cuanto a los resultados.

Capítulo 1



CAPITULO I. La respuesta de frecuencia y el Matlab.

El trabajo con la respuesta de frecuencia de un sistema tiene su expresión más generalizada en la representación gráfica de la magnitud y la fase de cualquier sistema representado por una Función de Transferencia , usualmente se hace de dos maneras distintas, en diagramas logarítmicos o diagramas de Bode y en diagramas polares o de Nyquist. Ambas contienen la misma información; la diferencia radica en la manera en que esta se presenta.

La respuesta en frecuencia es la respuesta en régimen permanente del sistema a entradas sinusoidales de frecuencia variable. La salida de un sistema lineal a una entrada sinusoidal es una sinusoide de la misma frecuencia pero con distinta magnitud y fase. La respuesta de frecuencia se define como las diferencias de magnitud y fase entre las sinusoides de entrada y salida. Se puede usar la respuesta de frecuencia de un sistema a lazo abierto para predecir su comportamiento a lazo cerrado.

Para graficar la respuesta de frecuencia, se crea un vector de frecuencias (entre cero e infinito), se calcula el valor de la Función de Transferencia de la planta a estas frecuencias.

Si llamamos $G(s)$ a la Función de Transferencia de un sistema a lazo abierto y w es el vector frecuencia, entonces sustituyendo s por jw tendremos la función en el campo de la frecuencia $G(jw)$. Como $G(jw)$ es ahora un número complejo, se puede graficar su magnitud y fase (diagrama de Bode) o su posición en el plano complejo (diagrama de Nyquist) en función de la frecuencia variable.

En varias asignaturas de la carrera de Automática, como son: Circuitos Eléctricos II, Electrónica II, Sistemas de Control I, Sistemas de Control II y en ocasiones en los Controles para Procesos I y II se utiliza el método de respuesta de frecuencia para el análisis y diseño, en mayor o menor medida.

En la última década la automatización del proceso de obtención de los diagramas a través de la computadora se ha hecho más frecuente y la herramienta matemática Matlab presenta un conjunto de funciones que permiten el análisis y el diseño de un sistema en este campo,

a saber, allmargin, bode, bodemag, evalfr, freqresp, interp, linspace, logspace, ltiview, margin, nichols, nyquist, sigma, ngrid, nichols. De todas ellas las que más se usan son:

Bode: Calcula y traza el diagrama de Bode.

Margin: Traza el diagrama de Bode y calcula los márgenes de ganancia y fase.

Nyquist: Traza el diagrama de Nyquist.

Allmargin: Calcula los márgenes de fase y ganancia.

Linspace: Crea un vector de frecuencia igualmente espaciados

Bodemag: Traza el diagrama de la magnitud de un sistema.

BodeAsym: Dibuja el diagrama de Bode asintótico.

Logspace: Crea un vector de frecuencias igualmente espaciados por década.

De este conjunto de funciones existen algunas que presentan ciertas limitaciones que restan objetividad al análisis que se haga, una vez que se usen en condiciones determinadas. A continuación se analizan.

1.1 Diagrama de Nyquist

El diagrama de Nyquist se traza con el objetivo de conocer la estabilidad de un sistema, el mismo es la representación en un plano de GH(jw) una vez que la frecuencia sigue una trayectoria en su plano, definida por Nyquist, esa trayectoria se divide en varias secciones como se aprecia en la figura 1.1.

En la sección I la frecuencia varía entre cero e infinito, la sección II es una circunferencia de radio infinito ($s = \mathrm{Re}^{j\theta}$) y θ va desde $\frac{\pi}{2}$ hasta $-\frac{\pi}{2}$, la sección III es igual que la

sección I pero en este caso la frecuencia va desde menos infinito hasta cero, se dice que es la imagen de la I considerando como referencia el eje real y la sección IV es entonces una circunferencia de radio infinitesimal y θ va desde $-\frac{\pi}{2}$ hasta $\frac{\pi}{2}$. Es bueno aclarar que la sección IV solo es necesario recorrerla si la función de Transferencia tiene polos y/o ceros en el eje imaginario.

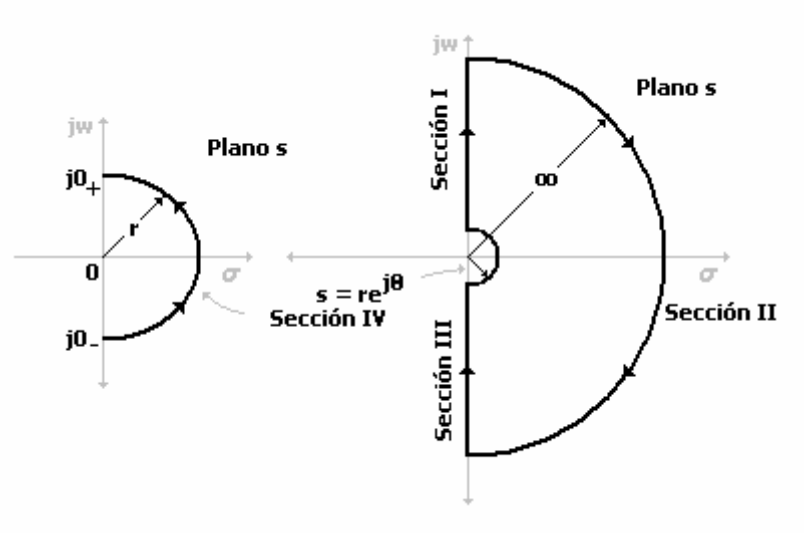


Figura 1.1: Contorno de Nyquist que evita polos y ceros en el origen del plano s

Una vez que se tiene el diagrama de Nyquist completo, para determinar la estabilidad según el teorema de Nyquist se aplica su ecuación o criterio,

$$Z = N + P$$

Donde

Z: Cantidad de ceros de la ecuación característica en la mitad derecha del plano S

P: Cantidad de polos de lazo abierto (GH) en la mitad derecha del plano.

N: Numero devueltas al punto (-1,0).

Partiendo de que para que un sistema sea estable no puede haber raíces de la ecuación característica en la parte derecha del plano S , $Z = 0$, entonces despejando

$N = -P$, el signo menos significa recorrido en el sentido contrario a las agujas del reloj, esto es: Se considera que un sistema es estable en lazo cerrado si el diagrama de Nyquist circula al punto $-1 + j0$ un número de veces en sentido antihorario, igual a la cantidad de polos que tenga el sistema en lazo abierto en el semiplano derecho del plano S, si el recorrido se hace en el sentido horario el sistema es inestable.

El MATLAB tiene una función que permite crear estos diagramas, pero no los traza completo, veamos esto a partir de un ejemplo, sea un sistema tal que puede ser representado por la siguiente Función de Transferencia de lazo abierto:

$$G(s)H(s) = K\frac{s+3}{s(s-1)}$$

Observemos que tiene un polo positivo por lo tanto $P = 1$, según Nyquist para que sea estable N tiene que ser igual a uno también. Si trazamos el diagrama de Nyquist, con K = 0.1 debe obtenerse el mostrado en la figura 1.2

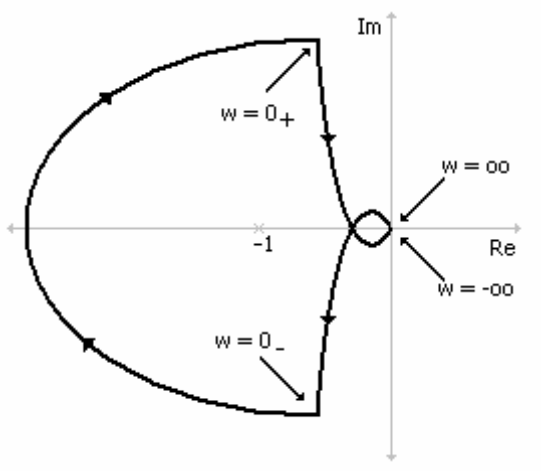


Figura1.2: Lugar de GH(jw) en el plano GH

El mismo es inestable, pues aunque hay una circulación al punto $-1+j0$, esta se hace en el sentido horario.

Si se utiliza el Matlab para graficar, obtendremos lo que se muestra en la figura 1.3.

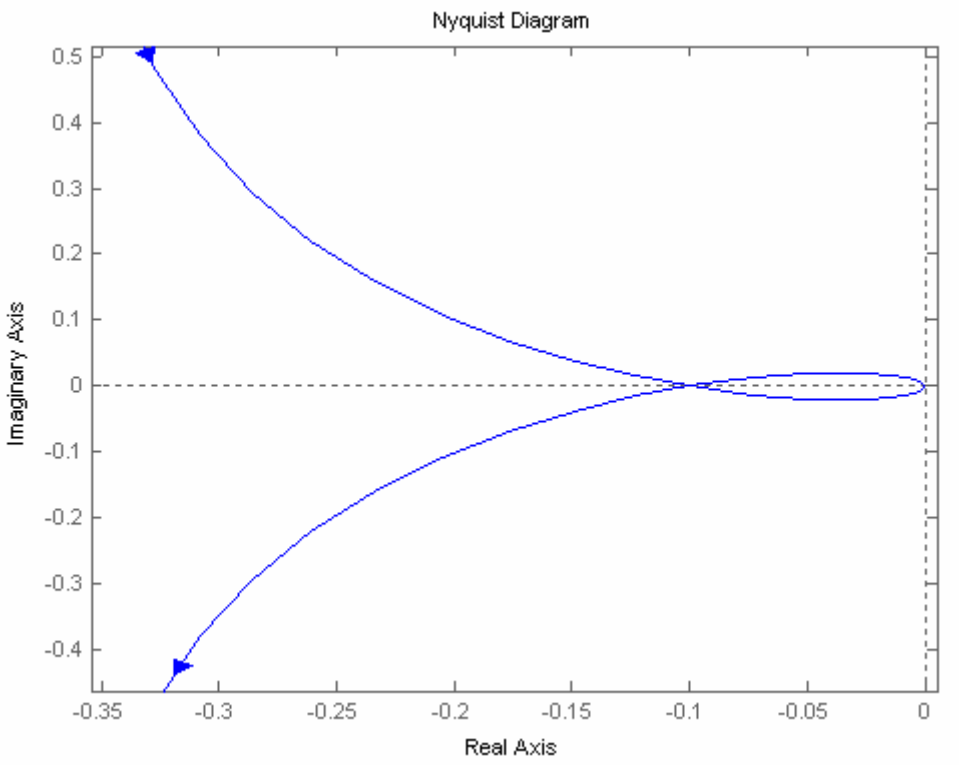


Figura 1.3: Diagrama de Nyquist de G(s)H(s) trazado por Matlab

Observemos la falta de información de que adolece esta figura, no se sabe si el diagrama circula o no al punto $-1+j0$, porque el mismo no está completo, se necesitaría una experiencia grande y conocimientos previos para inferir lo que va a pasar o continuar el trazo haciendo un análisis fuera de la máquina para completarlo y terminar el análisis de estabilidad, esto ocurre siempre que el sistema no sea tipo cero porque al tratar de representar la Sección IV de la trayectoria de Nyquist, definida anteriormente, se presenta una indefinición que el Matlab no soluciona.

Una solución a esta dificultad se presenta en este trabajo y la misma se explicará en detalles en el próximo capitulo.

1.2 Bandas de Gershgorin

La extensión del criterio de Nyquist directo o inverso al análisis de sistemas multivariables nos lleva a la obtención de las llamadas bandas de Gershgorin , el Matlab no cuenta con ninguna función que permita trazar estas bandas y es prácticamente imposible obtenerlas a mano. Nos propusimos hacer un algoritmo que trace las bandas y también que modifique la matriz de Transferencia del sistema para alcanzar la dominancia, para esto último se tomaron dos métodos propuestos por Rosenbrock [] .No es objetivo de este trabajo hacer un análisis profundo de esta técnica solo implementarla de manera que pueda automatizarse el proceso, pero con el propósito de que se entienda lo que hemos hecho plantearemos las definiciones fundamentales y las ecuaciones correspondientes a los métodos anteriormente mencionados

Sea G(s) la Matriz de Transferencia del sistema MIMO , y $\hat{G}(s)$ la inversa de ella.

$$G(s) = \begin{bmatrix} g_{11}(s) & g_{12}(s) & \cdots & g_{1n}(s) \\ g_{21}(s) & g_{22}(s) & \cdots & g_{2n}(s) \\ \vdots & \vdots & & \vdots \\ g_{n1}(s) & g_{n2}(s) & \cdots & g_{nn}(s) \end{bmatrix}$$

$$\hat{G}(s) = \begin{bmatrix} z_{11}(s) & z_{12}(s) & \cdots & z_{1n}(s) \\ z_{21}(s) & z_{22}(s) & \cdots & z_{2n}(s) \\ \vdots & \vdots & & \vdots \\ z_{n1}(s) & z_{n2}(s) & \cdots & z_{nn}(s) \end{bmatrix}$$

Entonces las bandas de Gershgorin son un conjunto de círculos con centro en $z_{ii}(s)$ y radio de:

$$d_i(s) = \sum_{\substack{j=1 \\ j \neq i}}^{n} \left| z_{ij}(s) \right|$$

Se dice que el sistema es dominante por fila si cada una de las bandas producidas excluyen el origen para i = 1, 2, 3,…, n, esto significa que ninguno de los círculos trazados puede tener al origen en su interior. A medida que una Matriz de Transferencia se acerca más a la dominancia por fila las interacciones entre los diferentes lazos en un sistema de control son menores.

1.3 Método de Seudo Diagonalización para lograr dominancia

Una matriz K(s) podría hacer KG y por tanto $\hat{K}\hat{G}$ diagonal, done G es la Matriz de Transferencia, si restringimos nuestra atención a un conjunto de todas las posibles matrices K(s) –por ejemplo a matrices independientes de s- podríamos preguntarnos cual matriz en este conjunto hace $\hat{K}\hat{G}$ lo mas diagonal posible. Esto significa que en cada fila de $\hat{K}\hat{G}$ la suma de los módulos de los elementos que no pertenecen a la diagonal, para determinados valores de s son tan pequeños como sea posible comparados con el módulo de los elementos de la diagonal. Este es el principal y complicado problema computacional que hay que resolver.

Escogemos s = iw y consideramos la fila j de $\hat{K}\hat{G}(iw)$ donde K es una matriz constante. Los elementos $\hat{q}_{jk}(iw)$ de esta fila son:

$$\hat{q}_{jk}(iw) = \sum_{i=1}^{n} \hat{k}_{ji}\hat{g}_{ik}(iw) \qquad (1)$$

$$= \sum_{i=1}^{n} \hat{k}_{ji}(\alpha_{ik} + i\beta_{ik}) \qquad (2)$$

Donde

$$\hat{g}_{ik}(iw) = \alpha_{ik} + \beta_{ik} \tag{3}$$

Ahora escogemos $\hat{k}_{j1}, \hat{k}_{j2}, \ldots, \hat{k}_{jn}$, así que

$$\sum_{\substack{k=1 \\ k \neq j}}^{n} \left| \hat{q}_{jk}(iw) \right|^2 \tag{4}$$

Sea tan pequeño como sea posible sujeto a la restricción

$$\sum_{i=1}^{n} \hat{k}_{ji}{}^2 = 1 \tag{5}$$

Usando el multiplicador de Lagrange λ, el cual requiere ser minimizado

$$\phi_j = \sum_{\substack{k=1 \\ k \neq j}}^{n} \left| \sum_{i=1}^{n} \hat{k}_{ji} \left(\alpha_{ik} + \beta_{ik} \right) \right|^2 + \lambda \left\{ 1 - \sum_{i=1}^{n} \hat{k}_{ji}{}^2 \right\} \tag{6}$$

$$= \sum_{\substack{k=1 \\ k \neq j}}^{n} \left\{ \left[\sum_{i=1}^{n} \hat{k}_{ji} \alpha_{ik} \right]^2 + \left[\sum_{i=1}^{n} \hat{k}_{ji} \beta_{ik} \right]^2 \right\} + \lambda \left\{ 1 - \sum_{i=1}^{n} \hat{k}_{ji}{}^2 \right\} \tag{7}$$

Derivando parcial con respecto a $\hat{k}_{jl}$ obtenemos

$$\frac{\partial \phi_j}{\partial \hat{k}_{jl}} = \sum_{\substack{k=1 \\ k \neq j}}^{n} \left(2 \left[\sum_{i=1}^{n} \hat{k}_{ji} \alpha_{ik} \right] \alpha_{lk} + 2 \left[\sum_{i=1}^{n} \hat{k}_{ji} \beta_{ik} \right] \beta_{lk} \right) - \lambda 2 \hat{k}_{jl} = 0 \tag{8}$$

$$l = 1,2,\ldots n$$

Ahora introducimos una matriz simétrica y real

$$A_j = \left(a_{il}{}^{(j)} \right) = \left(\sum_{\substack{k=1 \\ k \neq j}}^{n} \left[\alpha_{ik} \alpha_{lk} + \beta_{ik} \beta_{lk} \right] \right) \tag{9}$$

Lo cual como se puede ver es positiva semidefinida, así que sus valores propios son reales y no negativos. Además si se introduce el vector fila

$$\hat{k}_j = \left(\hat{k}_{jl}\right) \tag{10}$$

Podemos escribir (8) en la forma siguiente

$$A_j \hat{k}_j^{\,T} - \lambda k_j^{\,T} = 0 \tag{11}$$

Lo cual es un problema estándar de vectores propios. Cualquier vector propio de A_j hace la ecuación (8) verdadera. Las ecuaciones (7), (9), (11) y (5) se pueden escribir de la siguiente forma:

$$\sum_{\substack{k=1 \\ k \neq j}}^{n} \left|\hat{q}_{jk}(iw)\right|^2 = \hat{k}_j A_j \hat{k}_j^{\,T} \tag{12}$$

$$= \lambda \hat{k}_j \hat{k}_j^{\,T} \tag{13}$$

$$= \lambda \tag{14}$$

Para minimizar (14) debemos escoger $\hat{k}_j$ correspondiente al menor valor propio de A_j.

La solución de la ecuación (11) es un problema de vectores propios que no es difícil de resolver porque A_j es real y simétrica, esta solución da la j enésima fila de $\hat{K}$ la cual minimiza la suma de cuadrados, de las magnitudes de los elementos que no pertenecen a la diagonal, de la fila *j* de $\hat{K}\hat{G}(iw)$.

Podemos resolver un problema similar para cada fila de $\hat{K}$ usando el mismo o diferentes valores de *w* para cada una.

1.3.1 Método ampliado de Seudo Diagonalización

El procedimiento explicado anteriormente puede ser generalizado de varias formas. Primeramente en vez de minimizar la ecuación (4) podríamos minimizar una función más general.

$$\sum_{r=1}^{N}\left\{\sum_{\substack{k=1\\k\neq j}}^{n}\gamma_r\left|\hat{q}_{jk}(iw_r)\right|^2\right\} \tag{15}$$

Sujeto otra vez a (5). En (15) el factor γ_r es real y positivo, es una mesura de la cantidad a ser minimizada, es una medida de peso a las frecuencias $w_1, w_2, \ldots w_N$ de los elementos que no pertenecen a la diagonal en la fila j de $\hat{Q}$. Un análisis similar al dado anteriormente nos muestra que la fila j de la matriz $\hat{K}$ se obtiene otra vez solucionando un problema estándar de vectores propios.

$$B_j k_j{}^T - \lambda k_j{}^T = 0 \tag{16}$$

Donde

$$B_j = \left(b_{il}{}^{(j)}\right) = \left[\sum_{r=1}^{N}\gamma_r\left\{\sum_{\substack{k=1\\k\neq j}}^{n}\left[\alpha_{ik}{}^{(r)}\alpha_{lk}{}^{(r)} + \beta_{ik}{}^{(r)}\beta_{lk}{}^{(r)}\right]\right\}\right] \tag{17}$$

Y $\alpha_{ik}{}^{(r)}, \beta_{ik}{}^{(r)}$, son definidos por

$$\hat{g}_{ik}(iw_r) = \alpha_{ik}{}^{(r)} + i\beta_{ik}{}^{(r)} \tag{18}$$

Como se hizo antes, ahora se escoge el vector propio $\hat{k}_j{}^T$ correspondiente al menor valor propio de B_j. El mismo conjunto de $w_1, w_2, \ldots w_N$ puede ser usado para cada fila de K, o diferentes conjuntos pueden ser usados para diferentes filas. En la función donde se

implemento este algoritmo se uso el mismo vector de frecuencia para hallar todas las filas de K.

Este algoritmo implementado en Matlab se explica en el capítulo siguiente.

1.4 Margen de ganancia y margen de fase

La figura 1.4 muestra los diagramas polares de una función $G(jw)$ que no tiene polos positivos, para tres valores diferentes de ganancia (K) de lazo abierto. Para un valor grande de la ganancia, el sistema es inestable porque el diagrama encierra el punto -1,+j0. Al disminuir la ganancia hasta determinado valor, el lugar de $G(jw)$ pasa por el punto $-1+j0$.Esto significa que con este valor el sistema es exactamente inestable y que presentará oscilaciones mantenidas. Para un valor pequeño de la ganancia K el sistema es estable.

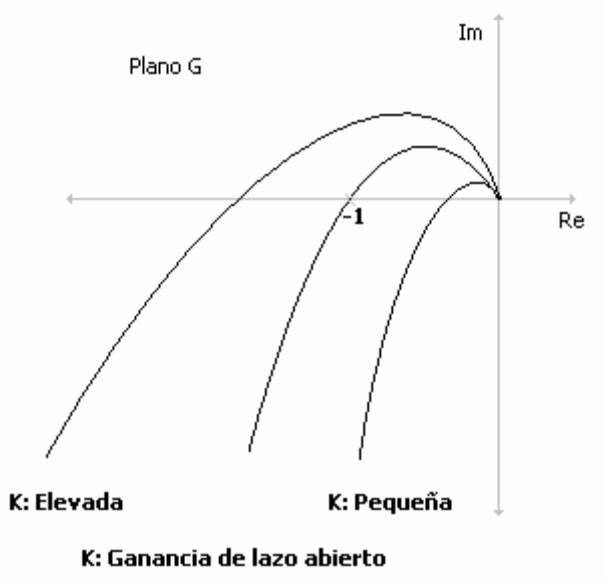


Figura 1.4: Diagramas polares

En general, cuando más cerca del punto $-1+j0$ pasa el lugar $G(jw)$, más oscilatorio se vuelve el sistema por lo tanto se puede utilizar esta proximidad del lugar $G(jw)$ al punto $-1+j0$ como una medida del margen de estabilidad. (No obstante esto no es válido en el caso de sistemas condicionalmente estables.) Es habitual representar esta cercanía en términos del margen de fase y del margen de ganancia.

El margen de ganancia, Gm , es la reciproca de $|G(jw_{cg})|$, donde $G(jw_{cg})$ es la Función de Transferencia del sistema en lazo abierto, a la frecuencia a la cual el ángulo de fase es -180 grados (w_{cg}) , ver figura 1.5

$$Gm = \frac{1}{|G(jw_{cg})|}$$

Expresado en decibelios

$$Gm(db) = 20\log Gm = -20\log|G(iw_{cg})|$$

El margen de ganancia expresado en decibelios es positivo si Gm es mayor que la unidad y negativo si es menor que la unidad. Por tanto, un margen de ganancia positivo (en decibelios) implica que el sistema es estable.

Para un sistema de fase mínima estable, el margen de ganancia indica cuanto se puede incrementar la ganancia antes de que el sistema se haga exactamente inestable. Para un sistema inestable, el margen de ganancia indica en cuanto hay que reducir la ganancia para que el sistema se haga estable.

El margen de fase es la cantidad de retardo de fase adicional necesaria a la frecuencia de cruce o de transición de ganancia (w_{cp}) para que el sistema quede al borde de la inestabilidad. La frecuencia de cruce de ganancia es aquella frecuencia para la cual el valor absoluto $|G(jw_{cp})|$ de la función Transferencia de lazo abierto, es la unidad. El margen de fase γ ver figura 1.5 puede definirse entonces como,

$$\gamma = 180^{\circ} + \phi$$

En el diagrama Nyquist o en el diagrama polar se puede trazar una línea desde el origen al punto en el cual el círculo de radio unidad cruza el lugar de $G(jw)$. El ángulo desde el eje real negativo a esta línea es el margen de fase. El margen de fase es positivo para $\gamma > 0$ y negativo para $\gamma < 0$. Para que un sistema de fase mínima sea estable el margen de fase debe ser positivo.

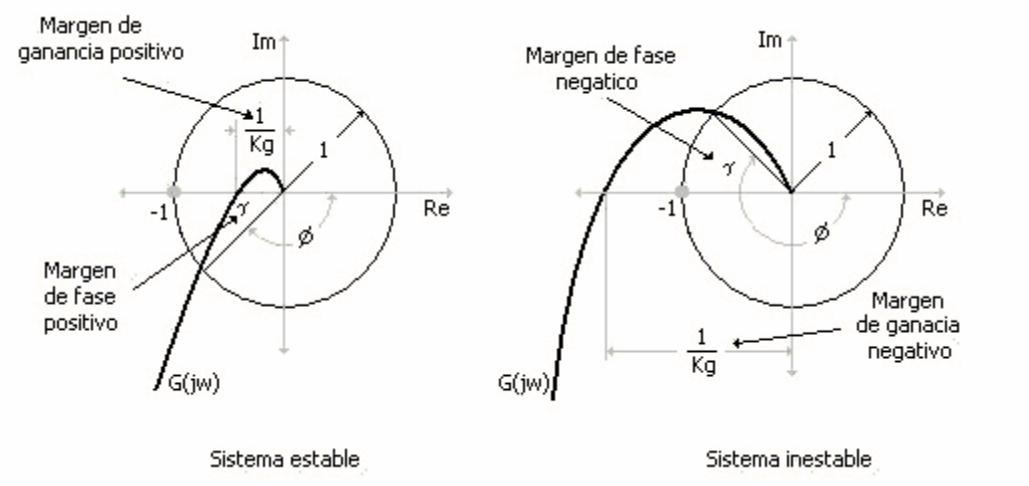


Figura 1.5: Margen de fase y Ganancia de sistemas estables e inestables

La herramienta computacional Matlab tiene dos funciones que permiten hallar los márgenes de fase y ganancia (margin.m o allmargin.m) así como sus respectivas frecuencias de cruce o de transición. En el caso de que la fase de un sistema no contemple el valor de -180 grados para ningún valor de frecuencia la función (margin) considerará que el margen de ganancia es infinito. Lo mismo ocurre con el margen de fase, pues si la magnitud de $|G(jw)|$ nunca es uno entonces en la función (margin) considerará que el margen de fase es infinito y por tanto la conclusión a que se llega es que el sistema es estable, lo cual puede ser cierto o no, veamos un ejemplo.

Sea G(s) la Función de Transferencia de lazo abierto de determinado proceso

$$G(s) = \frac{s^3 + 2s^2 + 3s + 4}{s^4 + 3s^3 + 5s^2 + 7.998s + 8.999}$$

Si hallamos los márgenes de ganancia y fase a este sistema con Matlab usando la función margin.m, da que:

Margen de Ganancia: 5.06 db a la frecuencia 1.67 rad/seg

Margen de Fase: Infinito

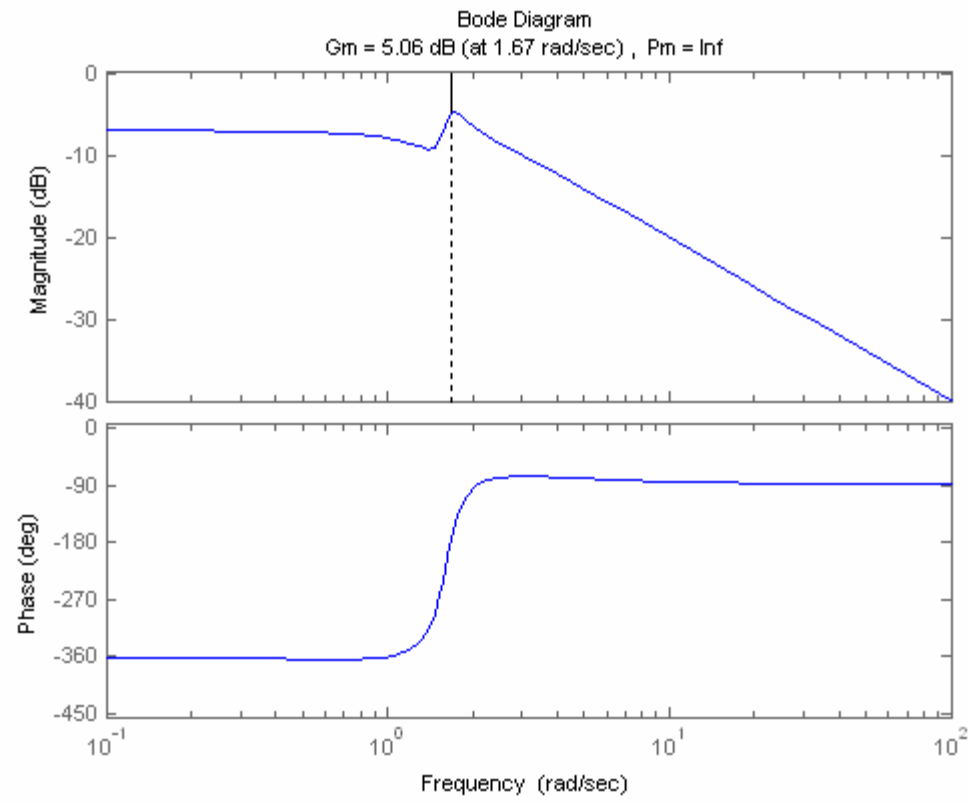


Figura 1.6: Diagrama de Bode de G(s) trazado por la función margin del Matlab

Con estos resultados pudiéramos afirmar que el sistema es estable lo cual no es cierto, si le aplicamos una entrada paso después de haber cerrado el lazo se obtiene lo que se muestra en la figura 1.7

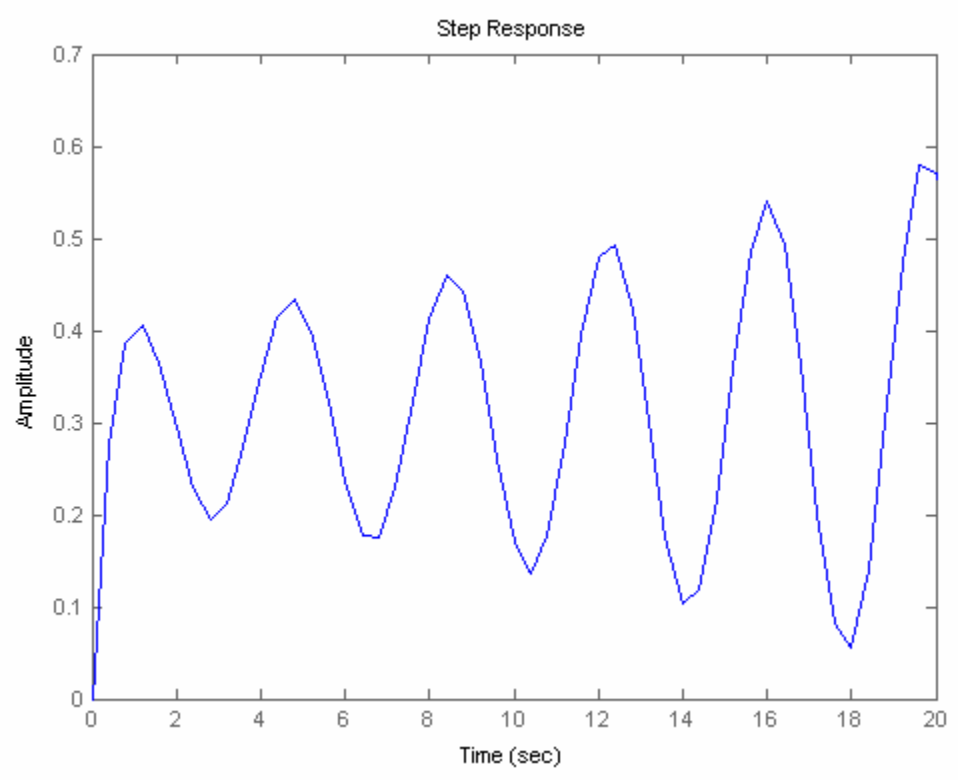


Figura 1.7: Respuesta al paso de un sistema inestable

Como podemos observar el sistema es inestable, los resultados son falsos. Una posible salida que se implementó para la solución de este problema es no decir que los márgenes son infinitos cuando no se pueden hallar, sino simplemente aclarar mediante una variable (NaN), que no se puede encontrar.

1.5 Diagrama de Bode

Se puede representar una Función de Transferencia sinusoidal, por dos diagramas distintos, uno que da la amplitud en función de la frecuencia y otro el ángulo de fase en función de la frecuencia. Un diagrama logarítmico o diagrama de Bode consta de dos trazados: uno es un diagrama del logaritmo del modulo de una función de Transferencia sinusoidal; el otro es un diagrama del ángulo de fase. Ambos son representados en función de la frecuencia en escala logarítmica.

1.5.1 Diagrama de Bode asintótico

La principal ventaja del diagrama de Bode asintótico es que se puede convertir la multiplicación de amplitudes en adición. Además se dispone de un método simple para trazar la curva aproximada del logaritmo de la amplitud. Esta basado en la aproximación asintótica. Esa aproximación por líneas rectas, asíntotas, es suficiente si solo se necesita una información global sobre las características de respuesta de frecuencia y en caso de necesitarse curvas exactas, se puede fácilmente efectuar correcciones a esas determinaciones asintóticas básicas. Además en todos los libros de texto, para el análisis y diseño de sistemas de control (análisis de estabilidad, compensación, etc) se trabaja con el diagrama asintótico y no con el real.

Para trazar el diagrama de Bode asintótico, primero se escribe la función Transferencia sinusoidal $G(jw)H(jw)$ como un producto de factores básicos, los cuales son:

Ganancia K

Factores integrales y derivativos $(jw)^{\pm 1}$

Factores de primer orden $(1+jwT)^{\pm 1}$

Factores cuadráticos $\left[1+2\zeta\left(\frac{jw}{w_n}\right)+\left(\frac{jw}{w_n}\right)^2\right]^{\pm 1}$

Luego se identifican las frecuencias de transición asociadas con estos factores básicos, $\frac{1}{T}$ para factores de primer orden y w_n para factores cuadráticos; a partir de cada frecuencia de transición se traza la asíntota con la pendiente que adiciona cada factor, ± 20 db/dec si se trata de un factor de primer orden, $\pm$ 40 db/dec si se trata de un término cuadrático. Los factores derivativos o integrales puros no presentan frecuencias de transición sino que

aportan la misma pendiente a lo largo de todo el espectro de frecuencia que estemos analizando y su influencia será más clara y definitoria en la asíntota de baja frecuencia.

Finalmente se trazan las curvas asintóticas del logaritmo de la amplitud con las pendientes adecuadas entre las frecuencias de transición, esto se logra sumando la ganancia de la Función de Transferencia (en decibelios), es decir $20\log(K)$ y todas las asíntotas correspondiente a cada factor de la Función de Transferencia.

MATLAB tiene una función que permite hallar el diagrama de Bode asintótico (bodeasym.m), ver figura 1.8 a modo de ejemplo, pero la misma es poco didáctica por varias razones, el ambiente donde aparece el diagrama es diferente al resto de los que comúnmente usa, el estudiante no puede interactuar con ella y exterioriza poca información, además no está documentada en la ayuda del Matlab. En el capitulo 2 se explica el algoritmo y funcionamiento de una función (nbode.m) que permite hallar el diagrama de Bode asintótico y hacerlo lo mas interactivo posible.

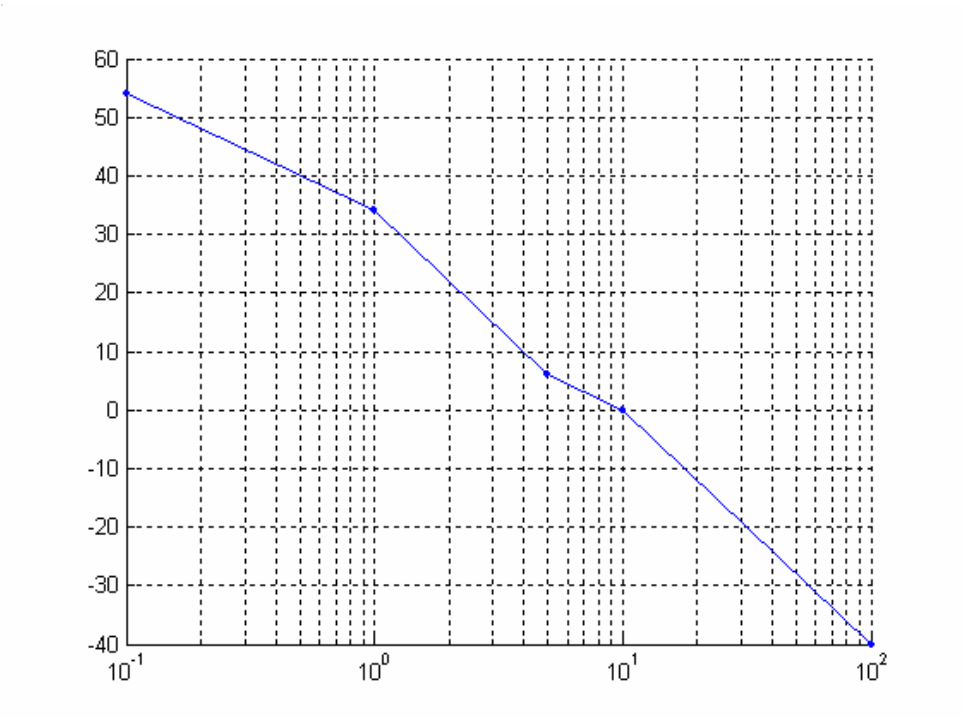


Figura 1.8: Diagrama de Bode asintótico trazado por la función bodeasym del Matlab

1.5.2 Diagrama de Bode real

El diagrama de Bode real se puede trazar o bien a partir del diagrama asintótico o evaluando la función Transferencia sinusoidal en cada valor de la frecuencia, para un rango de frecuencia dado.

El MATLAB presenta una función para trazar el diagrama de Bode real (bode.m) que en la mayor parte de los casos que se nos presentan puede dar un resultado apropiado, pero presenta dificultades cuando se trata de un sistema que tenga polos complejos conjugados sobre el eje imaginario, a continuación analizaremos este caso.

Sea el sistema de segundo orden

$$G(jw) = \frac{1}{1 + 2\zeta\left(j\frac{w}{w_n}\right) + \left(j\frac{w}{w_n}\right)^2}$$

Tiene una frecuencia de resonancia.

$$w_r = w_n\sqrt{1 - 2\zeta^2}$$

Y el valor del pico de resonancia.

$$Mr = |G(jw)|_{\max} = |G(jw_r)| = \frac{1}{2\zeta\sqrt{1-\zeta^2}}$$

En caso de que los polos de la función Transferencia estén sobre el eje *jw* la frecuencia de resonancia es igual a w_n, pues $\zeta = 0$, la magnitud de la función Transferencia sinusoidal $G(jw_n) = Mr$ es infinito por lo que el diagrama de Bode debe tender a una asíntota en la frecuencia w_n, es decir el diagrama debe abrirse a esa frecuencia.

Sin embargo cuando el Matlab traza el diagrama de Bode de un sistema cuya Función de Transferencia tenga polos complejos conjugados sobre el eje imaginario, no se abre a la frecuencia w_n, por ejemplo, sea $G(jw)$ una función Transferencia de lazo cerrado.

$$G(jw) = \frac{1}{(jw)^2 + 11}$$

$$G(jw) = \frac{\frac{1}{11}}{\left(j\frac{w}{11} + 1\right)}$$

Para esta Función de Transferencia el diagrama de Bode de magnitud debe tener la forma que se muestra en la figura que aparece a continuación.

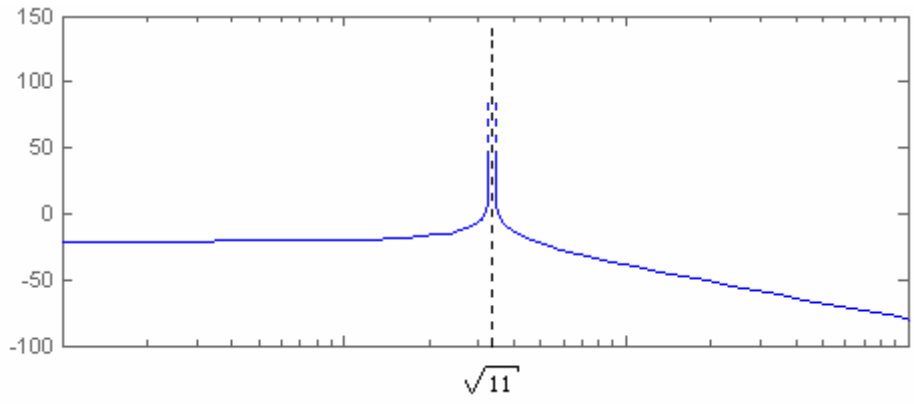


Figura 1.9: Diagrama de Bode de magnitud de un sistema con polos en ele eje imaginario

En este caso el diagrama de Bode se abre (Tiende a infinito) a la frecuencia $w_r = w_n = \sqrt{11}$ rad/seg, pues $Mr = \frac{1}{0} \to \infty$ debido a que $\zeta = 0$.

Usando el MATLAB para trazar el diagrama de Bode de esta función Transferencia se obtiene el siguiente grafico.

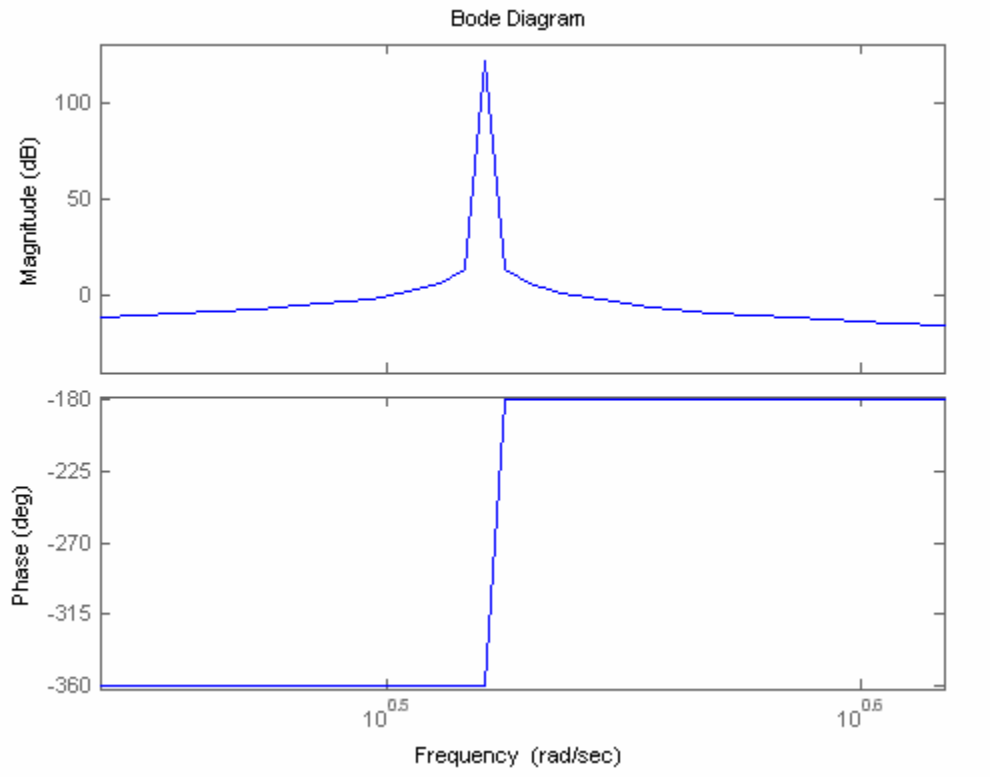
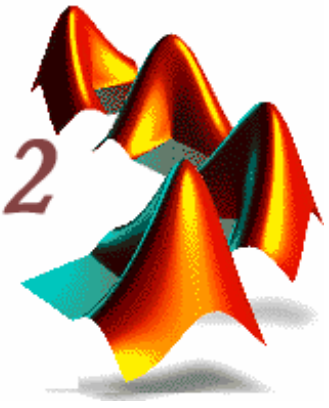


Figura 1.10: Diagrama de Bode de un sistema con polos en el eje imaginario trazado por la función bode del Matlab

Como se observa en el diagrama a la frecuencia $\sqrt{11} = 3.3166 = 10^{0.5207}$ rad/seg existe un valor real para la magnitud, que el estudiante puede obtener, lo cual no es cierto, pues a esa frecuencia no es posible obtener la magnitud, teóricamente debe ser infinita y el diagrama debe tender a una asíntota. Para resolver esta dificultad también se ha creado una función que se explicará en el capítulo siguiente.

Capítulo 2



CAPITULO II. Nuevas funciones

En este capitulo explicaremos los algoritmos elaborados para implementar cada función que hemos creado en el MATLAB con vistas a la obtención de la respuesta de frecuencia de sistemas lineales así como para el análisis de la misma y la utilización de esta para el diseño.

2.1 Diagrama de Nyquist

Para trazar el diagrama de Nyquist se decidió dividir el mismo en tres funciones, que son: mnyquist.m, dnyquist.m, nyfig.m.

mnyquist.m se encarga de manipular los datos de entrada y de salida.

dnyquist.m calcula los datos del diagrama de Nyquist, entiéndase como datos, los valores de la parte real, parte imaginaria y la frecuencia de todos los puntos que forman parte del diagrama de para un sistema dado.

nyfig.m manipula los gráficos para trazar el diagrama de Nyquist.

En el Anexo1.1 (b) se muestra la relación que existe entre las funciones mencionadas anteriormente. La principal función es mnyquist.m, ella es la que manipula las demás

2.1.1 Función mnyquist.m

La función mnyquist.m Anexo 1.1 se hace con el objetivo de poder entrar los datos del sistema de diferentes formas, ya sea pasando el numerador y denominador de la función Transferencia o pasando la función de Transferencia propiamente, y puede, obtener el diagrama de Nyquist o el conjunto de puntos de la sección I que forman el mismo (Parte Real, Parte Imaginaria, Frecuencia).

mnyquist(sys), **mnyquist(num,den)**: Traza el diagrama de Nyquist en una figura, sys es una variable del tipo Función de Transferencia "tf"; num, den son dos vectores que contienen los coeficientes del numerador y denominador de la Función de Transferencia.

Con intrusiones como **[Re,Im,w] = mnyquist(sys)**, **[Re,Im,w] = mnyquist(num,den)**, **[Re,Im] = mnyquist(sys)**, **[Re,Im] = mnyquist(num,den)** se pueden obtener los puntos que forman la sección I del diagrama de Nyquist.

Donde:

Re = Parte Real.

Im = Parte Imaginaria.

w = Frecuencia.

Cuando se ejecuta esta función Anexo 1.1 (b), primeramente se chequean los argumentos de entrada con el objetivo de comprobar si son correctos, en caso de serlo se obtiene la Función de Transferencia en una variable del tipo "tf", la cual se le pasa como parámetro de entrada a la función dnyquist.m que devuelve una variable de tipo estructura llamada ***Data*** que contiene toda la información necesaria para trazar el diagrama de Nyquist. Finalmente se chequea si existen variables de salida, en caso de que haya, se comprueba si la cantidad es correcta y se almacena, en la primera variable la parte real, en la segunda la parte imaginaria y si hay una tercera se almacena la frecuencia, todos de la sección I. Si no hay variables de salida entonces se procede a trazar el diagrama de Nyquist ejecutando la función NyFig.m, a la cual se le pasa como parámetro la variable ***Data***.

2.1.2 Función dnyquist.m

La función dnyquist.m Anexo 1.2 calcula los puntos que forman el diagrama de Nyquist, para ello se divide el diagrama en cuatro secciones, las mismas que define el recorrido de

Nyquist. Se chequea si el sistema tiene polos en el eje imaginario, si no tiene, se procede a hallar los puntos que forman cada sección.

Para calcular la sección I, en la función Transferencia se sustituye **s** por **jw** y se le halla el límite para cuando la frecuencia (w) tiende a cero y a infinito, esto se hace usando la función *limit.m* del MATLAB.

La función Allmargin del Matlab ofrece las posibles frecuencias a las cuales la fase de un sistema toma el valor -180º, la usamos entonces como un artificio del cual nos valemos para tener una frecuencia inicial de la cual partir para hacer los cálculos de la magnitud y fase con los cuales trazaremos los diagramas de Nyquist. Si la respuesta del Allmargin es vacía, la frecuencia inicial que tomamos es $w = 1$, si no, tomamos como referencia la menor de las frecuencias de cruce o transición de fase.

La frecuencia hallada anteriormente se comienza a multiplicar por diez de forma cíclica con el objetivo de hallar un valor de frecuencia en que la magnitud y fase del sistema estén bastante cerca de los valores que les corresponden cuando *w* tiende al infinito, considerando un margen de error. Para hacer el proceso contrario, se divide la frecuencia por diez hasta acercarnos con un margen de error también a la magnitud y fase correspondientes a la frecuencia cero.

Con los extremos de frecuencia se crea un vector comprendido en este rango y se calcula la parte real e imaginaria del sistema evaluado en dicho vector.

Para trazar la sección II se sustituye en la función Transferencia **s** por $re^{j\Theta}$ y se evalúa la función Transferencia para cuando r es igual al máximo valor de la frecuencia hallada en la sección I, y el ángulo va desde $\frac{\pi}{2}$ hasta $-\frac{\pi}{2}$. Con este conjunto de valores obtenemos parte real y parte imaginaria de todos los puntos que forman esta sección.

La sección III es igual que la sección I pero con el signo de la parte imaginaria invertido.

La sección IV se halla igual que la sección II pero tomando el primer valor de la frecuencia hallada en la sección I, en este caso el ángulo va desde $-\frac{\pi}{2}$ hasta $\frac{\pi}{2}$.

En caso de que el sistema tenga polos en el eje *jw* se uso un algoritmo que estaba implementado en una función llamada nyquist1.m que devuelve el conjunto de puntos de las cuatro secciones que forman el diagrama de Nyquist. Esta función esta publicada en el sitio Web. http://www.fiobera.unam.edu.ar/Materias/ControlDigital/lnyquist1.html

Al final, independientemente si el sistema tiene o no polos sobre el eje imaginario, todos los puntos se almacenan en una variable de tipo estructura que contiene los diferentes puntos de las cuatro secciones. Ver Anexo 1.2 (b)

2.1.3 función Nyfig.m

La función nyfig.m Anexo 1.3 es la encargada de visualizar el diagrama de Nyquist, para lo cual crea una figura, con un sistema de ejes de coordenadas en los cuales traza el eje real e imaginario con líneas discontinuas , marca al punto (-1;0) con una cruz y en un marco debajo se especifican notas aclaratorias del diagrama en cuestión. En el sistema de ejes de coordenadas se trazan las diferentes secciones del diagrama de Nyquist y cada sección tiene un color diferente. La sección I es de color azul, sección II, verde, sección III, roja y la sección IV, gris.

2.2 Bandas de Gershgorin

Para trazar las bandas de Gershgorin se usa la función gersh.m. Anexo 2

gersh(G,W) Dibuja las bandas de Gershgorin de los elementos de la diagonal de una

Matriz dinámica G cuadrada. Crea una figura para las bandas de cada elemento de la diagonal de la matriz dinámica.

gersh(G, W, ´single´) Dibuja las bandas de Gershgorin en una sola figura.

Donde:

G Es la inversa de la matriz dinámica del sistema, que debe ser cuadrada

W Es un vector de 2 elementos, [mínimo máximo] que contiene el inicio y fin del vector de frecuencia para el cual se quiere trazar las bandas de Gershgorin.

Para trazar las bandas de Gershgorin es necesario hallar los centros de los círculos y sus respectivos radios.

El centro de los círculos se halla evaluando los elementos de la diagonal $\hat{G}(i,i)$, $\hat{G}$ es la inversa de la Matriz de Transferencia del sistema, i es el numero de filas de la Matriz de Transferencia, con un vector de frecuencia $w = w_1, w_2, \ldots, w_N$ que se crea usando la función logspace.m del Matlab, Este vector define el rango para el cual es usuario quiere analizar la dominancia del sistema.

Los radios de los círculos se hallan evaluando los elementos de cada fila, con un vector de frecuencia, exceptuando los que pertenecen a la diagonal y sumándolos. El vector de frecuencia que se usa es el mismo que se uso en el caso anterior.

La función gersh traza una línea de color rojo que es la unión de todos los centros de los círculos y cada círculo los traza de color azul.

2.3 Seudo Diagonalización y su ampliación

La función findk.m Anexo 3 es la encargada de hallar la matriz $\hat{K}$ que hace a $\hat{G}(iw)$ dominante por fila, dentro de ella están implementados los dos algoritmos explicados en el capítulo anterior, es decir el método de la Seudo-Diagonalización y su ampliación.

K = findK(Gp, w, Gamma): Halla la matriz K que hace a la matriz dinámica inversa del sistema dominante por fila, donde Gp es la matriz dinámica del proceso, la cual debe ser cuadrada.

[K] = findK(Gp): es equivalente a K = Gp(0). Si Gp(0) es real.

[K] = findK(Gp, w): Halla la matriz K para un valor de w dado, (Seudo-Diagonalización).

[K] = findK(Gp, w, Gamma): Halla la matriz K para un conjunto de frecuencias (w) y un vector de peso Gamma, Gamma y w deben tener la misma longitud, (Ampliación de la Seudo-Diagonalización).

Esta función usa el mismo vector de frecuencia para calcular todas las filas de K. Al final da la opción de trazar las bandas de Gershgorin para chequear la dominancia.

La función findk.m cuando es ejecutada, en dependencia de la cantidad de variables de entrada decide que método ejecutar.Si se entra una variable, la Matriz de Transferencia, entonces la matriz K se obtiene evaluando esta con frecuencia cero. Si se entran dos variables, la Matriz de Transferencia y un valor de frecuencia, se usa el método de Seudo Diagonalización y si se entra tres variables, Matriz de Transferencia, vector de frecuencia y un vector de peso, se usa el método de la Seudo Diagonalización ampliado.

La función findk.m usa una función llamada invtf.m para hallar la inversa de la matriz dinámica. En la función invtf.m se transforma la Matriz de Transferencia a una variable de tipo simbólico, se halla la inversa a la transformación y después se vuelve a transformar a Matriz de Transferencia. Anexo 3.2

2.4 Diagramas de Bode

Para trazar el diagrama de Bode, el proceso se dividió en tres parte, una función que manipula los datos de entrada y de salida (nbode.m), otra función que calcula el conjunto de puntos que forman el diagrama de Bode real y asintótico (bode1.m), y la última función que dibuja el diagrama de Bode (vbode.m). Anexo 4.1 (a)

La función nbode es la que manipula las demás funciones y en dependencia de los parámetros de entrada y de salida traza el diagrama de Bode en una figura o devuelve el conjunto de puntos que lo forman, es decir, magnitud, fase y frecuencia

2.4.1 Función nbode.m

La función nbode.m Anexo 4.1 se creó con el objetivo de poder pasar los datos del sistema de varias formas además de poder entrar varios sistemas y que todos los diagramas puedan obtenerse en la misma figura.

nbode(sys) ó **nbode(num, den)**: Trazan diagramas de Bode asintóticos, donde sys es un función de Transferencia, y (num, den) son el numerador y denominador de la función de Transferencia respectivamente.

nbode(sys,'realtoo'), **nbode(num,den,'realtoo')**: Trazan los diagramas de Bode asintótico y real en una misma pantalla.

nbode(sys1,sys2,...):Traza el diagrama de Bode de varios sistemas, sys1, sys2,… son las respectivas Funciones Transferencia de cada sistema.

nbode(sys,sys2,...,'realtoo'): Traza el diagrama de Bode asintótico de varios sistemas conjuntamente con el real.

nbode(sys,sys2,...,w): Traza el diagrama de Bode en el rango de frecuencia especificado por w, w debe ser un vector.

[AF,fase] = nbode(...): Devuelve la magnitud y fase del diagrama de Bode asintótico, solo admite un solo sistema.

[AF,fase,frec] = nbode(...): Devuelve la magnitud, fase y frecuencia del diagrama de Bode asintótico.

[Gm, Pm, Wcg, Wcp] = nbode(...): Devuelve los márgenes de ganancia y fase y sus respectivas frecuentas de cruce.

Cuando se llama la función nbode, siempre se muestran en la ventana de comandos del Matlab los márgenes de ganancia y fase así como sus respectivas frecuencias de cruce, si al final de los parámetros de entrada se pasa la cadena '*realtoo*' el resultado es con respecto al diagrama de Bode real, de lo contrario los resultados son con respecto al diagrama de Bode asintótico.

La función nbode.m usa las funciones bode1.m, (ver epígrafe 2.4.2), para calcular los puntos que forman el diagrama de Bode, para lo cual le pasa como entrada una variable de tipo celda que contiene todos las Funciones de Transferencia de los sistemas que se quieran trazar, un vector de frecuencia vacío en caso de que el usuario no lo especifique, y una cadena de caracteres vacía en caso de que no se pase como parámetro de entrada ('realtoo'). La función bode1.m devuelve una variable de tipo celda que contiene la información necesaria para trazar el diagrama de Bode. Si no hay parámetros de salida se ejecuta la función vbode.m (ver epígrafe 2.4.3) que es la encargada de trazar el diagrama de Bode, de lo contrario vbode no se ejecuta y se almacenan los resultados en las variables de salida. Ver Anexo 4.1 (b)

2.4.2 Función bode1.m

La función bode1.m Anexo 4.2 es usada por la función nbode.m para calcular el diagrama de Bode asintótico y real. Para lo cual primeramente se hace un chequeo de la cantidad de Funciones de Transferencias que se le entraron a la función, en caso de que no se

especifique un vector de frecuencia hallar el mismo de forma tal que puedan verse todos los diagramas completamente. Después se procede a hallar cada diagrama por separado y al final todos se almacenan en una variable de tipo celda.

Para calcular el diagrama de Bode asintótico Anexo 4.2 (b) de un sistema determinado se normaliza el sistema, de forma tal que quede de siguiente forma.

$$sys = K_s \frac{a_n s^n + a_{n-1} s^{n-1} +, \cdots, + a_1 s + 1}{b_m s^m + b_{m-1} s^{m-1} +, \cdots, + b_1 s + 1}$$

Se hallan los polos y ceros y se organizan en dependencia si son reales o complejos o si están en el numerador o en el denominador. En función de esto se encuentran las respectivas frecuencias de transición así como las asíntotas que cada una incorpora. Al final sumamos todas las asíntotas y obtenemos el diagrama de Bode asintótico.

Esta función devuelve una variable tipo celda, que contiene un número de variables tipo estructuras igual a la cantidad de Funciones Transferencias que se le hayan pasado como entradas a la función. También calcula los márgenes de fase y de ganancia así como las frecuencias asociadas a estos.

2.4.3 Función vBode.m

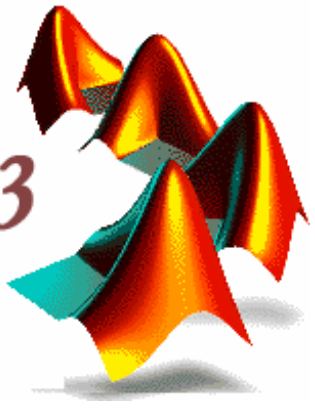
La función vbode.m Anexo 4.3 es usada por la función nbode.m y es la encargada de trazar el diagrama de Bode completo. Para lo cual primero se chequea si existe alguna figura activada y si está lista para adicionarle otros gráficos (*"hold on"),* en caso de que no lo esté entonces se crea una nueva figura con sus respectivos sistemas de coordenadas y se trazan los diagramas de Bode .Se visualiza en la ventana de comandos del Matlab los márgenes de ganancia y fase así como las frecuencias asociadas a estos.

Después se almacenan todas las Funciones de Transferencias, cuyos diagramas fueron pintados en la variable "UserData" de la figura, para posteriores usos.

Si hay una figura activa y la misma está lista para adicionarle nuevos gráficos entonces se chequea si lo que había anteriormente en ella eran diagramas de Bode, en caso de serlo entonces se busca la variable "UserData" de esta, que contiene las Funciones de Transferencias de los diagramas que están trazados y se les adicionan las nuevas funiones. Se recalculan los diagramas de Bode de cada sistema con el objetivo de obtener un vector de frecuencia que permita trazar completamente todos los diagramas, se visualizan los márgenes de ganancia y fase en la ventana de comandos del Matlab y se almacenan todas las Funciones de Transferencia en la variable "UserData de la figura. Ver Anexo 4.3 (b)

La función vbode.m traza el diagrama de Bode asintótico de color azul y el real en caso de que el usuario lo desee de color rojo.

Capítulo 3



CAPITULO III. Análisis de los resultados

En este capítulo se analiza y compara los resultados que se obtiene con la aplicación de las funciones creadas y las que actualmente ofrece el Matlab en varios casos de estudio.

3.1 Diagrama de Nyquist

Ejemplo 3.1.1

Sea G(s) la Función de Transferencia de lazo abierto de un sistema dado.

$$G(s) = \frac{100}{s(s+2)(s+10)}$$

Como se puede ver G(s) es una función de transferencia tipo uno, por lo que en la sección I del diagrama de Nyquist, cuando la frecuencia tiende a cero la magnitud tiende a infinito y la fase a -90^o y cuando la frecuencia tiende a infinito la magnitud tiende a cero y la fase a -270^o, la sección II es cero, la sección III es la imagen con respecto al eje real, de la sección I, y la sección IV es una circunferencia de radio infinito que va desde la fase -270º hasta -90º pasando por 0º.

Si trazamos el diagrama de Nyquist de G(s) usando la función nyquist.m del Matlab obtenemos lo que muestra la figura 3.1.

Podemos observar que en el diagrama de la figura 3.1 no se traza la sección IV, debido a la indefinición que se produce en el cálculo, esto deja incompleto el diagrama.

Si trazamos el diagrama de Nyquist de G(s) usando la función mnyquist.m entonces se obtiene lo que muestra la figura 3.2.

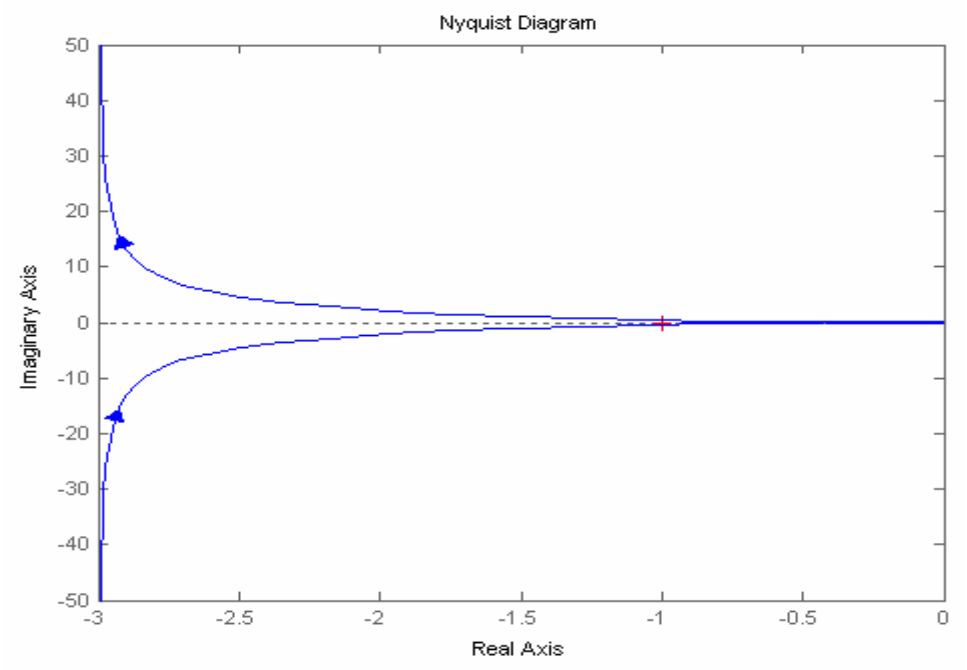


Figura 3. 1: Diagrama de Nyquist de G(s) usando la función nyquist del Matlab.

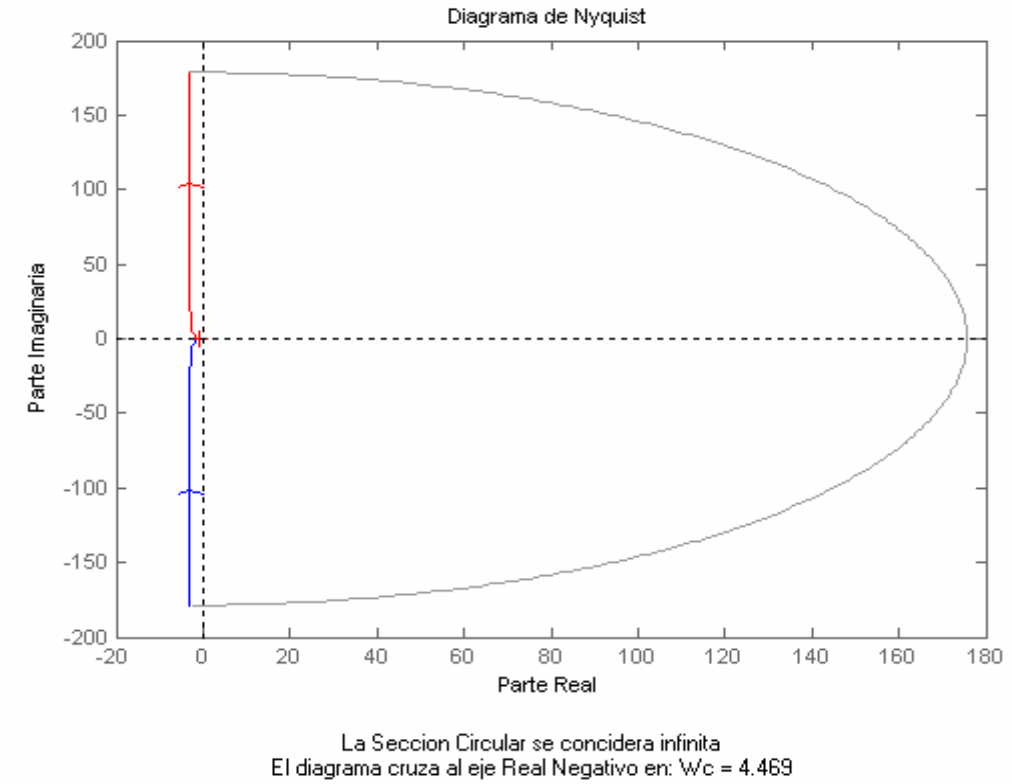


Figura 3. 2: Diagrama de Nyquist de G(s) usando la función mnyquist.m

Como podemos observar en la figura 3.2, si se traza la sección IV del diagrama de Nyquist además al final del gráfico se dan algunas notas aclaratorias, una de ella nos dice que el diagrama cruza el eje real negativo con frecuencia 4.47 rad/seg, lo cual llama la atención al estudiante para que se interese por saber que está pasando en esa zona del diagrama. Si se le aplica un "zoom" en la región próxima al punto (-1;0) entonces se observa lo que muestra la figura 3.3. Dando un "clic" encima del diagrama se puede escanear la sección completa y conocer la parte real, parte imaginaria y frecuencia de cada punto que la forma.

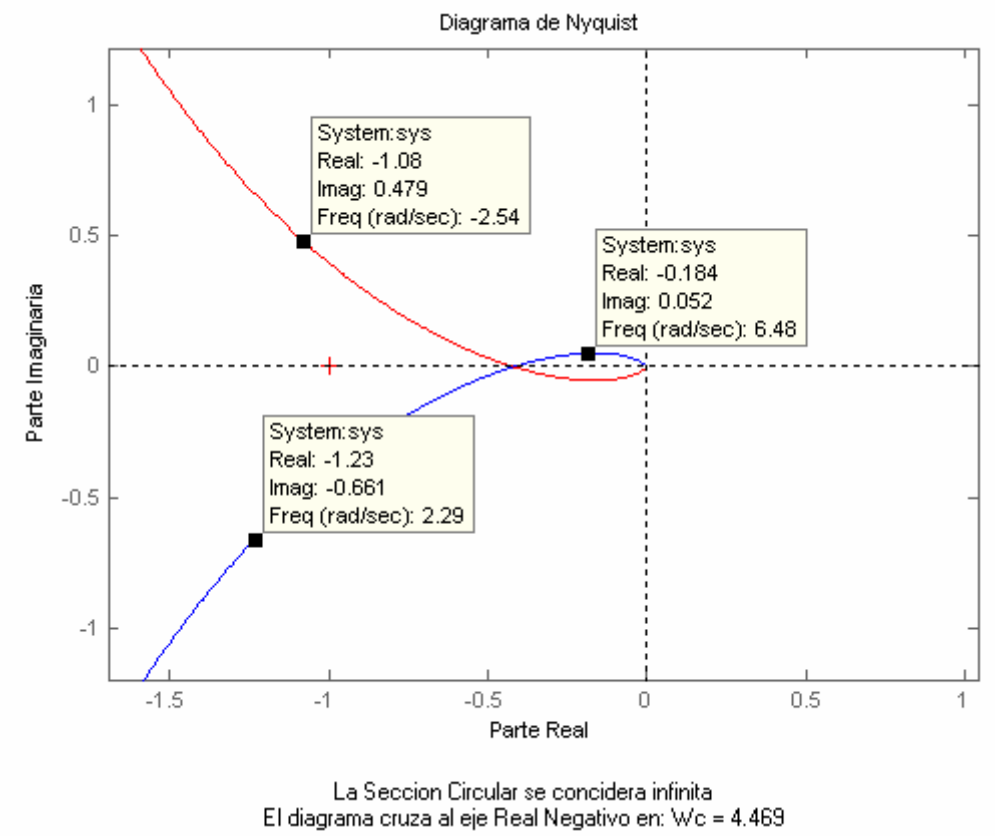


Figura 3. 3

Ejemplo 3.1.2

Sea G(s) la función de transferencia de lazo abierto de un sistema.

$$G(s) = 10\frac{s+5}{s(s-2)}$$

Como se puede observar el sistema tiene un polo en la mitad derecha del plano S por lo que para que el sistema en lazo cerrado sea estable, según el criterio de estabilidad de Nyquist, el diagrama de Nyquist debe circular una vez al punto (-1; 0) en sentido antihorario.

Si se usa la función Nyquist.m del Matlab para trazar el diagrama de Nyquist de G(s), se obtiene el diagrama mostrado en la figura 3.4, en este caso tampoco se traza la sección IV, aunque circula al punto (-1;0) una vez en sentido antihorario no podemos asegurar que el sistema sea estable o inestable pues el diagrama esta incompleto.

Si se traza el diagrama de Nyquist usando la función mnyquist.m se obtiene el diagrama que se muestra en la figura 3.5, en este caso el diagrama si esta completo y se puede asegurara que el sistema descrito por la Función de Transferencia G(s) es estable en lazo cerrado, además bajo el diagrama se dice que se cruza el eje real con una frecuencia de 3.15 rad/seg, haciendo “zoom” alrededor del punto (-1;0) y dando un “clic” sobre el diagrama en la parte azul (Sección I) y desplazando el “mouse” por encima se puede observar el conjunto de puntos que forman la sección I, de esta forma se puede saber cual es la parte real e imaginaria a la frecuencia de cruce (3.15 rad/seg). Figura 3.5.

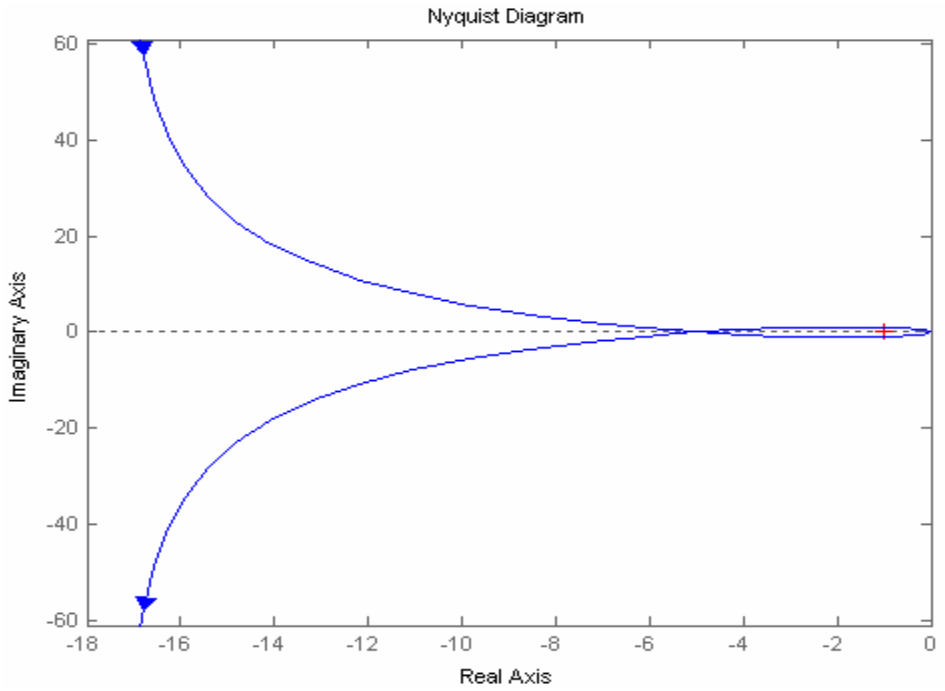


Figura 3. 4: Diagrama de Nyquist del ejemplo 3.1.2 usando la función nyquist.m del Matlab

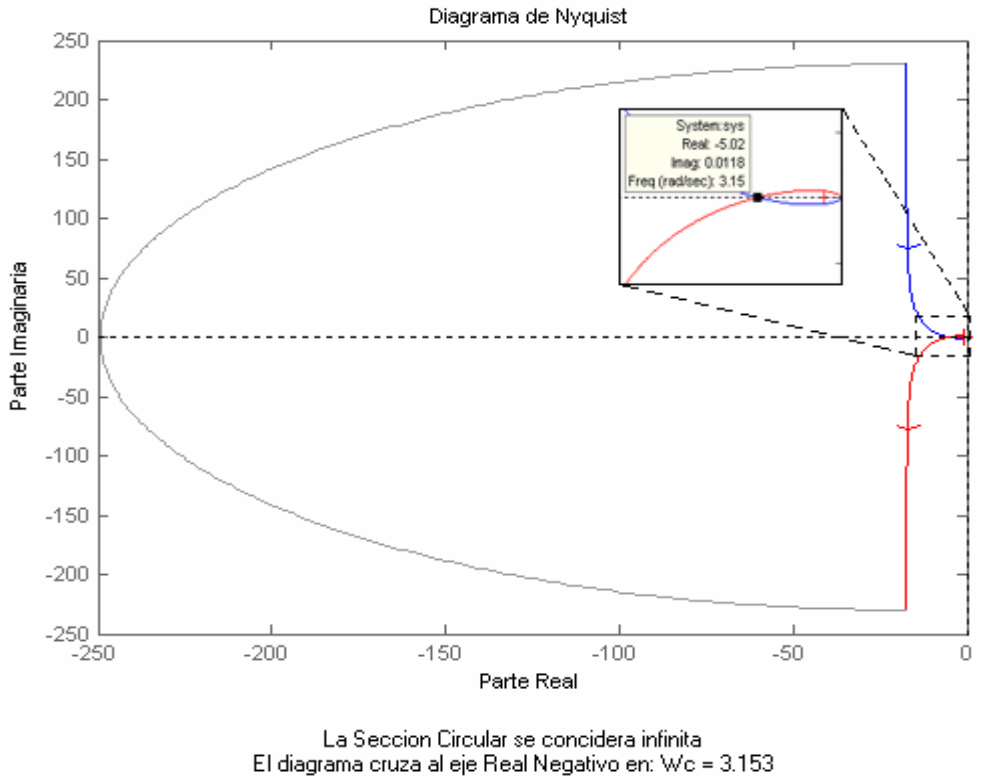


Figura 3. 5: Diagrama de Nyquist del ejemplo 3.1.2 usando la funciom mnyquist.m

Ejemplo 3.1.3

$$G(s) = \frac{10}{s^2 + 5}$$

Esta Función de Transferencia en lazo abierto tiene un par de polo complejos conjugados sobre el eje jw $s = 0 \pm j\sqrt{5}$ por lo que el sistema presenta una indefinición cuando la frecuencia es $w = \sqrt{5}$, en este caso para trazar el diagrama de Nyquist la sección I se divide en tres partes, una cuando la frecuencia va de cero a $\sqrt{5}$, otra que es $\lim_{R \to 0} G\left(s = \sqrt{5} + \mathrm{Re}^{j\theta}\right)$ donde θ va desde $-\frac{\pi}{2}$ hasta $\frac{\pi}{2}$, y la última es cuando la frecuencia va desde $\sqrt{5}$ hasta infinito.

Si analizamos estas tres subsecciones de la usual sección I de Nyquist a G(s), tendremos. Cuando la frecuencia tiende a cero la magnitud tiende a dos y la fase a 0º, si la frecuencia tiende a $\sqrt{5}$ por la izquierda la magnitud tiende a infinito y la fase a cero. Evaluando la expresión $\lim_{R \to 0} G\left(s = \sqrt{5} + \mathrm{Re}^{j\theta}\right)$ cuando θ va desde $-\frac{\pi}{2}$ hasta $\frac{\pi}{2}$, se obtiene una circunferencia de radio infinito que sale con ángulo cero, pasa por -270º y finaliza con ángulo -180º y cuando la frecuencia va desde $\sqrt{5}$ hasta infinito la magnitud toma valores desde menos infinito hasta cero y la fase siempre es -180º. La sección III del diagrama es la imagen de la sección I con respecto al eje real y las secciones II y IV son cero.

Si se traza el diagrama de Nyquist para este sistema usando la función nyquist.m del Matlab se obtiene lo que muestra la figura 3.6, como se puede observar esta dista mucho de la realidad. Con la función mnyquist.m si se puede obtener el diagrama de Nyquist completo, figura 3.7.

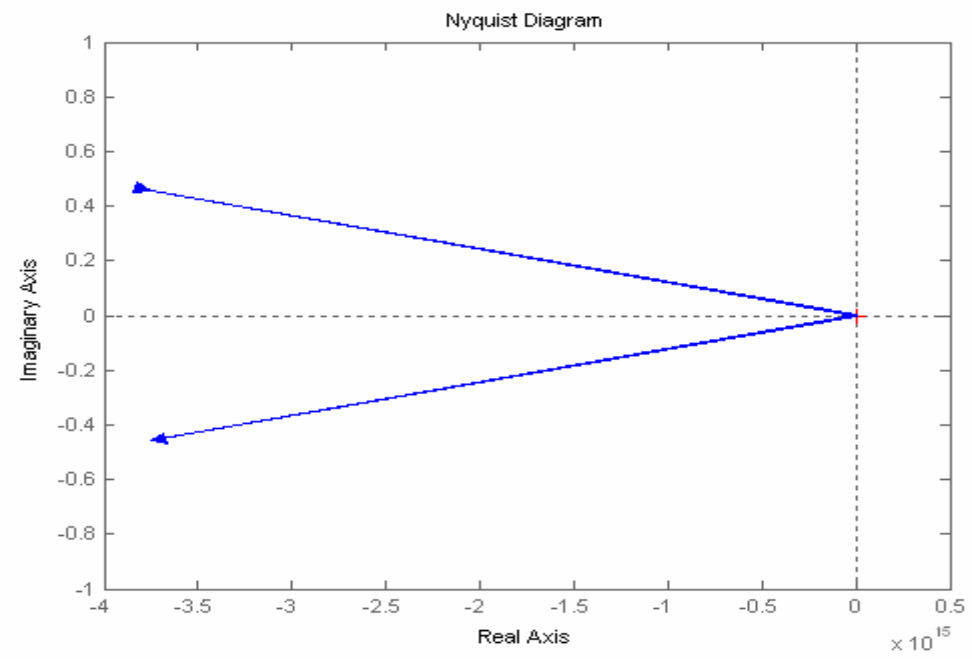


Figura 3. 6: Diagrama de Nyquist del ejemplo 3.1.3 usando la función nyquist.m del Matlab

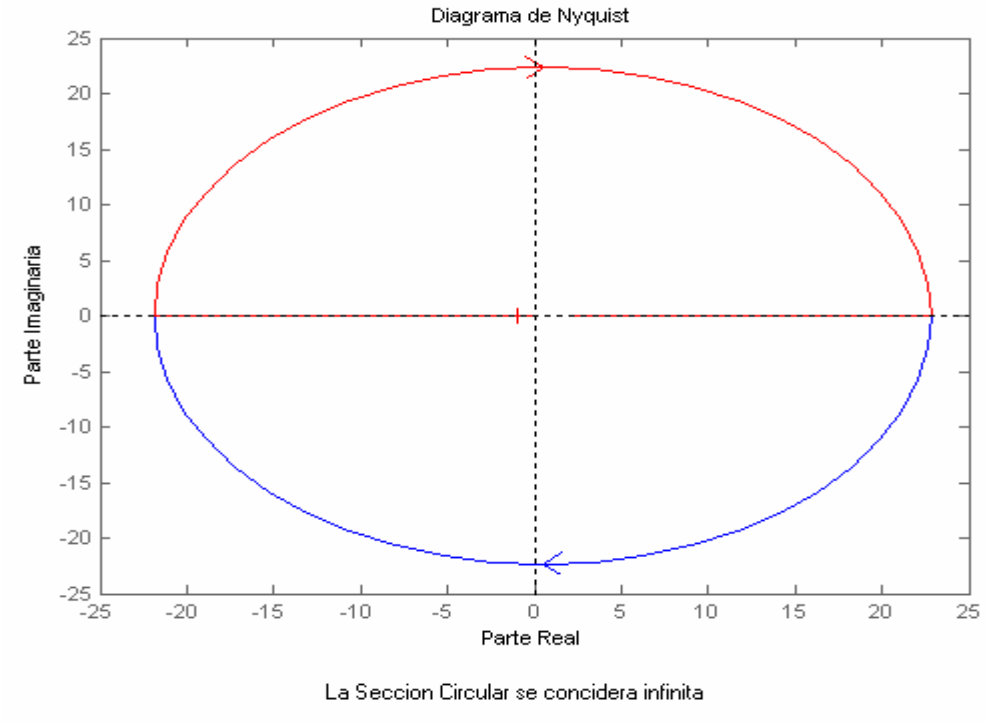


Figura 3. 7:Diagrama de Nyquist del ejemplo 3.1.3 usando la función mnyquist.m

3.2 Bandas de Gershgorin

Ejemplo 3.2.1

El modelo de una columna de destilación de treinta platos es descrito por la siguiente matriz de Funciones de Transferencia.

$$G(s) = \begin{bmatrix} \dfrac{0.088}{(1+75s)(1+722s)} & \dfrac{0.1825}{(1+15s)(1+722s)} \\ \dfrac{0.282}{(1+10s)(1+1850s)} & \dfrac{0.412}{(1+15s)(1+1850s)} \end{bmatrix}$$

Con el objetivo de analizar la dominancia del sistema se trazan las bandas de Gershgorin a la inversa de G(s) para un rango de frecuencias entre 0.0001 y 0.2 se obtiene lo que muestra la figura 3.8

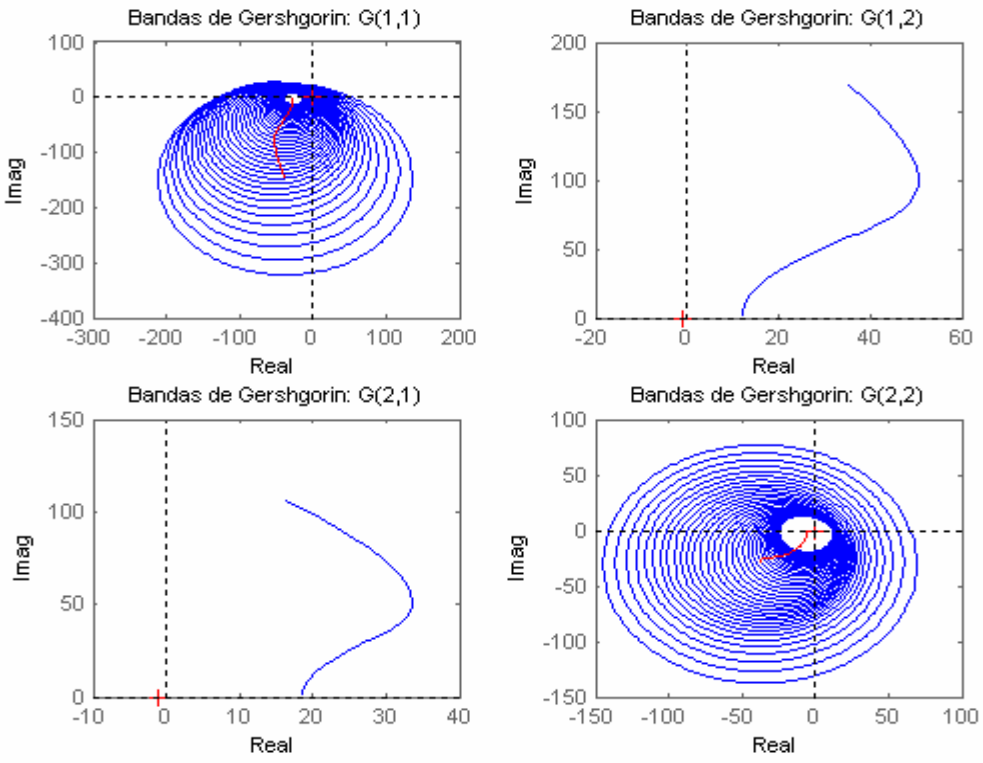


Figura 3. 8 Bandas de Gershgorin de la inversa de G(s)

Este modelo no es diagonalmente dominante por fila, los círculos de las bandas contienen al origen, por tanto es conveniente hallar una matriz K que haga que GK sea dominante por fila y por tanto $\hat{K}\hat{G}$ también.

Para hallar la matriz K se puede usar el método de la Seudo Diagonalización, o su ampliación, para este ejemplo se usara el método ampliado de la Seudo Diagonalización.

En la ventana de comandos del Matlab se crea un vector de frecuencia (w) entre 0.001 y 0.2, con un paso de 0.001 y un vector de peso Ganma que tomamos con valor uno para todas las frecuencias, y se llama la función findK.m para hallar la matriz $\hat{K}$ que hace $\hat{G}$ dominante por fila, figura 3.9. La función findK.m después de hallar la matriz $\hat{K}$ da la opción al usuario de trazar las bandas de Gershgorin de $\hat{K}\hat{G}$, en este caso se trazaron, figura 3.10.

```
>> w = 0.001:0.001:0.02; Ganma = ones(size(w));
>> K = findK(G,w,Ganma)
¿Desea chequear la dominancia? S/N [S]: s
Entre el rango de frecuencia:[0.001 0.02]

K =

         0.286534059754518         0.958070056207057
          0.56949410087425         0.821995419129224

>>
```

Figura 3. 9: Uso de la función findK en la ventana de comandos del Matlab

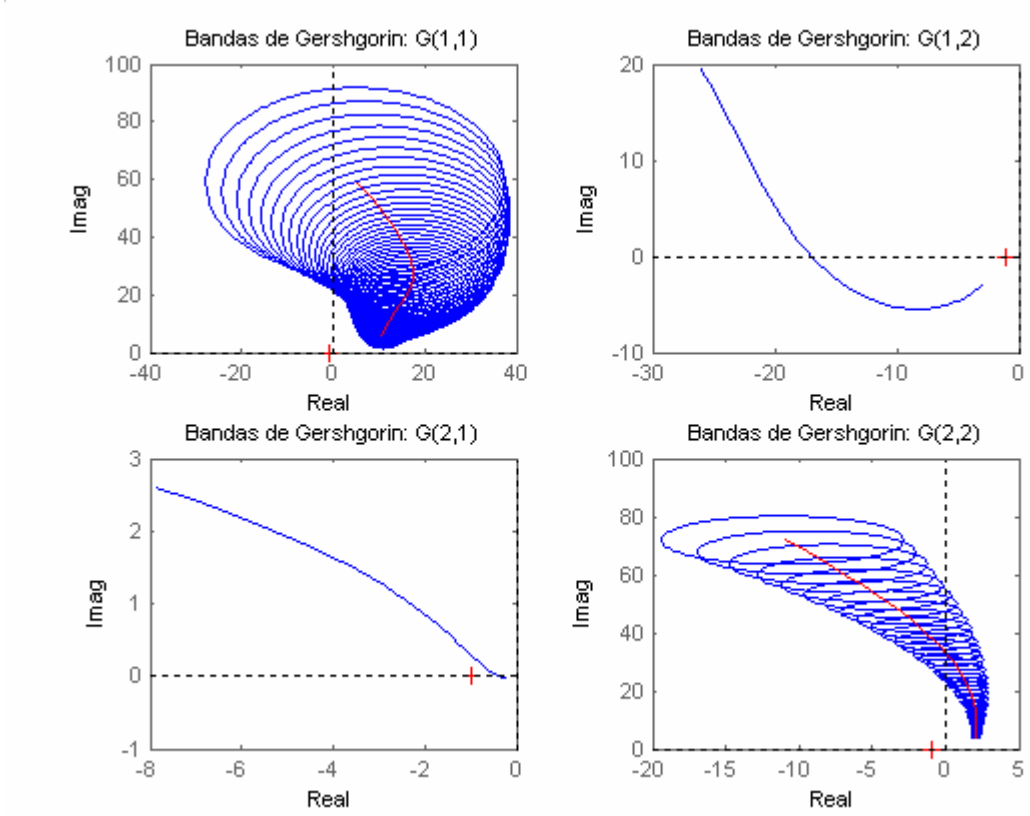


Figura 3. 10: Bandas de Gershgorin de $\hat{K}\hat{G}$

Como se observa $\hat{K}\hat{G}$ es diagonalmente dominante por fila

3.3 Diagrama de Bode

Ejemplo 3.3.1

Para trazar el diagrama de Bode asintótico de $GH(s) = \frac{100(s+5)}{s(s+1)(s+10)}$,el Matlab ofrece una función llamada bodeasym.m, usando esta función se obtiene lo que muestra la siguiente figura.

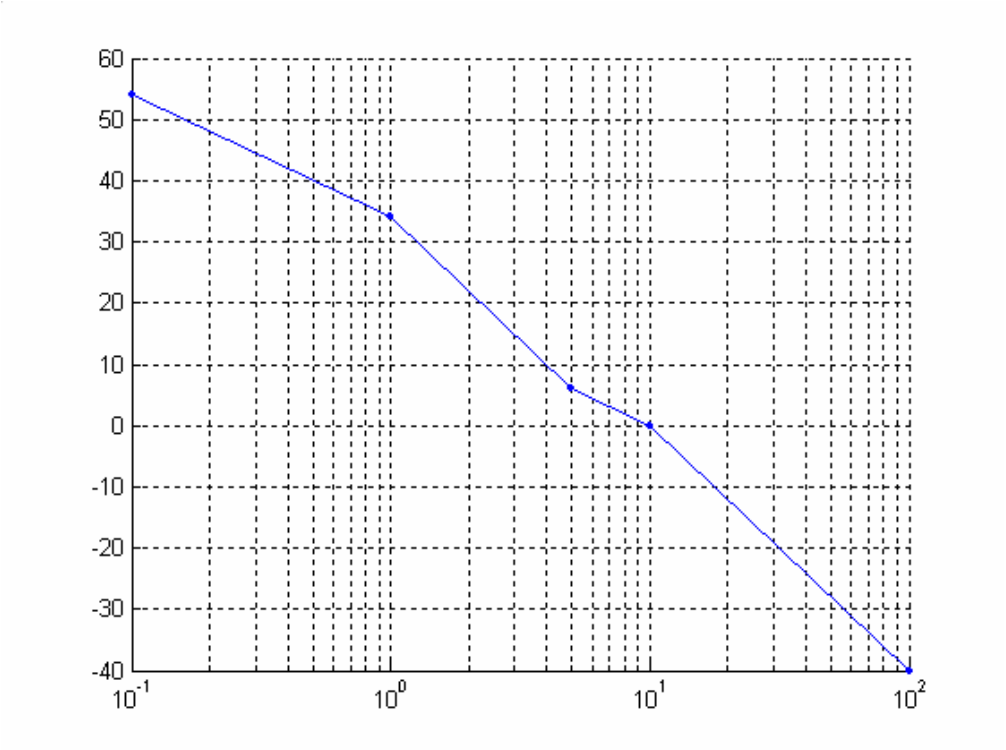


Figura 3. 11: Diagrama de Bode asintótico obtenido con la función bodeasym.m del Matlab

Observe que no se traza el diagrama de fase por tanto no se puede hacer ningún análisis serio de estabilidad**, solo inferir,** a partir de los cambios de pendiente. No es posible obtener ningún dato (magnitud, frecuencia) del gráfico "clicleando" sobre él como es usual en Matlab.y tampoco esta función ofrece la posibilidad de pasarle los datos del gráfico a un vector para trabajar con ellos.

Si utilizamos la función creada **nbode(sys)** obtendremos el diagrama de Bode siguiente.

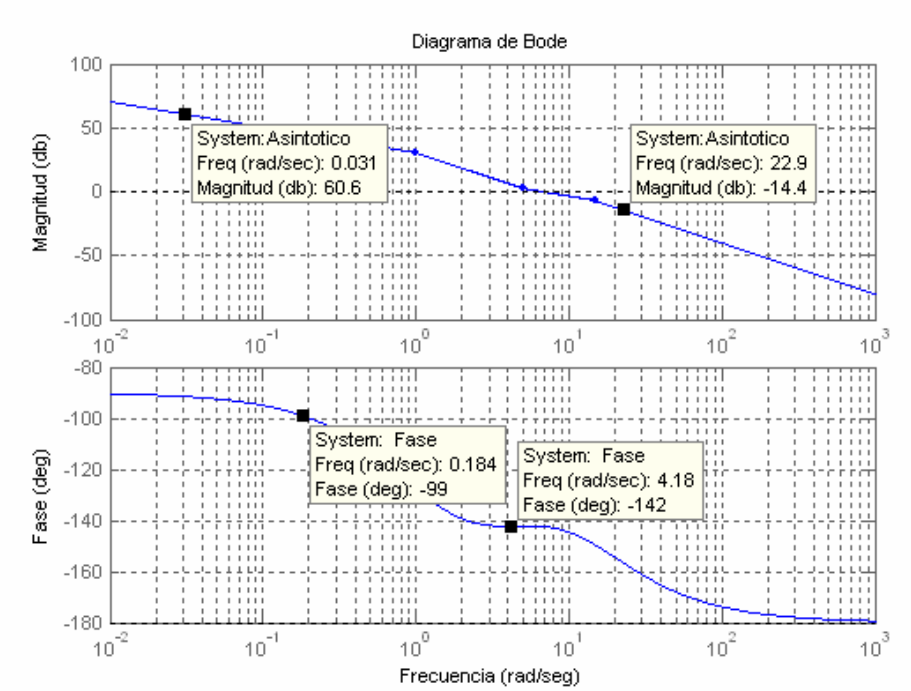


Figura 3. 12: Diagrama de Bode obtenido usando la función nbode.m

Observemos que:

El ambiente gráfico es el mismo que el estudiante ya conoce. Se tienen ambos diagramas, de magnitud y fase. Se puede obtener toda la información deseada y útil del grafico (observar en los recuadros).Adicionalmente aparecerá en la pantalla del Matlab los márgenes de fase y de ganancia del sistema, además con la función nbode.m se pueden almacenar todos los datos mencionados anteriormente en variables. Figura 3.13

```
>> [mg,fs] = nbode(sys);
>> [mg,fs,w] = nbode(sys);
>> [Gm,wcg,Pm,Wcp] = nbode(sys);
>>
```

Figura 3. 13

Ejemplo 3.3.2

Sea G(s) la Función de Transferencia en lazo abierto de un sistema.

$$G(s) = \frac{10}{(s^2 + 5)}$$

Como se puede observar G(s) presenta un par de polos complejos conjugados sobre el eje jw, por lo que cuando la frecuencia es $w = \sqrt{5}$ la magnitud de G(s) tiende al infinito por lo que el diagrama de Bode debe abrirse.

Usando la función bode.m del Matlab para trazar el diagramad de Bode se obtiene lo que se muestra en la figura.3.14

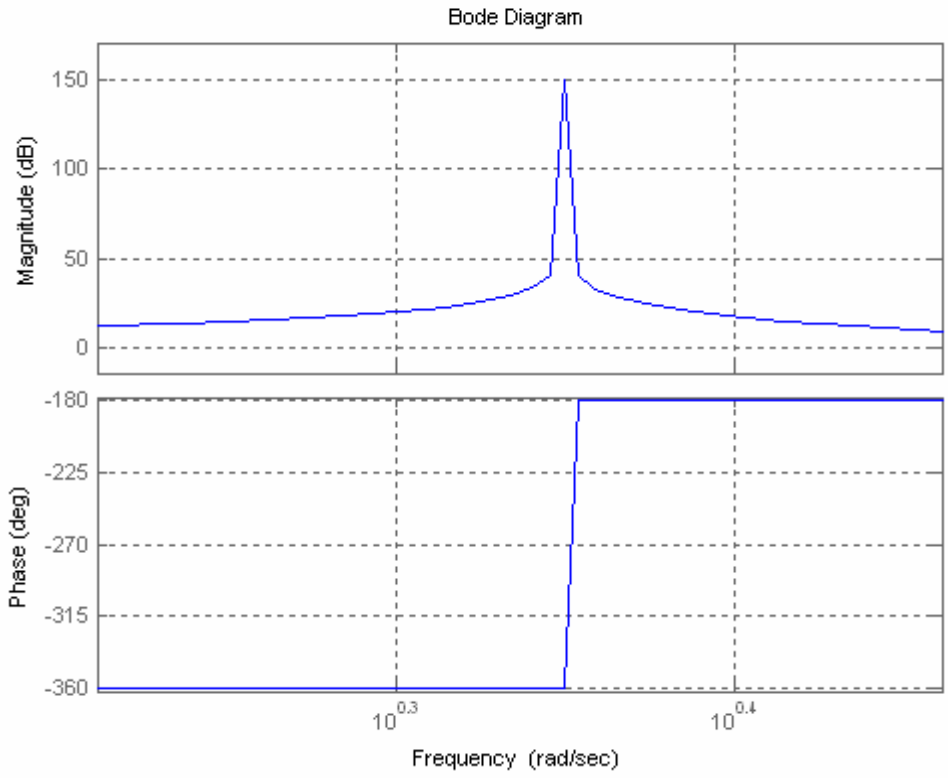


Figura 3. 14: Diagarama de Bode del ejemplo 3.3.2 trazado por la función bode.m del Matlab

Como se puede observar en el diagrama de Bode aunque hay cierta indicación de un valor muy grande por la forma que toma el diagrama en el pico, se puede inferir que este tiene un valor o sea que a la frecuencia $w = \sqrt{5} = 10^{0.35}$ rad/seg existe una magnitud finita, (aproximadamente 150 db), lo cual no es cierto, pues como se había dicho anteriormente a esa frecuencia la magnitud tiende a infinito.

Si se usa la función nbode.m, figura 3.15 para trazar el diagrama de Bode de G(s) se obtiene lo que muestra la siguiente figura.

```
>> sys  = tf(10, [1 0 5]);
>> nbode(sys,'realtoo')
>>
```

Figura 3. 15

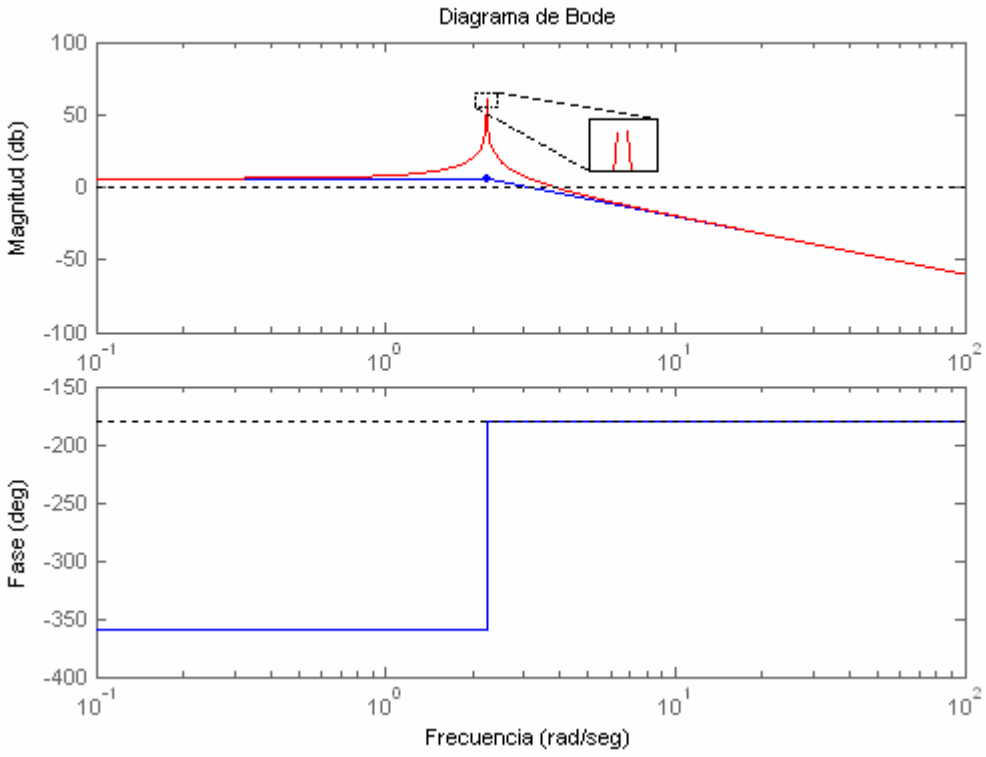


Figura 3. 16: Diagrama de Bode del ejemplo 3.3.2 hallado con la función nbode.m

Observemos que si se aplica "zoom" en la región del pico en el diagrama de magnitud este se abre, que es lo correcto.

Ejemplo 3.3.3

Sea GH(s) la Función de Transferencia de un sistema $GH(s) = \frac{(0.2s+1)(0.025s+1)}{s^3(0.001s+1)(0.005s+1)}$

Si le trazamos el diagrama de Bode usando la función margin.m del Matlab con el objetivo de hallar los márgenes de ganancia y fase se obtiene lo que muestra figura 3.17.

Observemos que existen dos posibles frecuencias de cruce por la fase -180º esto implica que hay dos márgenes de ganancias sin embargo la función margin.m solo devuelve el primer margen de fase que encuentra y no todos los que tenga el sistema.

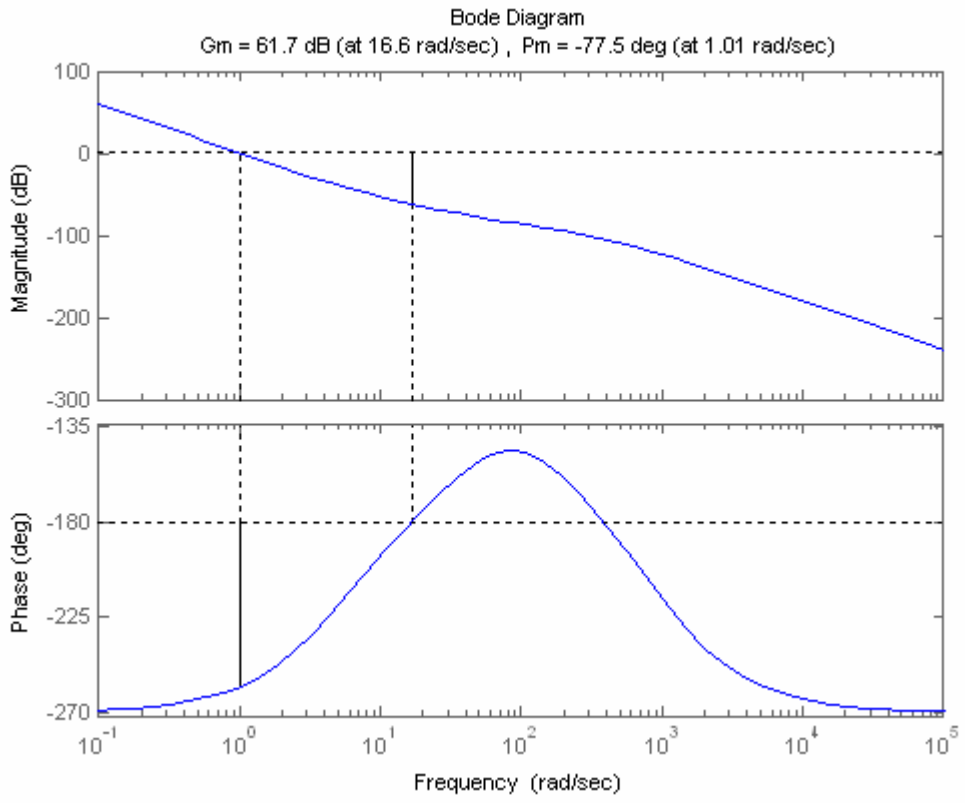


Figura 3. 17

Cuando se usa la función nbode.m se obtienen los márgenes de ganancia y fase en la ventana de comandos del Matlab, figura 3.18, o de lo contrario se pueden almacenar en variables, ver ultimo comando de la figura 3.13

```
>> sys = tf(conv([0.2 1],[0.025 1]),conv([0.001 1],[0.005 1 0 0 0]));
>> nbode(sys)
    'Gm = 62.68 db(a 16.5 rad/seg)'
    'Gm = 103.3 db(a 382 rad/seg)'
    'Pm = -77.84 deg(a 0.9998 rad/seg)'

>>
```

Figura 3. 18

Conclusiones

CONCLUSIONES

Al llegar al final del presente trabajo podemos concluir que:

1. Se puede contar con una serie de funciones para el análisis y diseño en el campo de la frecuencia que enriquecen el conjunto que aporta el Software Matlab para este fin.

2. La incorporación de estas funciones mejorará las posibilidades de ampliar la variedad de sistemas que pueden ser objeto de análisis en clases al quedar eliminadas una serie de imprecisiones en los resultados que se alcanzan cuando se utiliza margin.m y bode.m .

3. Con la función creada para obtener los diagramas de Nyquist ya es posible obtenerlos completamente, independientemente del tipo de función y de la complejidad que esta tenga.

4. La contradicción entre la utilización de los diagramas asintóticos para explicar todo el tema de diseño de compensadores como lo hace el libro de texto de la asignatura Sistemas de Control y la imposibilidad de implementársete en la maquina ya no existe más.

5. Queda implementada una función para el análisis de sistemas multivariables con la técnica de Nyquist inverso que puede utilizarse en las asignaturas que requieran de él en el postgrado, lo cual estaba limitado hasta el momento solo al plano teórico.

RECOMENDACIONES

RECOMENDACIONES.

Para futuros trabajos en esta línea se recomienda lo siguiente:

1. Hacer una función que permita a nbode.m obtener las características del sistema que sea objeto de estudio "cliqueando" sobre los gráficos.

2. Implementar la posibilidad de obtener varios diagramas de Nyquist en la misma figura.

3. Mejorar la función findk.m para poder hallar cada fila de K con un vector de frecuencia diferente en caso de que se quiera.

4. Dar a conocer este trabajo a otras universidades para que pueda ser utilizado por el todos los estudiantes que lo necesiten.

Bibliografía

BIBLIOGRAFIA

[1] Anónimo, Massachusetts Institute of Technology Department of Electrical Engineering & Computer Science, Nyquist Stability Criterion http://ocw.mit.edu/NR/rdonlyres/Electrical-Engineering-and-Computer-Science/6-241Fall2003/B89FC565-4A25-4002-BDAC-BED1CAD51384/0/rec8.pdf 22/5/2004

[2] Anónimo, *MATLAB the language of technical Computer* (1996) Natick, MA, The Math Works inc.

[3] Borrie, John A. (1986) *Modern Control Systems: A Manual of Design Methods*. Gran Bretaña, Prentice – Hall International.

[4] Brown, Robert G.; Thaler, George J. (1966) *Analysis and Design Feedback Control Systems, Servomechanism Analysis.* 2 ed. La Habana, Edición Revolucionaria.

[5] Buezo, Eduardo, Criterio de estabilidad de Nyquist, http://www.monografias.com/trabajos12/critestb/critestb.shtml 10/4/2004

[6] Campos, Neila, Algebra Lineal: Endomorfismos y Diagonalización, http://personales.unican.es/gonzalecn/endomorfismos.pdf 15/4/2004

[7] Chait, Yossi (2002) Multivariable Systems: Background, http://www.ecs.umass.edu/mie/courses/fall2002/mie642/ch10_MIMO_intro_handout.pdf 15/5/2004

[8] Deshpande, Pradeep B. Multivariable Process Control. Instrument Society of America

[9] Grantham, Walter J.; Vicent, Thomas L. (1993) *Modern Control System Analysis and Design,* EEUU, John Wiley & Sons inc.

[10] Iwata, Toshio (1999) Nyquist Diagram http://www.digitalfilter.com/niq.html 10/4/2004

[11] Kuo, Benjamin C. (1967) *Automatic Control Systems* 2 ed. La Habana, Edición Revolucionaria.

[12] Lourtie, Pedro (2003), Control de Sistemas: Diagramas de Nyquist y de Bode http://www.dem.ist.utl.pt/~m_csprod/files/Acetat_7.pdf 10/4/2004

[13] Mellon, Carnegie (1997) Tutoriales de Control con Matlab, http://www.fiobera.unam.edu.ar/Materias/ControlDigital/home.text.html, 25/3/2004

[14] Murillo, Carlos (2001) Curso Tutor para MATLAB http://jimulco.autonoma.edu.co/doc/tutmat.pdf 10/4/2004

[15] Ogata, Katsuhiko, (1985),*Ingeniería de Control Moderna*,La Habana,Edición Revolucionaria

[16] RosenBock, H. H. (1974) *Computer – Aided Control System Design*, London, Academic Press.

[17] Varela Marcelo, Maria Virginia y otros (1983) *Algebra Lineal*, Pueblo y Educación.

[18] Vázquez, Francisco. MATLAB TITO m-functions. http://www.uco.es/~in2vasef 2/5/2004

ANEXOS

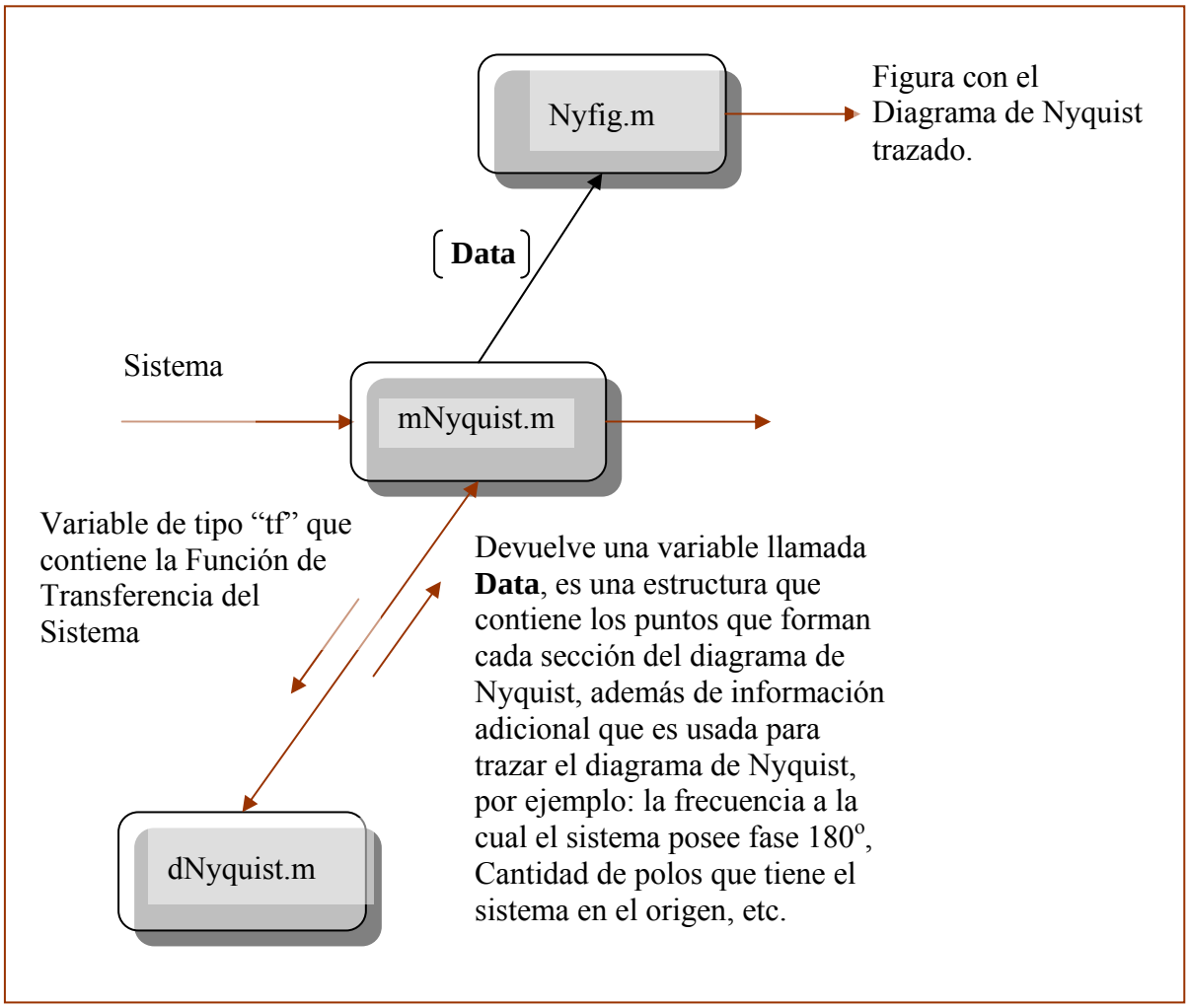
ANEXO 1: DIAGRAMA DE NYQUIST

1.1 Función mNyquist.m

(a) Código de la función

```
function[varargout] = Mnyquist(varargin)
 switch nargin
 case 0
    error('Insuficientes argumentos de entrada');
 case 1
    sys = varargin{1};
 case 2
    num = varargin{1}; den = varargin{2};
    sys = tf(num,den);
 otherwise
    error('Demasiados argumentos')
 end

 [Data] = DNyquist(sys);

 switch nargout
 case 0
    nyfig(Data);
 case 1
    error('Incorrectos parámetros de salida');
 case 2
    varargout{1} = Data.SECC_1.x';
    varargout{2} = Data.SECC_1.y';
 case 3
    varargout{1} = Data.SECC_1.x';
    varargout{2} = Data.SECC_1.y';
    varargout{3} = Data.SECC_1.Data.Freq';
 case 4
    varargout{1} = Data.SECC_1.x';
    varargout{2} = Data.SECC_1.y';
    varargout{3} = Data.SECC_1.Data.Freq';
 otherwise
    error('Parámetros de salida no válidos');
 end
```

(b) Diagrama de bloques



1.2 Función dNyquist.m

(a) Código de la función

```
function[Data] = DNyquist(sys)

 if nargout > 6
    error('Demasiados parámetros de salida');
 end
 if nargin ~= 1
    error('Parámetros de entrada no válidos');
 end

 [num,den] = tfdata(sys,'v');
  %Definiendo el tipo del sistema
 Tipo = 0;
 for I = length(den):-1:1,
    if (den(I) == 0)
       Tipo = Tipo + 1;
    else
       break
    end
 end

 %Definiendo la clase del sistema
 Clase = 0;
 for I = length(num):-1:1,
    if (num(I) == 0)
       Clase = Clase + 1;
    else
       break
    end
 end

%
 if (Clase > 0)&(Tipo > 0)
    if Clase > Tipo
       Clase = Clase - Tipo;
       while Tipo > 0
          den(end) = [];
          num(end) = [];
          Tipo = Tipo - 1;
       end
    elseif Clase < Tipo
       Tipo = Tipo - Clase;
```

```
        while Clase > 0
            den(end) = [];
            num(end) = [];
            Clase = Clase - 1;
        end
    else
        if Clase == Tipo
            while Clase > 0
                num(end) = [];
                den(end) = [];
                Clase = Clase - 1;
            end
            Tipo = Clase;
        end
    end
end
sys = tf(num,den); % actualizando el sistema  por si tiene ceros y polos en el origen

[z,p,k] = zpkdata(sys,'v'); p = roundn(p,-14); z = roundn(z,-15);
[num,den] = tfdata(sys,'v');
Polosjw = p(real(p) == 0); Polosjw = sort(Polosjw(imag(Polosjw) > 0));

% definiendo si el sistema es propio o impropio
if length(z) <= length(p)
    sysType = ['Impropio'];
elseif length(z) > length(p)
    sysType = ['Propio'];
end

if ~isempty(Polosjw)
    Tipo = Inf;
end

%SECCION I (0 < W < Inf)

[X,Y,w] = SeccionI(sys);

SECC_1.x = X;
SECC_1.y = Y;
[Wc] = FrecCruce(sys); % hallando la frecuencia de cruce del sistema
SECC_1.Data = struct('SysName','sys','Freq',w,'FreqCruce',Wc);

%SECCION II --> Lim( R*exp(i*Cita) ) ....R --> inf
if isempty(Polosjw)
    [ang,rho] = SeccionPolarII(sys);
    if (rho == Inf)|(Clase > 0)
        Fi = pi/2:-.01:-pi/2;
```

```
        Enum = polyval(num,w(end)*exp(i*Fi));
        EDen = polyval(den,w(end)*exp(i*Fi));
        for I = 1:length(Enum),
            Mg(I) = Enum(I)/EDen(I);
        end
        x = real(Mg);
        y = imag(Mg);
    else%if rho == 0
        [x,y] = pol2cart(rho, ang);
    end

    % Recuperando los datos de la SECCION_II
    SECC_2.x = x;
    SECC_2.y = y;
  else
    SECC_2.x = [];
    SECC_2.y = [];
  end

%SECCION III -Inf < W < 0
%La SECCION_III es igual a la SECCION_I pero con la Y invertida

  SECC_3.x = X;
  SECC_3.y = -Y;
  SECC_3.Data = struct('SysName','sys','Freq',-w,'FreqCruce',-Wc);

%SECCION IV Lim R*exp(i*Cita)....R --> 0
  if isempty(Polosjw)
    [ang,rho] = SeccionPolarIV(sys);
    if (rho == Inf) | (Tipo > 0)
        Fi = -pi/2:.01:pi/2;
        Enum = polyval(num,w(1)*exp(i*Fi));
        EDen = polyval(den,w(1)*exp(i*Fi));
        for I = 1:length(Enum),
            Mg(I) = Enum(I)/EDen(I);
        end
        x = real(Mg);
        y = imag(Mg);
    else
        [x,y] = pol2cart(rho,ang);
    end

    %Recuperando los datos de la SECCION_IV
    SECC_4.x = x;
    SECC_4.y = y;
  else
    SECC_4.x = [];
```

```
    SECC_4.y = [];
  end
  Info = struct('Tipo', Tipo, 'Clase', Clase, 'ProOImpro', sysType );
  Data = struct('SECC_1',SECC_1,'SECC_2',SECC_2,…
                'SECC_3',SECC_3,'SECC_4',SECC_4,'Info',Info);


'LimitCero' 'LimitInf' 'LimitPolo'});
 return


%_____________________Sección_Polar_II_____________________________
function[ang,rho] = SeccionPolarII(sys)
  Gs = tf2sym(sys);
  syms r f;
  s = r*exp(i*f);
  Gs = eval(Gs);
  lim = limit(Gs,r,Inf);
  A = [subs(lim,f,pi/2); subs(lim,f,pi/3) ;subs(lim,f,pi/4); subs(lim,f,pi/6); subs(lim,f,0);...
      subs(lim,f,-pi/6); subs(lim,f,-pi/4); subs(lim,f,-pi/3); subs(lim,f,-pi/2)];
  x = real(double(real(A)));
  y = imag(double(A));
  [ang,rho] = cart2pol(x,y);
  rho(2:8) = rho(1);
  ang = unwrap(ang);


%_____________________Seccion_Polar_IV_____________________________
function[ang,rho] = SeccionPolarIV(sys)
  Gs = tf2sym(sys);
  syms r f;
  s = r*exp(i*f);
  Gs = eval(Gs);
  lim = limit(Gs,r,0,'right');
  A = [subs(lim,f,pi/2); subs(lim,f,pi/3) ;subs(lim,f,pi/4); subs(lim,f,pi/6); subs(lim,f,0);...
      subs(lim,f,-pi/6); subs(lim,f,-pi/4); subs(lim,f,-pi/3); subs(lim,f,-pi/2)];
  x = real(double(real(A)));
  y = imag(double(A));
  [ang,rho] = cart2pol(x,y);
  rho(2:8) = rho(1);
  ang = unwrap(ang);
%_____________________SECCION_I_____________________________
function [Re,Im,w] = SeccionI(sys)

  Error = 1e-14;       %Definiendo cte de error
  no = nargout;
```

```
  % Hallamos si el sistema tiene polos en el eje jw
  [z,p,k] = zpkdata(sys,'v'); p = roundn(p,-14); z = roundn(z,-15); [num,den] =
tfdata(sys,'v');
  Polosjw = p(real(p) == 0); Polosjw = sort(Polosjw(imag(Polosjw) > 0));    %Tomamos los
polos mayores que cero sobre el eje jw


  if ~isempty(Polosjw)
     w = freqint2(num,den,30);
     tol = 1e-6;  % tolerancia para encontrar polos imaginarios
     %  Si todos los polos están en el origen,  tan solo los movemos un poco a la izquierda
     if norm(p) == 0
        length_p = length(p);
        p = -tol*ones(length_p,1);
        den = den(1,1)*[1 tol];
        for ii = 2:length_p
           den = conv(den,[1 tol]);
        end
     end
     zp = [z;p];        % combine los polos y ceros del sistema
     nzp = length(zp);  % número de polos y ceros
     ones_zp=ones(nzp,1);
     %Ipo = find((abs(real(p))<1e-6) & (imag(p)>=0)) %indica polos con cero real part +
non-neg parte imag.
     Ipo = find((abs(real(p)) < tol) & (imag(p)>=0)); %indica polos con cero real part + non-
neg parte imag.
     if  ~isempty(Ipo)   %
        % **** caso si tenemos tales polos hacemos lo siguiente:****
        po = p(Ipo); % polos con parte real 0 y parte imag. no negativa
        % check for distintos polos
        [y,ipo] = sort(imag(po));  % ordena por parte imaginaria
        po = po(ipo);
        dpo = diff(po);
        idpo = find(abs(dpo)>tol);
        idpo = [1;idpo+1];   % índice de distintos polos

        po = po(idpo);   % solo se usan los polos distintos
        nIpo = length(idpo); % # de tales polos
        originflag = find(imag(po)==0);  % busca polo en origen

        s = [];  % s es la respuesta en vector frecuencia
        w = sqrt(-1)*w;  % crea un vector jwo para evaluar todas las frecuencias con
        for ii=1:nIpo % para todos los polos imaginarios

           [nrows,ncolumns]=size(w);
           if nrows == 1
              w = w.';  % si w es un renglón, lo hace columna
```

```
        end;
        if nIpo == 1
           r(ii) = .1;
        else          % ver. Distancias a los otros polos y ceros
           pdiff = zp-po(ii)*ones_zp;  % halla diferencias entre
           % polos en verif. y otros polos y ceros
           ipdiff = find(abs(pdiff)>tol); % ipdiff todas dif. no nulas

           r(ii)=0.2*min(abs(pdiff(ipdiff))); % la mitad de eso
           r(ii)=min(r(ii),0.1);  % el mínimo entre esta dif. y 0.1
        end;
        t = linspace(-pi/2,pi/2,25);
        if (ii == originflag)
           t = linspace(0,pi/2,25);
        end;    % nos da un vector de puntos alrededor de cada polo imaginario
        s1 = po(ii)+r(ii)*(cos(t)+sqrt(-1)*sin(t));  % rodeo
        s1 = s1.';  % asegura es columna

        % reconstituye s - frecuencia compleja -
        % y recalcula.

        bottomvalue = po(ii)-sqrt(-1)*r(ii);  % magnitud de parte imaginaria.
        if (ii ==  originflag)  % si es un origen
           bottomvalue = 0;
        end;
        topvalue = po(ii)+sqrt(-1)*r(ii); % último valor rodeo se detiene
        nbegin = find(imag(w) < imag(bottomvalue)); %
        nnbegin = length(nbegin); % puntos menores de rodeos
        if (nnbegin == 0)& (ii ~= originflag)    % a polos en jw
           sbegin = 0;
        else sbegin = w(nbegin);
        end;
        nend = find(imag(w) > imag(topvalue));  % puntos mayores de rodeos
        nnend = length(nend);    % a raices en jw
        if (nnend == 0)
           send = 0;
        else send = w(nend);
        end
        w = [sbegin; s1; send];  % medio eje jw reconstruido
     end
  else
     w = sqrt(-1)*w;
  end
  %end  % esto termina el lazo para polos imaginarios
  % ********************************************************
  % fin de disgresión
  % calcula respuesta en frecuencia
```

```
    gt = freqresp(num,den,w);

    % ************************************************************

    nw = length(gt);
    mag = abs(gt);   % factor de escala
    ang = angle(gt);

    for n = 1:nw
       h(n,1) = mag(n,1)*(cos(ang(n,1))+sqrt(-1)*sin(ang(n,1)));
    end;  % recalcula G(jw) con factor de escala

    gt = h;
    % ************************************************************
    Re=real(gt);
    Im=imag(gt);
    w = abs(w);
  else

    % Buscamos el primer valor de w igual a la frecuencia cuando la
    % magnitud es uno
    marg = allmargin(sys);
    w = marg.PMFrequency;
    if (isempty(w))|(w == NaN)|(w == Inf)
       w = 1;
       maxMG = 10;
    else
       w = w(1);
       S = evalfr(sys, j*w);
       maxMG = 10*abs(S);
    end


    % hallando cuando w --> 0
    [ph,mg] = LimitCero(sys);
    [Re,Im,w] = EvalW(sys,w,mg,ph,0,Error,maxMG);

    % Hallando cuando w --> oo
    [ph,mg] = LimitInf(sys);
    [Re,Im,w] = EvalW(sys,w,mg,ph,Inf,Error,maxMG);
  end

%--------------------------------------------------------------------------------
  function[ang,rho] = LimitCero(sys)
    Gs = tf2sym(sys);
    syms w;
    s = j*w;
```

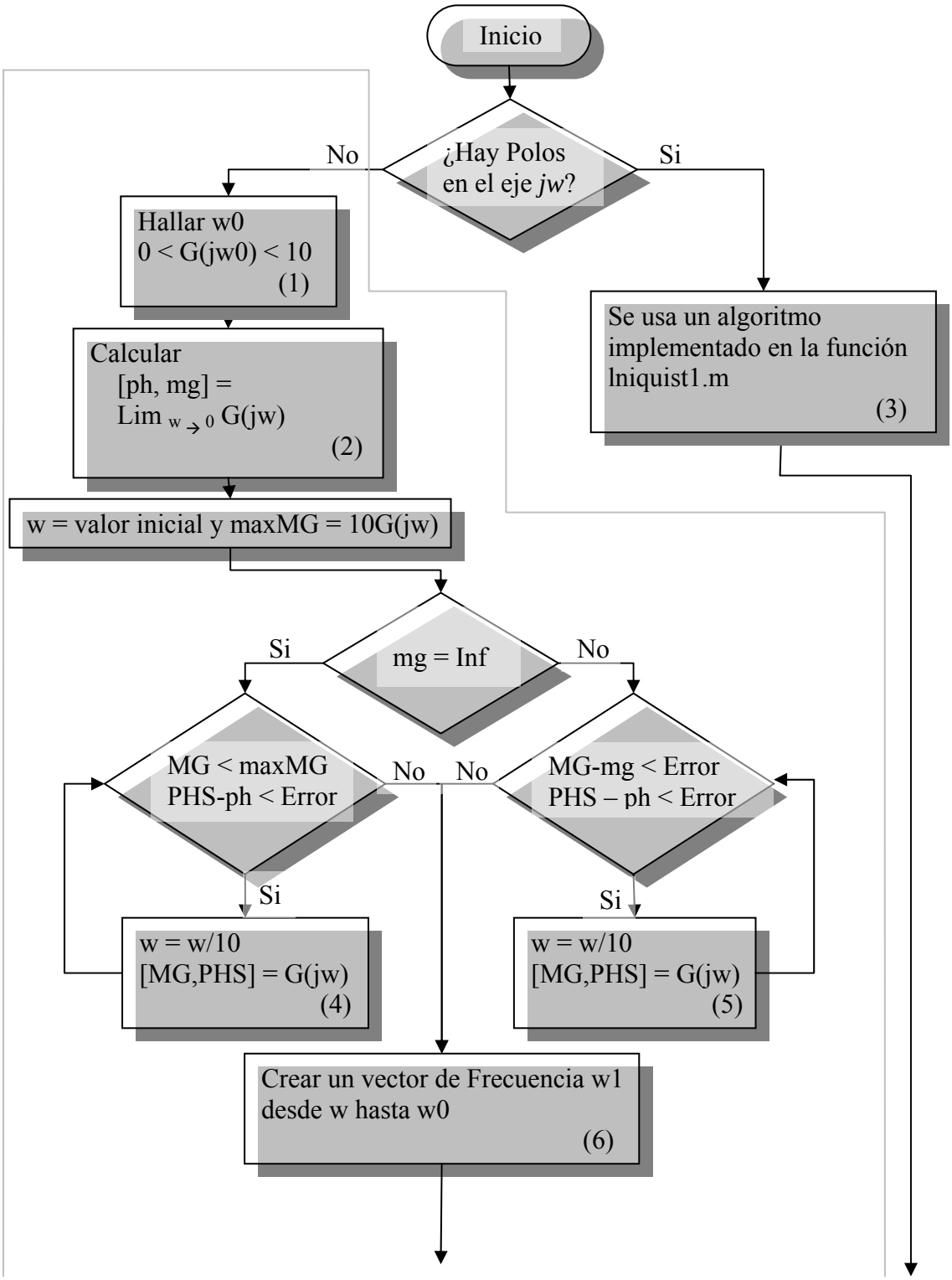
```
  Gs = eval(Gs);
  lim = limit(Gs,w,0,'right');
  x = double(real(lim));
  y = imag(double(lim));
  [ang,rho] = cart2pol(x,y);


function[ang,rho] = LimitInf(sys);
  Gs = tf2sym(sys);
  syms w;
  s = j*w;
  Gs = eval(Gs);
  lim = limit(Gs,w,Inf);
  x = double(real(lim));
  y = imag(double(lim));
  [ang,rho] = cart2pol(x,y);

function[Re,Im,w] = EvalW(sys,W,lim,ph,Sentido,Error,maxMG)
  % sys:      Sistema lti
  % W:        un valor o vector de w para partir
  % lim:      limite de la magnitud
  % ph:       limite de la fase
  % Sentido: Cero o Inf (valor al que tiende w)
  % Error:    error permitido

  S = freqresp(sys,W); S = reshape(S,[prod(size(S)) 1]);
  MG = abs(S); PHS = angle(S);

  if lim == Inf
     if Sentido == 0 % w --> 0
        w = W(1);
        while (abs(MG(1)) <= maxMG) | (abs(PHS(1) - ph) >= 0.1)
           % incremento de w = [... w]
           w = w/10;
           % Hallar la nueva Mag
           S = evalfr(sys,j*w);
           MG = abs(S); PHS = angle(S);
           if (sign(PHS)~=sign(ph))&(ph ~= 0),
              PHS = rem(PHS,2*pi);
              PHS = PHS - sign(PHS)*2*pi;
           end
        end
        w = [VectFrecR(w, W(1)) W];

     elseif Sentido == Inf % w --> oo
        w = W(end);
        while (abs(MG(end)) <= maxMG) | (abs(ph - PHS(end)) >= 0.1)
```

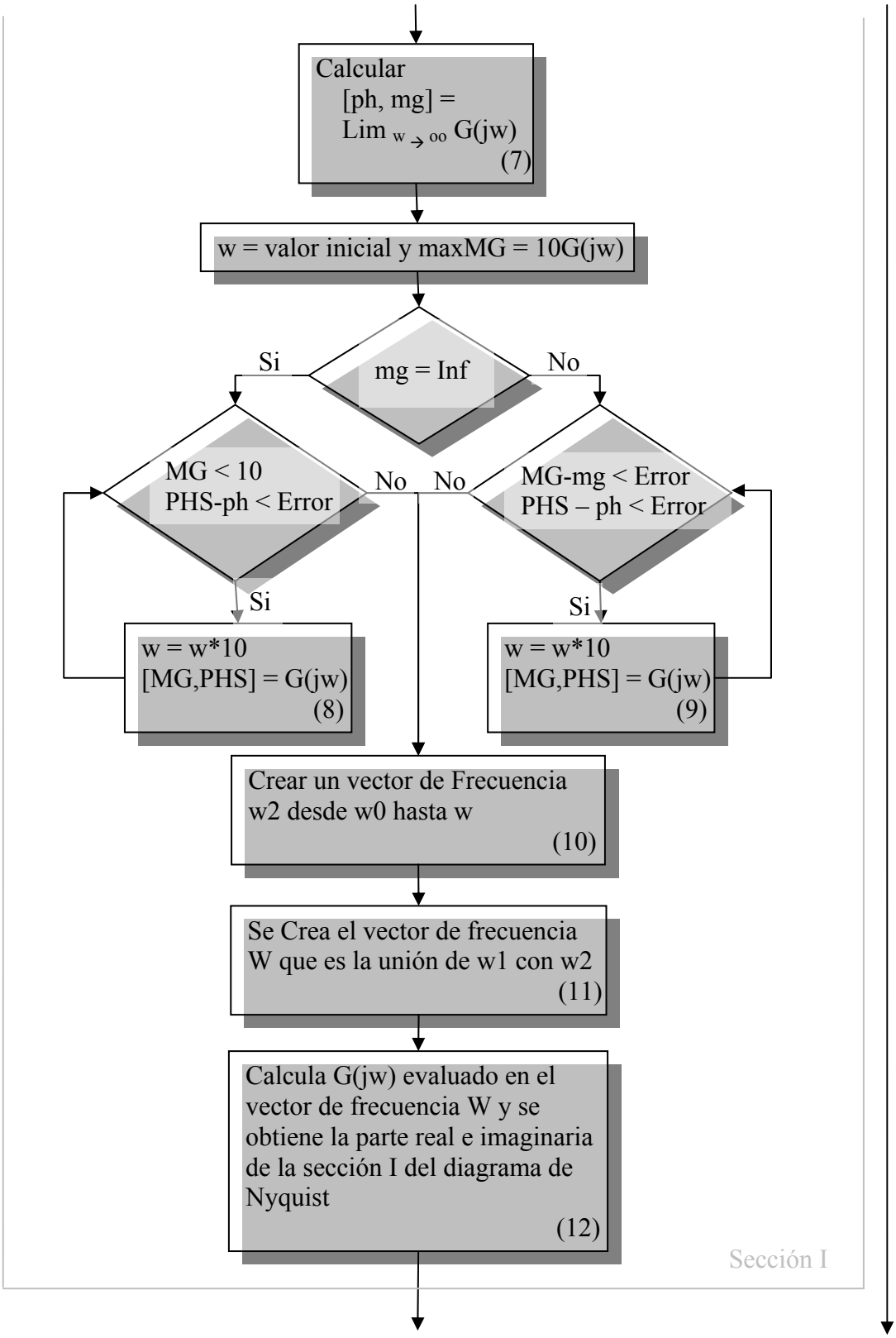
```
            % incremento de w = [w ...]
            w = w*10;
            % Hallar la nueva Mag
            S = evalfr(sys,j*w);
            MG = abs(S); PHS = angle(S);
            if (sign(PHS)~=sign(ph))&(ph ~= 0),
               PHS = rem(PHS,2*pi);
               PHS = PHS - sign(PHS)*2*pi;
            end
         end
         w = [W VectFrecR(W(end), w)];
      end

   else
      if Sentido == 0 % w --> 0
         w = W(1);
         while or((abs(MG(1) - lim) >= Error),and((abs(PHS(1) - ph) >= Error),(lim ~= 0)))
            % incremento de w = [... w]
            w = w/10;
            % Hallar la nueva Mag
            S = evalfr(sys,j*w);
            MG = abs(S); PHS = angle(S);
            if (sign(PHS)~=sign(ph))&(ph ~= 0),
               PHS = rem(PHS,2*pi);
               PHS = PHS - sign(PHS)*2*pi;
            end
         end
         w = [VectFrecR(w, W(1)) W];

      elseif Sentido == Inf % w --> oo
         w = W(end);
         while (abs(MG(end) - lim) >= Error)|(and((abs(PHS(1) - ph) >= Error),(lim ~= 0)))
            % incremento de w = [w ...]
            w = w*10;
            % Hallar la nueva Mag
            S = evalfr(sys,j*w);
            MG = abs(S); PHS = angle(S);
            if (sign(PHS)~=sign(ph))&(ph ~= 0),
               PHS = rem(PHS,2*pi);
               PHS = PHS - sign(PHS)*2*pi;
            end
         end
         w = [W VectFrecR(W(end), w)];
      end
   end

   S = freqresp(sys,w');
```

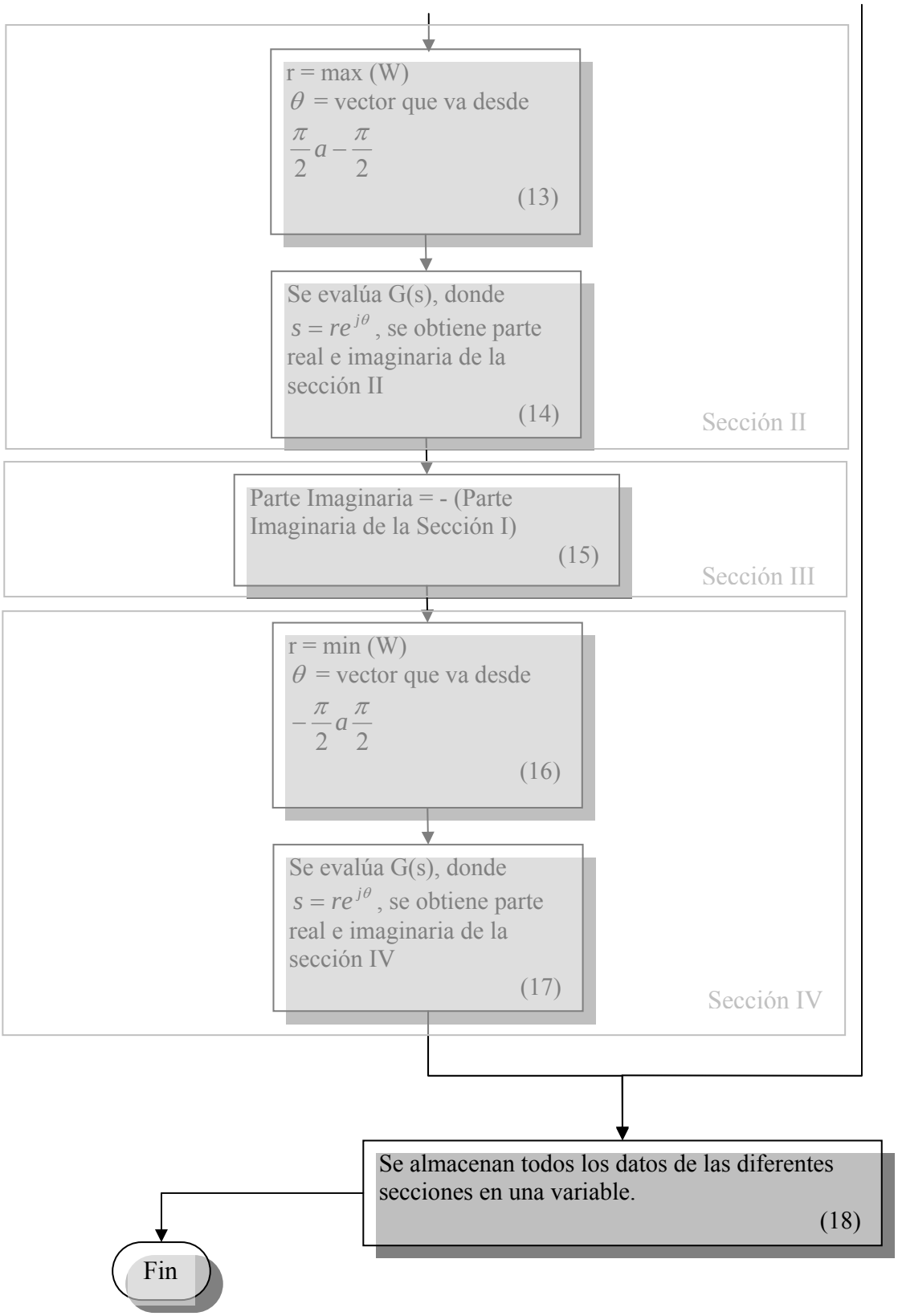
```
   S = reshape(S,[prod(size(S)) 1]);
   Re = real(S); Im = imag(S);


 %_____________________FrecCruce_____________________________
 function [Wc] = FrecCruce(sys)
% Devuelve las frecuencias (Wc) a la cual el sistema tiene fase -180 grados

 Error = 1.00e-003;
 [z,p,k] = zpkdata(sys,'v');

 marg = allmargin(sys);
 w = marg.PMFrequency;
 if (isempty(w))|(w == NaN)|(w == Inf)
    w = 1;
 else
    w = w(1);
 end
 S = evalfr(sys,j*w); MG = abs(S); PHS = angle(S);

 OK = 1; Wc = [];
 while OK

    Wc = [Wc HayCruce(PHS,w)];

    OKalto = 0; OKbajo = 0;PHS = [];
    %Chequeo parte alta de la frecuencia.
    frec = VectFrecR(w(end),10*w(end));
    S = freqresp(sys,frec); S = reshape(S,[prod(size(S)) 1]);
    PHS = angle(S); PHS = unwrap(PHS);
    if abs(PHS(1) - PHS(end)) >= Error
       w = [w frec];
       Wc = [Wc HayCruce(PHS,frec)];
    else
       OKalto = 1;
    end
    %Chequeo de la parte baja de la frecuencia.
    frec = VectFrecR(w(1)/10,w(1));
    S = freqresp(sys,frec); S = reshape(S,[prod(size(S)) 1]);
    PHS = angle(S); PHS = unwrap(PHS);
    if abs(PHS(1) - PHS(end)) >= Error
       w = [frec w];
       Wc = [Wc HayCruce(PHS,frec)];
    else
       OKbajo = 1;
    end
```

```
    if OKbajo & OKalto
       OK = 0;
    end
  end

  leng = length(Wc);
  if leng > 1
    Wc = sort(Wc); ind = [];
    I = 1;
    while I < leng
       for J = (I + 1):length(Wc),
          if Wc(I) == Wc(J)
             ind = [ind J];
          end
       end
       Wc(ind) = []; ind = [];
       leng = length(Wc);
       I = I + 1;
    end
  end

  %_____________________________HayCruce_____________________________


  function [Wc] = HayCruce(PHS,w)
    %Determina si un vector dado cruza el punto 180 grados
    Wc = [];
    for I = 1:length(PHS)-1,
       if ((abs(PHS(I)) > pi) & (abs(PHS(I+1)) < pi)) | ((abs(PHS(I)) < pi) & (abs(PHS(I+1))
> pi))
          Wc = [Wc w(I)];
       elseif abs(PHS(I)) == pi
          Wc = [Wc w(I)];
       end
    end
```

(b) Diagrama de bloques




Calcular
[ph, mg] =
Lim w → oo G(jw)
(7)
w = valor inicial y maxMG = 10G(jw)
mg = Inf
Si
No
MG < 10
PHS-ph < Error
No
No
MG-mg < Error
PHS – ph < Error
Si
Si
w = w*10
[MG,PHS] = G(jw)
(8)
w = w*10
[MG,PHS] = G(jw)
(9)
Crear un vector de Frecuencia
w2 desde w0 hasta w
(10)
Se Crea el vector de frecuencia
W que es la unión de w1 con w2
(11)
Calcula G(jw) evaluado en el
vector de frecuencia W y se
obtiene la parte real e imaginaria
de la sección I del diagrama de
Nyquist
(12)
Sección I


r = max (W)
θ = vector que va desde
$\frac{\pi}{2} a -\frac{\pi}{2}$
(13)
Se evalúa G(s), donde
$s = re^{j\theta}$, se obtiene parte real e imaginaria de la sección II
(14)
Sección II
Parte Imaginaria = - (Parte Imaginaria de la Sección I)
(15)
Sección III
r = min (W)
θ = vector que va desde
$-\frac{\pi}{2} a \frac{\pi}{2}$
(16)
Se evalúa G(s), donde
$s = re^{j\theta}$, se obtiene parte real e imaginaria de la sección IV
(17)
Sección IV
Se almacenan todos los datos de las diferentes secciones en una variable.
(18)
Fin

1.3 Función nyfig,m

(a) Código de la función

```
function varargout = nyfig(varargin)
% NYfIG DIBUJA EL DIAGRMA DE NYQUIST

if nargin == 1  % LAUNCH GUI
   % Creando la figura,axes y uicontrol
   fig = figure('Visible', 'off', 'Tag', 'FigDiagNyquist', 'Name', 'Diagrama de Nyquist',...
            'Color', get(0,'DefaultFigureColor'), 'NextPlot', 'replace',...
            'Resize', 'on', 'NumberTitle', 'off');
   pos = get(fig, 'Position');
   ToolBarDiagNyquist = uicontrol('Position',[1 1 pos(3) 30], 'Style', 'frame', 'Tag',…
                                  'ToolBarDiagNyquist');
   ToolBarTextDiagNyquist = uicontrol('Position',[2 2 pos(3)-2 30], 'Style', 'text', …
                                 'HorizontalAlignment', 'center', 'Tag',…
                                 'ToolBarTextDiagNyquist', 'BackgroundColor',…
                                  get(0,'DefaultFigureColor'));

   ax = axes;
   set(ax, 'Position', [0.1300 0.1900 0.7750 0.75], 'Tag', 'AxesDiagNyquist','Visible', 'on',...
      'FontSize', [8], 'XColor', [.4 .4 .4], 'YColor', [.4 .4 .4], 'Box', 'on');

   % Genera una estructura de de handles para pasar a callbacks, y almacenarlos.
   handles = guihandles(fig);
   guidata(fig, handles);
   % Poniendo la ToolBar
   LDoMenubar(handles.FigDiagNyquist);
   % Pintando el gráfico
   Data = varargin{1};
   VmNyquist(Data,handles)


   % Poniendo los títulos de los ejes de coordenadas
   mylabel(handles.AxesDiagNyquist,'Parte Imaginaria');
   mxlabel(handles.AxesDiagNyquist,'Parte Real');
   mtitle(handles.AxesDiagNyquist,'Diagrama de Nyquist');

   set(fig,'Visible','on');
   gca = handles.AxesDiagNyquist;

   %Ubicando los handles en la figura
   set(handles.FigDiagNyquist,'UserData',handles);
```

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

elseif ischar(varargin{1}) %invoca la subfunción nombrada o callback
        try
                if (nargout)
                        [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
                else
                        feval(varargin{:}); % FEVAL switchyard
                end
        catch
                disp(lasterr);
        end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%LDoMenubar%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function LDoMenubar(myFig)

   hTB = findall(myFig,'Tag','FigureToolBar');
   if isempty(hTB)
      oldFeature=feature('figuretools');
      feature('figuretools',1);
      set(myFig,'ToolBar','figure');
      feature('figuretools',oldFeature);

      hTB = findall(myFig,'Tag','FigureToolBar');

      %initialize the toolbar appropriately
      offon = {'off' 'on'};
      set(findall(hTB,'Tag','ScribeSelectToolBtn'),...
         'state',offon{plotedit(myFig,'isactive')+1});

      zoomInActive=0;
      zoomOutActive=0;
      switch zoom(myFig,'getmode')
      case {'in','on'}
         zoomInActive=1;
      case 'out'
         zoomOutActive=1;
      end
      set(findall(hTB,'Tag','figToolZoomOut'),...
         'state',offon{zoomOutActive+1});
```

```
        set(findall(hTB,'Tag','figToolZoomIn'),...
           'state',offon{zoomInActive+1});

        isRotate3dActive=~isempty(findall(myFig,'type','axes','Tag','rotaObj'));
        set(findall(hTB,'Tag','figToolRotate3D'),...
           'state',offon{isRotate3dActive+1});

    elseif strcmp(get(hTB,'visible'),'on')
        set(hTB,'visible','off');
    else
        set(hTB,'visible','on');
    end

    %jpropeditutils('jundo','stop',myFig);
    drawnow

function hh = mylabel(ax,string,varargin)
if nargin > 2 & (nargin-1)/2-fix((nargin-1)/2),
  error('Incorrect number of input arguments')
end

 isappdata(ax,'MWBYPASS_ylabel')
   h = mwbypass(ax,'MWBYPASS_ylabel',string,varargin{:});

else
   h = get(ax,'ylabel');

   set(h, 'FontAngle',  get(ax, 'FontAngle'), ...
         'FontName',   get(ax, 'FontName'), ...
         'FontSize',   get(ax, 'FontSize'), ...
         'FontWeight', get(ax, 'FontWeight'), ...
         'color',      'k',...
         'string',     string, varargin{:});
end

if nargout > 0
  hh = h;
end

function hh = mxlabel(ax,string,varargin)

if nargin > 2 & (nargin-1)/2-fix((nargin-1)/2),
  error('Incorrect number of input arguments')
end

if isappdata(ax,'MWBYPASS_xlabel')
   h = mwbypass(ax,'MWBYPASS_xlabel',string,varargin{:});
```

```
else
  h = get(ax,'xlabel');


  set(h, 'FontAngle',  get(ax, 'FontAngle'), ...
       'FontName',   get(ax, 'FontName'), ...
       'FontSize',   get(ax, 'FontSize'), ...
       'FontWeight', get(ax, 'FontWeight'), ...
       'Color',      'k',...
       'string',     string,varargin{:});
end

if nargout > 0
 hh = h;
end

function hh = mtitle(ax,string,varargin)
if nargin > 2 & (nargin-1)/2-fix((nargin-1)/2),
 error('Incorrect number of input arguments')
end

%---Check for bypass option
if isappdata(ax,'MWBYPASS_title')
  h = mwbypass(ax,'MWBYPASS_title',string,varargin{:});

%---Standard behavior
else
  h = get(ax,'title');

  set(h, 'FontAngle',  get(ax, 'FontAngle'), ...
       'FontName',   get(ax, 'FontName'), ...
       'FontSize',   get(ax, 'FontSize'), ...
       'FontWeight', get(ax, 'FontWeight'), ...
       'Rotation',   0, ...
       'string',     string, varargin{:});

end

if nargout > 0
 hh = h;
end

function  VmNyquist(Data,Hand)
  % Recuperando datos
  SECC_1 = Data.SECC_1; SECC_2 = Data.SECC_2;
  SECC_3 = Data.SECC_3; SECC_4 = Data.SECC_4;
```

```
Wc = Data.SECC_1.Data.FreqCruce;
sysType = Data.Info.ProOImpro;
Tipo = Data.Info.Tipo;
Clase = Data.Info.Clase;

Ax = Hand.AxesDiagNyquist;
%Creando cada sección.
SECC_1DiagNyquist = line('XData',          [SECC_1.x]  ,...
                    'YData',           [SECC_1.y]  ,...
                    'Parent',          Ax          ,...
                    'Color',           'b'         ,...
                    'UserData',        SECC_1.Data ,...
                    'ButtonDownFcn',   'Scan');

SECC_2DiagNyquist = line('XData',          [SECC_2.x]  ,...
                    'YData',           [SECC_2.y]  ,...
                    'Parent',          Ax          ,...
                    'Color',           [0 1 0]     ,...
                    'UserData',        []          ,...
                    'ButtonDownFcn',    '');

SECC_3DiagNyquist = line('XData',          [SECC_3.x]  ,...
                    'YData',           [SECC_3.y]  ,...
                    'Parent',          Ax          ,...
                    'Color',           'r'         ,...
                    'UserData',        SECC_3.Data ,...
                    'ButtonDownFcn',   'Scan');

SECC_4DiagNyquist = line('XData',          [SECC_4.x]  ,...
                    'YData',           [SECC_4.y]  ,...
                    'Parent',          Ax          ,...
                    'Color',           [.6 .6 .6]  ,...
                    'UserData',        []          ,...
                    'ButtonDownFcn',    '');

% Tratamiento del punto -1
line('XData',-1,'YData',0,'Marker','+','Color',[1 0 0],'Parent',Ax);


% Flechitas
%Creando la flecha de dirección
set(Ax, 'YLimMode', 'auto');
LimX = get(Ax,'XLim'); LimY = get(Ax,'YLim');
limits = [LimX LimY];
if ((Tipo >= 1)&(Tipo < Inf))|(Clase >= 1)|(strcmp(sysType, ['Propio']))
   %SECC_1
```

```
      [PX,PY] = Direccion(SECC_1.x,SECC_1.y,1);
      HfillSec1 =
line('XData',PX,'YData',PY,'Color',get(SECC_1DiagNyquist,'Color'),'Parent',Ax);

      %SECC_3
      [PX,PY] = Direccion(SECC_3.x,SECC_3.y,3);
      HfillSec3 = line('XData',PX,'YData',PY,…
                  'Color',get(SECC_3DiagNyquist,'Color'),'Parent',Ax);
   else
      % Flechitas
      g = SECC_1.x + j*SECC_1.y;
      im = SECC_1.y;
      sx = limits(2) - limits(1);
      [sy,muestra]=max(abs(2*im));
      arrow=[-1;0;-1] + 0.75*sqrt(-1)*[1;0;-1];
      muestra=muestra+(muestra==1);
      reim=diag(g(muestra,:));
      d=diag(g(muestra+1,:)-g(muestra-1,:));
      % gira flecha t. en cuenta factores sx y sy
      d = real(d)*sy + sqrt(-1)*imag(d)*sx;
      rot=d./abs(d);          % cuando flecha no horizontal
      arrow = ones(3,1)*rot'.*arrow;
      scalex = (max(real(arrow)) - min(real(arrow)))*sx/50;
      scaley = (max(imag(arrow)) - min(imag(arrow)))*sy/50;
      arrow = real(arrow)*scalex + sqrt(-1)*imag(arrow)*scaley;
      xy =ones(3,1)*reim' + arrow;
      xy2=ones(3,1)*reim' - arrow;
      [m,n]=size(g);
      HfillSec1 = line('XData',real(xy),'YData',-
imag(xy),'Color',get(SECC_1DiagNyquist,'Color'),'Parent',Ax);
      %           plot(real(xy),-imag(xy),'r-',real(xy2),imag(xy2),'b-')
      HfillSec1 =
line('XData',real(xy2),'YData',imag(xy2),'Color',get(SECC_3DiagNyquist,'Color'),'Parent',
Ax);
   end

   %ubicando el texto en la ToolBar
   %Creando el texto
   Tooltxt = MyText(Tipo,Clase,Wc,sysType);
   %Tomando el handle de la ToolBar
   ToolBarText = Hand.ToolBarTextDiagNyquist;
   set(ToolBarText,'String',Tooltxt);

   %Tratamiento de los Ejes de coordenadas (Eje X)
   x = get(Ax,'XLim');
   y = zeros(1,2);
```

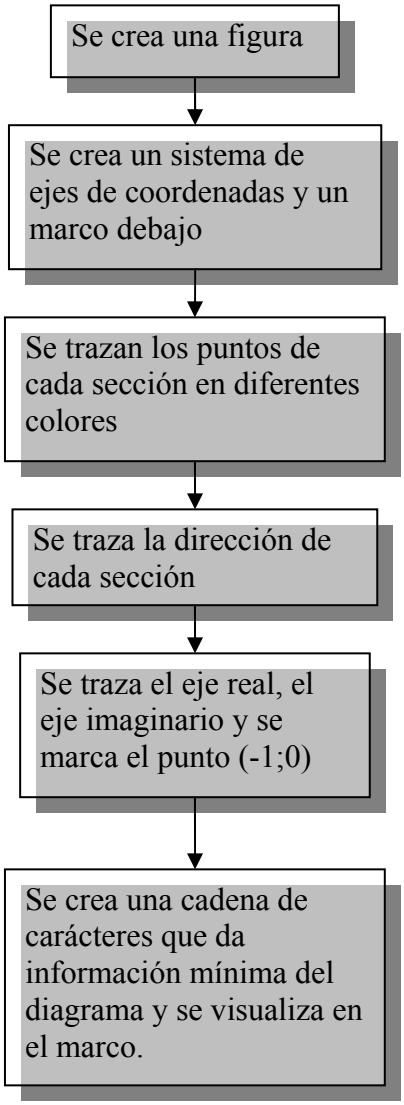
```
set(Ax,'DefaultLineLineStyle',':')      %definimos el estilo de la línea
line('XData',x,'YData',y,'Color','k','Parent',Ax)     %Se traza una línea horizontal por 0;Y

%Tratamiento del Eje Y
x = zeros(1,2);
y = get(Ax,'YLim');
set(Ax,'DefaultLineLineStyle',':')      %definimos el estilo de la línea
line('XData',x,'YData',y,'Color','k','Parent',Ax)     %Se traza una línea vertical por X;0;



function[str] = MyText(Tipo,Clase,Wc,sysType)
 str = {''};
 if (Tipo > 0)|(Clase > 0)|(strcmp(sysType, ['Propio']))
    str(1) = {'La Sección Circular se considera infinita'};
    if (~isempty(Wc))&(length(Wc) <= 15)
       s = ['El diagrama cruza al eje Real Negativo en: Wc = '];
       stemp = sprintf('%s%3.4g',s,Wc(1));
       % Informar sobre las frec. de cruce por fase = 180 deg
       if (length(Wc) > 6)
          stemp = sprintf('%s; ... ; %3.4g',stemp,Wc(end));
       else
          for I = 1:length(Wc)-1,
             stemp = sprintf('%s\t; %3.4g',stemp,Wc(I+1));
          end
       end
       str(2) = {stemp};
    end
 elseif Tipo == 0
    if ~isempty(Wc)
       s = ['El diagrama cruza al eje Real Negativo en: Wc = '];
       stemp = sprintf('%s%3.4g',s,Wc(1));
       % Informar sobre las frec. de cruce por fase = 180 deg
       if length(Wc) > 6
          stemp = sprintf('%s; ... ; %3.4g',stemp,Wc(end));
       else
          for I = 1:length(Wc)-1,
             stemp = sprintf('%s\t; %3.4g',stemp,Wc(I+1));
          end
       end
       str(2) = {stemp};
    end
 end

function [PX,PY] = Direccion(Px,Py,sen)
 % dado 2 vectores se dibuja el sentido de los mismos, si sen = 1 el
 % sentido es en la dirección en que aumenta el índice y si sen = 3 la
```

```
% dirección es en sentido de disminución del índice el vector

ni = nargin;
switch ni
case 0
   error('Falta de argumentos')
   return
case 1
   error('Falta de argumentos')
   return
end

d = [(abs(max(Px) - min(Px))/20), (abs(max(Py) - min(Py))/20)];
d = max(d);

[mx,Long] = max(abs(Px + j*Py));
if isempty(Long)
   Long = 50;
elseif Long > 300
   Long = Long - 200;
else
   Long = Long + 200;
end
X = Px(Long); Y = Py(Long);
I = Long; Paso = 1; Fin = length(Px); nA = []; nB = [];
while isempty(nA)
   for J = I:Paso:Fin-1,
      dp = ((X - Px(J))^2 + (Y - Py(J))^2);
      if dp > d
         m = (Py(J) - Y)/(Px(J) - X);          % Pendiente
         n = Py(J) - m*Px(J);                  % Intersecto eje y
         a = m^2 + 1; b = 2*(m*(n - Y) - X); c = X^2 + (n - Y)^2 - d;
         nA = [(-b - sqrt(b^2 - 4*a*c))/(2*a)...
              (-b + sqrt(b^2 - 4*a*c))/(2*a)];
         if (Px(J) > X)
            nA = max(nA);
         elseif Px(J) < X
            nA = min(nA);
         end

         if (m == Inf)
            nA = X;
            nB = Y + d;
         elseif (m == -Inf)
            nA = X;
            nB = Y - d;
         else
```

```
            nB = m*nA + n;
         end
         break
      elseif dp == d
         nA = Px(J); nB = Py(J);
         break
      end

   end
   if (Paso == 1)&(sen == 1) %sección I
      X = [X nA];
      Y = [Y nB];
   elseif (sen == 1)
      X = [nA X];
      Y = [nB Y];
   elseif (Paso == 1)&(sen == 3) %Sección II
      X = [nA X];
      Y = [nB Y];
   elseif (sen == 3)
      X = [X nA];
      Y = [Y nB];
   end
   Paso = -1; Fin = 2;
end


if (length(X) ~= length(Y)) | (length(X) ~= 2)
   disp('Error, vector X debe ser de la misma longitud de Y')
   PX = [];PY = [];
   return
elseif (X(1) == X(2)) | (Y(1) == Y(2))

   if (X(1) == X(2))
      PX = [X(1)-d^(1/2), X(2), X(1)+d^(1/2)];%, X(1)-d^(1/2)];
      PY = [Y(1),         Y(2),         Y(1)];%,         Y(1)];
   else
      PX = [         X(1), X(2),         X(1)];%,         X(1)];
      PY = [Y(1)-d^(1/2), Y(2), Y(1)+d^(1/2)];%, Y(1)-d^(1/2)];
   end
else
   m = (Y(2) - Y(1)) / (X(2) - X(1)); m = -1/m;
   n = (Y(1) - m*X(1));
   ax = 1 + m^2;
   bx = 2*(m*(n - Y(1)) - X(1));
   cx = X(1)^2 + (n - Y(1))^2 - (d);
   Px(1) = (-bx + sqrt(bx^2 - 4*ax*cx))/(2*ax);
   Px(2) = (-bx - sqrt(bx^2 - 4*ax*cx))/(2*ax);
```

```
    Py = m*Px + n;
    PX = [Px(1) X(2) Px(2)];% Px(1)];
    PY = [Py(1) Y(2) Py(2)];% Py(1)];

end
```

(b) Diagrama de bloques



ANEXO 2: BANDAS DE GERSHGORIN

2.1 Función Gersh.m

(a) Código de la Función

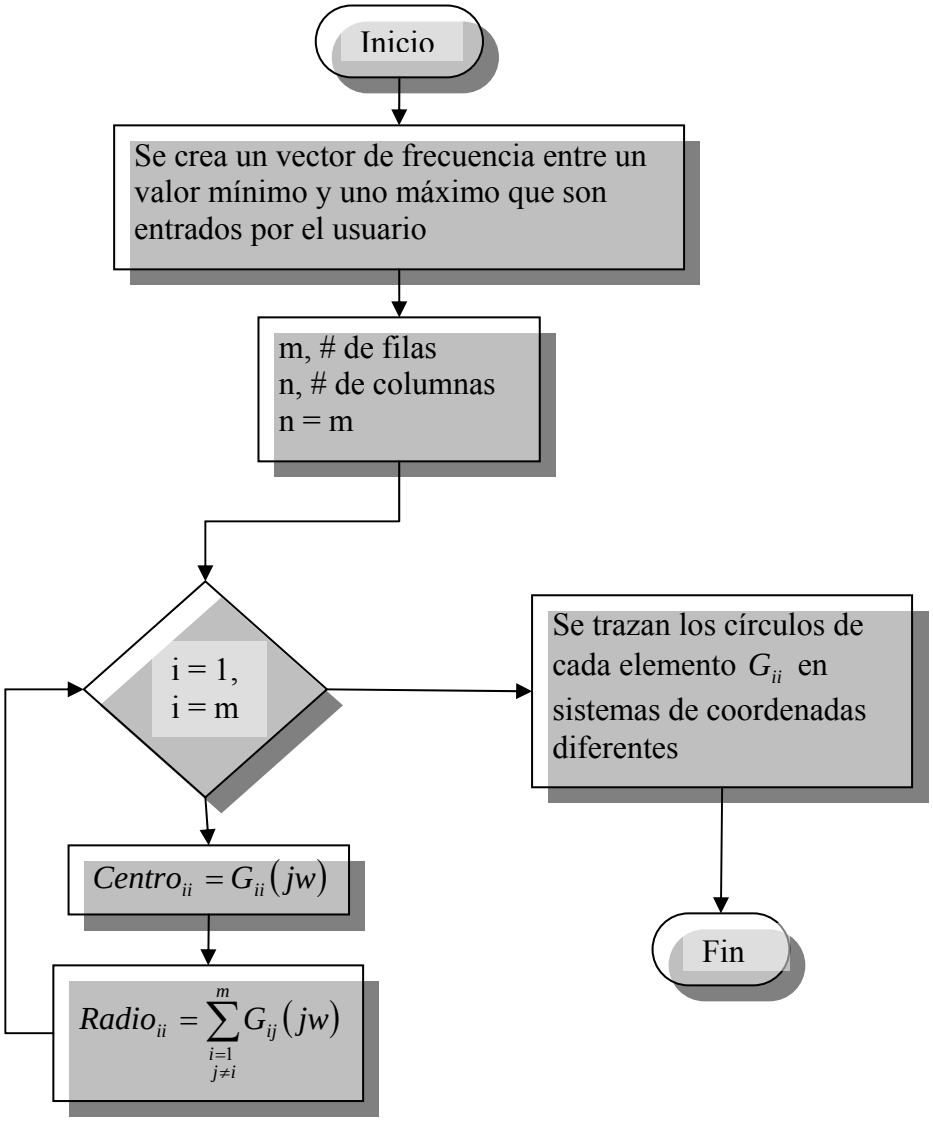
```
function gersh(G,W,Forma)
%   gersh(G,W) Dibuja las bandas de Gershgorin de los elementos de la diagonal de una
%   Matriz dinámica G, que debe ser cuadrada.
%
%   gersh(G,W,'Single')  Dibuja las bandas de Gershgorin en una sola figura.
%
%   G: Es la inversa de la matriz dinámica del sistema, que debe ser cuadrada
%   W: Es un vector de 2 elementos, [Min Max] que contiene el inicio y fin del vector de
%        frecuencia para el cual se quiere trazar las bandas de Gershgorin.


  ni = nargin;

  if class(G) ~= 'tf'
     error('Argumentos de entrada no valido, debe ser una matriz de funciones de
transferencia, cuadrada');
  end
  [m,n] = size(G); %m -- #filas; n -- #Columnas
  if m ~= n
     error('La matriz H debe ser cuadrada');
  elseif (length(W) < 2)
     error(' W debe tener como mínimo 2 elementos [min max]')
  end

  ini = 0;
  hWaitbar = waitbar(ini,'Gershgorin"s Band. Please wait...');

  % No se halla la inversa de la matriz dinámica
  H = G;

  % Hallando el vector de Frecuencia
  w = logspace(log10(W(1)),log10(W(2)));

  % Hallando los centros de los círculos de las tf. de la diagonal
  for I = 1:n,
     sys = H(I,I);
     [Re,Im] = nyquist(sys,w);
     [f,c] = size(Re); siz = [1 f*c]; Re = reshape(Re,siz); Im = reshape(Im,siz);
```

```
    CenterCircle{I} = struct('Name', ['sys' int2str(I) int2str(I)], 'Real', Re, 'Imag', Im);
end

% Hallando los radios de los círculos
for I = 1:m,
    radio = zeros(size(w));
    for J = 1:n,
        if I ~= J
            sys = H(I,J);
            [Re,Im] = nyquist(sys,w);
            [f,c] = size(Re); siz = [1 f*c]; Re = reshape(Re,siz); Im = reshape(Im,siz);

            radio = radio + abs(Re + i*Im);
            RadioCircle{I} = struct('Name', ['Fila-' int2str(I)], 'Radio', radio);
        else
            continue
        end
    end
end

%Pintando los círculos

switch ni
case 2
    inc = (1 - ini)/(length(w)*n);
    f = [];
    for I = 1:m,
        fon = figure(I);
        f =[f fon];
        set(f, 'Visible', 'off');

        for J = 1:length(w),
            ini = ini + inc;
            waitbar(ini,hWaitbar);
            [x,y] = Circle(RadioCircle{I}.Radio(J),CenterCircle{I}.Real(J),…
                           CenterCircle{I}.Imag(J));
            plot(real(x),real(y));
            hold on
        end
        han = plot(CenterCircle{I}.Real, CenterCircle{I}.Imag, 'r');
        Ax = get(han, 'Parent');
        set(Ax,'FontSize',[8],'XColor',[0.4 0.4 0.4],'YColor',[0.4 0.4 0.4]);
        ylabel('Imag','Color','k'); xlabel('Real','Color','k');
        titulo = ['Bandas de Gershgorin: G(' int2str(I) ',' int2str(I) ')'];
        title(titulo)

        %Pintando del punto -1
```

```
        line('XData',-1,'YData',0,'Color','r','Parent',Ax,'Marker','+');

        % Tratamiento de los Ejes de coordenadas
        % Eje Y
        x = zeros(1,2);
        y = get(Ax,'YLim');
        set(Ax,'DefaultLineLineStyle',':')      %definimos el estilo de la línea
        line('XData',x,'YData',y,'Color','k','Parent',Ax)    %Se traza una línea vertical por X;0;

        % Eje X
        x = get(Ax,'XLim');
        y = zeros(1,2);
        set(Ax,'DefaultLineLineStyle',':')      %definimos el estilo de la linea
        line('XData',x,'YData',y,'Color','k','Parent',Ax) %Se traza una linea horizontal por 0;Y
    end
    set(f, 'Visible', 'on');
    close(hWaitbar);
case 3 % Se trazan todas las bandas en una misma figura.
    if strcmp(Forma,'single')
        inc = (1 - ini)/(length(w)*n);
        hdf = figure('Visible', 'off');
        Cont = 0;
        for I = 1:m,
            for Z = 1:n,
                Cont = Cont + 1;
                Ax = subplot(m,n,Cont);
                if Z == I
                    for J = 1:length(w),
                        ini = ini + inc;
                        waitbar(ini,hWaitbar);
                        [x,y] = Circle(RadioCircle{I}.Radio(J),CenterCircle{I}.Real(J),…
                                       CenterCircle{I}.Imag(J));
                        plot(real(x),real(y));
                        hold on
                    end
                    plot(CenterCircle{I}.Real, CenterCircle{I}.Imag, 'r');
                else
                    frsp = [];
                    frsp = freqresp(G(I,Z),w);
                    f = size(frsp); siz = [1 prod(f)]; frsp = reshape(frsp,siz);
                    re = real(frsp); im = imag(frsp);
                    plot(re,im,'Parent',Ax);
                end


                set(Ax,'FontSize',[8],'XColor',[0.4 0.4 0.4],'YColor',[0.4 0.4 0.4]);
                ylabel('Imag','Color','k'); xlabel('Real','Color','k');
```

```
            titulo = ['Bandas de Gershgorin: G(' int2str(I) ',' int2str(Z) ')'];
            title(titulo);

            %Pintando del punto -1
            line('XData',-1,'YData',0,'Color','r','Parent',Ax,'Marker','+');

            % Tratamiento de los Ejes de coordenadas
            % Eje Y
            x = zeros(1,2);
            y = get(Ax,'YLim');
            set(Ax,'DefaultLineLineStyle',':')      %definimos el estilo de la línea
            %Se traza una línea vertical por X;0;
            line('XData',x,'YData',y,'Color','k','Parent',Ax)
            % Eje X
            x = get(Ax,'XLim');
            y = zeros(1,2);
            set(Ax,'DefaultLineLineStyle',':')      %definimos el estilo de la línea
            %Se traza una línea horizontal por 0;Y
            line('XData',x,'YData',y,'Color','k','Parent',Ax)
         end
      end
      set(hdf, 'Visible','on');
      close(hWaitbar);
   else
      close(hWaitbar);
      error('Tercer argumento no es valido')
   end
end

%********************Pintando un Circulo************************
function [x, y] = Circle(Radio, A, B)
% [x, y] = Circle(Radio, A, B) calcula todos los puntos (x;y) que forman un círculo
% con radio (Radio) y centro en (A;B)
   x1 = linspace((A - Radio),(A + Radio));
   y = sqrt(Radio^2 - (x1 - A).^2) + B;
   x2 = linspace((A + Radio),(A - Radio));
   y = [y -sqrt(Radio^2 - (x2 - A).^2) + B];
   x = [x1 x2];
```

(b) Diagrama de bloques



ANEXO 3: SEUDO DIAGONALIZACION

3.1 Función findK.m

(a) Código de la Función

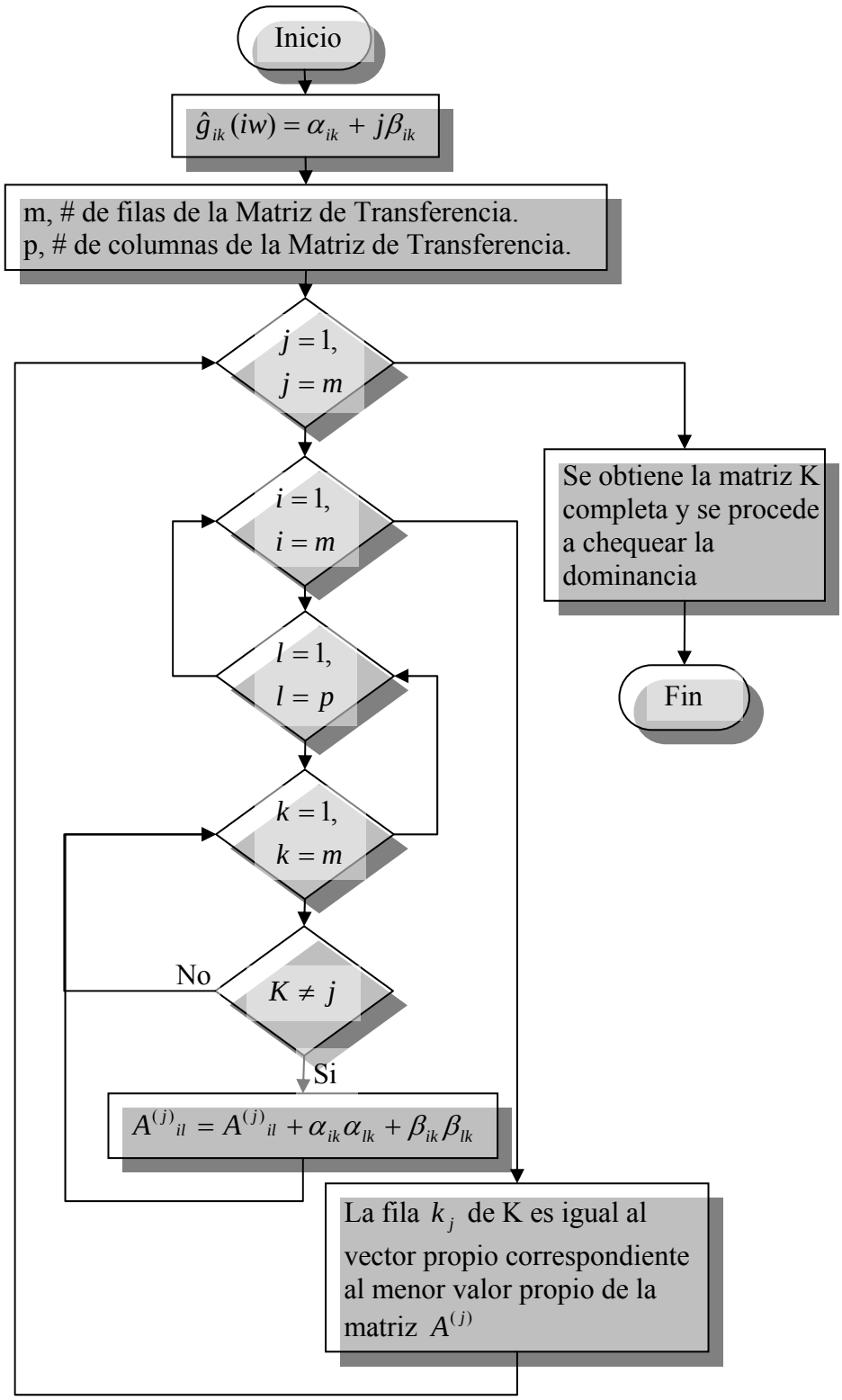
```
function [K] = findK(Gp, w, Gamma)
% K = findK(Gp, w, Gamma)
% Halla la matriz K que hace a la inversa de la matriz dinámica del sistema dominante por
% fila
%
% Gp es la matriz dinámica del proceso, debe ser cuadrada.
%
% [K] = findK(Gp)          ==> K = Gp(0)si G(0) es real.
% [K] = findK(Gp, w)       ==> Halla la matriz K para un valor de w dado
% [K] = findK(Gp, w, Gamma) ==> Halla la matriz K para un conjunto de frecuencias (w)
y un  vector de peso Gamma, Gamma y w deben ser de la misma longitud


[m,p] = size(Gp);       % m --> filas
if p ~= m               % p --> Columnas
   error('La matriz dinámica debe ser cuadrada');
end

G = invtf(Gp);  %se halla la inversa de la matriz dinámica
n = nargin;
switch n
case 1  %[K] = find(Gp)
   w = 0;
   for I = 1:m,
      for J = 1:p,
         frsp = evalfr(Gp(I,J),j*w);
         re(I,J) = real(frsp); im(I,J) = imag(frsp);
      end
   end
   K = re;

case 2  %[K] = findK(Gp, w)
   if length(w) == 1
      [m,p] = size(G);            % m --> filas
                                  % p --> Columnas
      re = {}; im = {};
      for I = 1:m,
         for J = 1:p,
            frsp = evalfr(G(I,J),j*w);
```

```
            re{I,J} = real(frsp); im{I,J} = imag(frsp);
        end
    end

    A = {};
    for i = 1:m,
        A{i} = [];
        for h = 1:m,
            for n = 1:p,
                A{i}(h,n) = 0;
                for l = 1:p,
                    if l ~= i
                        A{i}(h,n) = A{i}(h,n) + re{h,l}*re{n,l} + im{h,l}*im{n,l};
                    end
                end
            end
        end

        if det(A{i}) < 0
            warning('Matriz A no es Semidefinida Positiva')
        end
        neg = [];
        [V, D] = eig(A{i}); neg = find(D < 0);
        if ~isempty(neg)
            warning('valores propios negativos')
        end
        vp = diag(D); mnm = min(vp); ind = find(vp == mnm);
        K(i,:) = V(:,ind(1))';
        if (K(i,:) < 0)
            K(i,:) = -K(i,:);
        end
    end

elseif length(w) >= 2
    error('longitud de w debe ser igual 2')
end
case 3 %[K] = findK(Gp, w, Gamma)
if length(w) ~= length(Gamma)
    error('Gamma y W deben tener la misma longitud')
end

re = {}; im = {};
for I = 1:m,
    for J = 1:p,
        frsp = freqresp(G(I,J),w);
        f = size(frsp); siz = [1 prod(f)]; frsp = reshape(frsp,siz);
        re{I,J} = real(frsp); im{I,J} = imag(frsp);
```

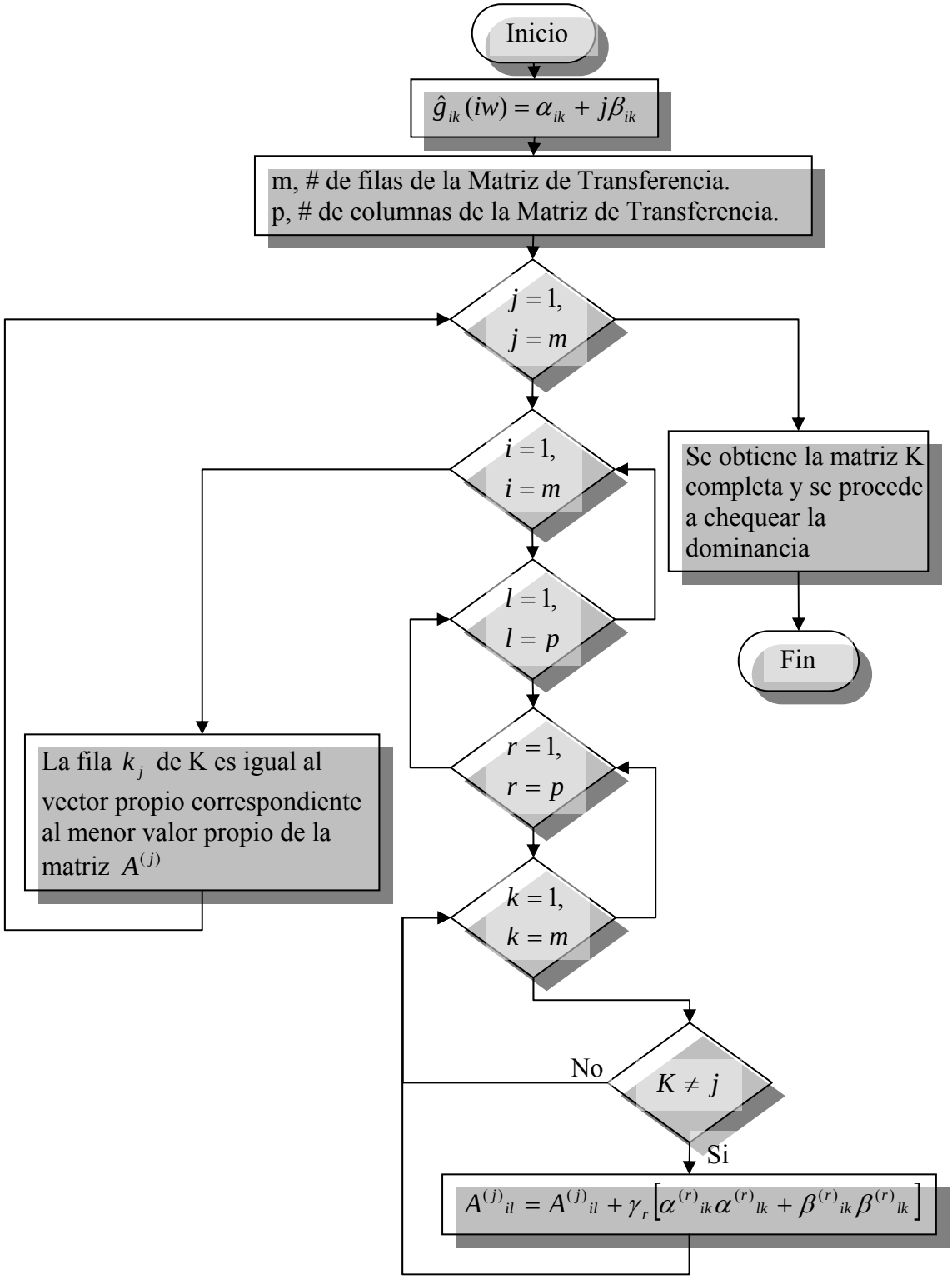
```
      end
   end

   B = {};
   for i = 1:p,
      B{i} = [];
      for h = 1:m,
         for n = 1:p,
            B{i}(h,n) = 0;
            for r = 1:length(w),
               for l = 1:p,
                  if l ~= i
                     B{i}(h,n) = B{i}(h,n) + Gamma(r)*(re{h,l}(r)*re{n,l}(r) + …
                                 im{h,l}(r)*im{n,l}(r));
                  end
               end
            end
         end
      end

      if det(B{i}) < 0
         warning('Matriz A no es Semidefinida Positiva')
      end
      neg = [];
      % Se hallan los valores propios
      [V, D] = eig(B{i}); neg = find(D < 0);
      if ~isempty(neg)
         warning('Valores propios negativos')
      end
      % Se hallan el vector propio correspondiente al mínimo valor propio
      vp = diag(D); mnm = min(vp); ind = find(vp == mnm);
      K(i,:) = V(:,ind(1))';
      if K(i,:) < 0
         K(i,:) = -K(i,:);
      end
   end
   if K < 0
      K = -K;
   end

otherwise
   error('Parámetros de entrada inválidos');
end

% Chequeo de dominancia.

reply = input('¿Desea chequear la dominancia? S/N [S]: ','s');
```

```
if (isempty(reply))|(reply == 's')|(reply == 'S')
   w = input('Entre el rango de frecuencia:');
   if m < 3
      gersh(K*G,w,'single');
   else
      gersh(K*G,w);
   end
end
```

(b.1) Diagrama de bloque (Metodo Seudo Diagonalización)



(b.2) Diagrama de Bloque (Metodo Seudo Diagonalización ampliado)



3.2 Función invtf.m

(a) Código de la Función

```
function [H] = invtf(G)
% Haya la inversa de una matriz dinámica, donde G es la Matriz Dinámica,
% la misma no debe tener retardos.

% Analizando los retardos
if hasdelay(G)
   error('La matriz de transferencia tiene retardos, haga una aproximación pade');
end

% Convirtiendo G a simbólica
[m,n] = size(G);
s = sym('s');

for I = 1:m,
   for J = 1:n,
      Gsim(I,J) = poly2sym(G.num{I,J},s)/poly2sym(G.den{I,J},s);
   end
end

% Hallando la inversa de la matriz simbólica
Ginv = inv(Gsim);
% Convirtiendo la inversa a 'tf'
[m, n] =size(Ginv);
s = tf('s');
for I = 1:m,
   for J = 1:n,
      H(I,J) = eval(Ginv(I,J)); % H es una matriz dinámica de 'tf'
   end
end
```

ANEXO 4: DIAGRAMA DE BODE

4.1 Función nBode.m

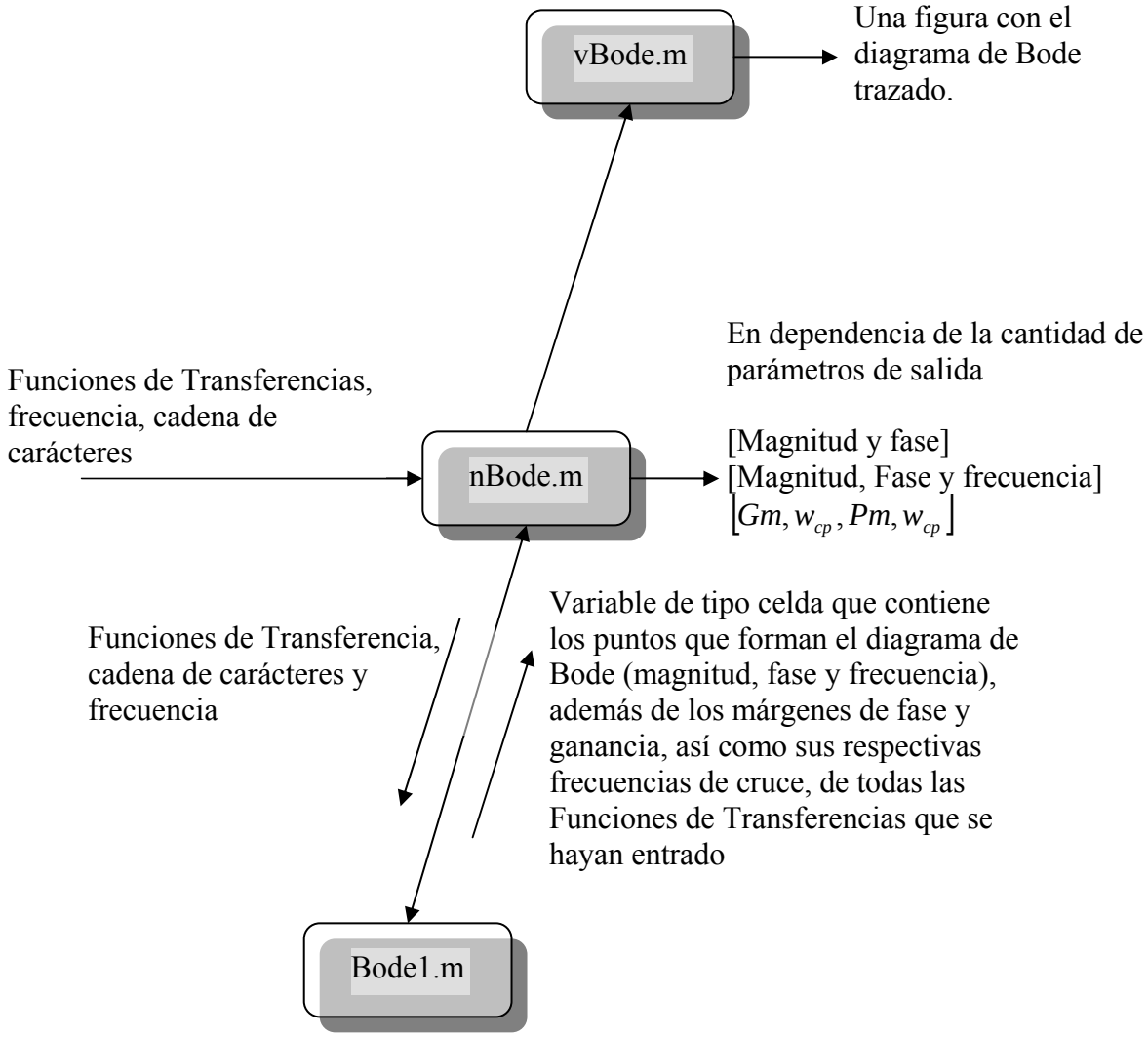
(a) Código de la Función

```
function[varargout] = nbode(varargin)
%  nbode(sys), nbode(num, den): Traza el diagrama de Bode asintótico
%
%  nbode(sys,'realtoo'), nbode(num, den, 'realtoo'): Traza el diagrama de
%  Bode asintótico y real.
%
%  nbode(sys1,sys2,...):Traza el diagrama de Bode de varios sistemas
%  nbode(sys,sys2,...,'realtoo'): Traza el diagrama de Bode asintótico
%  de varios sistemas conjuntamente con el real
%
%  [AF,fase] = nbode(...): Devuelve la magnitud y fase del diagrama de Bode
%  asintótico.
%
%  [AF,fase,frec] = nbode(...): Devuelve la magnitud, fase y frecuencia del
%  diagrama de Bode asintótico
%
%  [Gm, Wcg, Pm, Wcp] = nbode(...): Devuelve
%  Gm -- Margen de Ganancia
%  Pm -- Margen de fase
%  Wcg-- Frec. a la cual se calcula Gm
%  Wcp-- Frec. a la cual se calcula Pm
%
%  NOTA: Si se usa [...] = nbode(...,'realtoo'), la salida la da con
%  respecto al diagrama de Bode real, y si no es con respecto al diagrama
%  de bode asintótico. Además la salida es con respecto al primer sistema
%  que se le pasa como parámetro

  str = ['']; frec = [];
  ni = nargin; no = nargout;
  switch ni
     case 1
        sys = varargin{1};
     case 2
        for I = 1:length(varargin),
           sys1 = varargin{I};
           if strcmp(class(sys1),'tf')
              sys{I} = sys1;
           elseif strcmp(class(sys1),'double')
              if I == 1
```

```
            num = varargin{1}; den = varargin{2};
            sys = tf(num,den);
         elseif strcmp(class(sys1),'double')
            frec = sys1;
         end
      elseif strcmp(class(sys1),'char')
         str = sys1;
      end
   end
case 3
   for I = 1:length(varargin),
      sys1 = varargin{I};
      if strcmp(class(sys1),'tf')
         sys{I} = sys1;
      elseif strcmp(class(sys1),'double')
         if I == 1
            num = varargin{1};
            den = varargin{2};
            sys = tf(num,den);
            str = varargin{3};
         elseif strcmp(class(sys1),'double')
            frec = sys1;
         else
            error('Parámetros de entrada no validos')
         end
         break
      elseif strcmp(class(sys1),'char')
         str = sys1;
      else
         error('Parámetros de entrada no validos');
      end
   end
otherwise
   for I = 1:length(varargin),
      sys1 = varargin{I};
      if strcmp(class(sys1),'tf')
         sys{I} = sys1;
      elseif strcmp(class(sys1),'char')
         str = sys1;
      elseif strcmp(class(sysi),'double')
         frec = sys1;
      else
         error('Parámetros de entrada no validos');
      end
   end
end
% se calculan los puntos que forman el diagrama de Bode
```

```
Data = bode1(sys,str,frec);

switch no
   case 0
      vBode(Data,str);
   case 2
      if strcmp(str,'realtoo')
         varargout{1} = Data{1}.DiagReal.Magnitud;
         varargout{2} = Data{1}.DiagReal.Fase;
      else
         varargout{1} = Data{1}.DiagAsint.Magnitud;
         varargout{2} = Data{1}.DiagAsint.Fase;
      end
   case 3
      if strcmp(str,'realtoo')
         varargout{1} = Data{1}.DiagReal.Magnitud;
         varargout{2} = Data{1}.DiagReal.Fase;
         varargout{3} = Data{1}.DiagReal.Frec;
      else
         varargout{1} = Data{1}.DiagAsint.Magnitud;
         varargout{2} = Data{1}.DiagAsint.Fase;
         varargout{3} = Data{1}.DiagAsint.Frec;
      end
   case 4
      varargout{1} = Data{1}.Margenes.Gm;
      varargout{2} = Data{1}.Margenes.Wcg;
      varargout{3} = Data{1}.Margenes.Pm;
      varargout{4} = Data{1}.Margenes.Wcp;
   otherwise
      error('Inválidos parámetros de salida')
end
```

(b) Diagrama de bloques



4.2 Función Bode1.m

(a) Código de la Función

```
function[Data] = bode1(sys,rt, frec)
% Data contiene todos los puntos de bode
if isempty(frec)
   if iscell(sys)
      mx = 0;mn = inf;
      for I = 1:length(sys),
         [m,p,f]= bode(sys{I});
         if mx < max(f)
            mx = max(f);
         end
         if mn > min(f)
            mn = min(f);
         end
      end
      frec = VectFrecR(mn,mx);
      for I = 1:length(sys),
         Data{I} = bode11(sys{I},rt,frec);
      end
   else
      Data{1} = bode11(sys,rt);
   end
else
   for I = 1:length(sys),
      Data{I} = bode11(sys{I},rt,frec);
   end
end
 strF = str2func( {'bode11'} );

% Función para calcular el diagrama de bode de un solo sistema
function[Data] = bode11(sys,rt,frec)
 ni = nargin;
 if ni == 2
    frec = [];
 elseif (ni < 2)|(ni > 3)
    error('Demasiados parámetros de entrada');
 end
 if class(sys) ~= 'tf'
    error('Parámetro de entrada no valido')
 end

 [num,den] = tfdata(sys,'v');
```

```
  ind = [];
  for I = 1:length(num),
     if num(I) == 0
        ind = [ind I];
     else
        break
     end
  end
  num(ind) = []; ind = [];
  for I = 1:length(den),
     if den(I) == 0
        ind = [ind I];
     else
        break
     end
  end
  den(ind) = [];

  [Num,Den,K] = NORML(num,den);

  N = -(roots(Num));
  D = -(roots(Den));

  if length(N) <= length(D), SysType = 'propio'; else, SysType = 'impropio'; end
  raices = [N; D]; raices = raices(raices ~= 0);
  if min(frec) >= min(abs(raices))
    warning('El mínimo valor del vector frecuencia debe ser menor que la mínima frecuencia
esquina');
    frec = VectFrecR(min(abs([N; D]))/2, frec(end));
  end
  if max(frec) <= max(abs(raices))
    warning('El máximo valor de la frecuencia debe ser mayor que la máxima frecuencia
esquina');
    frec = VectFrecR(frec(1), 2*max(abs([N; D])));
  end

  NumCeros = 0;
  RN = []; CN = [];

  for I = 1:length(N),
     if N(I) == 0
        NumCeros = NumCeros + 1;
        %continue;
     elseif imag(N(I)) == 0
        RN = [RN N(I)];
     else
        if mod(I,2) == 1
```

```
            CN = [CN abs(N(I))];
        end
    end
end

RD = []; CD = [];

for I = 1:length(D)
    if D(I) == 0
        NumCeros = NumCeros - 1;
        %continue;
    elseif imag(D(I)) == 0
        RD = [RD D(I)];
    else
        if mod(I,2) == 1
            CD = [CD abs(D(I))];
        end
    end
end



% Ya se tienen las raíces reales en R
%           y las raíces complejas en C

SYS = tf(num,den);
if isempty(frec)
    [MAG,fase,frec] = BODE(SYS);
end
mn = frec(1); mx = frec(end); frec = VectFrecR(mn,mx);
[MAG,fase] = BODE(SYS,frec);
fase = reshape(fase,[prod(size(fase)) 1]);
if (fase >= 0)&(strcmp(SysType,'propio')), fase = fase - 360*ones(size(fase)); end

ABF = []; ARN = []; ARD = []; ACN = []; ACD = [];


%para las soluciones que son cero

ABF = NumCeros*20*log10(frec);

%Para las raíces reales del numerador.

for I = 1:length(RN),
    w = VectFrecR(RN(I),mx);
    A = 20*log10(w/RN(I));
    A = [zeros(1,length(frec) - length(A)) A];
```

```
    if I == 1
        ARN = A;
    else
        ARN = ARN + A;
    end
end

%Para las raíces reales del denominador.

for I = 1:length(RD),
    w = VectFrecR(RD(I),mx);
    A = -20*log10(w/RD(I));
    A = [zeros(1,(length(frec)-length(A))) A];
    if I == 1
        ARD = A;
    else
        ARD = ARD + A;
    end
end

%Para las raíces complejas del numerador

for I = 1:length(CN),
    w = VectFrecR(CN(I),mx);
    A = 40*log10(w/CN(I));
    A = [zeros(1,length(frec) - length(A)) A];
    if I == 1
        ACN = A;
    else
        ACN = ACN + A;
    end
end

%Para las raíces complejas del denominador.

for I = 1:length(CD),
    w = VectFrecR(CD(I),mx);
    A = -40*log10(w/CD(I));
    A = [zeros(1,length(frec) - length(A)) A];
    if I == 1
        ACD = A;
    else
        ACD = ACD + A;
    end
end
```

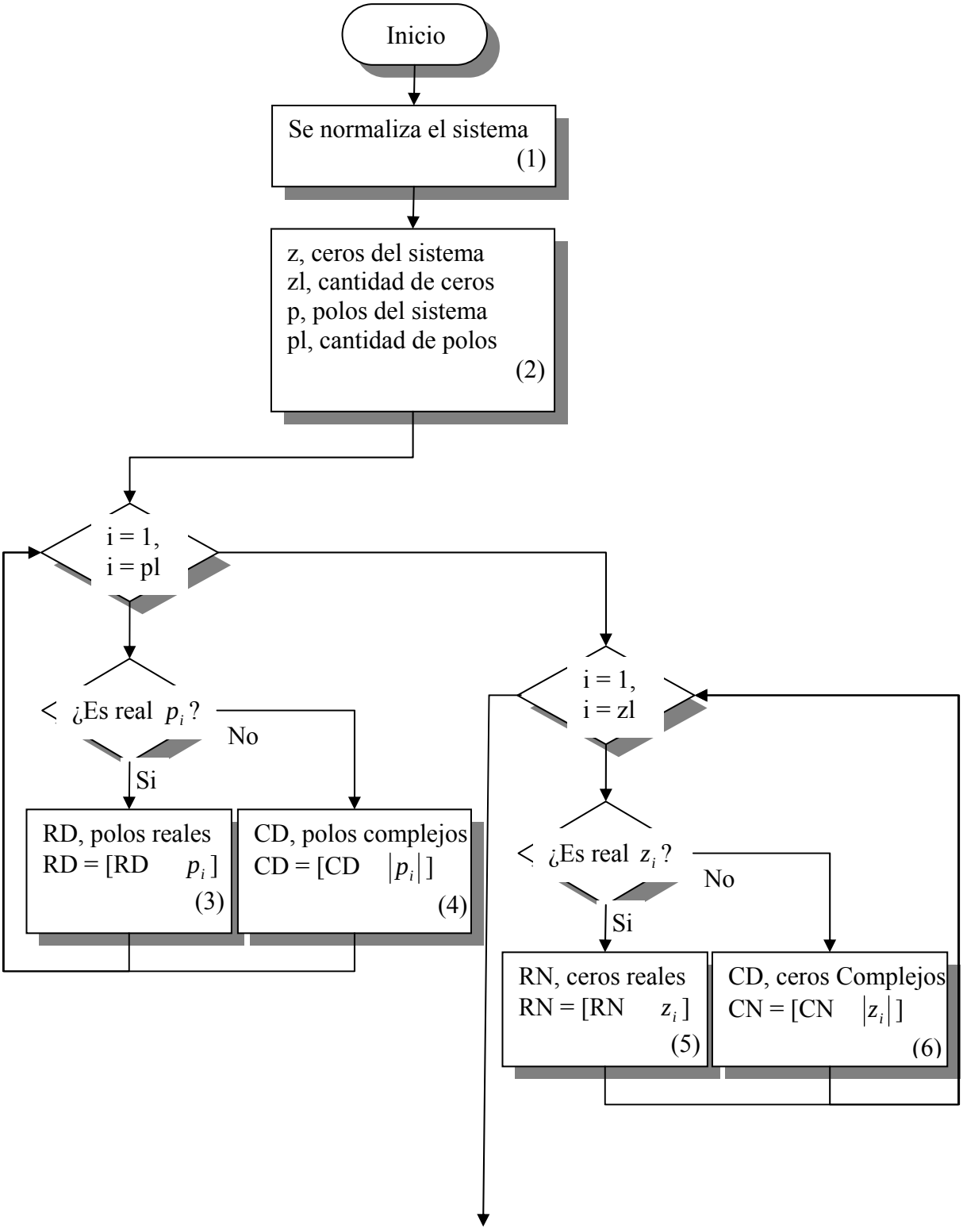
```
AF = zeros(1,length(frec)) + 20*log10(K);

if not(isempty(ABF))
   AF = AF + ABF;
end
if not(isempty(ARN))
   AF = AF + ARN;
end
if not(isempty(ARD))
   AF = AF + ARD;
end
if not(isempty(ACN))
   AF = AF + ACN;
end
if not(isempty(ACD))
   AF = AF + ACD;
end
AF = AF'; frec = frec';


W = abs([RN CN RD CD]); mgAF = [];W = sort(W);
for I = 1:length(W),
   mgAF = [mgAF buscar(frec,W(I),AF)];
end

DiagAsin.Magnitud = AF;
DiagAsin.Fase = fase;
DiagAsin.Frec = frec;
DiagAsin.Data.FrecEsq = W;
DiagAsin.Data.MgFrcEq = mgAF;

[mag, phase, wr] = respbode(SYS,frec);
mag = reshape(mag, [prod(size(mag)) 1]); mag = 20*log10(mag);
phase = reshape(phase,[prod(size(phase)), 1]);
if (phase > 0)&(strcmp(SysType,'propio')), phase = phase - 360*ones(size(phase)); end

DiagReal.Magnitud = mag;
DiagReal.Fase = phase;
DiagReal.Frec = wr;


if strcmp(rt,'realtoo')
   [Gm,Pm,Wcg,Wcp,str, ispositive] = Margen(mag,phase,wr);
else
   [Gm,Pm,Wcg,Wcp,str, ispositive] = Margen(AF,fase,frec);
end
```

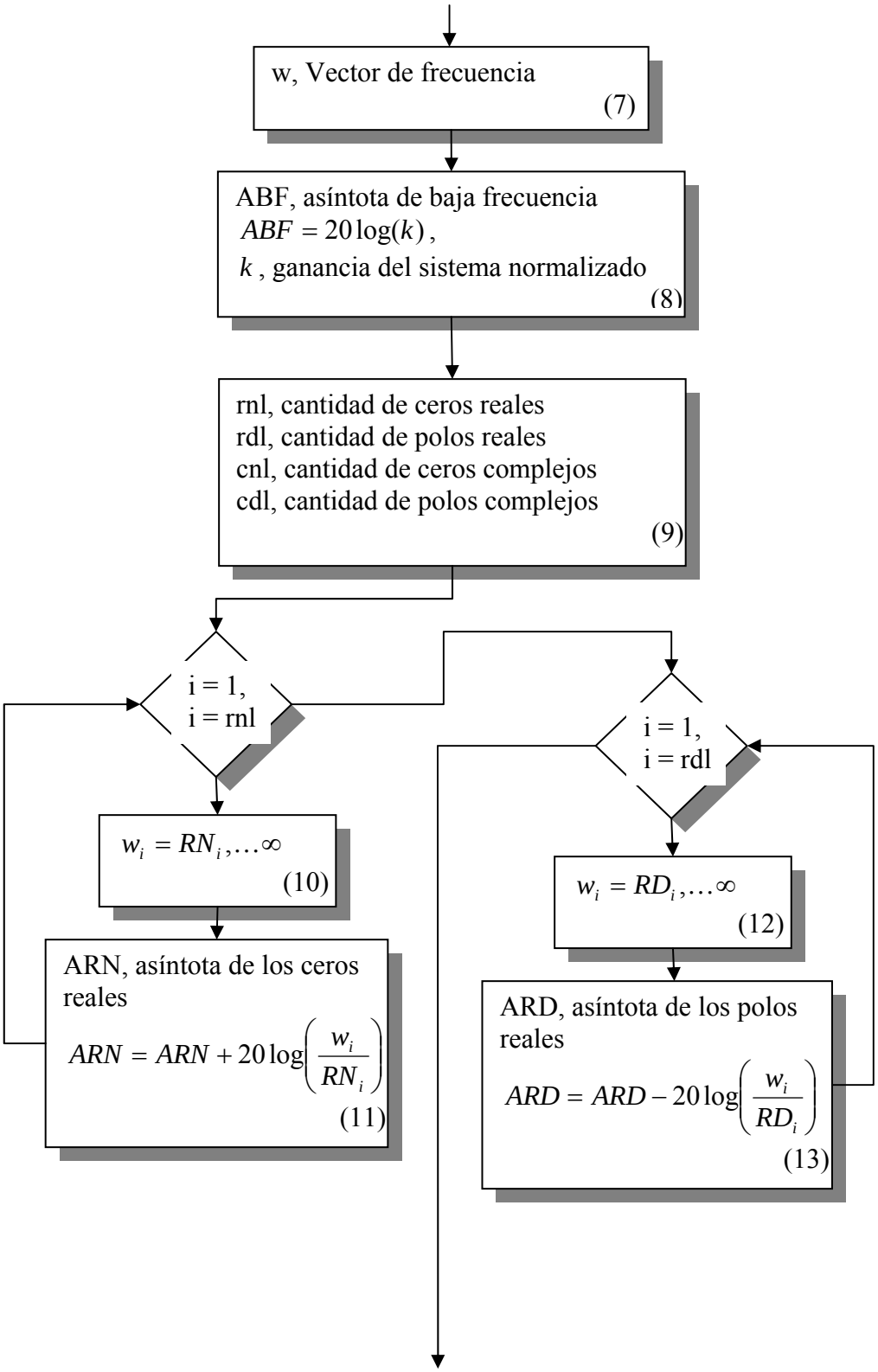
```
margen.Gm = Gm;
margen.Wcg = Wcg;
margen.Pm = Pm;
margen.Wcp = Wcp;
margen.ispositivo = ispositive;
margen.String = str;


Data = struct('sys',sys,'DiagAsint', DiagAsin, 'DiagReal', DiagReal, 'Margenes', margen);

%-------------------------LOCAL FUNCTION-----------------------------

function[nB] = buscar(A,nA,B)
%Dado un número nA en un Vector A,
%la función le devuelve el número que le corresponde en el vector B

if length(A) ~= length(B)
   error('Los vectores A y B deben ser de la misma longitud')
end
nB = [];
for I = 1:length(A) - 1,
   cond1 = (A(I) > nA)&(A(I+1) < nA);
   cond2 = (A(I) < nA)&(A(I+1) > nA);
   if (cond1) | (cond2)
      m = (B(I+1) - B(I))/(A(I+1) - A(I));  % Pendiente
      n = B(I) - m*A(I);                    % Intersecto eje y
      nB = m*nA + n;
      return
   elseif A(I) == nA
      nB = B(I);
      return
   end
end


%---------------Margen-------------------------Margen----------------------------


function[Gm,Pm,Wcg,Wcp,str, ispositive] = Margen(Mag,Phse,w)
Wcp = [];
Wcg = [];
Pm = [];
Gm = [];
ispositive = 0;

for I = 1:length(w)-1,
   % Hallando margen de fase
```

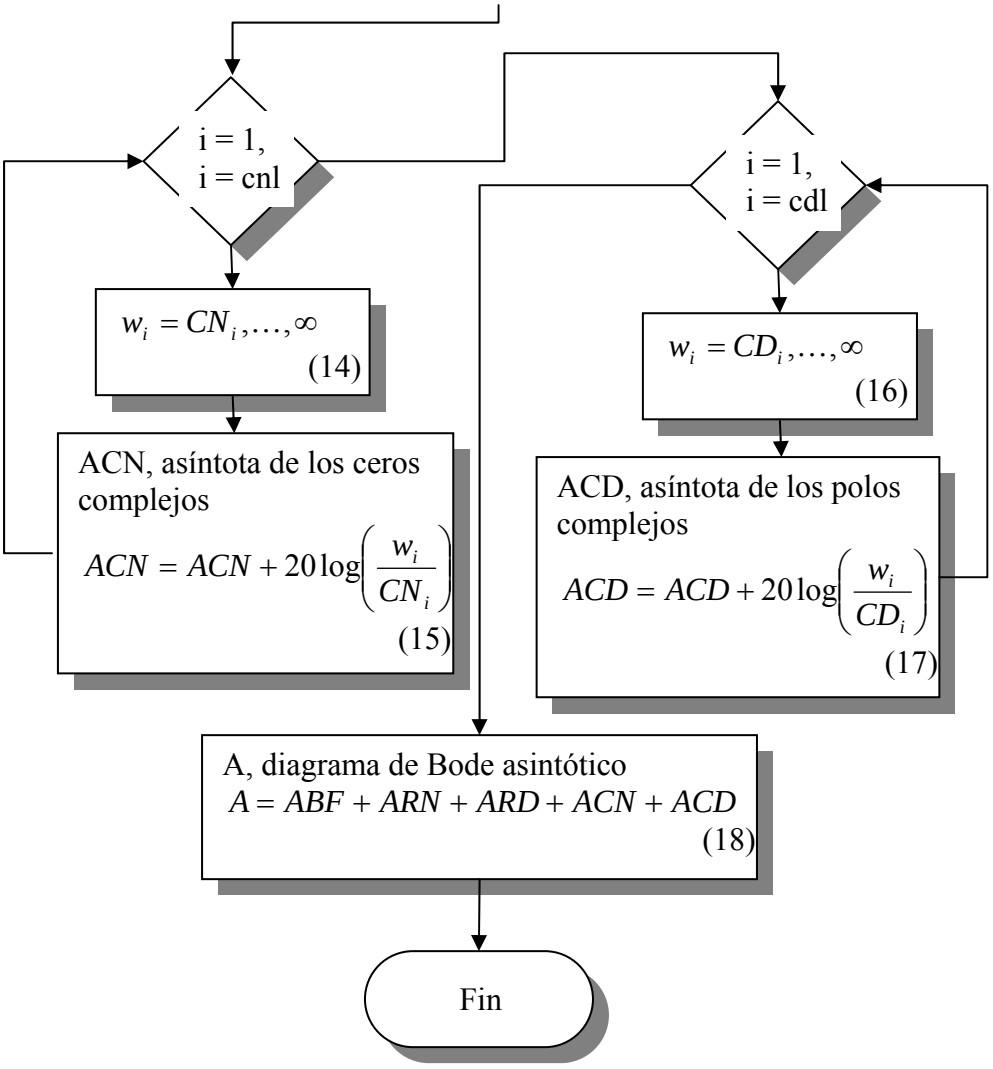
```
    if ((round(Mag(I)) > 0)&(round(Mag(I+1)) <= 0))|((round(Mag(I)) <
0)&(round(Mag(I+1)) >= 0))
        m = (Mag(I)-Mag(I+1))/(w(I)-w(I+1));
        n = Mag(I)-m*w(I);
        Wcp =[Wcp -n/m];
        if Phse(I) < 0, Pm = [Pm Phse(I) + 180];
        else Pm = [Pm 180 - Phse(I)]; ispositive = 1; end
    end
    % Hallando margen de ganancia
    if  ((Phse(I) >= -180)&(Phse(I+1) <= -180))|((Phse(I) <= -180)&(Phse(I+1) >= -180))
        Wcg = [Wcg w(I)];
        Gm = [Gm -Mag(I)];
    elseif ((Phse(I) >= 180)&(Phse(I+1) <= 180))|((Phse(I) <= 180)&(Phse(I+1) >= 180))
        Wcg = [Wcg w(I)];
        Gm = [Gm -Mag(I)];
    end
end
% creando la cadena para visualizar los márgenes.
if isempty(Wcp)
    Pm = NaN;
    sp = {sprintf('Pm = NaN ')};
else
    [Wcp,Ind] = ClearAprx(Wcp); Pm(Ind) = [];
    sp = cell(length(Pm), 1);
    for I = 1:length(Pm),
        sp{I} = sprintf('Pm = %0.4g deg(a %0.4g rad/seg)', Pm(I), Wcp(I));
    end
end

if isempty(Wcg)
    Gm = NaN;
    sg = {sprintf('Gm = NaN ')};
else
    [Wcg,Ind] = ClearAprx(Wcg); Gm(Ind) = [];
    sg = cell(length(Gm),1);
    for I = 1:length(Gm),
        sg{I} = sprintf('Gm = %0.4g db(a %0.4g rad/seg)', Gm(I), Wcg(I));
    end
end

stmp = {sg; sp};
ind = 1;
for I =1:length(stmp),
    for J = 1:length(stmp{I}),
        str{ind,1} = stmp{I}{J,1};
        ind = ind + 1;
    end
```

```
end

function [D,Ind] = ClearAprx(G)
  Ind = []; E = 1e-3;
  leng = length(roundn(G,-10));
  if leng > 1
     D = sort(G); ind = [];
     I = 1;
     while I < leng
        for J = (I + 1):length(D),
           if (D(I) <= D(J) + E)&(D(I) >= D(J) - E)
              ind = [ind J];
           end
        end
        Ind = [Ind ind];
        D(ind) = []; ind = [];
        leng = length(D);
        I = I + 1;
     end
  else
     D = G;
  end
```

(b) Diagrama de bloques




w, Vector de frecuencia
(7)
ABF, asíntota de baja frecuencia
$ABF = 20\log(k)$,
k, ganancia del sistema normalizado
(8)
rnl, cantidad de ceros reales
rdl, cantidad de polos reales
cnl, cantidad de ceros complejos
cdl, cantidad de polos complejos
(9)
i = 1,
i = rnl
$w_i = RN_i, \ldots \infty$
(10)
ARN, asíntota de los ceros reales
$ARN = ARN + 20\log\left(\frac{w_i}{RN_i}\right)$
(11)
i = 1,
i = rdl
$w_i = RD_i, \ldots \infty$
(12)
ARD, asíntota de los polos reales
$ARD = ARD - 20\log\left(\frac{w_i}{RD_i}\right)$
(13)


i = 1,
i = cnl
$w_i = CN_i, \ldots, \infty$
(14)
ACN, asíntota de los ceros complejos
$ACN = ACN + 20\log\left(\frac{w_i}{CN_i}\right)$
(15)
i = 1,
i = cdl
$w_i = CD_i, \ldots, \infty$
(16)
ACD, asíntota de los polos complejos
$ACD = ACD + 20\log\left(\frac{w_i}{CD_i}\right)$
(17)
A, diagrama de Bode asintótico
$A = ABF + ARN + ARD + ACN + ACD$
(18)
Fin

4.3 Función vBode.m

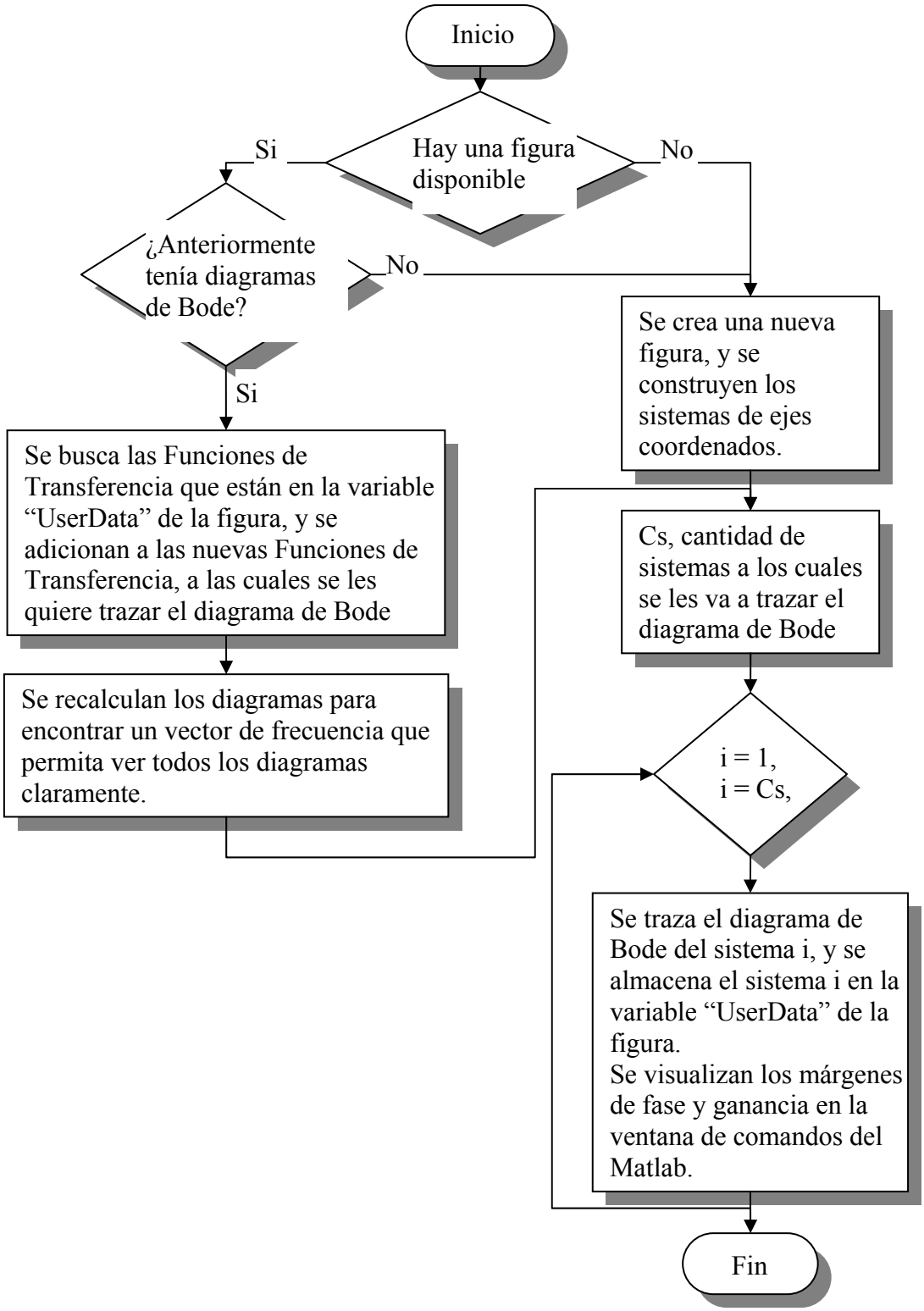
(a) Código de la Función

```
function vbode(Data,rt)

 fig = gcf;
 sys = get(fig,'UserData');
 % chequeando si existe un axes disponible.
 Ax = findobj(fig,'Type','axes'); add = 0;

 for I = 1:length(Ax),
    add = (add) | (strcmp(get(Ax(I), 'NextPlot'), 'add'));
 end

 if ~(iscell(sys))&(add)
    warning('El diagrama no se pudo trazar en la figura activa')
    delete(fig); fig = gcf; add = 0;
 end

 if (length(Ax) == 2) & (add)
    cline = findobj('Type','line');
    for I = 1:length(cline)
       UsDt = get(cline(I),'UserData');
       if iscell(UsDt)
          if strcmp(UsDt{1},'Fase')
             Ax(2) = get(cline(I), 'Parent');
          elseif strcmp(UsDt{1},'Mag')
             Ax(1) = get(cline(I), 'Parent');
          end
       end
    end

    for I = 1:length(Data)
       sys{end + 1,1} = Data{I}.sys;
    end
    Data = bode1(sys,rt,[]);
 else % no existe un axes disponible
    % creando los axes
    set(fig, 'UserData', []);
    Ax(2) = subplot(2,1,2,'remplace');
    set(Ax(2), 'Tag', 'BodeFase');
    Ax(1) = subplot(2,1,1,'remplace');
    pos = get(Ax(1), 'Position'); pos(2) = 0.52;
    set(Ax(1), 'position', pos, 'Tag', 'BodeMag');
```

```
  end

  for I = 1:2,
     % Definiendo tamaño y color de los ejes y letras
     set(Ax(I), 'FontSize',[8], 'XColor', [.4 .4 .4], 'YColor', [.4 .4 .4], 'XScale', 'log', 'Box',
'on');
  end
  % Definiendo los nombres de los ejes
  set(get(Ax(1),'Title'), 'String', {'Diagrama de Bode'}, 'FontSize', [8]);
  set(get(Ax(1),'yLabel'), 'String', 'Magnitud (db)', 'Color', 'k', 'FontSize', [8]);
  set(get(Ax(2),'yLabel'), 'String', 'Fase (deg)', 'Color', 'k', 'FontSize', [8]);
  set(get(Ax(2),'xLabel'), 'String', 'Frecuencia (rad/seg)', 'Color', 'k', 'FontSize', [8]);

  % trazando cada uno de los gráficos
  Gm = []; Pm = []; Wcg = []; Wcp = [];
  Color = ['brgkmy']; k = 1; UserData = cell(length(Data),1);
  if length(Data) > 6
     error('Demasiados diagramas');
  end
  for I = 1:length(Data),
     % Organizando datos
     DiagAsin = Data{I}.DiagAsint;
     DiagAsin.SysType = ['Asintotico'];
     PointAsin = Data{I}.DiagAsint.Data;
     if strcmp(rt,'realtoo')
        DiagReal = Data{I}.DiagReal;
        DiagReal.SysType = ['Real'];
     end
     Margenes = Data{I}.Margenes;
     Pm = [Pm Margenes.Pm]; Gm = [Gm Margenes.Gm];
     Wcp = [Wcp Margenes.Wcp]; Wcg = [Wcg Margenes.Wcg];

     UserData{I} = Data{I}.sys;

     %Plotear el diagrama de magnitud asintótico
     HMagAsint(I) = line(  'XData'          ,   DiagAsin.Frec,...
                           'YData'          ,   DiagAsin.Magnitud,...
                           'Parent'         ,   Ax(1),...
                           'Color'          ,   Color(k),...
                           'LineStyle'      ,   '-',...
                           'UserData'       ,   {'Mag' 'Asintotico'},...
                           'ButtonDownFcn'  ,   'ScanBode');
     %Plotiando el diagrama de fase
     HFaseAsint(I) = line( 'XData'          ,   DiagAsin.Frec,...
                           'YData'          ,   DiagAsin.Fase,...
                           'Parent'         ,   Ax(2),...
                           'UserData'       ,   {'Fase'},...
```

```
                    'Color'          ,   Color(k),...
                    'LineStyle'      ,   '-',...
                    'ButtonDownFcn'  ,   'ScanBode');
    %Marcando las frecuencias esquinas
    HPntAsint(I) = line('XData'          ,   PointAsin.FrecEsq,...
                    'YData'          ,   PointAsin.MgFrcEq,...
                    'Parent'         ,   Ax(1),...
                    'Color'          ,   Color(k),...
                    'Marker'      ,   '.');

    % Si se quiere plotear el diagrama de Bode real
    if strcmp(rt,'realtoo')
       HMagReal = line('XData'           ,   DiagReal.Frec,...
                    'YData'           ,   DiagReal.Magnitud,...
                    'Parent'          ,   Ax(1),...
                    'UserData'        ,   {'Mag' 'Real'},...
                    'Color'           ,   Color(k+1),...
                    'LineStyle'       ,   '-',...
                    'ButtonDownFcn'   ,   'ScanBode');
       k = k + 2;
    else
       k = k + 1;
    end
    if prod(size( Margenes.String )) <= 50
       disp( Margenes.String );
    end
  end

  set(fig,'UserData', UserData);

  % para pintar una línea que pasa po 0db
  x = get(Ax(1),'XLim');
  y = zeros(1,2);
  line('XData',x,'YData',y,'Color','k','Parent',Ax(1), 'LineStyle', ':')   %Se traza una línea
horizontal por 0.0 db

  %Para pintar líneas horizontales discontinuas que pasen por 180 grados
  y = -180*ones(1,2);
  line('XData',x,'YData',y,'Color','k','Parent', Ax(2), 'LineStyle', ':')
```

(b) Diagrama de bloques


Inicio
Hay una figura disponible
Si
No
¿Anteriormente tenía diagramas de Bode?
No
Si
Se busca las Funciones de Transferencia que están en la variable "UserData" de la figura, y se adicionan a las nuevas Funciones de Transferencia, a las cuales se les quiere trazar el diagrama de Bode
Se recalculan los diagramas para encontrar un vector de frecuencia que permita ver todos los diagramas claramente.
Se crea una nueva figura, y se construyen los sistemas de ejes coordenados.
Cs, cantidad de sistemas a los cuales se les va a trazar el diagrama de Bode
i = 1,
i = Cs,
Se traza el diagrama de Bode del sistema i, y se almacena el sistema i en la variable "UserData" de la figura.
Se visualizan los márgenes de fase y ganancia en la ventana de comandos del Matlab.
Fin