Chapter 1

Aplicaciones en control y electrónica

En esta sección presentamos algunas aplicaciones de MATLAB en sistemas de control. MATLAB cuenta para esto con un toolbox de control.

Ejemplo 9.16 Estabilidad de un sistema de retroalimentación

MATLAB cuenta con el paquete CONTROL TOOLBOX que permite realizar una gran cantidad de funciones para sistemas retroalimentados. Por ejemplo, consideremos el diagrama de la figura 9.20

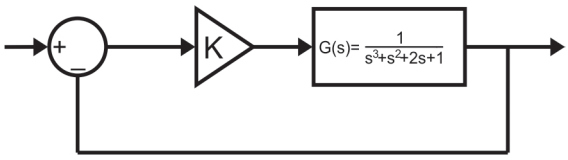

Figure 1.1: Diagrama a Bloques de una planta con retroalimentación.

La planta se define en MATLAB con:

```
nump = [1];
denp = [1 1 2 1];
Planta = tf(nump, denp)
```

Con lo que MATLAB nos da como salida Transfer function:

$$\frac{1}{s^3+s^2+2s+1}$$

Si la ganancia del sistema es K = 1, el sistema con retroalimentación negativa se define con

sistema = feedback(Planta,[1]) con lo que obtenemos la función de transferencia del sistema como

$$\frac{1}{s^3+s^2+2s+2}$$

Para verificar la estabilidad del sistema obtenemos los polos con

polos = pole(sistema)

para obtener

polos =

0.0000 + 1.4142i
0.0000 - 1.4142i
-1.0000

Vemos que este sistema tiene un polo en -1 y una par de polos sobre el eje en $\pm$ 1.4142i y por lo tanto es inestable. Podemos averiguar más sobre la estabilidad de este sistema obteniendo su root locus, que es una gráfica del lugar geométrico de los polos del sistema contra la ganancia K. El root locus lo podemos calcular con

rlocus(planta)

Con lo que obtenemos la figura 9.21. Usando el cursor y presionándolo dos veces sobre un punto cercano al eje vemos que para un valor de $K < 1$ el sistema es estable pero para $K > 1$ el sistema es inestable.

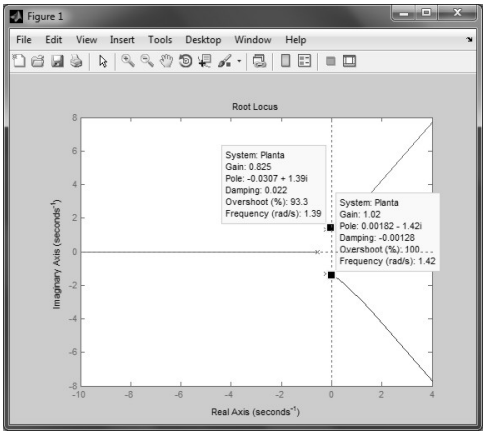

Figure 1.2: Root locus de un sistema retroalimentado.

Ejemplo 9.17 Comparación de métodos de compensación

Consideremos la planta Gp(s)

$$G_p(s) = \frac{1}{s(s+1)(s+4)}$$

El objetivo es comparar los diferentes métodos de control para optimizar alguno de los parámetros de la respuesta tales como el tiempo de subida, el tiempo de establecimiento, sobretiro, etc.

Suponiendo que el tiempo de establecimiento es el importante podemos comparar tres métodos distintos: controlador proporcional, adelanta de fase y PID. Primeramente verificamos la respuesta de la planta con retroalimentación unitaria y sin compensador, Esto lo hacemos con el siguiente archivo-m:

```
%Este es el archivo Ejemplo9_17a.m
%Calcula la respuesta al escalón para
%una planta usando retroalimentación unitaria
%y con compensación proporcional.

Nump = [1]; Demp = [1 5 4 0 ];

%Definimos la planta
Planta = tf ( nump , demp );

%Definimos el sistema
Sist = feedback (planta , [ 1 ] );

%checamos los polos Pole ( sist )

%checamos la respuesta al escalón
t = [ 0 : 0.1 : 40 ];
step ( sist , t )
```

con lo que obtenemos, la figura 9.22 es donde vemos que el tiempo de subida es Tr = 7.41 seg y que el tiempo de establecimiento es Ts $\approx$ 8.67 seg y que no hay sobretiro.

Ahora vamos a usar el controlador proporcional con ganancia K como se muestra en la figura 9.23

Si realizamos un análisis de root locus con

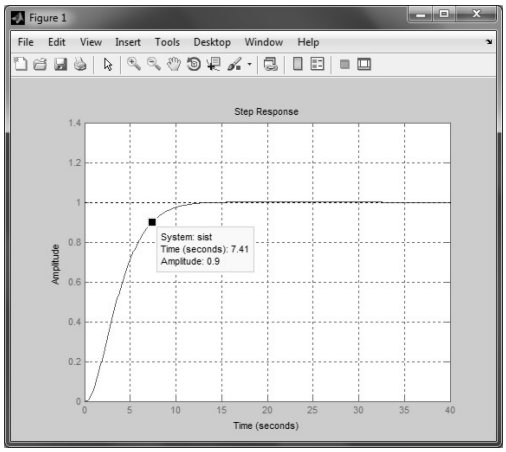

Figure 1.3: Respuesta al escalón sin controlador.

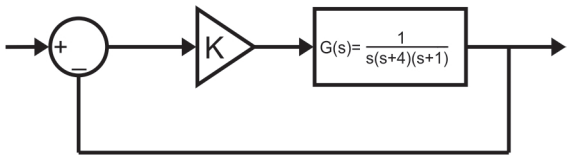

Figure 1.4: Planta con controlador proporcional.

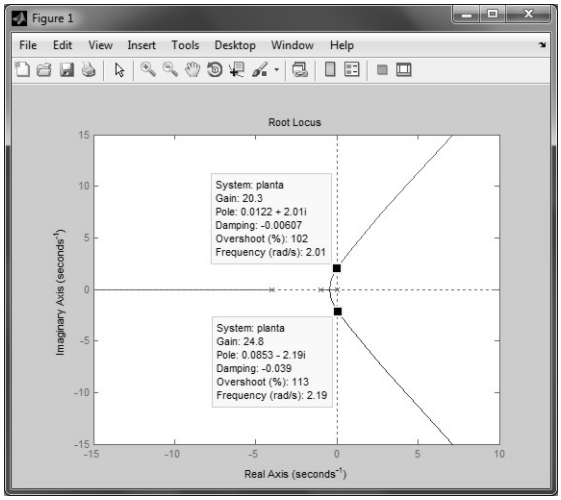

Figure 1.5: Root locus de sistema con controlador proporcional.

```
rlocus(nump,demp)
```

obtenemos la gráfica de la figura 9.24 donde vemos que para K = 20 se obtienen los polos sobre el eje jw y para K > 20 el sistema se vuelve inestable. La función de transferencia se puede obtener con

```
sist = feedback (K*planta , [ 1 ] );
```

Definimos el vector de ganancia con K en el rango de estabilidad como

```
K = [ 0 : 2 : 20 ];
```

Ahora con este archivo-m calculamos la respuesta al escalón del sistema con compensador proporcional.

```
%Este es el archivo Ejemplo9_17b.m
%Calcula la respuesta al escalón para
%una planta con controlador proporcional
clear
clc
close all
nump = [ 1 ];
denp = [ 1 5 4 0 ];
planta = tf( nump, denp );
t = [ 0 : 0.1 : 30 ];
K = [ 0 : 2 : 20 ];
for i = 1: length(K) ;
    sist = feedback ( K ( i )*planta , [ 1 ] );
    step ( sist, t )
    hold on
end
hold off
```

Al correr este programa obtenemos la figura 9.25. Ahí vemos que para valores de K cercanos a 20 se obtiene el mejor tiempo de subida pero el tiempo de establecimiento es peor al igual que el sobretiro. Entonces repetimos el análisis pero para valores de K entre 1 y 4, que se realiza con el archivo Ejemplo9_17c. Obtenemos la gráfica de la figura 9.26. Ahí vemos que para K = 2, se obtiene Ts $\approx$ 7.8 seg y Tr$\approx$3.6 seg mientras que para K = 3 se obtiene Ts $\approx$ 6.46 y Tr $\approx$ 2.7 seg

```
%Este es el archivo Ejemplo9_17c.m
%Calcula la respuesta al escalón para
%una planta usando retroalimentación unitaria
%y con compensación proporcional.
clc
```

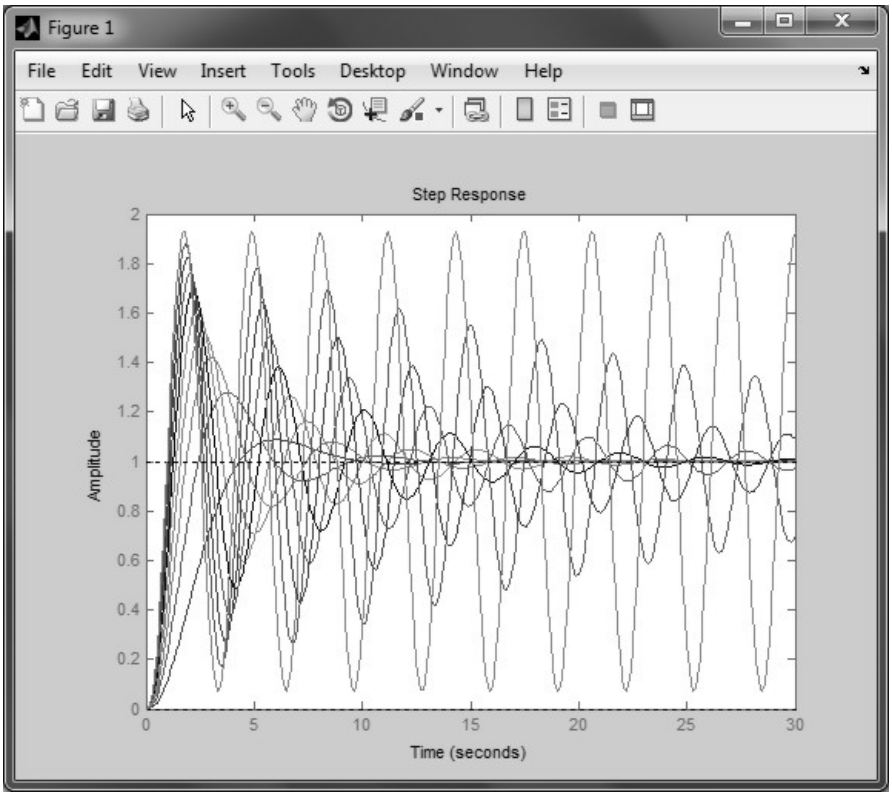

Figure 1.6: Respuesta al escalón para ganancias K entre 1 y 20.

```
clear
close all

nump = [ 1 ];
denp = [ 1 5 4 0 ];
planta = tf( nump, denp );
k = [ 1: 1 :4 ];
for i = 1 : 4;
    sist = feedback( k(i)*planta, [1] );
    step( sist )
    hold on
end
legend ( ' K=1' , 'K=2', 'K=3' , 'K=4' )
hold off
grid on
```

Ahora probamos un controlador proporcional-derivativo (PD) cuya función de transferencia es

$$\mathrm{G}_c(s) = K_P + K_d s = K_P(s + \tfrac{K_d}{K_P})$$

de modo que la función de transferencia de la planta y controlador en cascada es

$$\mathrm{G}_C(s)G_P(s) = \tfrac{K_P(s+\frac{K_d}{K_P})}{(s(s+1)(s+4))}$$

Con el siguiente archivo-m investigamos su respuesta al impulso para $\mathrm{K}_P = 4$,

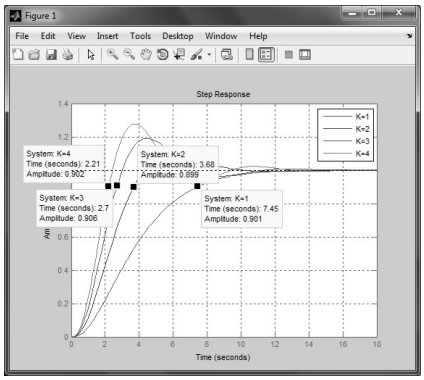

Figure 1.7: Respuesta al escalón para distintos valores de K.

```
%Este es el archivo Ejemplo9_17d.m
%Calcula la respuesta al escalón para
%una planta
%y con compensación proporcional derivativa.
close all
clear
clc
Kp=4;
KD=[2:1:4]
for I = 1:3;
    nump = [ Kp KD(I)];
    denp = [ 1 5 4 0 ];
    planta = tf( nump, denp );
    sist = feedback( planta, [1] );
    step( sist )
    hold on
end
```

```
legend ( ' KD=2' , 'KD=3' , 'KD=4' )
hold off
grid on
```

que nos da la figura 9.27

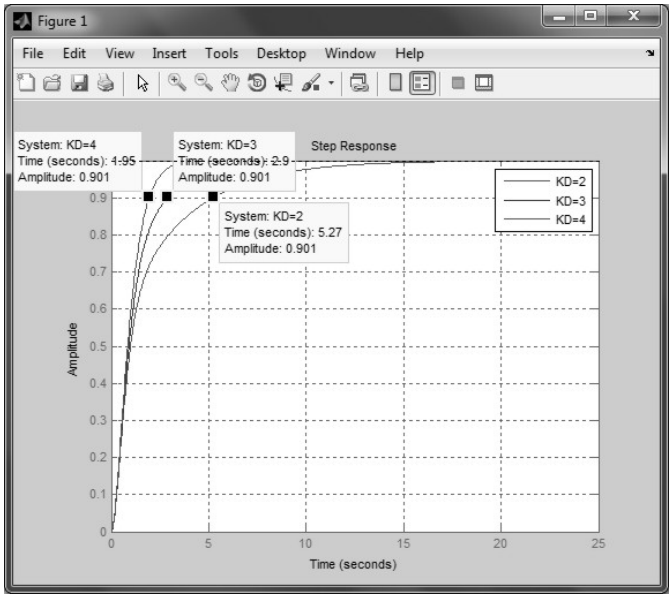

Figure 1.8: Respuesta al escalón para sistema retroalimentado con controlador PD.

y que para $K_D = 4$ nos da una $T_r = 1.19$ seg y $T_s = 2.25$ seg, además de no tener sobretiro.

EJEMPLO 9.18 Control del Telescopio espacial Hubble

Un modelo del telescopio espacial Hubble que se encuentra en órbita alrededor de la tierra, junto con un sistema posicionador se muestra en la figura 9.28

El objetivo es encontrar valores de K_1y K tal que el sobretiro de una entrada escalón r(t) es menor o igual al 5cuando r(t) es una rampa y que el efecto de un escalón de perturbación también se reduce.

El sistema tiene dos entradas, R(s) y P(s), con lo que:

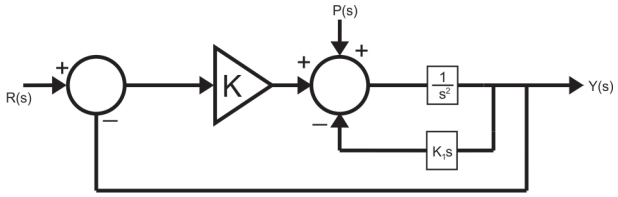

Figure 1.9: Modelo del telescopio espacial Hubble.

$$Y(s) = \frac{KG(s)}{1+KG(s)}R(s) + \frac{G(s}{1+KG(s)}P(s)$$

El error E(s) está dado por:

$$E(s) = \frac{1}{1+KG(s)}R(s) - \frac{G(s}{1+KG(s)}P(s)$$

Primero buscamos satisfacer el requisito de sobretiro de 5%. Para esto sólo tomamos en cuenta el sistema de la entrada a la salida para una entrada de escalón de amplitud A como:

$$\frac{Y(s)}{R(s)} = \frac{KG(s)}{1+KG(s)} = \frac{K}{s(s+K_1)+K} = \frac{K}{s^2+K_1s+K}$$

Para un sobretiro de 5% se necesita seleccionar = 0.7 Para una rampa r(t) = Bt, el error de estado estacionario está dado por:

$$\mathrm{e}_{ee}^{R} = lim_{s\to 0}\frac{B}{KG(s)} = \frac{B}{(K/K_1}$$

Y para una perturbación de escalón unitario:

$$\mathrm{e}_{ee}^{R} = lim_s(\frac{-1}{(s+K)+K}) = \frac{-1}{K}$$

Para que el error debido a la perturbación se pueda reducir, necesitamos aumentar k. También para reducir el error de estado estacionario cuando la entrada es una rampa se necesita aumentar $\mathrm{k/k_1}$. Además se requiere ξ=0.7 La ecuación característica es:

$$\mathrm{s}^2 + 2\xi\ w_n s + w_n^2 = s^2 + K_1 s + K = s^2 + 2(0.7)w_n s + K$$

De modo que $w_n = \sqrt{K} y K_1 = (0.7)w_n, dedonde K_1 = 1.4\sqrt{K}$. Por lo tanto:

$$\frac{K}{K_1} = \frac{K}{1.4\sqrt{K)}} = \frac{\sqrt{K}}{1.4} = \frac{\sqrt{K}}{2\xi}$$

Si K = 36, K_1 = 1.4 X 6 = 8.4 y K/K_1 = 36/ 8.4 =4.28 y si K = 100, K_1 = 1.4, la respuesta al escalón para r(t) y para la perturbación se muestra en la figura 9.62

```
% Este es el archivo Ejemplo9_18.m
% Se evalúa el comportamiento del telescopio Hubble.
clear
clc
close all

% Primero se calcula la planta.
K = 100;
K1 = 14;
nump = [ 1 ];
denp = [ 1 K1 0 ];
% función de transferencia desde r(t)
planta = tf( nump, denp );
% función de transferencia desde r(t)
sistema = feedback ( K*planta, [ 1 ] );
% perturbación
% función de transferencia desde la perturbación
perturbación = feedback (planta, [ K ] )
step ( sistema )
hold on
step ( perturbación )
hold off
```

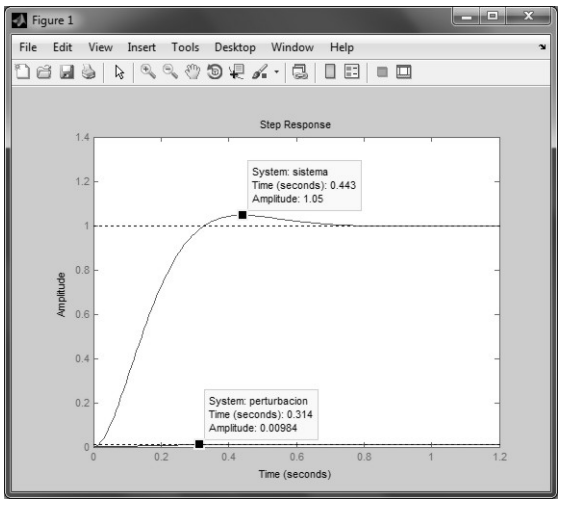

Figure 1.10: Respuesta del sistema del telescopio Hubble.

EJEMPLO 9.19 Control del ángulo de inclinación de una aeronave*

Se desea controlar el ángulo de inclinación de una aeronave como se muestra en la figura 9.30a.

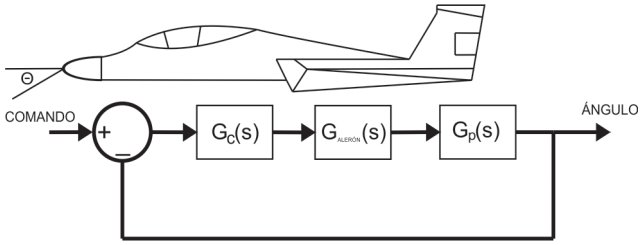

Figure 1.11: Sistema de control del ángulo de inclinación de una aeronave. a) Definición del ángulo de inclinación, b)Sistema de control.

La figura 9.30b muestra el lazo de alimentación unitario del control de ángulo de un Jet, donde para la planta

$$\mathrm{G}_p(s) = \frac{((s+6)(s+15))}{(s(s+3-12j)(s+3+12j)}$$

Este problema fue resuelto y programado por el Ing. Antonio Arizaga, estudiante de Maestría en Electrónica en la Universidad de las Américas, Puebla.

Y para el sistema del alerón

$$\mathrm{G}_{alerón}(s) = \frac{4000}{(s+8)}$$

se desea diseñar un compensador que cumpla con estas especificaciones:

a)Constante de error de velocidad $\mathrm{K}_v >= 130$
b)Sobretiro $\mathrm{M}_p \leq 15\%$
c)Tiempo del primer máximo $\mathrm{T}_p \leq 0.7s$

Para hacer uso de otras funciones de MATLAB se resolverá este problema utilizando variables de estado.

Se define el sistema en términos de los polos, ceros y ganancia tomando en cascada las expresiones G_py $\mathrm{G}_{alerón}$

```
Z = [ -6 -15 ];
P = [ -8 -3+ 12i -3-12i 0 ];
K = 4000;
```

Se transforma a variables de estado

[A, B, C, D] = zp2ss(Z', P', K)

Dados Tp y Mp se encuentra el sistema de Segundo orden que cumple con estos requerimientos.

Tp = .5;
Mp = .15; % sobretiro
si = (-log(Mp))/(sqrt((pi ^{2}) + $(log(Mp))^2$));
wd = pi/t_p;
wn = wd/(sqrt(1- (si ^{2})))
[NUM, DEN] = ORD2(wn, si)

Se obtienen las raíces del denominador.

r = roots(DEN)

El sistema es de cuarto orden por lo que se adicionan 2 polos más a las raíces del denominador los cuales serán -6 y -15 para cancelar los ceros del sistema y dejarlo de segundo orden

polos = [r(1) r(2) -15 -6]

Definimos k como el vector de las ganancias de la retroalimentación de estados.

k = acker(A, B, polos)

Hacemos la retroalimentación de estados

Anueva = A -B*K

El nuevo sistema queda definido como

sys2 = ss(Anueva, B, C, D)

T = tf(sys2)

Se cancelan los términos comunes.

T = minreal(T)

Se verifica la respuesta al escalón

step (sys2)

El error de estado estacionario es muy grande por lo que se hace necesario el uso de un integrador.

La respuesta del escalón se eleva hasta 74. Para hacerla unitaria cambiamos por una ganancia.

k1 = 0.0135

lo que deja al sistema como

sys2 = sys2*k1

step(sys2)

El archivo-m que realiza esto es

```
% Este es el archivo Ejemplo9_19.m
% Controla el ángulo para mantener horizontal un avión.
%
clear
close all
Z = [ -6 -15 ];
P = [ -8 -3+12i -3-12i 0 ];
K = 4000;
[ A, B, C, D ] = zp2ss( Z', P'. K )
Tp =.5;
Mp =.15; % sobretiro
si = ( -log( Mp ) )/( sqrt( ( pi ^2) + (log(Mp))^2));
wd = pi/Tp;
wn = wd/ ( sqrt( 1- ( si ^2)))
[ NUM, DEN ] = ord2w(wn, si )
r = roots( DEN )
polos = [ r(1) r(2) -15 -6 ]
k = acker( A, B, polos )
Anueva = A - B*k
sys2 = ss( Anueva, B, C, D )
T = tf( sys2 )
T = minreal( T )
step (sys2)
k1 = 0.0135
sys2 = sys2*k1
step( sys2, 2.5)
grid on
```

La respuesta al escalón se muestra en la figura 9.31 donde vemos que el primer máximo ocurre en 0.5 seg y el sobretiro es de 1.527-1.0023 = 15.04%

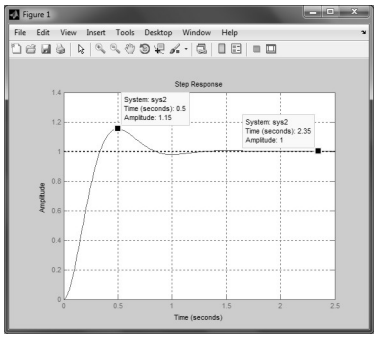

Figure 1.12: Respuesta al escalón del sistema.

9.5 Aplicaciones en Electrónica

EJEMPLO 9.19 Sumador digital (Full adder)

La versatilidad de MATLAB se hace patente al manejar variables Booleanas definidas por MATLAB como lógicas. Una variable se vuelve lógica con solo asignarle uno de los dos posibles valores True o False. Las variables lógicas pueden ser vectores o matrices. Las operaciones básicas con variables lógicas pueden ser NAND, OR y XOR, las cuales se realizan con & para nand, | para nor, y xor para xor, respectivamente.

Deseamos ahora realizar un sumador completo (full adder) usando dos medios sumadores (half adder). El circuito que realiza esto se muestra en la figura 9.32a y la realización de un medio sumador se muestra en la figura 9.32b. Un medio sumador se puede realizar por la siguiente función:

```
function [ s c ] = medio_sum ( a, b )
% s es el bit de la suma y c es el carry resultante.
% las entradas son los bits a y b.
s = xor ( a , b );
c = a & b;
```

La interconexión de los medios sumadores y la compuerta or se realiza con el siguiente archivo-m y la figura 9.33 muestra las formas de onda de salida.

```
% Este es el Archivo Ejemplo9_20.m
```

```
% Realiza un sumador completo a partir de medios sumadores.
close all
clc
clear

x = [0 0 0 0 0 0 true true true 0]; % entrada x.
y = [0 0 0 true true 0 0 0 true true ]; % entrada y
carry_en = [ 0 0 0 true true true 0 0 0 true ]; % carry de entrada.
t=[0:1:9];

[ s1 c1 ] = medio_sum ( x , y ); % Resultados del primer medio sumador.
[ ss c2 ] = medio_sum ( carry_en , s1 );% Resultados del segundo medio
carry_sal = c2—c1; % carry de salida.
subplot(5, 1, 1)
plot(t, x )
axis([0 10 -1 2])
xlabel ( 'entrada x' )
subplot(5, 1, 2)
plot(t, y )
axis([0 10 -1 2])
xlabel ( 'entrada y' )
subplot(5, 1, 3)
plot (t, carry_en )
axis([0 10 -1 2)]
xlabel ( 'carry de entrada' )
subplot( 5, 1, 4)
plot(t, ss )
xlabel ( 'suma' )
axis([0 10 -1 2])
subplot(5, 1, 5)
plot(t, carry_sal )
xlabel ( 'carry de salida' )
axis([0 10 -1 2])
```

Figura 9.32 Sumador completo a) Diagrama de bloqueo. b) Medio sumador.

Figura 9.33 Resultados de sumar las señales x, y y carry$_e n$.

EJEMPLO 9.21 Análisis de MonteCarlo para un transitor en configuración de emisor común.

Consideremos el circuito de la figura 9.34. Para este circuito el punto de operación se obtiene de la siguiente manera: del lazo colector-emisor

V_CC=(R_C+R_E) I_C+V_CE

Del lazo base-emisor

V_BB=R_B I_B+V_BE+R_B I_B

Figura 9.34 Transistor en configuración de emisor común.

Si suponemos que I_C=I_E, I_E=? I_B y V_BE=0.7 V, entonces de la segunda ecuación,

I_CQ=(V_BB-V_BE)/(R_B/?+R_E)=(4-0.7)/((12 K)/75+16 K)?(4-0.7)/(16 K)=3.3/(16K)=0.206mA

de la primera ecuación, V_CEQ=V_CC-(R_C+R_E) I_CQ=4.4 V

Ahora deseamos realizar un análisis de MonteCarlo. En este tipo de análisis los componentes del circuito varían aleatoriamente en una base estadística y el circuito se analiza con estos valores. Se generan conjuntos aleatorios de los parámetros y se construye el comportamiento estadístico del circuito. Por ejemplo, para un resistor el valor aleatorio se puede generar con:

R=R_nom [1+2 ? (aleatorio-0.51)]

Donde R_nom es el valor nominal de R y el ancho de la distribución es 2 ? R_nom. La función rand genera números aleatorios uniformemente distribuidos entre 0 y 1.

Supongamos que la tolerancia V_CC es de 5

V_CC=12 (1+0.1 rand-0.5)
R_1=18000 (1+0.2 rand-0.5)
R_C=22000 (1+0.2 rand-0.5)
R_E=16000 (1+0.2 rand-0.5)
R_B=12000 (1+0.2 rand-0.5)
?=75 (1+ rand-0.5)

El siguiente archivo-m calcula los puntos Q correspondientes a cada conjunto de valores y los grafica.

```
% Este es el archivo Ejemplo9_21.m
% Realiza un análisis de MonteCarlo
% cuando algunos elementos varían de su valor nominal.
clear
clc
close all

VCC = 12*(1+0.005*(rand(2000, 1)-0.5));
RC = 22e3*(1+0.05*(rand(2000, 1)-0.5));
RE = 16e3*(1+0.05*(rand(2000, 1)-0.5));
RB = 12e3*(1+0.05*(rand(2000, 1)-0,5));
beta = 75*(1+0.1*(rand(2000, 1)-0.5));
VBB = 4;VBE= 0.7;
for I = 1:2000;
    ICQ(i) = (VBB-VBE)/(RB(i)/beta(i)+RE(i));
    VCEQ(i) = VCC(i)-(RC(i)+RE(i)*ICQ(i);
end
a = [ICQ, VCEQ];
hist(ICQ)
figure
hist(VCEQ)
```

Figura 9.35 Variación de I_CQ

Figura 9.36 Variación de V_CEQ

En la figura 9.35 vemos que casi todos los resultados de I_CQ están muy cerca de 0.2mA mientras que en la figura 9.36 vemos que la mayoría de los análisis resultan en un valor deV_CEQ alrededor de 4.2 volts. Las figuras 9.35 y 9.36 pueden variar ligeramente en otras máquinas ya que se usan números aleatorios.

EJEMPLO 9.21 Cálculo de funciones trigonométricas usando el algoritmo CORDIC*

La técnica de CORDIN (Cordinate Rotation DIgital Computer) es un algoritmo diseñado para realizarse en hardware binario y que solamente usa operaciones básicas como sumas y corrimientos. Utiliza unidades de memoria y tablas de datos (Look Up Tables) previamente calculadas. Este algoritmo es capaz de calcular varias funciones incluyendo las trigonométricas, hiperbólicas, raíces y funciones logarítmicas, entre otras.

Algoritmo General Para entender este algoritmo, se describe el funcionamiento utilizando la figura 9.37,

Figura 9.37 Ángulos en el sistema CORDIC

La diagonal OA está a un ángulo inicial de ?? que por simplicidad se elige igual a cero, la línea OB es la línea OA, pero con una rotación contraria a las manecillas del reloj de ? grados para llegar a un ángulo de ??, que es el ángulo de interés del cual obtendremos el seno y el coseno. Para el algoritmo CORDIC, esta rotación de ?? a ?? se hace en un número definido de pequeñas rotaciones cuyos ángulos se eligen de una manera conveniente, de forma que al sumar cada incremento de X y Y de cada pequeña rotación se obtendrá del cos ?? y el sen ??, respectivamente, claro, si se eligió ?? = 0.

Siguiendo con el análisis matemático: dada cada rotación, la nueva coordenada X2, Y2 está relacionada con las coordenadas anteriores X1, Y1 en la forma siguiente: X2 = X1 * cos ? - Y1 * sin ? Y2 = X1 * sin ? + Y1 * cos ?

De este par de ecuaciones que definen la transformación por rotación se deriva el algoritmo CORDIC. Las ecuaciones anteriores pueden reescribirse de la manera siguiente:

X2 = cos ? * [X1 - Y1 * tan ?]
Y2 = cos ? * [X1 * tan ? + Y1]

Este problema fue resuelto y programado por el Ing. Daniel Ortega, estudiante de Maestría en Electrónica en la Universidad de las Américas, Puebla. Donde ? es el ángulo con el que estaremos rotando el vector inicial hasta llegar al ángulo de interés. Estos incrementos se eligen de tal forma que el valor tan ? es una potencia fraccional de 2, esto es, el valor de tan ? y el ángulo ? cambiará de la siguiente manera:

tan ? = 1/1 ' 1/2 ' 1/4 ' 1/8 ' 1/16 ' 1/32
? = 45 ' 26.56 ' 14.03 ' 7.12 ' 3.57 ' 1.78

De esta manera se ha remplazado la multiplicación de tan ? por el uso de una rápida y sencilla operación de corrimiento. Estos valores son guardados en una pequeña memoria ROM o Look Up Table.

Por otro lado el valor de cos ? que multiplicará a cada incremento, se reduce a una multiplicación final por un factor que tendrá el valor de la multiplicación de los cosenos, por ejemplo, para 5 iteraciones, se tendrá un factor de:

cos(45)*cos(26.56)*cos(14.03)*cos(7.12)*cos(3.57)*cos(1.78) = 0.0607352

Más aún, para evitar la multiplicación de este valor al resultado final, se puede incializar X con este valor, en lugar de 1.Finalmente, haciendo esta consideración, sustituyendo tan ? por $2^{(} - 1)$ en las ecuaciones de rotación, el algoritmo se puede representar por las siguientes ecuaciones:

X(i+1) = X (i)-m*Y(i)*d(i) * $2^{(} - 1)$
Y(i+1) = Y (i)-X(i)*d(i) * $2^{(} - 1)$

Donde

+sgn [Z(i)] para modo de creación
d(i)=
-sgn [Z (i)] para modo vectorial
E(i) = arctan(i) para seno y coseno

y m depende también del tipo de operación que se quiere realizar, m = 1 para funciones trigonométricas, m = -1 para funciones hiperbólicas, o m = 0 para funciones aritméticas.

El algoritmo CORDIC brevemente descrito aquí se ha realizado en SIMULINK y se muestra en la figura 9.38. El valor de ángulo se da en el bloque 1/z con la leyenda doble pulsación; aquí se ha dado el valor inicial de 60, como se muestra en la figura 9.39, para obtener la repuesta en los bloques de seno y coseno.

Figura 9.38 Sistema que realiza el algoritmo CORDIC para cálculo de seno y coseno.

Figura 9.39 Ventana para indicar el ángulo

9.6 Aplicaciones en Ingeniería de Alimentos EJEMPLO 9.23 Interpolación de datos experimentales del proceso freído de donas

En el proceso de freído de donas a 180 se recabaron datos experimentales del proceso de transferencia de masa (Vélez y Sosa-Morales, 2003). En particular aquí consideramos el total de aceite ganado en las donas durante el proceso. Los datos experimentales se muestran en la Tabla 9.3. Los datos de la tabla se van a ajustar a una curva. Los autores reportan como la mejor aproximación a estos datos la ecuación:

O_G=1.313+?10.160?$_t$−?0.001?$_(t^2)$

Al realizar un ajuste de curvas en MATLAB podemos buscar distintos grados de los polinomios y observar que tan fino es el ajuste. El siguiente archivo-m realiza el ajuste de las curvas para polinomios de distintos grados.

```
% Este es el archivo Ejemplo9_23.m
% Realiza el ajuste de curvas para los datos del aceite
% ganado durante el freído de donas a 180 grados.
% Se usan grados 2 3 y 4.

t= [0 15 30 45 60 75 90 105 120];
gain= [ 0 5 6 6.2 7.2 6.8 6.7 7.9 6.1];
stem(t,gain)
hold on
```

```
p2 = polyfit(t,gain,2)
p3 = polyfit(t,gain,3)
p4 = polylfit(t,gain,4)
pp2 = polyval(p2,t);
pp3= polyval(p3,t);
pp4= polyval(p4,t);
plot(t,pp2,1,pp3,t,pp4)
hold off
legend ( 'datos', 'n=2, 'n=3', 'n=4' )
xlabel( 'Tiempo de freído -seg' )
ylabel ( 'Aceite ganado (%) ' )
```

La salida en MATLAB al correr el archivo es:

```
p2 =
-0-0010 0.1576 1.3509
p3 =
0.0000 -0.0033 0.2604 0.6354
p4 =
-0.0000 0.0001 -0.0114 0.4506 0.0857
```

Esto indica que los polinomios que se obtienen son:

OG = 1,359+0.1576t - 0,004t n=2
OG = 0.6354+0.2604t - 0.0033t n=3
O = 0.0857+0.4506t - 0.0114t +0.001t n=4

Observaciones: 1) el caso n = 2 es muy parecido al de la Ref.1; 2) aunque se pidió el resultado para un polinomio de grado 3, MATLAB sólo entrea un polinomio de segundo grado pero con distintos coeficientes y mejor ajuste para n = 3; 3) para n = 4 se obtiene un polinomio de grado 3 para el cual el ajuste es mejor como se observa en la figura 9.40.

Figura 9.40 Datos experimentales y polinomios de ajuste.

EJEMPLO 9.24 Ley de Fick

Cuando se agrega una gota de tinta en un vaso con agua se efectúa un proceso de de difusión. Este y otros fenómenos están gobernados por las leyes de Fick. Si la difusión es en estado estable, la primera ley de Fick establece que el perfil de la concentración C(x) y el flujo de la difusión J, que es la masa transportada por unidad de área por unidad de tiempo están relacionadas por:

$$J=-D\,dC/dx \qquad \text{Primera Ley de Fic}$$

donde D es el coeficiente de difusión. En esta ecuación suponemos que la difusión es en la dirección X. Si la difusión no es en estado estable, entonces la difusión está gobernada por la segunda ley de Fick.

?C/?t=-D $(?^2C)/(?x^2)$ Segunda Ley deFick

donde C = C(x, t)

Una solución para esta ecuación es:

(C_x-C_0)/(C_s-C_0)=6/?^{2}?_$(n = 1)^?$??$1/n^2 exp?(? - n?^2(D?^2)/r^2)$

donde C_ses la concentración superficial del material que se va a difundir, C_0 es la concentración incial, C_x es la concentración a una distancia X de la superficie, D es el coeficiente de difusión, n es el número de términos que tomamos en la serie y r es la distancia de la difusión. Otra solución dada por funciones error como:

(C_x-C_0)/(C_s-C_0)=1-erf(x/(2?(D t)))

donde erf es la función error. La ley de Fick es usada para cálculos de contenido de humedad durante el proceso de secado en el periodo de deshidratación decreciente (Soriano y Vélez, 2003). En particular, se desea saber el contenido de humedad en el betabel después de ser sometido a un secado durante un tiempo t. El betabel se encuentra en forma de esferas de 2.5cm de diámetro y las condiciones de operación del secador son: temperatura de 60C, humedad de equilibrio de 0.066 y la inicial de 0.25kg_agua/kg_ss. La solución de la ecuación de Fick para este caso está dada por:

(X-X_6)/(X_i-Xe)=6/?^{2}?_$(n = 1)^?$?$1/n^2 exp(-n^2(D?^2)/r^2)$

donde D es la difusividad que para el betabel es de 7=?10?$^(-10)m^2/s$. El siguiente archivo-m calcula las gráficas del contenido de humedad durante los primeros 60 segundos. La figura 9.41 muestra el resultado cuando se toman desde un solo término de la serie hasta 13 términos. De la figura 9.41 podemos apreciar que a partir del quinto término de la serie las curvas de humedad son casi iguales.

```
% Este es el archivo Ejemplo9_24.m
% Calcula el comportamiento, según la
% ley de Fick, del secado de betabel.
clear
clc
close all
y=zeros(11,3601);
Xi = 0.25;
```

```
Xe = 0.066;
x1 = (6/pi ^2);
t = [0:1:3600];
D = 7e-10;
r = 1.25e-2;
x = 0
for n = 1:13;
      x = x+x1*(1/n ^2) * exp(-n^2 * D * pi^2 * t/r^2);
      X = x*(Xi-Xe)+ Xe;
      y(n,:) = X;
end
plot ( t/60, y(1:2:13,:))
legend ( 'n=1','n=3','n=5','n=7','n=9','n=11','n=13' )
ylabel( 'Curvas de secado' )
xlabel( 'tiempo-minutos' )
```

Figura 9.41 Curvas de secado para distinto número de términos de la serie.

EJEMPLO 9.24 Transferencia de calor*

Se requiere determinar la temperatura de puré de chícharo contenido en un recipiente cilíndrico que se calienta a cierta temperatura. El punto donde se desea calcular está a una distancia r del eje del cilindro. Los recipientes cilíndricos de dimensiones finitas se modelan mediante las intersecciones de planos y cilindros infinitos como se muestra en la figura 9.42.

Figura 9.42 Intersección de planos paralelos y cilindro infinito.

Este ejemplo fue propuesto por el Dr. Jorge Welti Chanes, profesor del Departamento de Ingeniería Química y Alimentos de la Universidad de las Américas-Pueblas. Para determinar la temperatura de un recipiente cilíndrico se puede proceder de la siguiente manera: primero se calcula la temperatura residual de un cilindro infinito de radio r_m, la cual está dada por:

((T_a-T)/(T_a-T_b)) (_infinito$^{c}ilindro$) = $2/r\_m?\_(m = 1)^{?}?(J\_0(B\_nr/r\_m)[exp(?-$
$B?\_n^2K?/(?C\_pr\_m^2))]])/((1 + (kB\_n^2)/(h^2r\_m^2))B\_n/r\_mJ\_1(B\_n))$

donde B_n son las soluciones de mJ_0 (B)=BJ_1 (B). J_0 y J_1son las funciones de Bessel de orden cero y uno, respectivamente. Después se calcula la temperatura residual de un volumen acotado por dos planos infinitos paralelos. En este caso:

((T_a-T)/(T_a-T_b)) (_?(infinitos@paralelos)$^{p}lanos$) = $2/r\_m?\_(n = 1)^{?}??sen(B\_n)cos(B\_nr/r\_$
$sen(B\_n)cos?(B\_n))exp(? - B?\_n^2K?/(?C_pr_d^2))?$

donde la distancia entre los planos es $2r_d y B_n son las soluciones de mcot B = B. La temperatura residual del cilindro finito está dada entonces por$:

$((T_a - T)/(T_a - T_b))(_f inito^c ilindro) = ((T_a - T)/(T_a - T_b))(_i nfinito^c ilindro) x ((T_a - T)/(T_a - T_b))(_? (infinitos @ paralelos)^p lanos)$

Los parámetros son el coeficiente de transferencia de calor del vapor h_o, la conductividad térmica del puré K, el calor específico del puré C_p, la densidad ?, la temperatura ambiente T_ay la temperatura del vapor T_V, dados por:

h_o=670 Btu/(h x pie$^2 xF$)
K=0.48 Btu/(libra x F)
?=68 libras/pie^3
T_a=86 F T_v=240 F

El radio del cilindro es de 0.112 pies y la altura es de 0.167 pies. Las ecuaciones dadas antes se programan en el siguiente archivo-m:

```
% Este es el archivo Ejemplo9_25.m
% Este archivo calcula la temperatura en el eje de un cilindro de radio
% rcm conteniendo puré de chícharo y calentándose a una temperatura
% de Tcal grados Farenheir, la temperatura ambiente es Ta,
% el tiempo de calentamiento es teta_cal y el come up time es teta_come_up

clear
close all
clc

b1(1) = fsolve(@funcion,1);
b1(2) = fsolve(@funcion,5);
b1(3) = fsolve(@funcion,10);
b(1) = fsolve(@funcionbessel,1);
b(2) = fsolve(@funcionbessel,5);
b(3) = fsolve(@funcionbessel,10);

rm= 0.167; % radio del stub
rmc = 0.112;% radio del cilindro
%r = 0.0; %distancia del eje del cilindro
%
r = [0 : rmc/25 :rmc ];
K = 0.48; %conductividad térmica del slab
Ro = 68; % densidad
Cp = 0.91;% calor específico del puré
h = 670; %coeficiente de transferencia del vapor
teta_cal = 20; % minutos
teta_come_up = 5;
teta = (teta_cal + 0.42*teta_come_up)/60;
```

```
teta1 = [(0.42*teta_come)/60)/25 : teta ];

Ta = 86;% temperatura ambiente
Tcal = 240;%temperatura del vapor
jj = [1:3];
T = 0;
Tc = 0;

% CÁLCULOS PARA r VARIABLE DESDE EL CENTRO DEL CILINDRO
% HASTA LA SUPERFICIE.
% cálculos para el stub
for kk = 1:length(r);
T = 0;
Ts = 0;

for n= 1 :length(jj) ;
Ts = 2*sin(b1(n))*cos(b1(n)*r(kk)/rm)/
(b1(n)+sin(b1(n))*cos(b1(n)));
Ts = Ts*exp(-b1(n) ^ 2 * K * teta/(ro * Cp * rm^2));
T = T + Ts;
%pause
end

TR(kk) = T;
end
%for ij = 1:
%nn = n/rmc;
%cálculos para el cilindro infinito
for j = 1:length(r);
Tc = 0;
for n = 1:lenght(jj);
Tcs = 2*besselj(0,b(n)*r(j)/rmc);
Tcs = Tcs*exp((-b(n)^2) * K * teta/(ro * Cp * rmc^2));
```

Bibliography

[1] ...

[2] ...