DISEÑO DE UNA INTERFAZ GUI EN MATLAB PARA EL CALCULO DE UN CONTROL OPTIMO DISCRETO

JOHN HAROLD BERMEO AYA

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERIA ELECTRONICA
NEIVA
2007

DISEÑO DE UNA INTERFAZ GUI EN MATLAB PARA EL CALCULO DE UN CONTROL OPTIMO DISCRETO

JOHN HAROLD BERMEO AYA

Director:
AGUSTIN SOTO OTALORA
Docente Programa Ingeniería Electrónica

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERIA ELECTRONICA
NEIVA
2007

Nota de Aceptación

_____
_____
_____
_____
_____
_____

_____
Director

_____
Firma del Jurado

_____
Firma del Jurado

Neiva, marzo 22 de 2007

A Dios, a mis padres Jairo y Martha, a mis hermanas Laura y Pilar, a toda mi familia, y a todos los que han confiado en mí en el transcurso de la carrera.

John Harold

AGRADECIMIENTOS

El autor expresa agradecimiento:

A Dios

Al ingeniero Agustín Soto Otalora, Director de tesis

Al cuerpo de profesores del programa de Ingeniería electrónica.

A todos los compañeros de estudio.

CONTENIDO

LISTA DE FIGURAS

Pág

RESUMEN

La teoría de control clásica, aunque es muy completa, presenta ciertas desventajas como los son los cálculos tediosos e innumerables para llegar a una respuesta óptima. Por tal motivo los métodos de análisis basados en variables de estado proporcionan la manera más fácil y adecuada para trabajar el diseño de controladores de sistemas actualmente.

En muchas ocasiones, existen plantas o sistemas en los que no simplemente basta con realizar un controlador en el que se obtenga el mejor tiempo de respuesta o la mejor respuesta del sistema ante una señal de entrada determinada, sino que es mas importante como se desempeñara el sistema, esto permite saber que tan eficiente es por ejemplo un sistema en el uso de la energía que lo mantiene funcionando. Es aquí donde el control óptimo juega un papel importante, porque el problema de control ya no consiste en encontrar los parámetros comunes, en lugar de ello se pretende es encontrar la señal de entrada necesaria para que todo el sistema mantenga un buen rendimiento permitiendo el ahorro de energía o un buen costo energético.

Por tal motivo pareció razonable, al menos académicamente hablando, implementar una manera fácil de calcular los parámetros necesarios de las ganancias y los valores de la señal de entrada indispensables en la implementación de un control óptimo.

Para iniciar el desarrollo del presente proyecto, se colocaron en práctica los conocimientos adquiridos sobre matlab y su interfaz grafica de usuario GUI, así como de la teoría de control en espacio de estados, permitiendo integrar todo este conocimiento y implementar el programa deseado. Permitiendo de buena manera, abrir las bases necesarias para realizar un software mas especializado con un paquete de programación especifico como puede ser Visual Basic, Borland C, Visual .NET u otro entorno de programación comercial.

ABSTRACT

Theory of classic control, though it's very complete, it has some disadvantages like many and difficult estimates to get an optimal response. Therefore, the based state variables analysis methods are actually providing easier manner and appropriate to work in the design of system controllers.

In many occasions, plants or systems exist in which not simply it is enough with making a controller in whom the best response time or the best answer of the system is obtained before a signal of determined entrance, but that is but important as the system evolved, this allows to know that so efficient it is for example a system in the use of the energy that maintains it working. It is here where the optimal control plays an important role, because the control problem no longer consists of finding the parameters common, instead of it it is tried is to find the signal of necessary entrance so that all the system maintains a good yield allowing the energy saving or a good power cost.

This reason looked rational, less academic speaking, to implement one easy manner to calculate required parameters of gains and the essential input signal values in the implementation of a optimal control.

To begin to develop the present project, it has put practice the acquired knowledge about matlab and its graphic user interface, in addition to control theory in space states, enabling to implement all this knowledge and to implement the desired software program. Allowing opening the necessary bases to make most skilled software with a specify software package like could be Visual Basic, Borland C, visual .NET or any trade software environment.

INTRODUCCION

En la mayoría de las ramas de la Matemática Aplicada, el objetivo de los problemas es analizar una situación determinada, para ser capaces de tener una descripción matemática o modelo de las situaciones reales.

Las áreas clásicas de la Matemática Aplicada, como Mecánica (de partículas, sólidos, fluidos), la Teoría Electromagnética, la Termodinámica, etc., han o desarrolladas ampliamente durante los últimos 200 años y generalmente reflejan este énfasis en análisis. Sin embargo, muchos problemas de gran importancia en el mundo actual, como lo es controlar un sistema para que se comporte de una forma determinada, requieren una visión matemática algo diferente.

La palabra “sistema” se utiliza aquí para indicar una colección de objetos que están relacionados mediante interacciones y que proporciona diferentes tipos de salidas ante diferentes tipos de entradas. Los problemas de control asociados con estos sistemas pueden ser la producción de algún producto químico o siderúrgico de la forma más eficiente posible, aterrizajes automáticos de aviones, alunizajes suaves sobre la Luna, posicionamiento de satélites artificiales, regulaciones de funciones corporales; tales como los latidos del corazón, presión sanguínea, temperatura, y el problema más cercano y actual de controlar la inflación económica.

Sea cual sea el sistema controlado, generalmente es posible dirigir de diversas formas el sistema desde un estado hasta otro mediante la aplicación de dispositivos de control.

Para elegir entre las diferentes estrategias, podemos asignar algún coste a cada una de ellas, tal como el tiempo transcurrido en alcanzar el objetivo, el gasto de combustible, o la cantidad de energía utilizada. La decisión que hay pues que tomar se hace de entre todas las posibles estrategias aquella que se desarrolla con un coste mínimo. Esta es la esencia de los problemas de *Control Óptimo*. Por supuesto, no hay garantía de que el control óptimo sea único o ni si quiera de que exista.

La no-existencia usualmente sucede cuando el coste mínimo matemático no puede ser alcanzado, pero pueden alcanzarse costes arbitrariamente cerca del mínimo. De nuevo, la adición de un componente extra al coste puede hacer el mínimo alcanzable.

1 .EL CONTROL ÓPTIMO CUADRATICO

Cuando se habla de la solución óptima de un problema, intuitivamente se piensa en que ésta es 'la mejor solución', es decir 'insuperable'. De hecho, éste es el significado que puede encontrarse en el Diccionario de la Real Academia Española:

> ***óptimo****: forma procedente del superlativo latino optimus, que significa 'bueno en grado sumo', que no puede ser mejor. Por tanto, es incorrecto su empleo en combinación con muy, más, menos o tan: *muy óptimo, *más óptimo; *menos óptimo, *tan óptimo.*

Sin embargo, como muchos otros adjetivos, la palabra óptimo tiene un alto grado de subjetividad. Efectivamente, un pésimo control desde el punto de vista del comportamiento dinámico podría ser óptimo desde el punto de vista económico y viceversa. Luego, para calificar la bondad de un control (en particular para poder decir que es óptimo) es necesario asociarlo a un 'índice' de desempeño. En términos de control diremos que un control es óptimo si minimiza un funcional de costo en el que claramente se manifiesta un compromiso entre distintas especificaciones y restricciones. A este funcional lo llamaremos índice de desempeño y normalmente lo indicaremos con J. Obviamente, el mismo control evaluado con otro índice de desempeño J no será óptimo.

Todos los conceptos y la formulación del problema de control óptimo son análogos al caso de sistemas continuos en variables de estado.

Todos los objetivos de respuesta temporal (coeficiente de amortiguamiento, elongación, tiempo de asentamiento, etc.) se pierden para centrarse en un único objetivo: la minimización de un índice de desempeño J.

El problema que viene a solucionar el control óptimo es el de saber donde han de reubicarse los autovalores para conseguir unos fines determinados, como puede ser la minimización de una función de coste escalar J. Esta función de coste, o índice de desempeño, suele estar relacionada con la energía y por lo tanto con el coste económico del proceso.

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

$$x(k+1) = Ax(k) + Bu(k) \tag{1}$$

$$y(k) = Cx(k) + Du(k) \tag{2}$$

De donde:
A = matriz de n x n
B = matriz de n x r
C = matriz de m x n
D =matriz de n x r
x(k) = vector de estado (vector -n)
u(k) = vector de control (vector -r)

y definido un cierto índice de desempeño escalar, en el que siempre suelen estar incluidas la señales de control u(k):

$$J = J\left[x(k), y(k), u(k)\right] \tag{3}$$

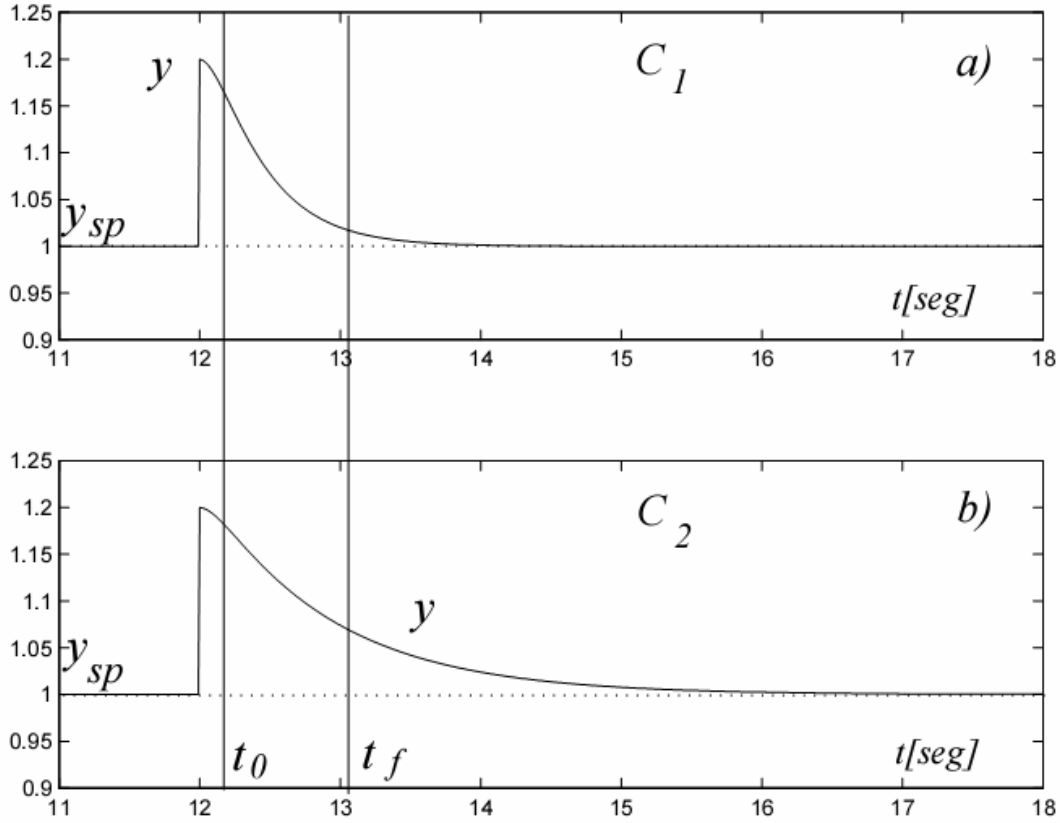
el control óptimo consiste en obtener el vector de las señales de control $\mathbf{u_o}(k)$, que haga mínimo el índice de desempeño J. A este vector se le denomina trayectoria óptima de control.

Cuando el índice de desempeño es función de las variables de estado **x**(k), se trata de un problema de regulación de estado. Cuando es función de las variables de salida **y**(k), se dice que es una regulación óptima de salida. En cualquier caso las variables de control **u**(k) siempre estarán dentro del índice de desempeño.

1.1 El índice de desempeño

Considere la Figura 1. Tanto la parte a) como la parte b) de la figura muestran la variable controlada de un determinado proceso en dos casos diferentes, cuando es controlada empleando los controladores C_1 y C_2 respectivamente. En ambos casos se produce una perturbación en t = 12seg, la cual es rechazada con los transitorios mostrados. Aceptemos que la calidad de un determinado producto depende de esta variable. Luego, no parece desacertado pensar que la bondad de dicho producto en t_0 dependerá de la amplitud del error $e(t_0) = y_{sp} - y(t_0)$, siendo y_{sp} el valor de referencia.

Si usamos esta gráfica para determinar el desempeño de los controladores C_1 y C_2 podría decirse que el controlador C_1 es mejor que el controlador C_2.



Fuente: Introducción al control optimo – Ricardo Julian Mantz – Universidad de la Plata

Figura 1 Comparación del desempeño de 2 controladores

Como en el problema de control optimo se desea minimizar el índice de desempeño para encontrar el vector de control u(k), como puede ser u(0), u(1), u(2),…., u(N-1). Un ejemplo de índice de desempeño cuadrático podría ser:

$$J = \frac{1}{2} x^T(N) S x(N) + \frac{1}{2} \sum_{k=0}^{N-1} \left[x^T(k) Q x(k) + u^T(k) R u(k) \right] \qquad (4)$$

donde N es el instante final de control, que puede ser infinito.

El producto $\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k)$ con **Q** diagonal sería de la forma:

$$x(k)^T Q x(k) = q_{11} x_1^2 + q_{22} x_2^2 + \ldots + q_{nn} x_n^2 \qquad (5)$$

y en este caso se daría más importancia a minimizar unas componentes que otras. El término que sería más importante minimizar sería el de mayor q_{ii}

El término $\frac{1}{2}\sum_{k=0}^{N-1} x^T(k)Qx(k)$ indica que se intenta minimizar el cuadrado del módulo ponderado del vector de estado durante todos los instantes de muestreo desde el instante inicial 0 hasta el instante N-1.

El término $\frac{1}{2}x^T(N)Sx(N)$ minimiza el cuadrado del módulo ponderado del vector de estado en el instante final N. Dependiendo de los valores de la matriz **S**, se pueden seleccionar las variables que se desean que sean mínimas en el instante final. En un problema de regulación **x**(k) tiende a cero por lo que **S** suele elegirse igual a cero.

El significado del tercer la sumatoria es análogo, pero respecto a las señales de control. La razón de incluir las variables de control en el índice de desempeño se debe a razones energéticas, lo cual es muy importante puesto que minimiza la energía de las señales de entrada que son las que realizan los actuadores y por lo tanto son las que cuestan dinero. Cuanto más se desee limitar la excursión de una cierta variable de control u_J, mayor se escogerá el elemento r_{ii} de la matriz **R**.

Normalmente las matrices **S**, **Q** y **R** son matrices diagonales con $\gamma_{ij} = 0$ si $i \neq j$ y además son matrices definidas positivas. Cuando hay alguna variable que no se desea minimizar entonces se hace el correspondiente término $\gamma_{ii} = 0$

En el caso de sistemas discretos los índices más utilizados son:

$$J = \frac{1}{2}\sum_{k=0}^{\infty}\left[x^T(k)Qx(k) + u^T(k)Ru(k)\right] \qquad (6)$$

que correspondería a un índice de regulación de estados en tiempo infinito. En este caso se estaría minimizando el cuadrado del módulo ponderado del vector de estado, por lo que tendería a cero y al mismo tiempo la energía ponderada del vector de entrada a la planta.

Si fuese en tiempo finito:

$$J = \frac{1}{2}\sum_{k=0}^{N}\left[x^T(k)Qx(k) + u^T(k)Ru(k)\right] \qquad (7)$$

Un índice de desempeño para control óptimo de salida sería:

$$J = \frac{1}{2}\sum_{k=0}^{N}\left[y^T(k)Qy(k) + u^T(k)Ru(k)\right] \qquad (8)$$

y dependiendo de N sería en tiempo finito o infinito.

Un índice de control óptimo de seguimiento es:

$$J = \frac{1}{2}\sum_{k=0}^{N}\left[\varepsilon^T(k)Q\varepsilon(k) + u^T(k)Ru(k)\right] \qquad (9)$$

siendo el error: $\varepsilon(k) = x(k) - X_{REF}(k)$ En este caso además de minimizar la energía de entrada a la planta se estaría minimizando el cuadrado ponderado del vector error, para evitar que los errores positivos y negativos se anularan entre sí.

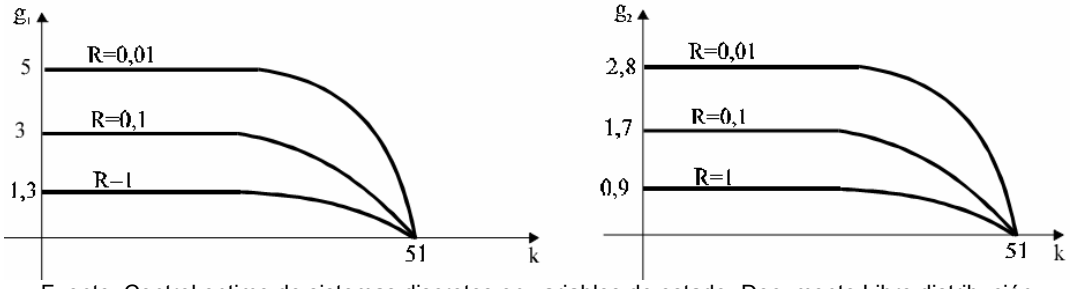
En general en todos los índices de desempeño los coeficientes de las matrices **Q** y **R** inciden directamente sobre los valores de las señales del sistema **x**(k) y **u**(k).

Por ejemplo, para un índice de desempeño dado, como el siguiente:

$$J = \frac{1}{2}\sum_{k=0}^{51}\left[x^T(k)Qx(k) + u^T(k)Ru(k)\right] \qquad (10)$$

Como se verá mas adelante, la matriz de ganancia de control óptimo **G** depende de la matriz **R.** Luego entonces, se podrían obtener diferentes ganancias de control óptimo ($\mathbf{G}_0$) según se hiciese **R** mayor o menor.

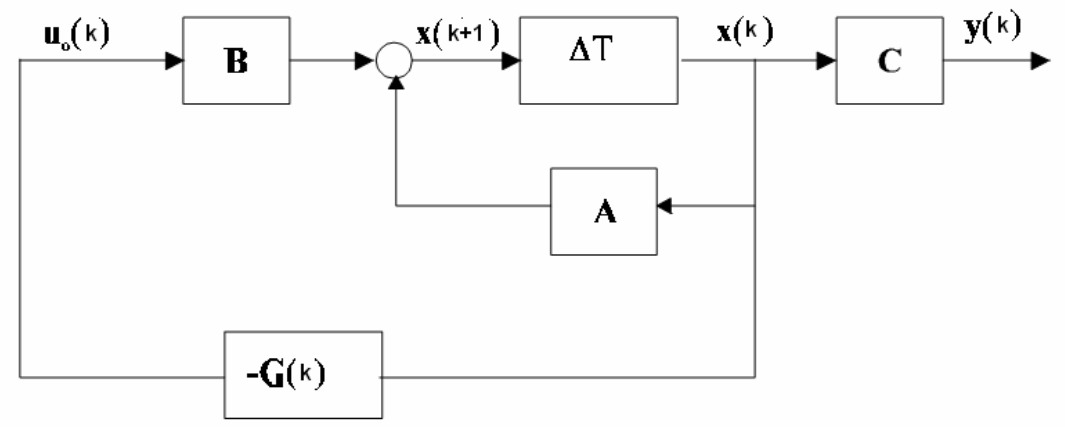
Así, si tuviéramos **G** = [g_1 g_2] para diferentes valores de la matriz **R** se podría obtener:



Fuente: Control optimo de sistemas discretos en variables de estado- Documento Libre distribución

Figura 2 Ganancias en función del coeficiente R del índice de desempeño

y como la matriz de ganancia de control óptimo va realimentada el sistemas en espacio de estado de la siguiente forma:



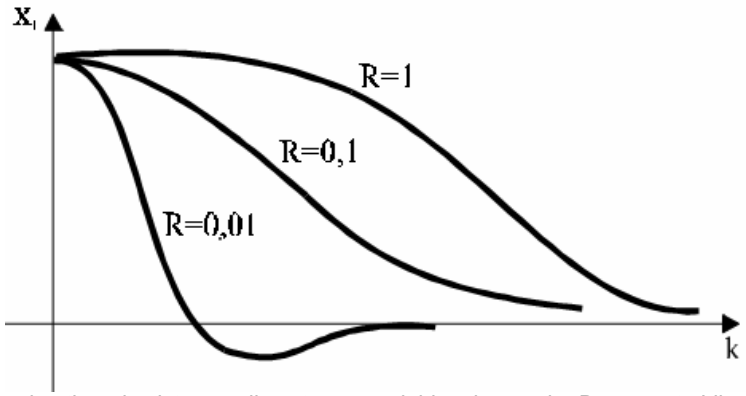
Fuente: Control optimo de sistemas discretos en variables de estado- Documento Libre distribución

Figura 3 sistema de control óptimo

Entonces la señal de control es:

$$u_o(k) = -g_1 x_1(k) - g_2 x_2(k) \tag{11}$$

cuanto mayores sean g_1 y g_2 mayor será la señal de control y con ella el coste energético. Por lo tanto si la matriz **R** aumenta, las ganancias disminuyen y también disminuye el coste energético de control. Pero existe un compromiso ya que con valores grandes de la matriz **R** empeora el comportamiento del sistema:



Fuente: Control optimo de sistemas discretos en variables de estado- Documento Libre distribución

Figura 4 Comportamiento de una variable de estado en función del coeficiente **R**

Este fenómeno era de esperar ya en la expresión del índice de desempeño:

$$J = \frac{1}{2}\sum_{k=0}^{N}\left[x^T(k)Qx(k) + u^T(k)Ru(k)\right] \tag{12}$$

si **R** >> **Q** la contribución del control **u**(k) es muy grande comparándola con la contribución del estado **x**(k). Dicho de otra forma, disminuye el coste del control pero empeora el desempeño del sistema.

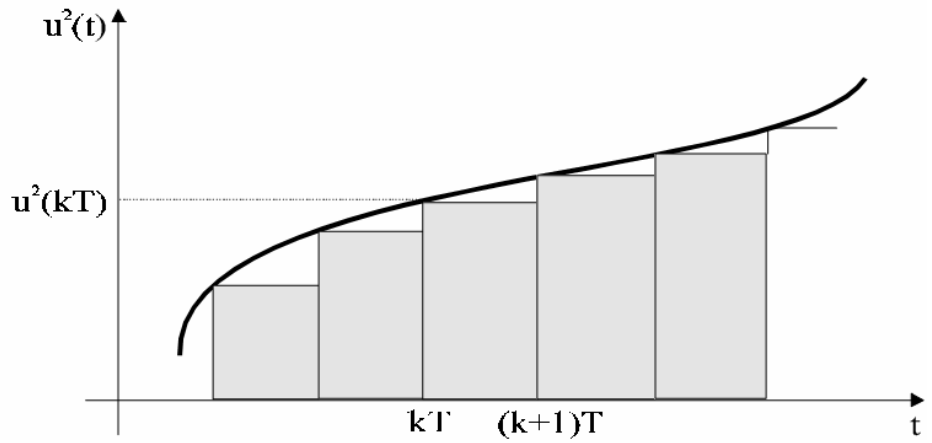
Si **Q** >> **R** mejora la respuesta del sistema pero aumenta el coste energético de control. El vector de estado estará muy controlado, pero las señales de control **u**(k) pueden ser muy grandes.

En general cuanto mayor es sean los coeficientes, más se optimiza la variable que corresponda.

El periodo de muestreo es muy importante en los problemas de control óptimo. En general, el índice de desempeño o coste aumenta cuadráticamente o disminuye según aumenta o disminuye el periodo de muestreo, por lo que existe un límite para el que en la práctica no merece la pena reducir más el periodo de muestreo.

1.2 Discretización de Índice de desempeño continúo

Para discretizar un índice de comportamiento continuo se suele aproximar la integral al área que se indica en la figura 5:



Fuente: Control optimo de sistemas discretos en variables de estado- Documento Libre distribución

Figura 5 Aproximación de una integral

Así por ejemplo, las integrales del índice de comportamiento:

$$J=\frac{1}{2}\int_0^5\left[x^2(t)+u^2(t)\right]dt=\frac{1}{2}\int_0^T\left[x^2(t)+u^2(t)\right]dt+\frac{1}{2}\int_T^{2T}\left[x^2(t)+u^2(t)\right]dt+...$$

$$...+\frac{1}{2}\int_{4T}^{5T}\left[x^2(t)+u^2(t)\right]dt \qquad (13)$$

Donde $x^2(t)+u^2(t)$ representa la función en el tiempo dada para un índice de desempeño a discretizar.

se aproximarían, en un intervalo genérico por:

$$J=\frac{1}{2}\int_{kT}^{(k+1)t}\left[x^2(t)+u^2(t)\right]dt\approx\frac{1}{2}\left[x^2(kT)+u^2(kT)\right]T \qquad (14)$$

y suponiendo que el periodo de muestreo está normalizado (T=1), resultaría:

$$J_M=\frac{1}{2}\sum_{k=0}^{4}\left[x^2(k)+u^2(k)\right] \qquad (15)$$

1.3 Trayectoria de control óptima en tiempo finito

La acción de control está limitada en el tiempo, entre el instante inicial k=0 y el final k=N.
La ecuación dinámica de un sistema discreto es:

$$x(k+1)=Ax(k)+Bu(k)$$

y el índice de comportamiento a minimizar:

$$J=\frac{1}{2}x^T(N)Sx(N)+\frac{1}{2}\sum_{k=0}^{N-1}\left[x^T(k)Qx(k)+u^T(k)Ru(k)\right] \qquad (16)$$

es decir, se trata de que **x**(k) alcance un valor mínimo pero empleando al mismo tiempo una señal de entrada **u**(k) mínima. Es decir, que tiendan a cero y por lo tanto es un problema de regulación. Para conseguirlo es necesario introducir una secuencia de control óptima $\mathbf{u}_o$ (k), que deseamos calcular.

Análogamente al caso continuo, se aplica el principio de optimalidad de Bellman, es decir se va obteniendo el vector de control óptimo entre dos muestreos consecutivos del sistema. El índice de comportamiento entre los muestreos N-1 y N sería:

$$J_{N-1,N}=\frac{1}{2}x^T(N)Sx(N)+\frac{1}{2}x^T(N-1)Qx(N-1)+\frac{1}{2}u^T(N-1)Ru(N-1)$$

$$=\frac{1}{2}\left[Ax(N-1)+Bu(N-1)\right]^T S\left[Ax(N-1)+Bu(N-1)\right]+$$

$$+\frac{1}{2}x^T(N-1)Qx(N-1)+\frac{1}{2}u^T(N-1)Ru(N-1) \qquad (17)$$

$$=\frac{1}{2}\left[Ax(N-1)\right]^T SAx(N-1)+\frac{1}{2}\left[Ax(N-1)\right]^T SBu(N-1)+$$
$$+\frac{1}{2}\left[Bu(N-1)\right]^T SAx(N-1)+\frac{1}{2}\left[Bu(N-1)\right]+\frac{1}{2}x^T(N-1)Qx(N-1)+$$
$$+\frac{1}{2}u^T(N-1)Ru(N-1) \qquad (18)$$

La condición necesaria para que J sea mínimo es:

$$\frac{\partial J_{N-1,N}}{\partial u(N-1)}=0 \qquad (19)$$

es decir:

$$\frac{\partial J_{N-1,N}}{\partial u_1(N-1)}=0 \qquad (21)$$

$$\frac{\partial J_{N-1,N}}{\partial u_2(N-1)}=0 \qquad (22)$$

$$\cdots\cdots\cdots\cdots$$

$$\frac{\partial J_{N-1,N}}{\partial u_m(N-1)}=0 \qquad (23)$$

$$\frac{\partial}{\partial u}(u^TA)=A \qquad \frac{\partial}{\partial u}(Au)=A^T \qquad \frac{\partial}{\partial u}(u^TRu)=Ru+R^Tu$$

teniendo en cuenta que las matrices **R** y **S** son simétricas, se obtiene:

$$\frac{\partial J_{N-1,N}}{\partial u(N-1)}=0=\frac{1}{2}\left[BSAx(N-1)+B^TSAx(N-1)+2B^TSBu(N-1)+2Ru(N-1)\right] \qquad (24)$$

De donde:

$$Ru_O(N-1)+B^TS\left[Ax(N-1)+Bu_O(N-1)\right]=0$$
$$u_O(N-1)=-\left[R+B^TSB\right]^{-1}B^TSAx(N-1)$$

Se comprueba fácilmente que se trata de un mínimo ya que:

$$\frac{\partial J_{N-1,N}}{\partial u(N-1)} = R + B^T SB > 0$$

por ser las matrices **R** y **S** definidas positivas (ver anexo A para saber qué es una matriz definida positiva).

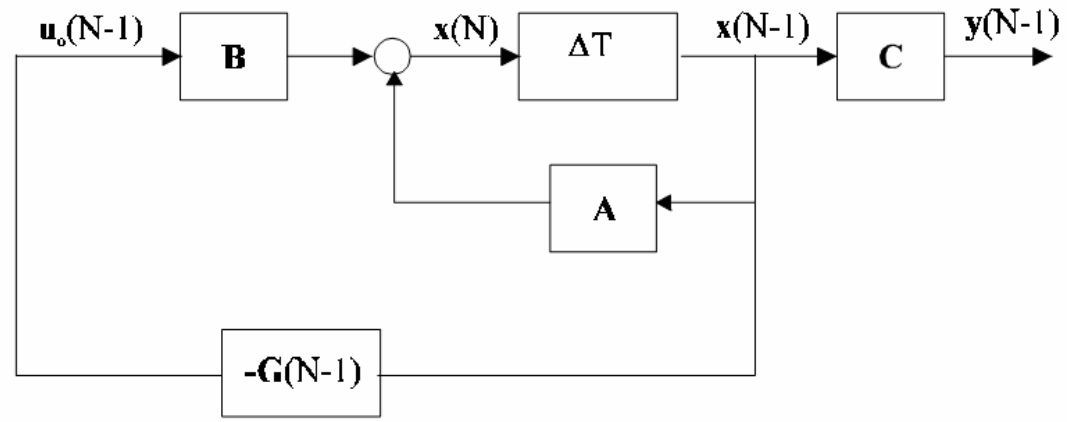
Llamando **S** = **M**(0) para realizar un algoritmo, se puede poner:

$$u_O(N-1) = -\left[R + B^T M(0)B\right]^{-1} B^T M(0)Ax(N-1) = -G(N-1)x(N-1) \quad (25)$$

Siendo:

$$G(N-1) = \left[R + B^T M(0)B\right]^{-1} B^T M(0)A \quad (26)$$

que puede interpretarse como una realimentación del estado del sistema con una ganancia variable en cada instante de muestreo de valor **G**(N-1):



Fuente: Control optimo de sistemas discretos en variables de estado- Documento Libre distribución

Figura 6 Regulación óptima entre los instantes N-1 y N

Es muy importante notar que el problema del regulador óptimo se reduce finalmente a un problema de regulación normal, con la diferencia de que la matriz **G** depende del tiempo k, es decir, es una matriz dinámica que cambia con el tiempo.

Sustituyendo el valor de $\mathbf{u}_o$(N-1) puede obtenerse el índice de comportamiento óptimo entre los instantes de muestreo N-1 y N:

$$J_{N-1,N}\,|_0 = \frac{1}{2}\left[Ax(N-1)\right]^T M(0)Ax(N-1) - \frac{1}{2}\left[Ax(N-1)\right]^T M(0)BG(N-1)x(N-1) -$$

$$-\frac{1}{2}\left[BG(N-1)\right] + \frac{1}{2}\left[BG(N-1)x(N-1)\right]^T M(0)BG(N-1) + \frac{1}{2}x^T(N-1)Qx(N-1) +$$

$$+\frac{1}{2}\left[G(N-1)x(N-1)\right]^T B \qquad (27)$$

$$= \frac{1}{2}x^T(N-1)M(1)x(N-1)$$

Con

$$M(1) = \left[A - BG(N-1)\right]^T M(0)\left[A - BG(N-1)\right] + G^T(N-1)RG(N-1) + Q \qquad (28)$$

donde ya puede apreciarse la forma recursiva de calcular la matriz **M**(N) conociendo el valor de **M**(0).

Aplicando el principio de optimalidad entre N-2 y N, las señales de control optimizarían a:

$$J_{N-2,N} = \frac{1}{2}x^T(N)M(0)x(N) + \frac{1}{2}\sum_{k=N-2}^{N-1}\left[x^T(k)Qx(k) + u^T(k)Ru(k)\right] =$$

$$= J_{N-1,N}\,|_0 + \frac{1}{2}x^T(N-2)Qx(N-2) + \frac{1}{2}u^T(N-2)Ru(N-2) \qquad (29)$$

y sustituyendo el índice óptimo en el intervalo N-1, N por la expresión:

$$= J_{N-1,N}\,|_0 = \frac{1}{2}x^T(N-1)M(1)x(N-1)$$

Y como:

$$x(N-1) = Ax(N-2) + Bu(N-2)$$

Resulta:

$$J_{N-2,N} = \frac{1}{2}x^T(N-2)Qx(N-2) + \frac{1}{2}u^T(N-2)Ru(N-2) +$$

$$+\frac{1}{2}\left[Ax(N-2) + Bu(N-2)\right]^T M(1)\left[Ax(N-2) + Bu(N-2)\right] \qquad (30)$$

El mínimo se obtendría de :

$$\frac{\partial J_{N-2,N}}{\partial u(N-2)} = 0$$

Que conduce a

$$u_O(N-2) = -\left[R + B^T M(1) B\right]^{-1} B^T M(1) A x(N-2) = -G(N-2) x(N-2) \qquad (31)$$

Sustituyendo esta expresión en el índice de desempeño se obtiene:

$$= J_{N-2,N} \mid_0 = \frac{1}{2} x^T (N-2) M(2) x(N-2)$$

Con

$$M(2) = \left[A - BG(N-2)\right]^T M(1) \left[A - BG(N-2)\right] + G^T(N-2) R G(N-2) + Q \qquad (32)$$

Repitiendo este proceso se irían obteniendo las variables de control en los diferentes instantes de muestreo, que en general resultan ser:

$$u_O(N-j) = -G(N-j) x(N-j) \qquad j = 1,2,..., N \qquad (33)$$

Siendo las matrices de ganancia:

$$G(N-j) = \left[R + B^T M(j-1) B\right]^{-1} B^T M(j-1) A \qquad (34)$$

y las matrices **M**(j) se obtienen a partir de **M**(0) = **S** según la ecuación de Riccati:

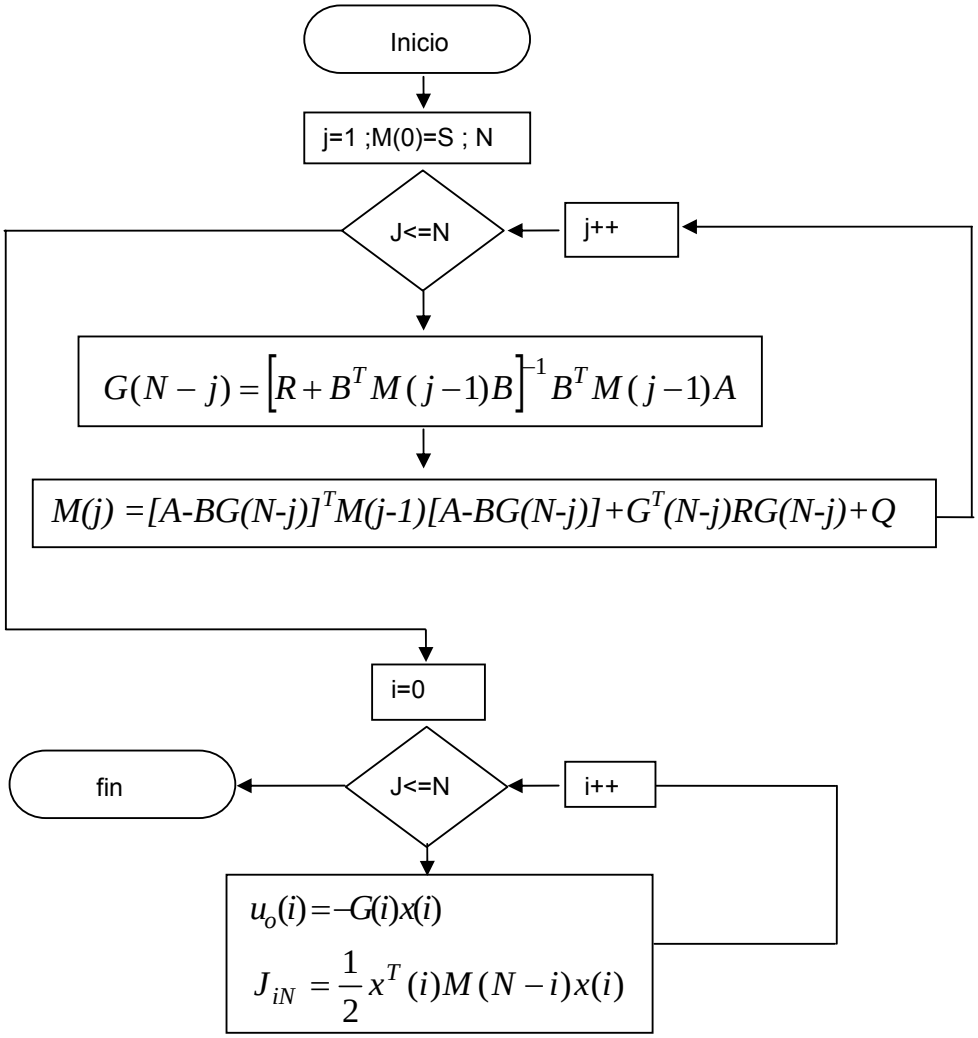
$$M(j) = \left[A - BG(N-j)\right]^T M(j-1) \left[A - BG(N-j)\right] + G^T(N-j) R G(N-j) + Q \qquad (35)$$

Con estas expresiones estamos optimizando el índice desde el momento inicial hasta el final, es decir se minimiza J durante el régimen transitorio y el régimen permanente.

El índice óptimo de comportamiento es:

$$J_{N-j,N} \mid_0 = \frac{1}{2} x^T (N-j) M(j) x(N-j)$$

El organigrama para implementar el algoritmo sería:



Supuesto N=5 en el primer ciclo for se calculan las ganancias **G** y las matrices **M** en este orden:

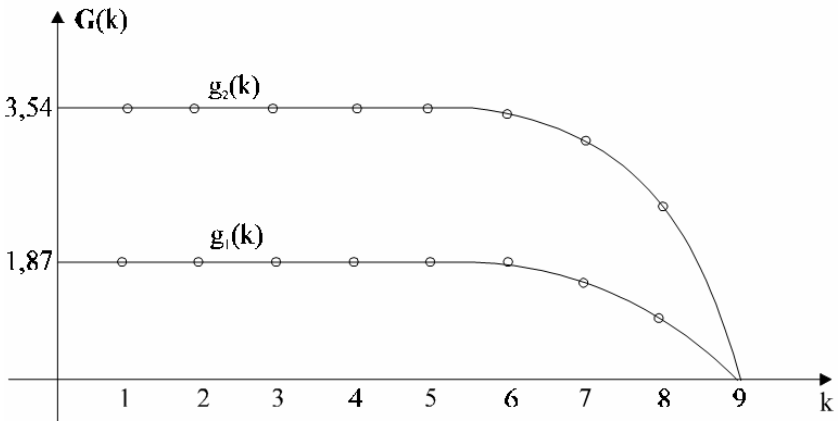
$$\begin{array}{ll} \mathbf{G}_4 & \mathbf{M}_1 \\ \mathbf{G}_3 & \mathbf{M}_2 \\ \mathbf{G}_2 & \mathbf{M}_3 \\ \mathbf{G}_1 & \mathbf{M}_4 \\ \mathbf{G}_0 & \mathbf{M}_5 \end{array}$$

es decir, integrando la ecuación de Riccati de atrás hacia delante, lo que es típico en estas ecuaciones.

En el segundo ciclo for se calculan las entradas óptimas en el orden en el que se van a ir aplicando y, si se considera necesario, el valor del índice de comportamiento mínimo en cada periodo de muestreo. El orden sería:

$$\mathbf{u}(0) \qquad J_{0,5}$$
$$\mathbf{u}(1) \qquad J_{1,5}$$
$$\mathbf{u}(2) \qquad J_{2,5}$$
$$\mathbf{u}(3) \qquad J_{3,5}$$
$$\mathbf{u}(4) \qquad J_{4,5}$$

Supongamos que las soluciones encontradas después de resolver la ecuación de Riccati sean de la forma de la figura 6:



Fuente: Control optimo de sistemas discretos en variables de estado- Documento Libre distribución

Figura 7

en el instante de muestreo inicial k=0 habría que aplicar la matriz de regulación **G**(0)=[1,87 3,54], lo mismo que en los instantes de muestreo siguientes hasta k=5. Durante los instantes de muestreo 6, 7 y 8 la ganancia del regulador **G**(k) es variable con el tiempo, para valer finalmente cero en el instante k=9.

Con estos valores se habría minimizado el índice J en los nueve periodos de muestreo, incluyéndose tanto el régimen permanente como el régimen transitorio. La resolución de la ecuación de Riccati se produce del instante final al inicial por lo que para obtener la secuencia de control óptimo $\mathbf{u}_o(k)$ es necesario procesar todos los instantes de muestreo. Esta carga computacional, la mayoría de las veces, no es asumible, por lo que en la práctica se recurre a obtener la matriz **G** que optimiza al índice en régimen permanente, perdiéndose su dependencia del tiempo. Es lo que se conoce como regulador óptimo en tiempo infinito.

1.4 Regulador óptimo en tiempo infinito (régimen permanente)

Si el valor de N lo hacemos tender a infinito, el valor de **G** permanece constante durante todos los periodos de muestreo. La ecuación de Riccati converge hacia el valor que **G** tiene en el régimen permanente, en muy pocos periodos de muestreo. Por ello la solución que se adopta es la de calcular el valor **G** en régimen permanente y dejarla fija desde el principio.

Al estar en régimen permanente **M**(k)=**M**(k-1)=**M**(k-2)= y la ecuación de Riccati en régimen permanente resulta ser:

$$M = \left[A - BG\right]^T M \left[A - BG\right] + G^T RG + Q \tag{36}$$

en donde ya se han quitado las dependencias del tiempo. Por otra parte:

$$G = \left[R + B^T MB\right]^{-1} B^T MA \tag{37}$$

y hay que tener en cuenta que las matrices $\left[R + B^T MB\right]$ y M son simétricas y por tanto iguales a sus transpuestas.

Sustituyendo el valor de **G** en la ecuación de Riccati:

$$M = \left[A^T - A^T MB(R + B^T MB)^{-1} B^T\right] M \left[A - B(R + B^T MB)^{-1} B^T MA\right] +$$
$$+ A^T MB(R + B^T MB)^{-1} R(R + B^T MB)^{-1} B^T MA + Q$$

$$= A^T MA - A^T MB(R + B^T MB)^{-1}(2B^T MA - B^T MA) + Q$$
$$= A^T MA - A^T MB(R + B^T MB)^{-1} B^T MA + Q$$

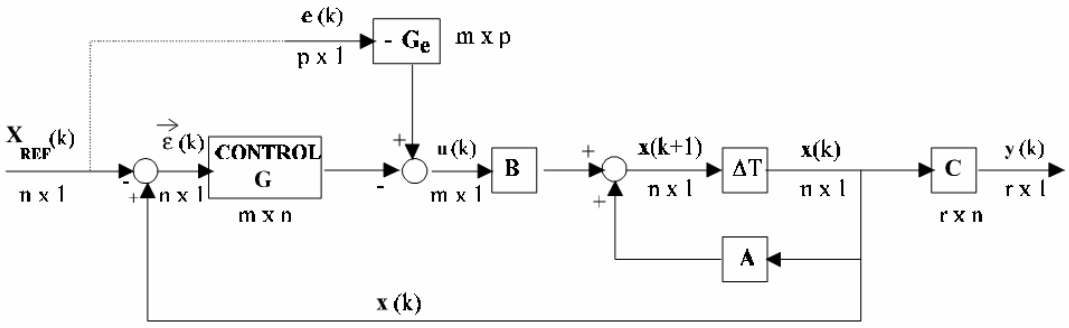
La ecuación de Riccati en régimen permanente resulta ser:

$$M - A^T MA + A^T MB(R + B^T MB)^{-1} B^T MA - Q = 0 \tag{38}$$

La solución de esta ecuación, así como el valor de **G** lo proporciona Matlab con la orden dlqr.

1.5 Control óptimo con señales de referencia

Exactamente igual que se realizó el control por medio de reubicación de polos puede efectuarse el control óptimo.



Fuente: Control optimo de sistemas discretos en variables de estado- Documento Libre distribución

Figura 8 Control óptimo con señal de referencia

La matriz **G** es igual a la encontrada en el caso del regulador óptimo, con la diferencia que ahora la entrada al controlador es **ε**(k) y por lo tanto el índice que se está minimizando es:

$$J = \frac{1}{2}\sum_{k=0}^{\infty}\left[a^T(k)Qa(k) + u^T(k)Ru(k)\right] \tag{39}$$

que es justamente el que interesa en los problemas de seguimiento.

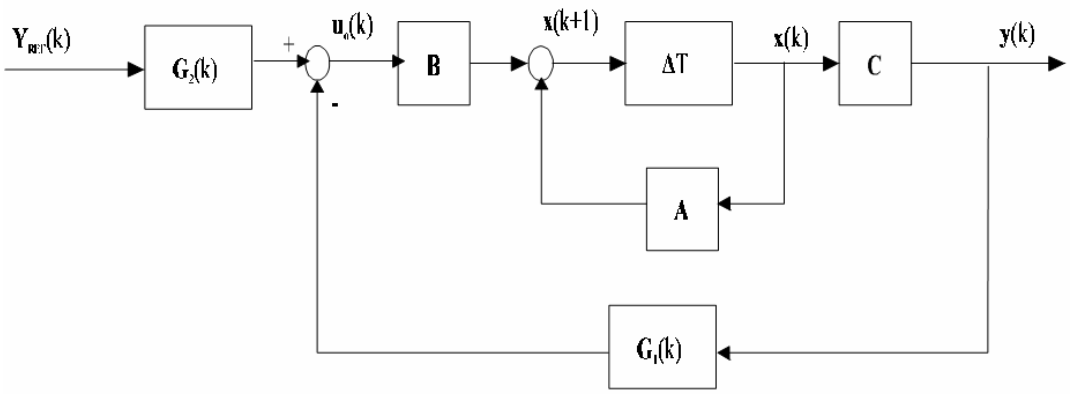
Respecto al error en régimen permanente puede anularse, muchas veces, con la matriz **Ge** que al igual que en el caso de reubicación de autovalores se obtiene de:

$$\left[I-(A-BG)\right]^{-1}(BG_e - A + A_{REF})X_{REF}(k) = 0 \tag{40}$$

La resolución de esta ecuación es posible con Matlab. Para ello:

$$\left[I-(A-BG)\right]^{-1}BG_eX_{REF}(k) = \left[I-(A-BG)\right]^{-1}(A-A_{REF})X_{REF}(k)$$
$$G_eX_{REF}(k) = M = \left[I-(A-BG)\right]^{-1}B/\left[I-(A-BG)\right]^{-1}(A-A_{REF})X_{REF}(k)$$
$$G_e = M / X_{REF}(k) \tag{41}$$

En el caso de un control realimentando la salida que se ajustase al esquema de la figura:



Fuente: Control optimo de sistemas discretos en variables de estado- Documento Libre distribución

Figura 9 Control óptimo de salida con set-point

$G_1(k)$ sería la solución de ganancia de control óptimo sin señal de referencia y:

$$G_2(k) = \left\{C\left[I - A + BG_1(k)\right]^{-1} B\right\}^{-1} \tag{42}$$

Es evidente que esta solución es válida solamente para cuando $\mathbf{G}_2$ es cuadrada. La matriz $\mathbf{G}_2$ minimiza el error en régimen permanente y $\mathbf{G}_1$ minimiza y(k).

1.6 Aspectos prácticos de implementación

La implementación del control óptimo suele ser off-line porque la resolución de n(n+1)/2 ecuaciones no lineales, no suele ser sencilla. La resolución, al no tener problemas de tiempo porque se resuelve off-line, puede hacerse analíticamente por métodos numéricos.

La segunda alternativa consistiría en integrar paso a paso la ecuación de Riccati, pero esta solución no es realista y para sistemas invariantes debe rechazarse. La solución más apropiada puede consistir en aplicar una ganancia de control constante, que corresponda a la solución en régimen permanente de la ecuación de Riccati. Esta alternativa es la de implementación más simple y está tanto más justificada cuanto más cortos sean los transitorios de los elementos de la solución de Riccati.

2. LA INTERFAZ GRAFICA DE USUARIO DEL MATLAB –GUI

MATLAB permite desarrollar fácilmente un conjunto de pantallas (paneles) con botones, menús, ventanas, etc., que permiten utilizar de manera muy simple programas realizados dentro de este entorno. Este conjunto de herramientas se denomina interfaz gráfica de usuario (GUI). Las posibilidades que ofrece MATLAB no son muy amplias, en comparación a otras aplicaciones de Windows como Visual Basic, Visual C. La elaboración de GUIs puede llevarse a cabo de dos formas, la primera de ellas consiste en escribir un programa que genere la GUI (script), la segunda opción consiste en utilizar la herramienta de diseño de GUIs, incluida en el Matlab, llamada GUIDE. En esta sección se abordarán ambas formas de crear GUIS. Para poder hacer programas que utilicen las capacidades gráficas avanzadas de MATLAB hay que conocer algunos conceptos que se explican en las siguientes secciones.

El panel GUI se crea en una ventana, identificada como figura y está formada por los siguientes elementos:

- Menú de interfaz con el usuario
- Dispositivos de control de la interfaz con el usuario
- Ejes para desplegar las gráficas o imágenes.

Mediante la GUI, el flujo de información está controlado por las acciones (eventos) que sucedan en la interfaz. Comparando con los scripts, en estos los comandos están en un orden preestablecido, mientras que en la GUI no lo están. Los comandos para crear una GUI se escriben en un script, pero una vez que se ejecuta la GUI, esta permanece en la pantalla aunque se haya terminado la ejecución del script. La interacción con el usuario continúa hasta que se cierra la GUI.

2.1 Objetos gráficos en MATLAB

El objeto más general es la pantalla o panel (screen). Dicho objeto es la raíz de todos los demás y sólo puede haber un objeto pantalla. Una pantalla puede contener una o más ventanas (figures). A su vez cada una de las ventanas puede tener uno o más ejes de coordenadas (axes) en los que se puede representar otros objetos de más bajo nivel. Una ventana puede tener también controles (uicontrols) tales como botones, barras de desplazamiento, botones de selección o de opción, etc.) y menús (uimenus). Finalmente, los ejes pueden contener los cinco tipos de elementos gráficos que permite MATLAB: líneas (line), polígonos (patch), superficies (surfacs), imágenes tipo bitmap (image) y texto (text). La jerarquía de objetos indica que en MATLAB hay objetos padres e hijos. Por ejemplo, todos los objetos ventana son hijos de pantalla, y cada ventana es padre de los objetos ejes, controles o menús que están por debajo.

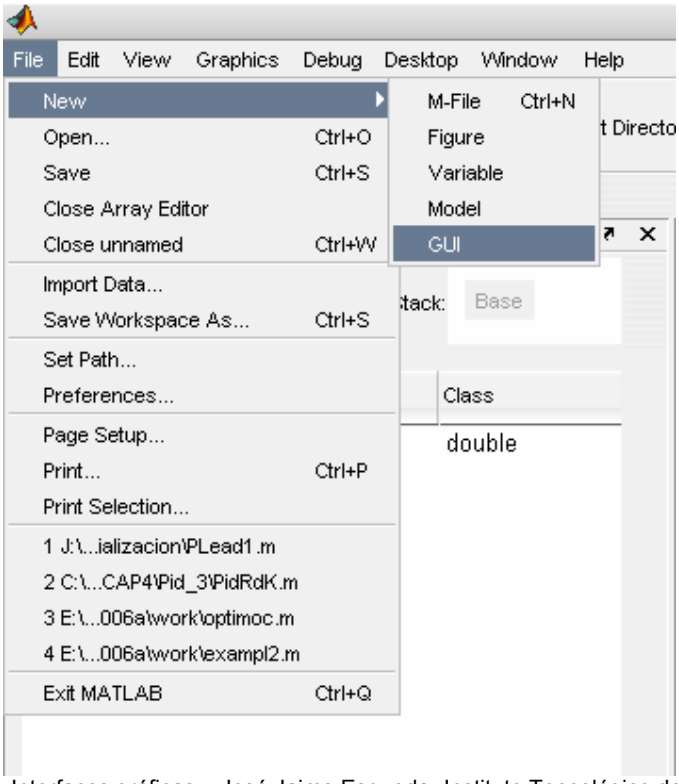
A su vez los elementos gráficos (líneas, polígonos, etc.) son hijos de un objeto ejes, y no tienen otros objetos que sean sus hijos. Cuando se borra un objeto de MATLAB automáticamente se borran todos los objetos que son sus descendientes. Por ejemplo, al borrar unos ejes, se borran todas las líneas y polígonos que son hijos suyos.

2.2 Utilizando GUIDE

A la herramienta GUIDE se accede de varias maneras, la primera de ellas es tecleando guide en la ventana de comando.

>> guide

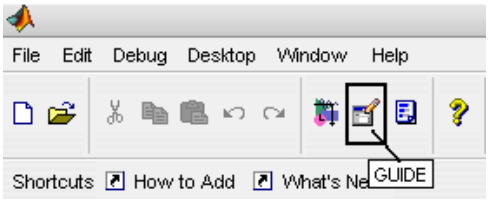
Otra manera de entrar a GUIDE es través del File opción New y por ultimo el GUI, (como se muestra en la figura 9).



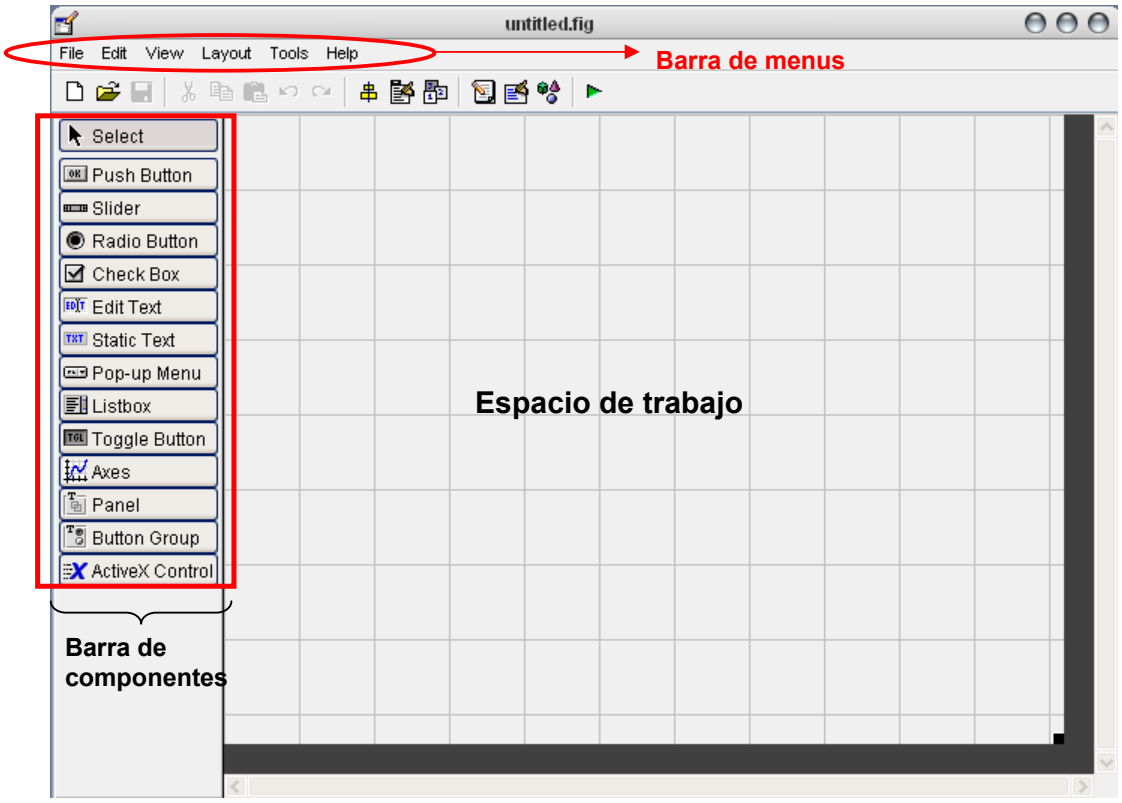
Fuente: Matlab e Interfaces gráficas – José Jaime Esqueda- Instituto Tecnológico de Ciudad Madero

Figura 10 Acceso al GUI

O presionando el botón de inicio del GUIDE



La ventana principal de GUIDE es la siguiente:



Fuente: Matlab e Interfaces gráficas – José Jaime Esqueda- Instituto Tecnológico de Ciudad Madero

Figura 11 Ventana principal del GUIDE

Las Componentes principales de GUIDE son:

Barra de Menús: Aquí se encuentran las funciones elementales de Edición de GUI's.

Paleta de Componentes (component Palette): Aquí se encuentras los uicontrols estos componentes permite seleccionar los controles (objetos) que son los que se muestra en la figura.10

La Barra de Herramienta: En ella se encuentran lo siguientes botones

Botón de ejecución (Run button): Al presionarse este botón se crea la figura de la interfaz diseñada en el área de diseño.

Alineación de Componentes (Alignment tool): esta opción permite alinear los componentes que se encuentra en el área de trabajo (Layout Área) de manera personalizada.

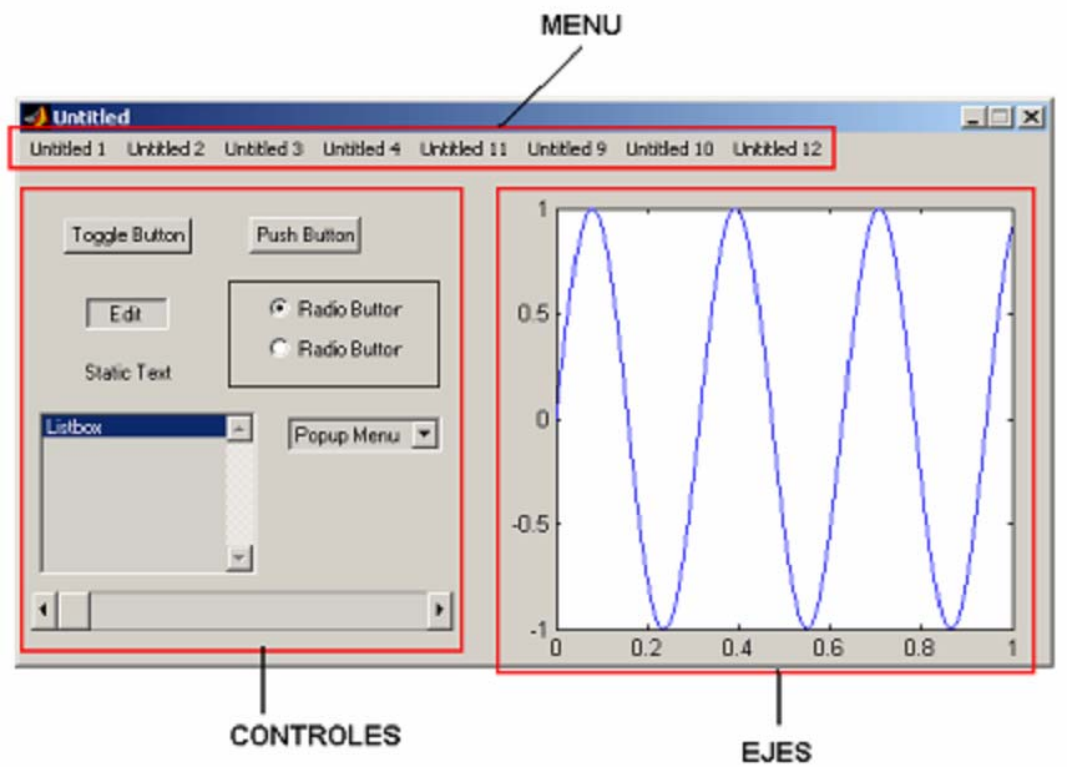
Propiedades del Inspector (Property Inspector): con esta opción se asignan y modifican las propiedades de cada objeto en forma personalizada.

Navegador de Objetos (Object Browser): Muestra todos los objetos que se encuentra en la figura (en forma de árbol) y a través de Object Browser se puede seleccionar los objetos.

Editor de Menús (Menú Editor): El redactor de Menú crea menús de ventana y menús de contexto.

La Interfaz de Grafica de Usuario (GUI) se crea en una ventana de figura que consta de los siguientes componentes:

1. Menú de interfaz con el usuario.
2. Dispositivo de control de interfaz con el usuario.
3. Ejes para exhibir graficas o imágenes



Fuente: Matlab e Interfaces gráficas – José Jaime Esqueda- Instituto Tecnológico de Ciudad Madero

Figura 12 componentes de una ventana GUI

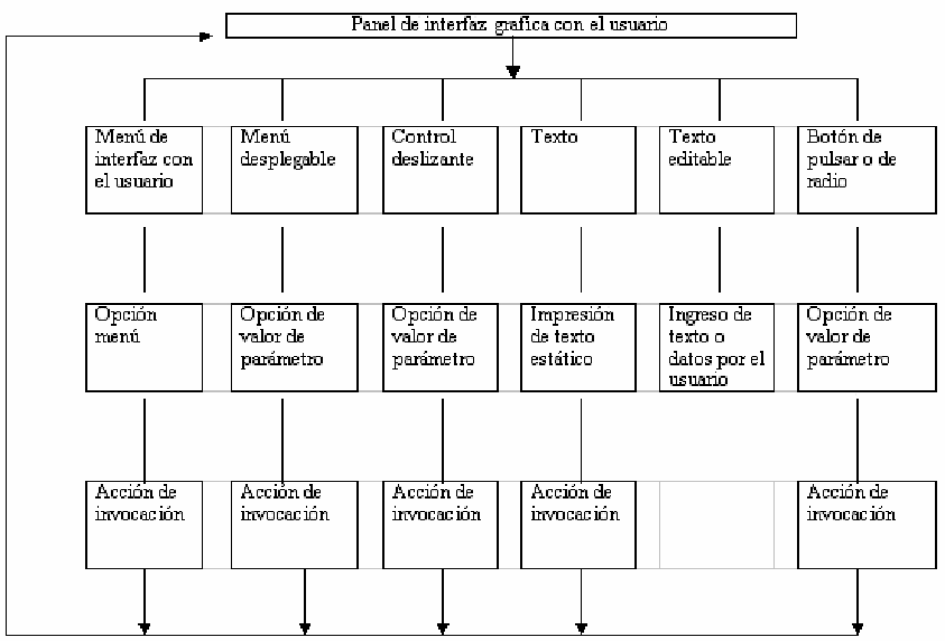
Puede personalizar el GUIDE en la opción hallada Preferences en el menú File, ahí es posible desplegar los nombres de los componentes hallados en la paleta y de presentar las herramientas.

2.3 Flujo de operación con GUI

Con una GUI, el flujo de computo esta controlado por las acciones en la interfaz. Mientras que en un guión el flujo de comandos esta predeterminado, el flujo de operaciones con una GUI no lo está. Los comandos para crear una interfaz con el usuario se escribe en un guión, la interfaz invoca el guión que se ejecute, mientras la interfaz del usuario permanece en la pantalla aunque no se haya completado la ejecución del guión.

Cuando se interactúa con un control, el programa registra el valor de esa opción y ejecuta los comandos prescritas en la cadena de invocación. Los menús de interfaz con el usuario, los botones, los menús desplegables, los controladores deslizantes y el texto editable son dispositivos que controlan las operaciones del software. Al completarse la ejecución de las instrucciones de la cadena de invocación, el control vuelve a la interfaz para que puedan elegirse otra opción del menú. Este ciclo se repite hasta que se cierra la GUI.

El control guarda un string que describe la acción a realizar cuando se invoca puede consistir en un solo comando de MATLAB o una secuencia de comandos, o en una llamada a una función. Es recomendable utilizar llamadas a funciones, sobre todo cuando se requieren de más de unos cuantos comandos en la invocación.



Fuente: Matlab e Interfaces gráficas – José Jaime Esqueda- Instituto Tecnológico de Ciudad Madero

Figura 13 Flujo de Operación con GUI

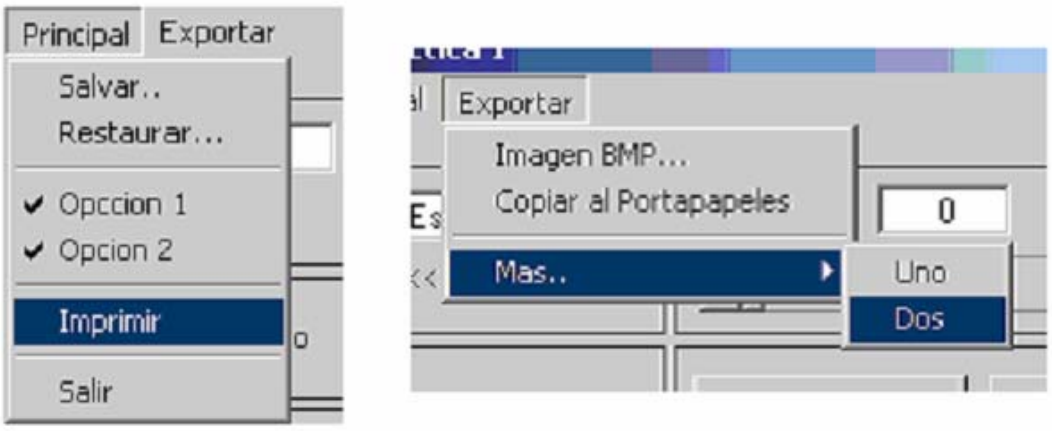
Básicamente solo se necesita entender cinco comandos para poder describir una GUI: uimenu, uicontrol, get, set y axes. No obstante, lo que hace

relativamente complicadas a estos comandos es el gran número de formas de uso que tiene. Es imposible describir todos los tipos de situaciones, pues requiere demasiado espacio y seria muy laborioso leerlo.

Por tanto, solo se tratará de explicar los elementos básicos de una GUI a través de ejemplos. Los lectores que deseen información mas detallada sobre los comandos consultar MATLAB: building a graphical user interface.

Menú de interfaz con el usuario

El menú de interfaz con el usuario es un menú o un grupo de menús que se encuentran en la parte superior de una ventana de la figura. El menú se desarrolla de arriba hacia abajo cuando se hace clic con el ratón y se muestra una lista de opciones siguiente figura. Cuando se elige una opción de la lista, es posible que se desenrolle otro nivel de menús (si el menú se diseño para ello).



Fuente: Matlab e Interfaces gráficas – José Jaime Esqueda- Instituto Tecnológico de Ciudad Madero

Figura 14 Ejemplos de menús de interfaz

Los menús de interfaz con el usuario se especifican con 'uimenu' cuya sintaxis es la siguiente:

```
m1 = uimenu (gcf, ...
          'Label', 'cadena de rótulos 1', ...
          'Position' [números de prioridad (entero)], ...
          'Backgroundcolor' [r, g, b], ...
          'CallBack' 'cadena de invocación')

m2 = uimenu (gcf, ...
          'Label', 'cadena de rótulos 2', ...
          'Position' [números de prioridad (entero)], ...
          'Backgroundcolor' [r, g, b], ...
          'CallBack' 'cadena de invocación')

m3 = uimenu (gcf, ...
```

```
'Label', 'cadena de rótulos 1', ...
'Position' [números de prioridad (entero)], ...
'Backgroundcolor' [r, g, b], ...
'CallBack' ' cadena de invocación')
```

Aquí se esta suponiendo que hay tres menús en una ventana de figura. En los comandos, m1 m2,… son mangos (get) de los menús que a menudo son necesarios en CallBack y otros comandos. Los argumentos que siguen a gcf son las propiedades del menú y tiene el siguiente significado:

a. `'Label', 'cadena de rótulos'` especifica el rotulo que aparecerá en el menú.
b. `'Position'`, **k** determina la posición secuencial del rotulo en el menú, donde k es un numero entero que indica en orden de prioridad.
c. `'Backgroundcolor'`, [r, g, b], especificas el color de segundo plano del menú.
d. `'CallBack' ' cadena de invocación'`, especifica los comandos que se ejecutan cuando se selecciona el rotulo.

En la sintaxis anterior es posible omitir las líneas correspondientes a position y backgroundcolor si son aceptables los valores por omisión. Además, callBack no es necesario cuando el menú va seguido de una lista de opciones que se abre cuando se hace clic en el menú.

También podemos especificar otras opciones; sin embargo, la mejor forma de aprenderlas es ejecutando el comando get (mango) después de ejecutarse el comando uimenu, donde mango deber ser como m1 o m2 de la explicación de la sintaxis anterior. El comando get (mango) devuelve las propiedades actuales del menú, que en su mayoría se establecen por omisión pero que pueden modificarse en los argumentos del enunciado del comando uimenu

Si una opción del menú tiene subopciones, estas también se especifican como uimenu Si consideramos la lista de opciones para el primer menú cuyo mango es m1, la sintaxis con tres opciones seria la siguiente:

```
m1sA = uimenu (m1, ...
           'Label', 'opcion A', ...
           'CallBack' ' cadena de invocación')

m1sB = uimenu (m1, ...
           'Label', 'opcion B', ...
           'CallBack' ' cadena de invocación')

m1sC = uimenu (m1, ...
           'Label', 'opcion C', ...
           'CallBack' ' cadena de invocación')
```

Las tres opciones pertenecen al primer menú con mango m1. se omitieron las propiedades de Position y BackgroundColor, pero pueden incluirse si se desea.

La propiedad CallBack es importante aquí si uicontrol es para el nivel terminal del menú. La cadena de invocación es una cadena que consiste en un comando, un conjunto de ordenes o una llamada de función. En esta cadena se especifican todas las tareas de cómputo que deben ejecutarse al elegirse la opción. Las instrucciones pueden ser un solo comando o varios; sin embargo, a fin de simplificar la programación es recomendable no escribir más de unos cuantos comandos o más de una llamada de función en la cadena de invocación. Todos los detalles de los cálculos se pueden escribir en un archivo M.

Nota: el archivo M puede o no ser de función. En cualquier caso, 'cadena de invocación' es remplazado por 'nombre del archivo M' la diferencia entre utilizar un archivo M de función y uno no de función es la siguiente. Si se emplea un archivo M de función, es necesario proporcionar como argumentos las variables necesarias para el archivo M de función, pues de lo contrario este no podría ver las variables del archivo M que invoca. Por otro lado, si se utiliza un archivo M no de función, todas las variables del archivo M invocado estarán visibles.

2.4 CONTROLES DE LA INTERFAZ DE GUIDE.

Los controles de la interfaz con el usuario en MATLAB se especifican con la orden uicontrol. Estos controles tienen mucho en común con los menús de la interfaz con el usuario, pero los primeros tienen muchos estilos. La sintaxis de uicontrol es

```
k = uicontrol ('Style', 'especificacoin de estilo', …
               'String', 'cadena para exhibir', …
               'Value', [valor], …
               'BackgroundColor', [r,g,b], …
               'Max' [valor], …
               'Min' [valor], …
               'Position', [izq, base, ancho, alto], …
               'CallBack', 'cadena de invocacion')
```

donde `'especificación de estilo'` es una de las siguientes cadenas:

```
popup
push
radio
checkbox
slider
edit
text
frame
```

Las propiedades de uicontrol son similares a las de uimenu las propiedades que aparecen aquí por primera vez son:

a. '`Value`', valor: especifica el valor por omisión de ajuste. En el caso de interruptores de encendido/apagado, valor es 0 o 1. En el caso de un

control deslizante (‘slider’), puede ser cualquier valor entre el mínimo y el máximo.

b. ‘Min’, Valor: establece el valor mínimo. Su significado difiere dependiendo del estilo.
c. ‘Max’, Valor: establece el valor máximo. Su significado difiere dependiendo del estilo.

Hay muchas más propiedades que pueden incluirse en los comandos de uicontrol, tal como sucede con las propiedades de uimenu, aunque al programar conviene minimizar el número de propiedades a fin de simplificar el guión. Si se desea saber mas acerca de las propiedades adicionales, investigue utilizando el comando get

Texto estático. Un static text puede exhibir símbolos, mensajes o incluso valores numéricos de una GUI, y puede colocarse en lugar deseado. El texto estático no tiene cadenas de invocación. A continuación mostramos un ejemplo de texto estático.

```
k1 = uicontrol ('Style', 'text', …
                'String', 'cadena para exhibir', …
                'Position', [20, 50, 140, 30])
```

El contenido de un texto exhibido puede modificarse si es necesario. Esto se hace con el comando set. Por ejemplo, ejecute el comando que sigue desde la ventana de comandos mientras está vigente el ejemplo anterior de orden uicontrol:

```
set (k1, 'string','Ahora aparece un texto modificado.')
```

Menú desplegable. Los pop-up menús difieren de los menús de interfaz con el usuario en que pueden aparecer en cualquier punto de la ventana de figura, mientras que los menús de interfaz con el usuario solo se localizan en la parte superior.

Los Push button generan una acción cuando das click con el puntero del ratón sobre ellos. Cuando usted da click en un push button, aparece presionado; Cuando sueltas el botón del ratón, el botón aparece levantado; y su rutina de llamada se ejecuta.

Casilla de verificación . Las casillas de verificación están diseñadas para realizar operaciones de encendido/apagado. La casilla activa o desactiva la aparición de los ejes. Las posiciones de encendido/apagado se registran en ‘Value’ que puede examinarse con get(handles,’Value’). Los comandos axis on y axis off se escriben en la cadena de invocación.

Botón de radio. Cuando solo se usa un botón de radio, no existe diferencia funcional alguna con respecto a una casilla de verificación. Por otro lado, los botones de radio en grupo son mutuamente exclusivos (es decir, si un botón esta encendido, todos los demás botones se apagan), mientras que las casillas de verificación son independientes entre sí. Sin embargo, esta características exclusiva de los botones de radio sólo puede implementarse mediante la programación del usuario en la cadena de invocación.

Barra deslizadora. Los sliders aceptan datos de entrada numéricos con un rango especifico. Los usuarios mueven la barra dejando presionado el botón del mouse y arrastrándola, Dando click en el canal, en la flecha. La posición de la barra indica un valor numérico.

Texto editable. El dispositivo de texto editable permite al usuario teclear una cadena de entrada. Se pueden escribir varios valores numéricos en forma de vector o matriz como cadena mediante el mismo dispositivo; esta cadena se convertirá posteriormente en valores numéricos con el comando str2num.

Un ejemplo de uicontrol para texto editable es:

```
ed1 = uicontrol (gcf, 'Style', 'edit', …
                 'Position', [10, 260, 110, 20], …
                 'CallBack', inp_txt = get (ed1, 'string'))
```

Las palabras clave en el comando anterior son ´Style´, ´edit´ y `get` (`mango`, ´string´) que capturan el texto introducido.

Marcos. El estilo marcos puede servir para agrupar dispositivos como lo botones de radio o las casillas de verificación.

Botón de palanca El toggle button genera una acción que indica un estado binario (on o off). Cuando das click en un toggle button , aparece presionado y permanece así hasta que sueltes el boton de el mouse, y en ese momento ejecuta la llamada. Un click del mouse subsecuente regresa al toggle button a su estado original y vuelve a ejecutar la rutina de llamada.

Cajas de lista. El componente List Box muestra una lista de artículos y permite a usuarios seleccionar unos o más artículos.

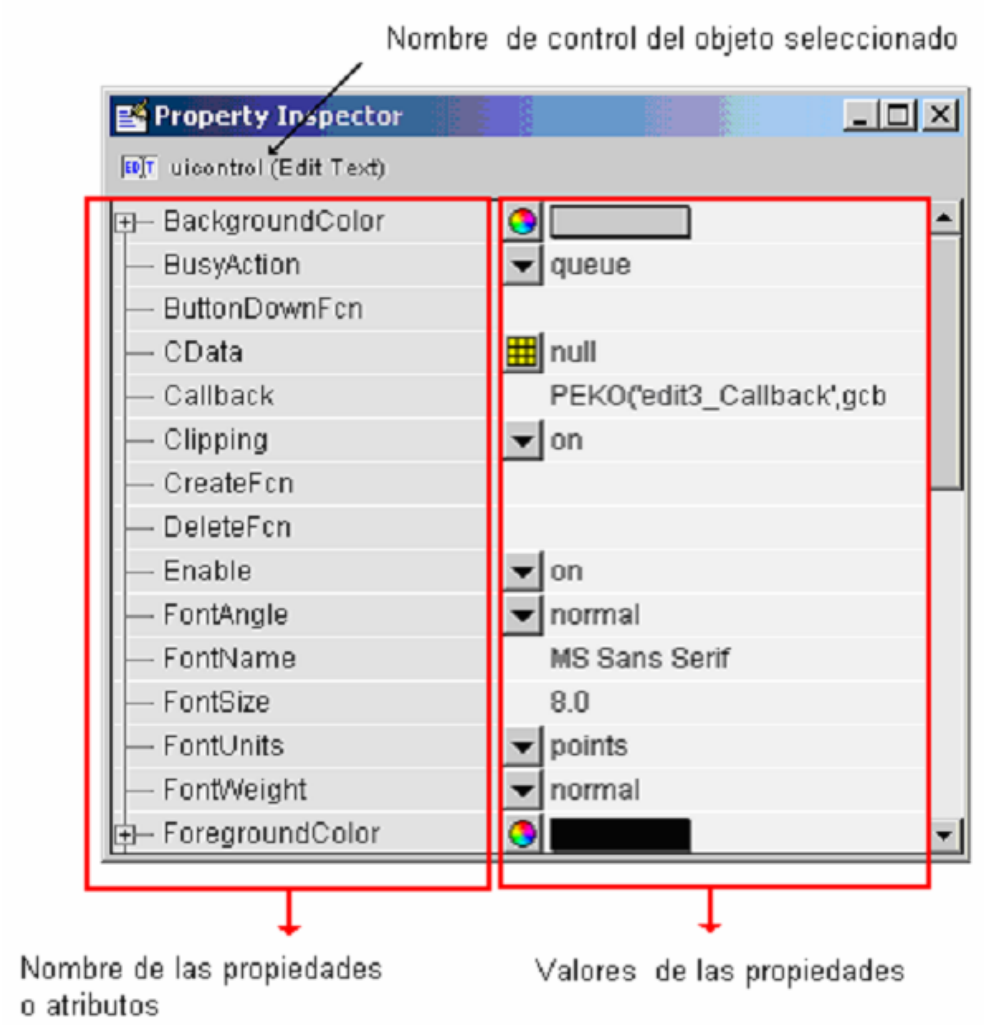
Ejes múltiples

Al crear una interfaz con el usuario, a menudo hay necesidad de trazar una o más gráficas dentro de la interfaz. Podemos usar el comando subplot para este fin, pero el comando axes es más flexible y ofrece opciones versátiles a los programadores.

El comando axes abre un eje en un punto especificado dentro de una ventana de figura. Aunque podemos abrir varios ejes en una ventana de figura con axes, primero consideramos únicamente uno.

Property Inspector

El inspector de propiedades esta compuesta de la siguiente forma como se muesta en la figura 14.



Fuente: Matlab e Interfaces gráficas – José Jaime Esqueda- Instituto Tecnológico de Ciudad Madero

Figura 15 Property inspector

3. DESARROLLO DE LA INTERFAZ GUI

Para desarrollar una aplicación en el que se aplique GUI, se diseñará un control óptimo cuadrático para un sistema de péndulo invertido, a partir del ejemplo propuesto en la sección 8-4, “control óptimo cuadrático de un sistema de seguimiento” del libro Sistemas de control en tiempo discreto de Katsuhiko Ogata (ver anexo B).

Una vez creado un GUI en blanco (de acuerdo a la sección 2 **LA INTERFAZ GRAFICA DE USUARIO DEL MATLAB –GUI**) se procede a colocar los distintos controles de interfaz que se van a utilizar y ordenarlos de tal manera que se organicen de la forma más amigable para el usuario y al gusto del diseñador como se muestra a continuación:

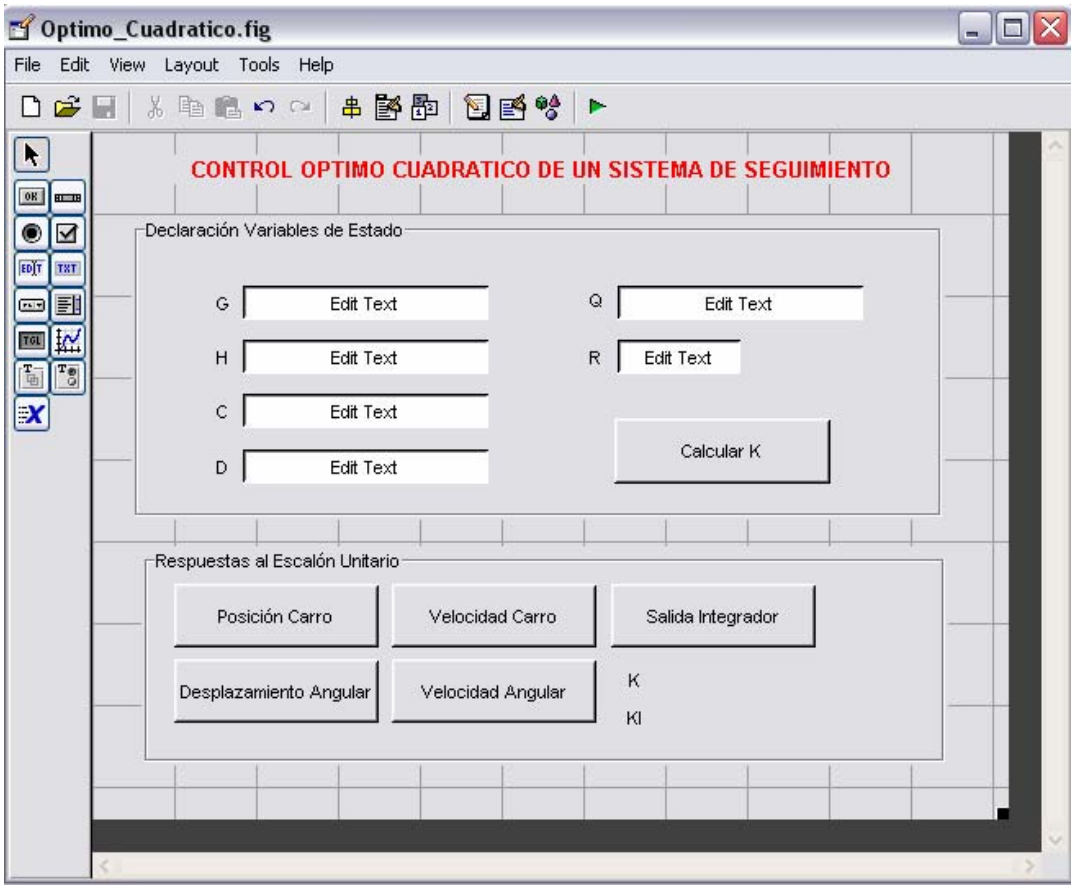


Figura 16 Diseño de la interfaz de control óptimo para un péndulo invertido

Ya desarrollada la interfaz final, se continua con la programación del botón que permitirá calcular las ganancias K y KI del control óptimo y del integrador respectivamente. Procedemos a dar clic derecho en el botón con la etiqueta

“Calcular K” y del menú que aparece seleccionamos “view callbacks” y de allí seleccionamos la opción “callback”, en donde colocaremos el código necesario para el calculo de las ganancias K y KI del modo que sigue:

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

G = str2num(get(handles.edit1,'String'));
H = str2num(get(handles.edit2,'String'));
C = str2num(get(handles.edit3,'String'));
D = str2num(get(handles.edit4,'String'));
Q = str2num(get(handles.edit5,'String'));
R = str2num(get(handles.edit6,'String'));

G1 = [G zeros(4,1); -C*G 1];
H1 = [H; -C*H];
% Se comienza a resolver la ecuacion de Riccati
P=diag(0,4);
P = Q+G1'*P*G1-G1'*P*H1*inv(R+H1'*P*H1)*H1'*P*G1;
W = Q+G1'*P*G1-G1'*P*H1*inv(R+H1'*P*H1)*H1'*P*G1;
%
while P~=W
    P = Q+G1'*P*G1-G1'*P*H1*inv(R+H1'*P*H1)*H1'*P*G1;
    W = Q+G1'*P*G1-G1'*P*H1*inv(R+H1'*P*H1)*H1'*P*G1;
    P = Q+G1'*W*G1-G1'*W*H1*inv(R+H1'*W*H1)*H1'*W*G1;
end
KK = inv(R+H1'*P*H1)*H1'*P*G1
K  = [KK(1) KK(2) KK(3) KK(4)];
KI = -KK(5);
k1=num2str(KK(1));
k2=num2str(KK(2));
k3=num2str(KK(3));
k4=num2str(KK(4));
ki=num2str(KI);
set(handles.text10,'String',k1);
set(handles.text11,'String',k2);
set(handles.text12,'String',k3);
set(handles.text13,'String',k4);
set(handles.text14,'String',ki);
```

Paso a seguir se programarán los botones para la respuesta al escalón unitario que se encuentran en el panel creado en la parte inferior de nuestra interfaz.

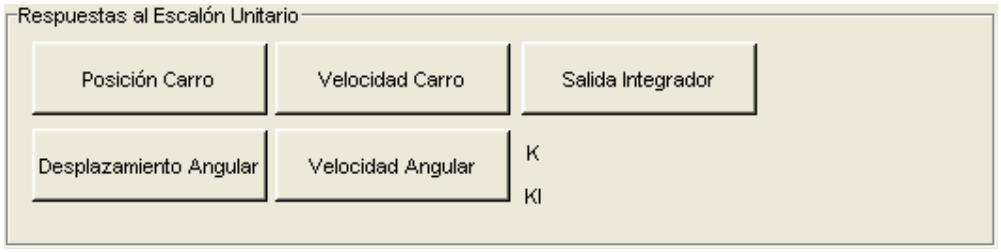


Figura 17 Botones para la respuesta al escalón unitario

Procedemos a dar clic derecho en el botón con la etiqueta “Posición Carro” y del menú que aparece seleccionamos “view callbacks” y de allí seleccionamos la opción “callback”, en donde colocaremos el código necesario para observar la respuesta de la variable de estado que me representa la posición del carro del modo que sigue:

```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global G H C D Q R K KI
figure(1)
GG = [G-H*K H*KI;-C*G+C*H*K 1-C*H*KI];
HH = [0;0;0;0;1];
CC = [0 0 1 0 0];
DD = [0];
[num,den] = ss2tf(GG,HH,CC,DD);
r = ones(1,101);
axis([0 100 -0.2 1.2]);
k = 0:100;
y = filter(num,den,r);
plot(k,y,'o',k,y,'-')
grid
title('Posición del Carro: y(k)= x3(k)')
xlabel('k')
ylabel('y(k)=x3(k)')
```

Siguiendo el mismo método con el que se programo el botón anterior se programan los botones restantes. Así, se tiene que el código para programar el botón “Velocidad carro” es:

```
% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global G H C D Q R K KI
figure(2)
GG = [G-H*K H*KI;-C*G+C*H*K 1-C*H*KI];
HH = [0;0;0;0;1];
LL = [0 0 0 1 0];
DD = [0];
[num,den] = ss2tf(GG,HH,LL,DD);
r = ones(1,101);
axis([0 100 -0.2 1.2]);
k = 0:100;
y = filter(num,den,r);
plot(k,y,'o',k,y,'r-')
grid
title('Velocidad del Carro: x4(k)')
xlabel('k')
ylabel('y(k)=x3(k)')
```

para el botón “Desplazamiento angular” el código será:

```
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global G H C D Q R K KI
figure(3)
GG = [G-H*K H*KI;-C*G+C*H*K 1-C*H*KI];
HH = [0;0;0;0;1];
FF = [1 0 0 0 0];
DD = [0];
[num,den] = ss2tf(GG,HH,FF,DD);
r = ones(1,101);
axis([0 100 -0.2 1.2]);
k = 0:100;
y = filter(num,den,r);
plot(k,y,'o',k,y,'r-')
grid
title('Desplazamiento Angular: x1(k)')
xlabel('k')
ylabel('x1(k)')
```

para el botón “Velocidad angular” tenemos el siguiente código:

```
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global G H C D Q R K KI
figure(4)
GG = [G-H*K H*KI;-C*G+C*H*K 1-C*H*KI];
HH = [0;0;0;0;1];
JJ = [0 1 0 0 0];
DD = [0];
[num,den] = ss2tf(GG,HH,JJ,DD);
r = ones(1,101);
axis([0 100 -0.2 1.2]);
k = 0:100;
y = filter(num,den,r);
plot(k,y,'o',k,y,'r-')
grid
title('Velocidad Angular: x2(k)')
xlabel('k')
ylabel('x2(k)')
```

y por último para el botón “Salida Integrador” será programado con el siguiente código:

```
% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global G H C D Q R K KI
figure(5)
GG = [G-H*K H*KI;-C*G+C*H*K 1-C*H*KI];
```

```
HH = [0;0;0;0;1];
MM = [0 0 0 1 0];
DD = [0];
[num,den] = ss2tf(GG,HH,MM,DD);
r = ones(1,101);
axis([0 100 -0.2 1.2]);
k = 0:100;
y = filter(num,den,r);
plot(k,y,'o',k,y,'r-')
grid
title('Salida del Integrador: x5(k)')
xlabel('k')
ylabel('x5(k)')
```

Una vez desarrollado la interfaz y la programación del programa, se procede a realizar una prueba con los datos existente en el ejemplo presentado en la sección 8-4 del libro Sistemas de control en tiempo discreto de Katsuhiko Ogata (anexo B)

G = 1.1048 0.1035 0 0; 2.1316 1.1048 0 0; -0.0025 -0.0001 1 0.1; -0.0508 -0.0025 0 1
H= -0.0051;-0.1035; 0.0025; 0.0501
C= 0 0 1 0
D= 0
Q = 10 0 0 0 0;0 1 0 0 0;0 0 100 0 0;0 0 1 0 0; 0 0 0 0 1 ; este matriz puede variar para mejorar la respuesta
R= 1

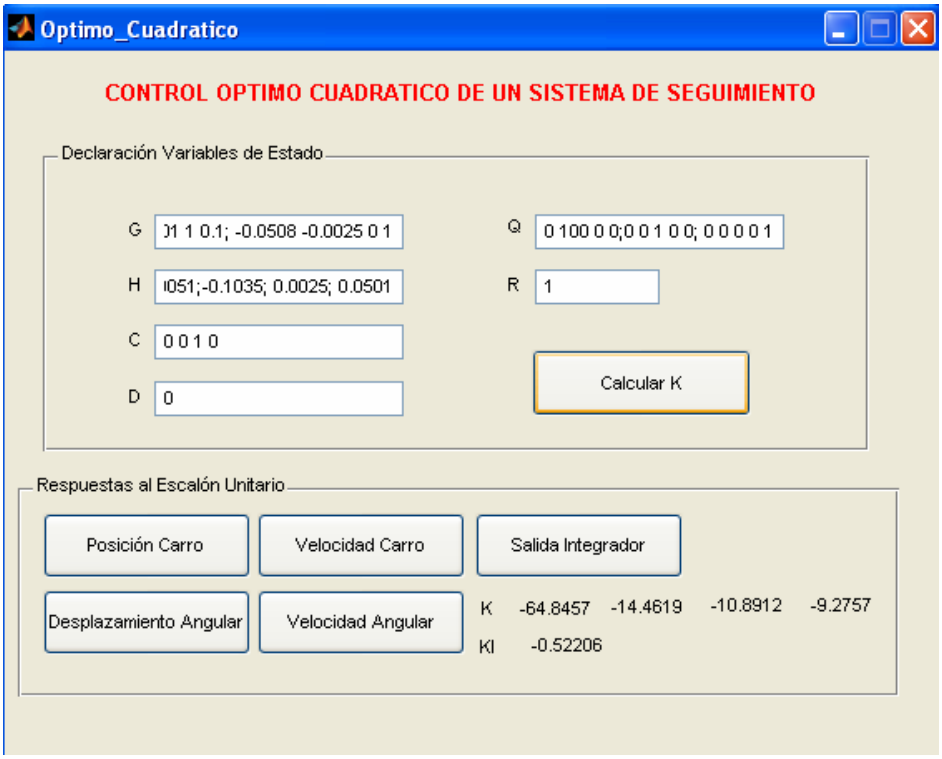


Figura 18 Pantalla principal del software desarrollado

Al ejecutar el programa diseñado en el modo Layout se introducen los datos teniendo en cuenta que la matriz Q es un valor de prueba, por lo tanto esta matriz puede variar de acuerdo al criterio deseado, es decir para llegar al valor deseado es necesario recurrir al método de prueba y error.

Puede notarse que al presionar el botón “Calcular K” se muestran los valores de K y KI para el ejemplo del péndulo invertido mencionado.

Para la matriz Q colocada en la prueba se tendrá la siguiente respuesta al escalón unitario al presionar el botón “Posición Carro”:

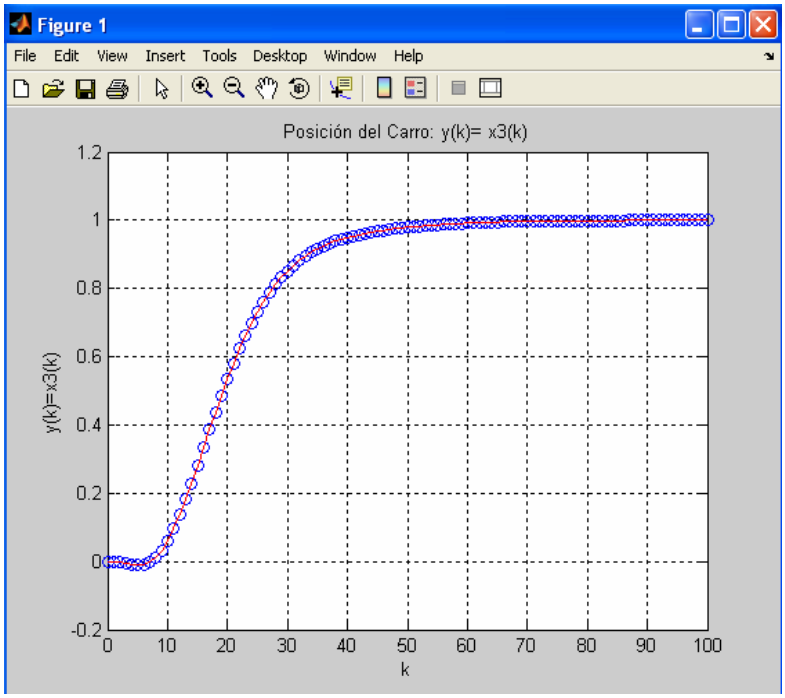


Figura 19 respuesta al escalón unitario para la variable de la posición del carro

Al presionar el botón para observar el comportamiento de la velocidad del carro se obtendrá una respuesta como la siguiente:

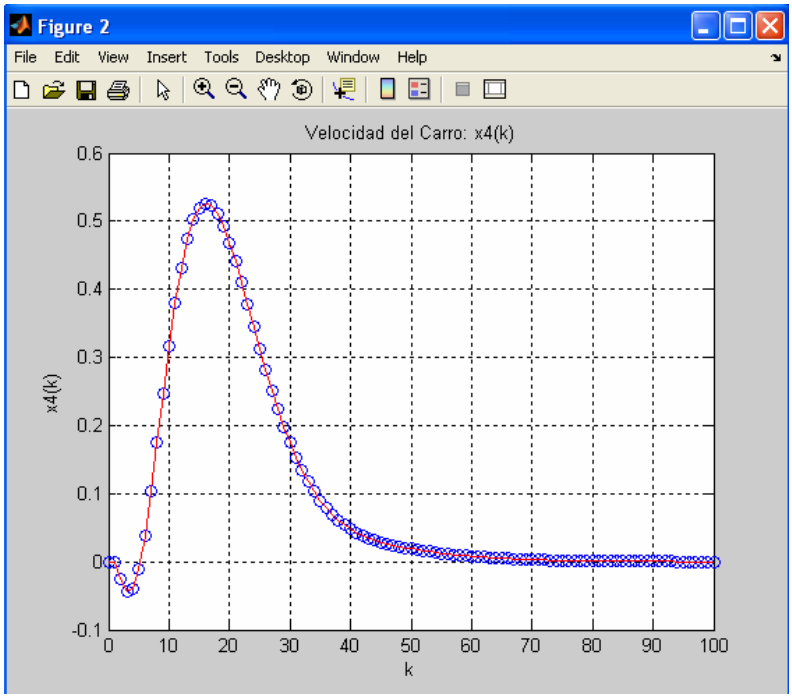


Figura 20 respuesta al escalón unitario para la variable de la velocidad del carro

Para obtener la respuesta gráfica para el desplazamiento angular, se presiona el botón “Desplazamiento Angular” con se obtiene:

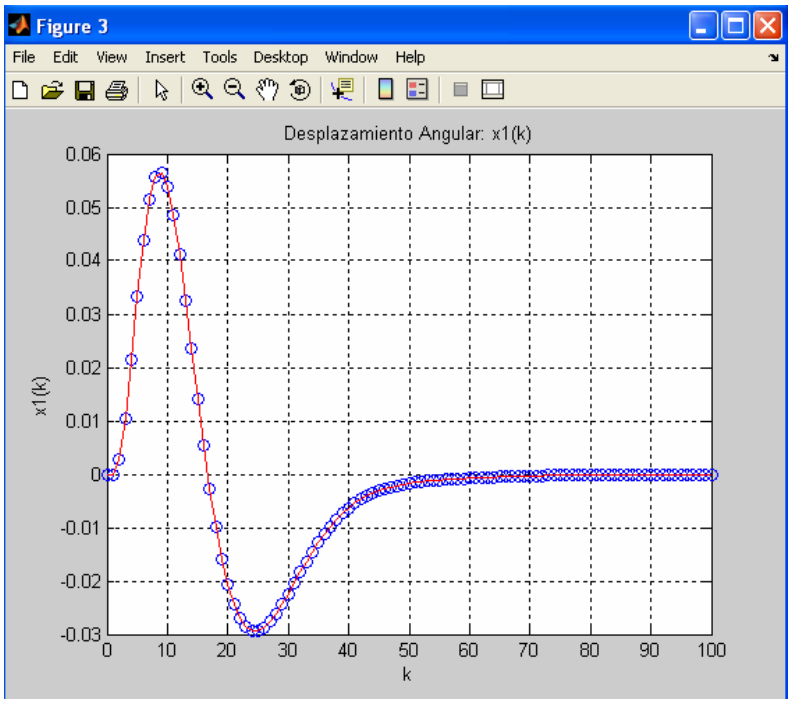


Figura 21 respuesta al escalón unitario para la variable del desplazamiento angular

El comportamiento de la velocidad angular se obtiene presionando el botón "Velocidad angular"

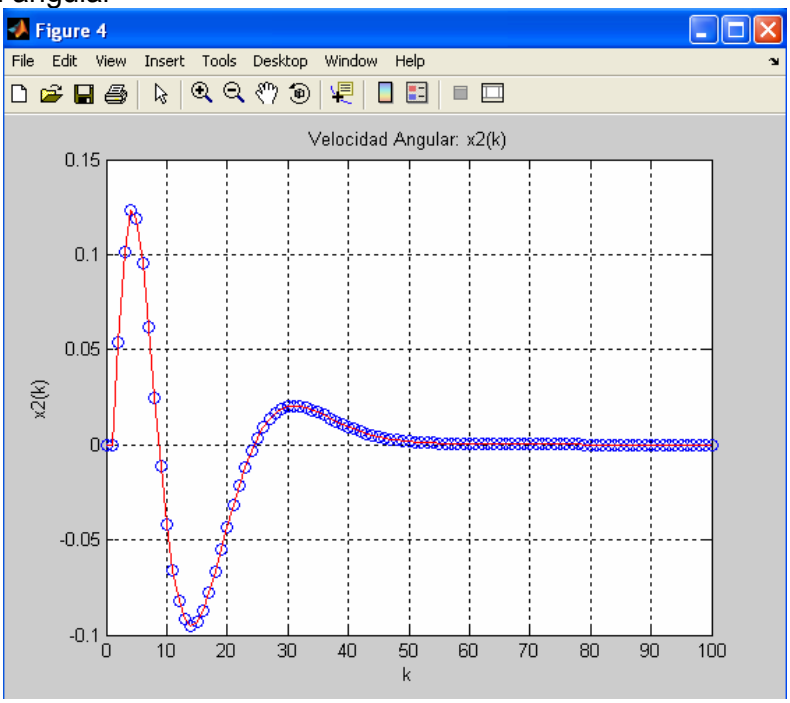


Figura 22 respuesta al escalón unitario para la variable de la velocidad angular

Y por último la salida del integrador se puede apreciar si se hace clic en el botón "Salida integrador" como se muestra a continuación:

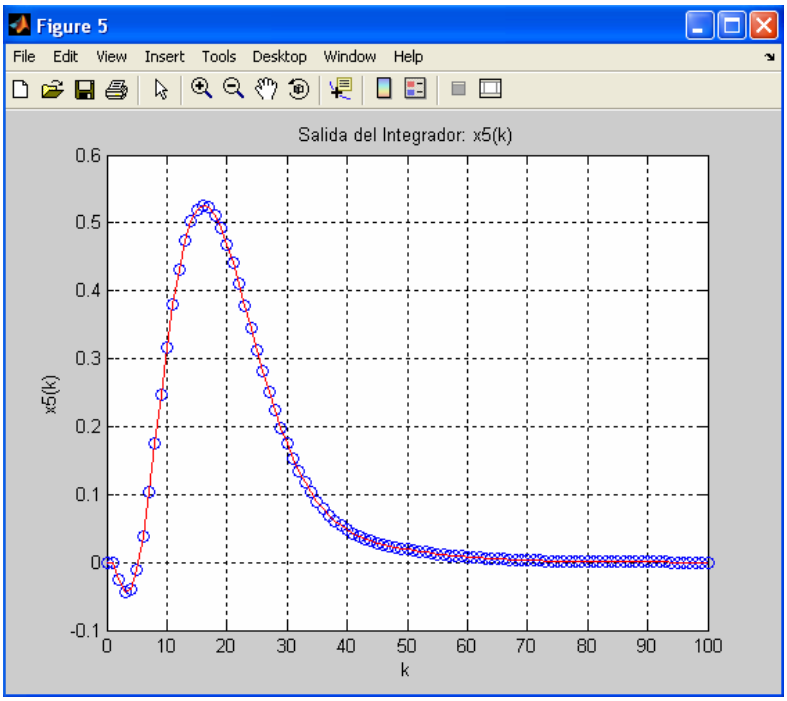


Figura 23 respuesta al escalón unitario para la salida del integrador

Siguiendo e mismo procedimiento anterior de diseñar y programar los controles, se realizará otro software también para control óptimo pero un poco más general y no tan especifico como lo es con un péndulo invertido.

Para el desarrollo de esta interfaz grafica en el GUI de matlab se tuvo en cuenta las matrices de estado necesarias para hacer los cálculos. Dichas matrices de estado deben ser introducidas a la interfaz ya en estado discreto, es decir, la discretización de la ecuación de entrada en espacio de estados continuo incluido el tiempo de muestreo.

Así como también se deben digitar los datos necesarios de los parámetros de la función de costo, encargados de calcular el índice de desempeño para el control óptimo, las ganancias y los valores para la señal de control **u**.

Inicialmente, se partió de la necesidad de los reguladores óptimos de una sola variable de estado, por tanto el software incluye la opción de seleccionar este tipo de sistemas como se muestra en la figura 15.

Puede observarse que las matrices A y B de la ecuación de estado discreta de entrada son de tamaño 1x1, además se incluye el campo que permite entrar la condición inicial de la variable de estado a optimizar.

Las matrices de los parámetros de la función de costo Q, S y R, como he de esperarse, también son de 1x1 para que halla concordancia a la hora de realizar los cálculos.

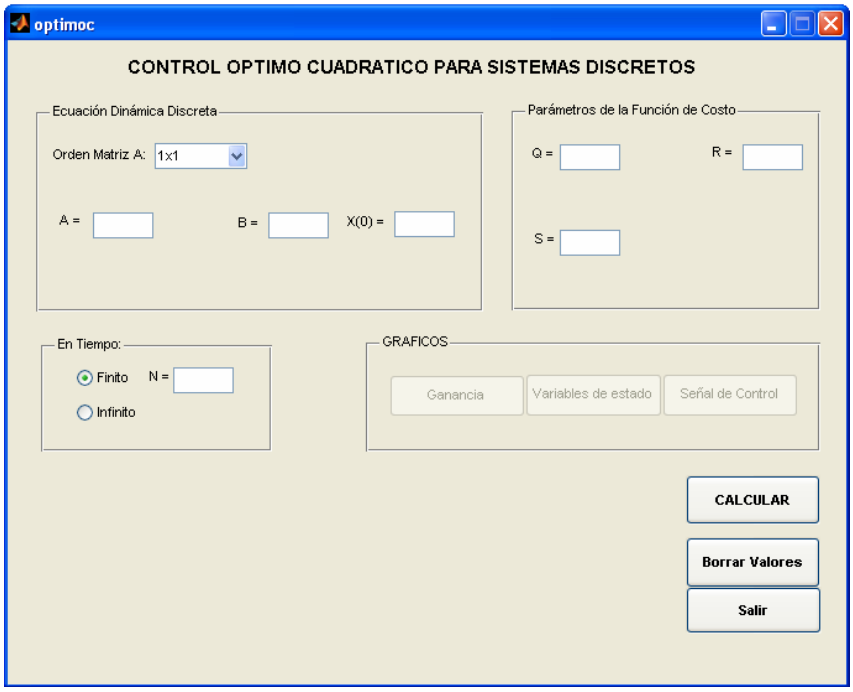


Figura 24 interfaz para el cálculo de sistemas de primer orden

Para observar el comportamiento del programa se hará la siguiente prueba:

Sea un sistema cuya ecuación en el espacio de estados discreto es:

$$x_1(k+1) = 0.3679x_1(k) + 0.6321u(k)$$

los parámetros para la función de costo y condición inicial son :

Q = 1
R = 1 $x_1(0) = 1$
S = 1

Como la parte mas importante para el control optimo es el estado transitorio entonces para tiempo finito con N= 15 se tiene:

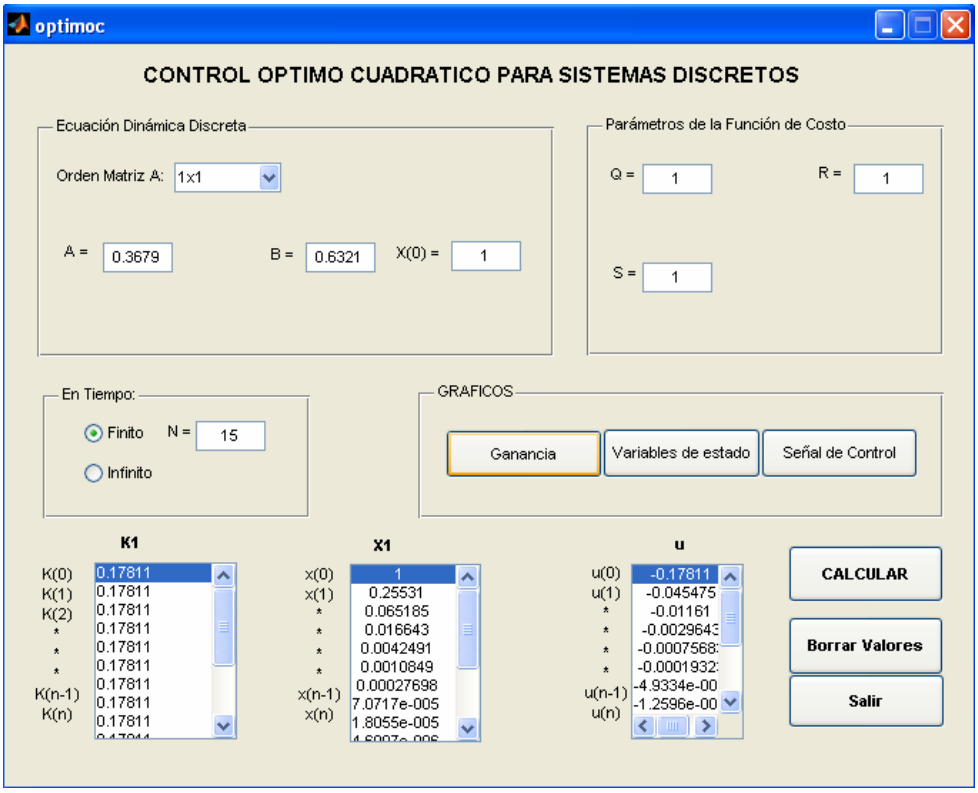


Figura 25 Resultado de un cálculo para un sistema de primer orden

En donde la ganancia K toma los valores de:

K(10)=0,K(9)=0.1662,K(8)=0.1773,K(7)=0.1781,.....,K(0)=0.1781

Estos valores se pueden observar en la grafica de Ganancia como se ve en la figura 26

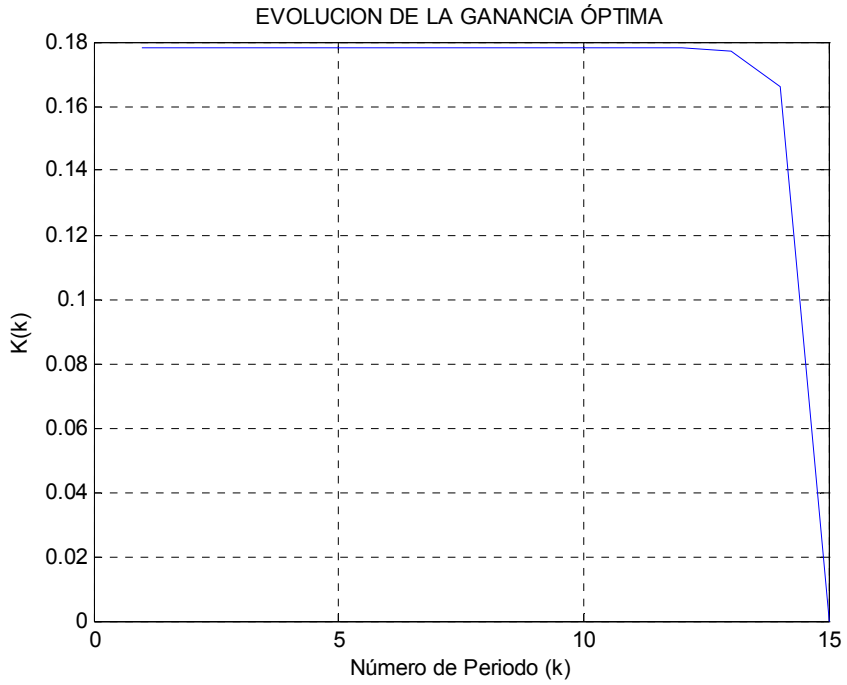


Figura 26 Evolución de la Ganancia óptima para sistema de primer orden

La grafica de la variación de la variable de estado se muestra presionando el botón Variables de estado en el recuadro GRAFICOS como se muestra en la figura 27

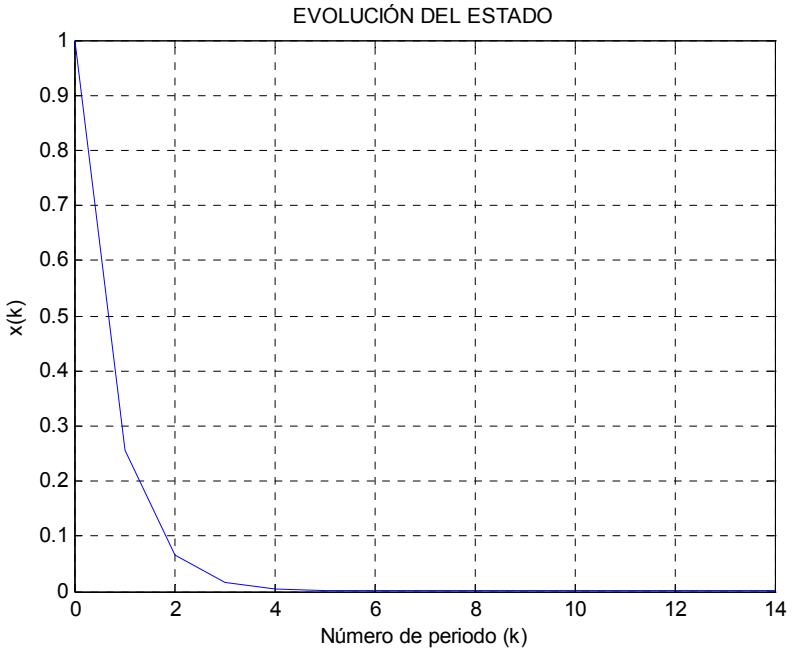


Figura 27 Evolución variable de estado para sistema de primer orden

Por último se muestran los valores para la señal de control u, en donde también se puede ver su evolución a través de grafica presionando el botón Señal de control en el recuadro GRAFICAS, (Figura 28)

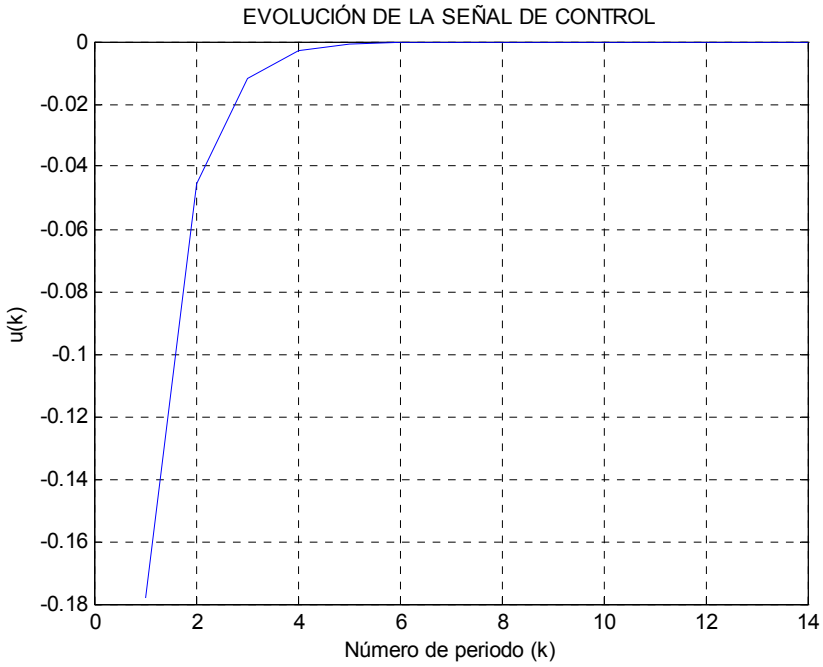


Figura 28 Evolución de la Señal de control para sistema de primer orden

Posteriormente el estudio a los reguladores de control óptimo con dos variables de estado (sistemas de segundo orden), se realiza con el software seleccionando el orden de la matriz A de la ecuación de estado discreta como se ve en la figura 29.

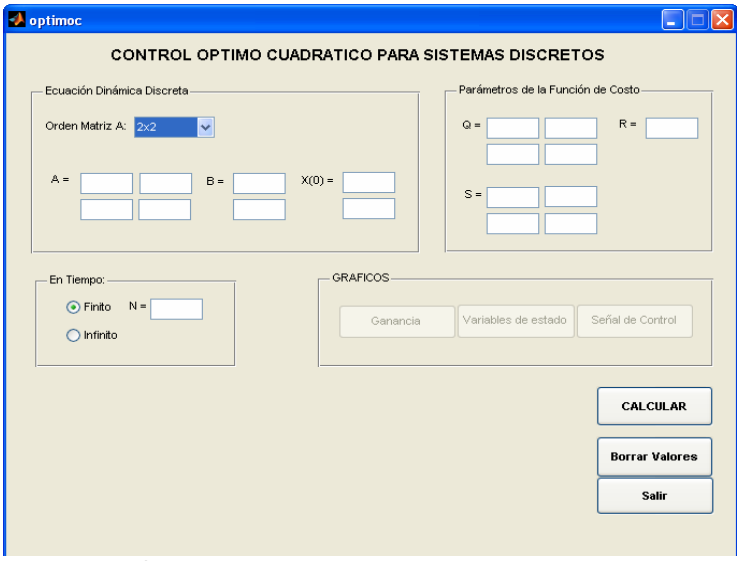


Figura 29 interfaz para el cálculo de sistemas de segundo orden

Se observa que la matriz A es de tamaño 2x2, mientras que la matriz B es de tamaño 2x1, que corresponde precisamente a una ecuación de estado discreta de segundo orden.

Así mismo las matrices para el calculo de los parámetros de la función de costo Q y S deben ser del mismo tamaño de la matriz A, es decir 2x2, mientras que R es una matriz de 1x1.

Para la prueba se tiene un sistema:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0.6277 & 0.3597 \\ 0.0899 & 0.8526 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0.025 \\ 0.115 \end{bmatrix} u(k)$$

los parámetros para la función de costo y condición inicial pudieran ser:

$$Q = \begin{bmatrix} 50 & 0 \\ 0 & 10 \end{bmatrix} \qquad S = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad R = 1$$

Para tiempo finito se tiene que N=9 (aunque podría ser otro)

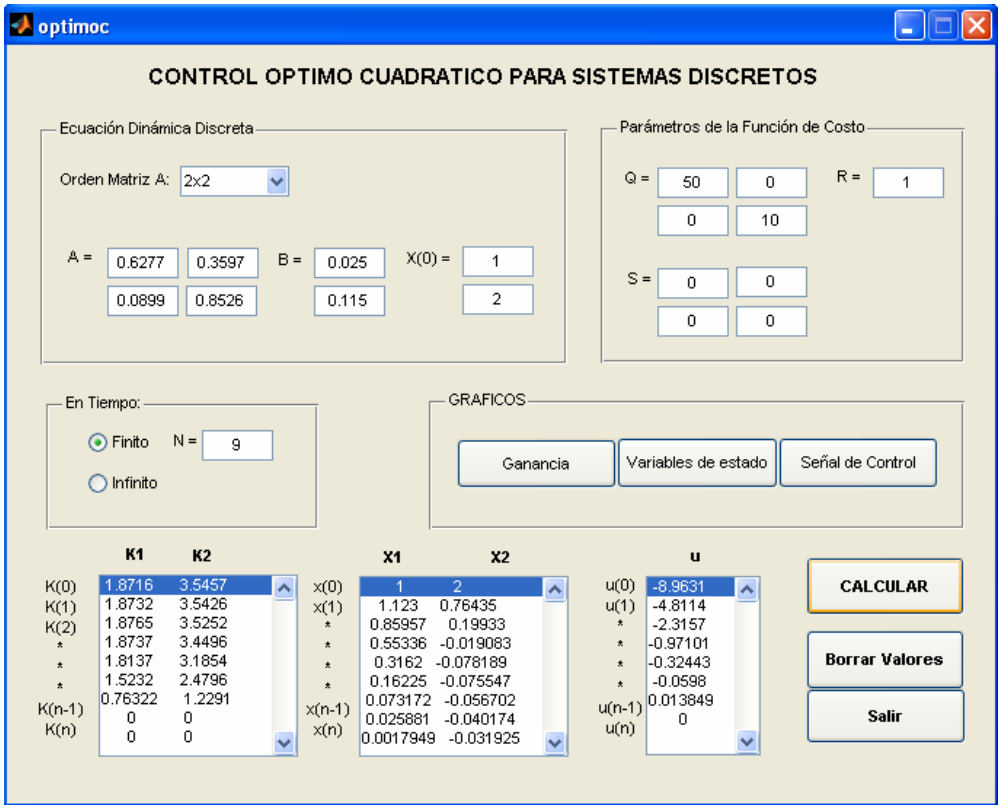


Figura 30 Resultado de un cálculo para un sistema de segundo orden

En donde K toma los valores mostrados y evoluciona según la figura 31

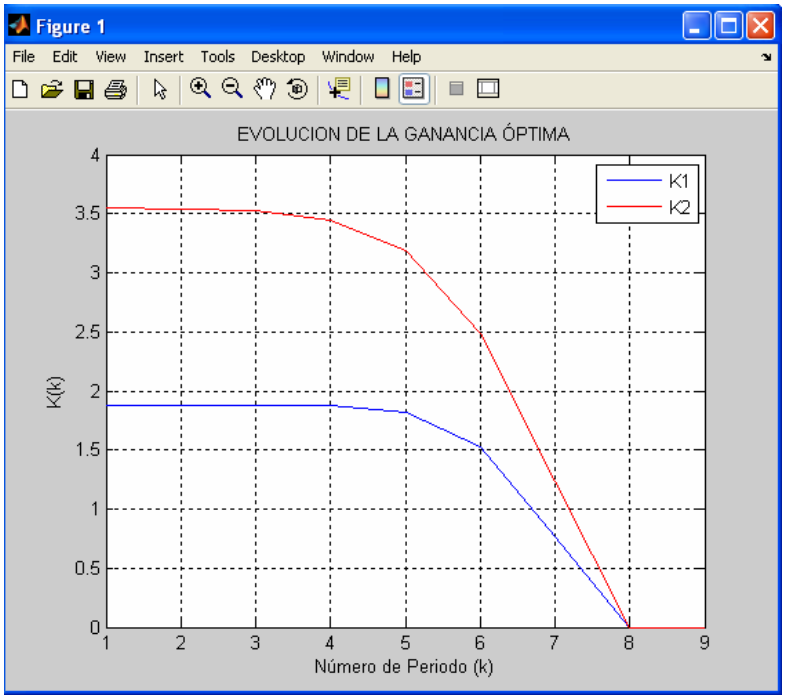


Figura 31 Evolución de la Ganancia óptima para sistema de segundo orden

Las variables de estado del sistema muestran su comportamiento a partir de las condiciones iniciales, permitiendo determinar el coste energético del control como se ve en la figura 32:

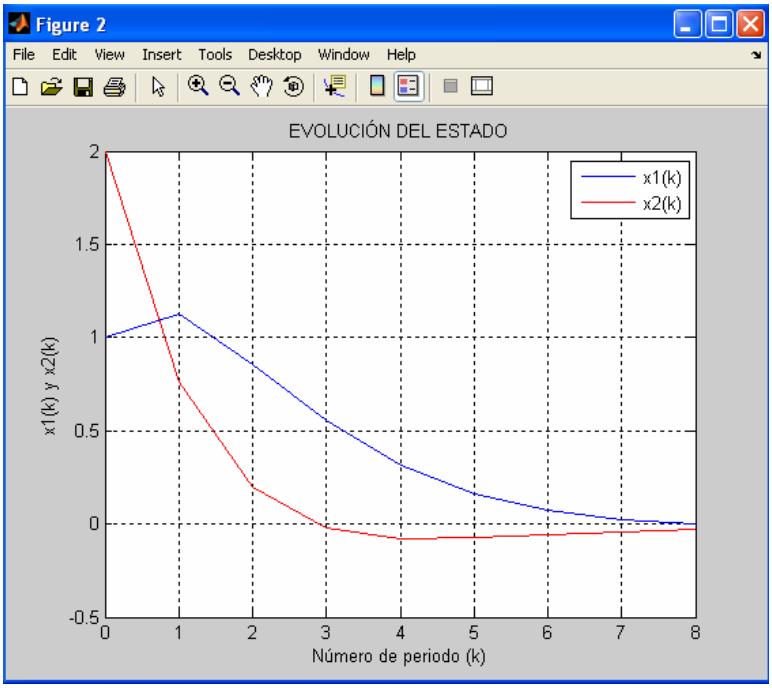


Figura 32 Evolución variables de estado para sistema de segundo orden

Para la evolución de la señal de control se presiona el botón Señal de control del recuadro Gráficos (figura 33) donde se muestra la señal que será introducida al sistema para obtener un control óptimo

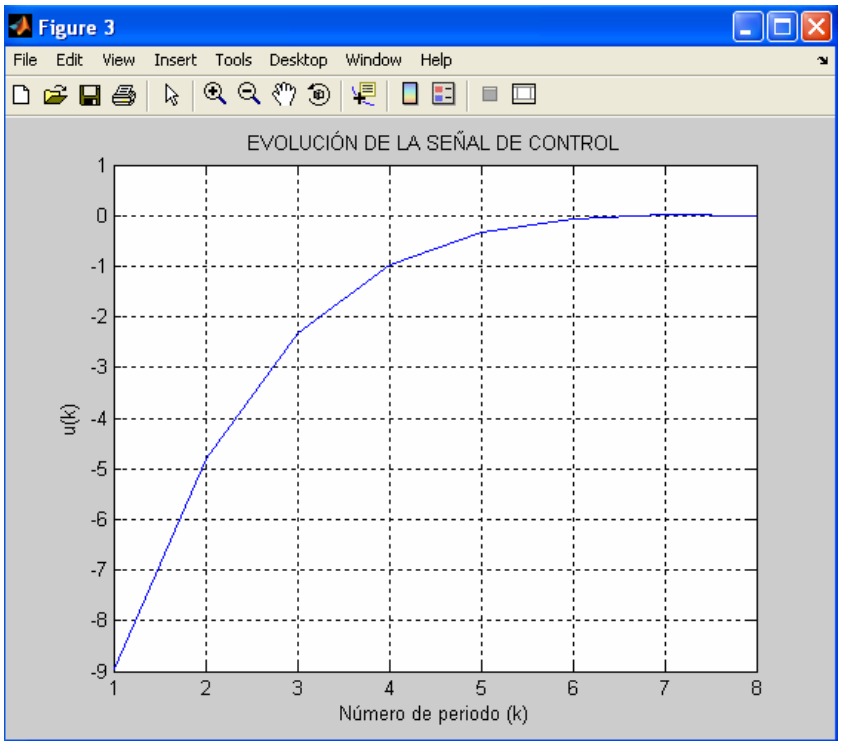


Figura 33 Evolución de la Señal de control para sistema de segundo orden

4. CONCLUSIONES

Se realizo una interfaz de usuario capaz de desarrollar de manera sencilla el cálculo necesario para poder encontrar la señal de control más apropiada para ser aplicada a un sistema que pretenda implementar un control basado en la idea del control óptimo.

La idea del control óptimo es un concepto muy importante en el desarrollo de reguladores, ya que permite la obtención de una función de costo mínima y apropiada para así reducir la cantidad de energía requerida en un sistema que necesita ser controlado, proporcionándole un buen rendimiento del trabajo realizado por el sistema a controlar.

Matlab además de ser una poderosa herramienta de cálculo, posee un admirable paquete de desarrollo capaz de permitir crear interfaces de usuario de una manera gráfica (GUI), recomendable para aquellos usuarios que no necesitan saber los procedimientos internos en los cálculos de ciertos problemas.

La interfaz GUI desarrollada para este proyecto, especifica explícitamente que es de uso único para sistemas discretos y como máximo de segundo orden, que son los sistemas que frecuentemente se utilizan en la práctica, por tanto se deben tener en cuenta estas dos condiciones en el momento de ingresar los datos para realizar los cálculos.

La importancia de poder contar con un programa como éste, radica en el hecho que se cuenta con una herramienta capaz de proporcionar todos los datos necesarios para un diseñador de reguladores óptimos, además permite visualizar dichos datos de una manera grafica lo que mejora la comprensión y el entendimiento de hacia donde se quiere llegar con un diseño especifico.

5. BIBLIOGRAFIA

OGATA, Katsuhiko. Ingeniería de control moderna. Tercera edición. México: Prentice – Hall, 1994. 946 p.

FRIEDLAND, Bernard Control System Design, McGraw-Hill IE. Capítulos 6 y 8.

KWAKERNAAK, H. & Sivan, R. (1972). Linear Optimal Control Systems, Willey-Intercience.

Control optimo de sistemas discretos en variables de estado- Documento Libre distribución

MathWorks Inc MATLAB User´s Guide, Versión 5.2, 1998. Toolbox de control.

Matlab GUI Documentation. Disponible en Internet en : http://www.opendap.org/user/mgui-html/mgui.html

José Jaime Esqueda- Matlab e Interfaces gráficas –Instituto Tecnológico de Ciudad Madero

ANEXOS

ANEXO A: MATRIZ DEFINIDA POSITIVA

En el álgebra lineal, una matriz definida positiva es una matriz hermitiana que es análoga a los números reales positivos.

Sea M una matriz hermitiana cuadrada n × n. De ahora en más notaremos la transpuesta de una matriz o vector a como a^T, y el conjugado transpuesto, a * . Esta matriz M se dice definida positiva si cumple con una (y por lo tanto, las demás) formulaciones equivalentes:

1. Para todos los vectores no nulos $z \in \mathbb{C}^n$ tenemos que $\mathbf{z}^* M \mathbf{z} > 0$.
   Nótese que z * Mz es siempre real.
2. Todos los autovalores λi de M son positivos. (Recordamos que los autovalores de una matriz hermitiana o en su defecto, simétrica, son reales.)
3. La función $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^* M \mathbf{y}$
   define un producto interno $\mathbb{C}^n$.
4. Todas las siguientes matrices tienen determinante positivo.
   - la superior izquierda de M de dimensión 1x1
   - la superior izquierda de M de dimensión 2x2
   - la superior izquierda de M de dimensión 3x3
   - ...
   - M en sí misma

Análogamente, si M es una matriz real simétrica, se reemplaza $\mathbb{C}^n$ por $\mathbb{R}^n$, y la conjugada transpuesta por la transpuesta.

Propiedades

- Toda matriz definida positiva es inversible (su determinante es positivo), y su inversa es definida positiva.
- Si M es una matriz definida positiva y r > 0 es un número real, entonces rM es definida positiva.
- Si M y N son matrices definidas positivas, entonces la suma M + N, el producto MNM y NMN son definidas positivas, además si MN = NM, entonces MN es también definida positiva.
- Toda matriz definida positiva M, tiene al menos una matriz raíz cuadrada N tal que $N^2 = M$.

ANEXO B: CONTROL ÓPTIMO CUADRATICO DE UN SISTEMA DE SEGUIMIENTO (Tomado: Capitulo 8, sección 8-4 de Sistemas de control en tiempo discreto por Katsuhiko Ogata página 596)

La planta considerada es el péndulo invertido mostrado en la figura 1B. Por lo tanto se diseñará un sistema de control para éste sistema mencionado.

El péndulo invertido es inestable, ya que puede caer en cualquier momento y en cualquier dirección a menos de que se le aplique una fuerza de control. Solamente se considera el problema en dos dimensiones en el cual el péndulo solo se mueve en el plano de la página. Se supone que la masa del péndulo se concentra al final de la varilla. La fuerza de control u se aplica al carro.

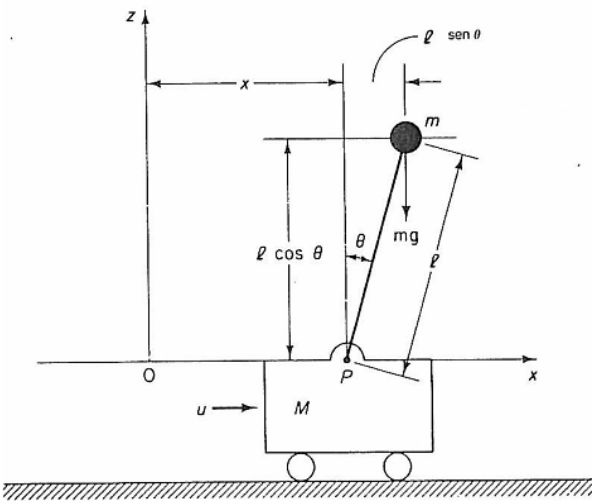


Figura 1B Sistema de péndulo invertido

Es preferible mantener el péndulo en posición vertical tanto como sea posible y mantener el control de la posición del carro, por ejemplo, mover el carro en forma de escalón. Para controlar la posición del carro, se necesita construir un sistema de seguimiento tipo 1. El sistema del péndulo invertido montado sobre un carro no tiene un integrador. Por ende, se alimenta la señal de posición y de regreso a la entrada y se inserta un integrador en la trayectoria directa, como se muestra en figura 2B. Se escoge el periodo de muestreo T=0.1 seg

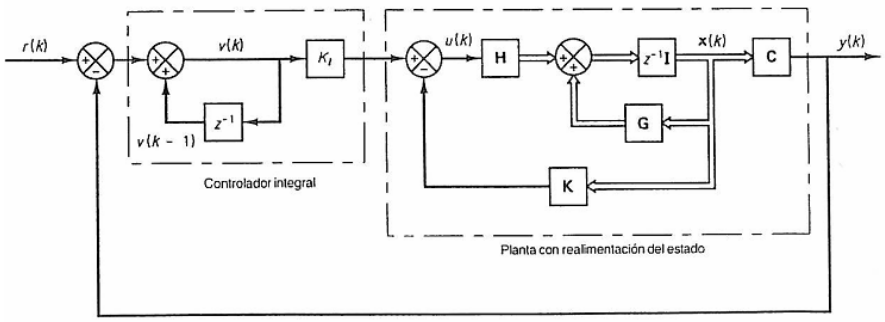


Figura 2B Diagrama bloques sistema de seguimiento

El sistema de control a diseñar incluirá realimentación de estado y un integrador en el lazo cerrado. Las variables de diseño son la constante KI y la matriz de ganancia de realimentación K.

Las variables de estado se definen como:

$$x_1 = \theta \quad x_2 = \dot{\theta} \quad x_3 = x \quad x_4 = \dot{x}$$

El desplazamiento del carro es la variable de salida entonces:

$$y = [0 \quad 0 \quad 1 \quad 0]\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Y $M = 2\ \text{kg}, \quad m = 0.1\ \text{kg}, \quad l = 0.5\ \text{m}$

Para diseñar el sistema de control, se utilizará el sistema discretizado:

$$\mathbf{x}(k+1) = \mathbf{G}\mathbf{x}(k) + \mathbf{H}u(k)$$

$$y(k) = \mathbf{C}\mathbf{x}(k) + Du(k)$$

Donde:

$$\mathbf{G} = \begin{bmatrix} 1.1048 & 0.1035 & 0 & 0 \\ 2.1316 & 1.1048 & 0 & 0 \\ -0.0025 & -0.0001 & 1 & 0.1 \\ -0.0508 & -0.0025 & 0 & 1 \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} -0.0051 \\ -0.1035 \\ 0.0025 \\ 0.0501 \end{bmatrix}, \quad \mathbf{C} = [0 \ 0 \ 1 \ 0], \quad D = [0]$$

De la figura 2B la representación en el espacio de estado de todo el sistema esta dada por:

$$\mathbf{x}(k+1) = \mathbf{G}\mathbf{x}(k) + \mathbf{H}u(k)$$

$$y(k) = \mathbf{C}\mathbf{x}(k)$$

$$v(k) = v(k-1) + r(k) - y(k)$$

$$u(k) = -\mathbf{K}\mathbf{x}(k) + K_I v(k)$$

Donde:

$$\mathbf{K} = [k_1 \quad k_2 \quad k_3 \quad k_4]$$

Ya que:

$$\begin{aligned} v(k+1) &= v(k) + r(k+1) - y(k+1) \\ &= v(k) + r(k+1) - \mathbf{C}[\mathbf{G}\mathbf{x}(k) + \mathbf{H}u(k)] \\ &= -\mathbf{C}\mathbf{G}\mathbf{x}(k) + v(k) - \mathbf{C}\mathbf{H}u(k) + r(k+1) \end{aligned}$$

Se tiene

$$\begin{bmatrix} \mathbf{x}(k+1) \\ v(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{G} & \mathbf{0} \\ -\mathbf{C}\mathbf{G} & 1 \end{bmatrix}\begin{bmatrix} \mathbf{x}(k) \\ v(k) \end{bmatrix} + \begin{bmatrix} \mathbf{H} \\ -\mathbf{C}\mathbf{H} \end{bmatrix}u(k) + \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}r(k+1)$$

Se supone que la entrada r es una función escalón, es decir:

$$r(k) = r(k+1) = r$$

Entonces cuando k se aproxima a infinito:

$$\begin{bmatrix} \mathbf{x}(\infty) \\ v(\infty) \end{bmatrix} = \begin{bmatrix} \mathbf{G} & \mathbf{0} \\ -\mathbf{CG} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}(\infty) \\ v(\infty) \end{bmatrix} + \begin{bmatrix} \mathbf{H} \\ -\mathbf{CH} \end{bmatrix} u(\infty) + \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} r(\infty)$$

Se define:

$$\mathbf{x}_e(k) = \mathbf{x}(k) - \mathbf{x}(\infty)$$

$$v_e(k) = v(k) - v(\infty)$$

Entonces la ecuación del error es:

$$\begin{bmatrix} \mathbf{x}_e(k+1) \\ v_e(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{G} & \mathbf{0} \\ -\mathbf{CG} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_e(k) \\ v_e(k) \end{bmatrix} + \begin{bmatrix} \mathbf{H} \\ -\mathbf{CH} \end{bmatrix} u_e(k)$$

Observe que:

$$u_e(k) = -\mathbf{K}\mathbf{x}_e(k) + K_I v_e(k) = -[\mathbf{K} \quad -K_I] \begin{bmatrix} \mathbf{x}_e(k) \\ v_e(k) \end{bmatrix}$$

Ahora se define:

$$\hat{\mathbf{G}} = \begin{bmatrix} \mathbf{G} & \mathbf{0} \\ -\mathbf{CG} & 1 \end{bmatrix}, \qquad \hat{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ -\mathbf{CH} \end{bmatrix}, \qquad \hat{\mathbf{K}} = [\mathbf{K} \quad -K_I], \qquad w(k) = u_e(k)$$

$$\xi(k) = \begin{bmatrix} \mathbf{x}_e(k) \\ v_e(k) \end{bmatrix} = \begin{bmatrix} x_{1e}(k) \\ x_{2e}(k) \\ x_{3e}(k) \\ x_{4e}(k) \\ x_{5e}(k) \end{bmatrix}$$

Donde $x_{5e}(k) = v_e(k)$, entonces se tiene:

$$\xi(k+1) = \hat{\mathbf{G}}\xi(k) + \hat{\mathbf{H}}w(k)$$

$$w(k) = -\hat{\mathbf{K}}\xi(k)$$

Hay que tener en cuenta que como el sistema es continuo, se considera el índice de desempeño continuo. Pero si se considera un índice cuadrático discreto seria mas simple la solución, para ello se debe hallar la matriz K de tal manera que minimice el índice de desempeño:

$$J = \frac{1}{2}\sum_{k=0}^{\infty} [\xi' \mathbf{Q} \xi + w' R w]$$

La matriz Q se selecciona de manera intuitiva hasta que la respuesta del sistema sea satisfactoria y la matriz R se tomara como valor unitario para este caso:

$$\mathbf{Q} = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad R = [1]$$

Ahora se encontrará la matriz P a partir de la ecuación de Ricatti en estado estacionario, la cual esta dado por:

$$P = Q + \hat{G}^* P \hat{G} - \hat{G}^* P \hat{H} (R + \hat{H}^* P \hat{H})^{-1} \hat{H}^* P \hat{G}$$

Utilizando matlab se interactúa para encontrar el valor de la matriz P hasta que permanezca constante.
Una vez encontrado P, se procede a encontrar el valor de la matriz de ganancia de realimentación óptima K con la ecuación de Ricatti para estado estacionario:

$$\hat{K} = (R + \hat{H}^* P \hat{H})^{-1} \hat{H}^* P \hat{G}$$