ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA

TESIS DE GRADO

(ANEXOS)

"ANÁLISIS Y APLICACIÓN DE SISTEMAS LINEALES UTILIZANDO MATLAB"

TESIS PREVIA LA OBTENCIÓN DEL TÍTULO DE INGENIERA EN ELECTRÓNICA Y CONTROL

INÉS DEL CARMEN JURADO MORENO

JULIO, 1998

ANEXO A

MANUAL DEL USUARIO

ANEXO A

MANUAL DEL USUARIO

El *Toolbox para Análisis de Sistemas Lineales* fue escrito en el lenguaje del programa computacional *Matlab con Simulink Versión 4.2.* Por lo tanto su ambiente de trabajo es de Windows. Consiste de un grupo de archivos con extensión ***. m***, los cuales se los debe invocar desde el Matlab. Este manual del usuario es una guía a través de las diferentes pantallas de menúes. Explicando de una manera sencilla la operación del programa.

a.1 REQUERIMIENTOS DEL SISTEMA

- Sistema Operativo Microsoft Windows 3.1, Microsoft Windows 95 o Windows NT 4.0.
- Computador personal basado en procesador Intel 486, Pentium, Pentium Pro, o Pentium II.
- Unidad de disco flexible de 3 ½ pulgadas.
- MegaBytes de memoria RAM mínimo, 16 MegaBytes (o más) recomendable.
- Matlab para Windows Versión 4.2 y Simulink Versión 1.3
- Adaptador gráfico de 8 bits y monitor VGA mínimo, Adaptador gráfico y monitor SVGA recomendable.
- Copiar SISLIN en el toolbox del Matlab: c:\MATLAB\TOOLBOX\Sislin
- Modificar el path del Matlab, para que conste el toolbox implementado SISLIN.
- Se requiere la tarjeta LAB-PC1200 para realizar la adquisición de datos.

a.2 PROGRAMA PRINCIPAL

Invocando el archivo ***menup.m*** se inicia la ejecución del programa que permite el fácil manejo de los archivos ***. m***. La estructura general del programa se puede observar en la figura A.1.

La pantalla principal del programa se observa en la figura A.2. Desde esta pantalla se invocan los menúes para trabajar en el dominio del tiempo, en el dominio de la frecuencia o para trabajar con la aplicaciones. Estas pantallas se pueden observar en las figuras A.3, A.4, y A.5 respectivamente.

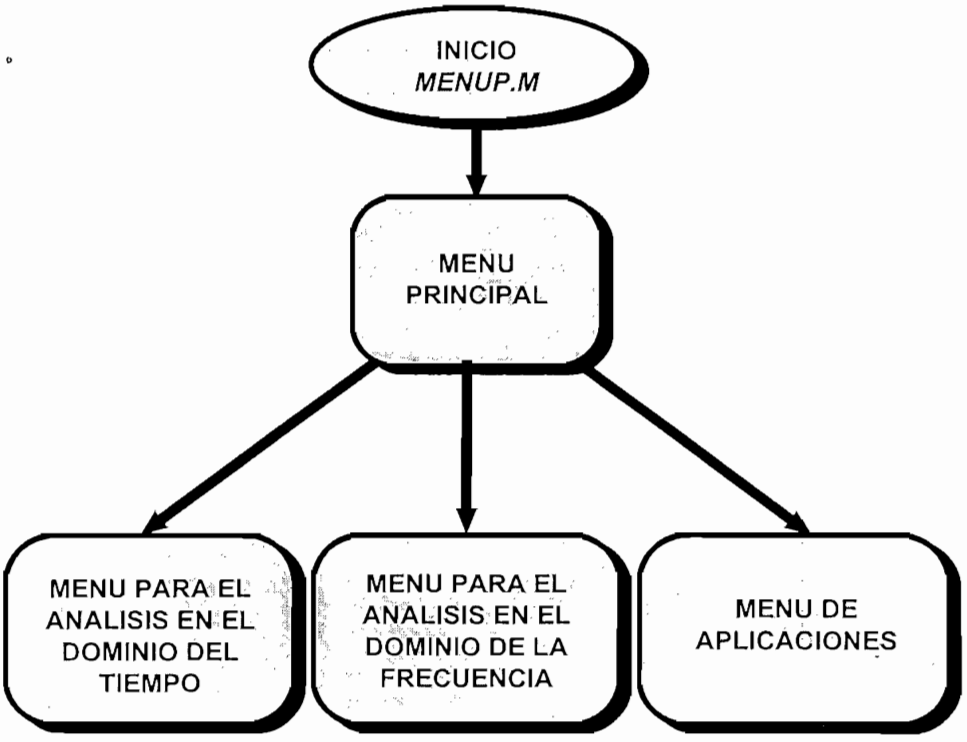


Figura A.1. Estructura del programa.

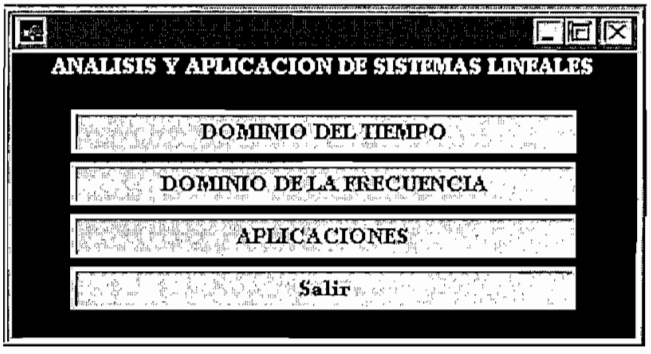


Figura A.2. Menú Principal

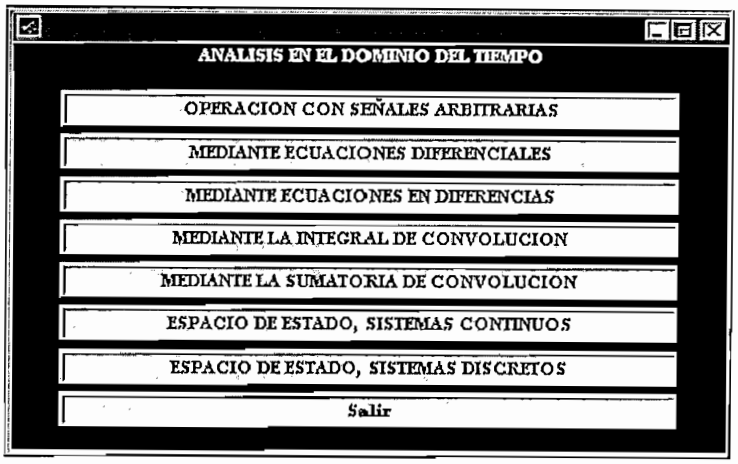


Figura A.3. Menú para acceder al análisis en el dominio del tiempo.

Figura A.4. Menú para acceder al análisis en el dominio de la frecuencia.

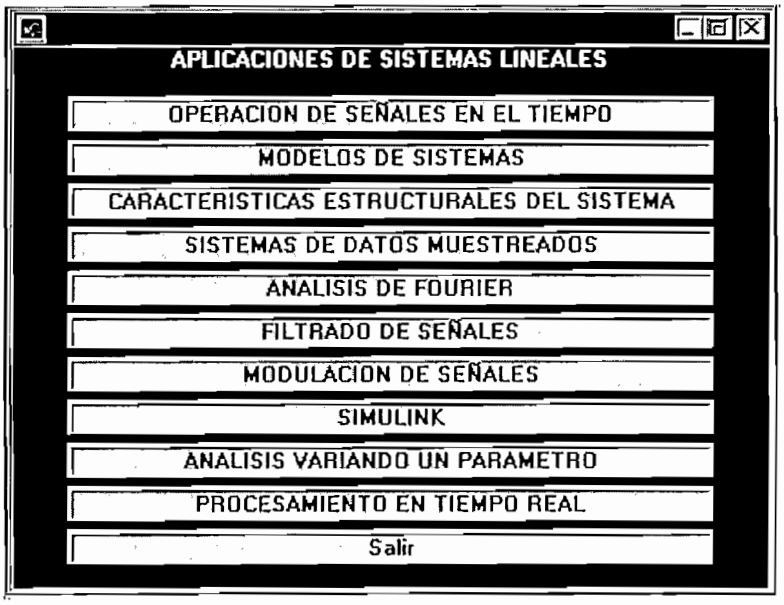


Figura A.5. Menú para acceder a las aplicaciones.

a.3 ANÁLISIS EN EL DOMINIO DEL TIEMPO

Este menú permite accesar a los diferentes tipos de análisis que se pueden realizar en el dominio del tiempo. Una estructura de este menú de opciones se observa en la figura A.6.

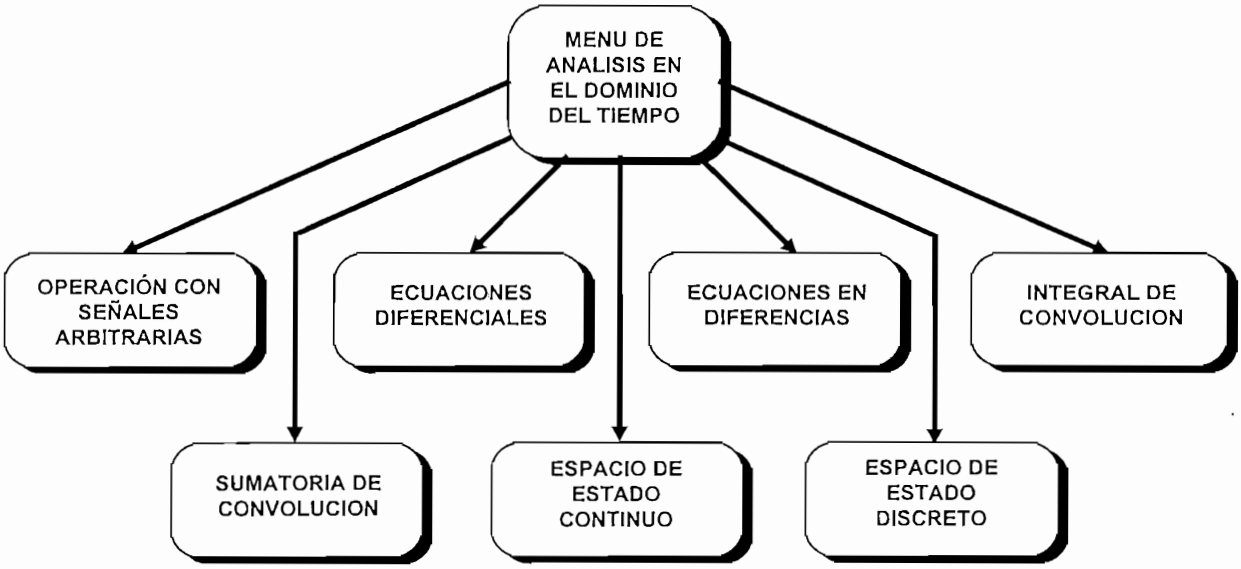


Figura A.6. Estructura del menú de análisis en el dominio del tiempo.

a.3.1. Operaciones con señales arbitrarias

Al ingresar a la opción para realizar las operaciones con señales arbitrarias se despliega el menú de la figura A.7.

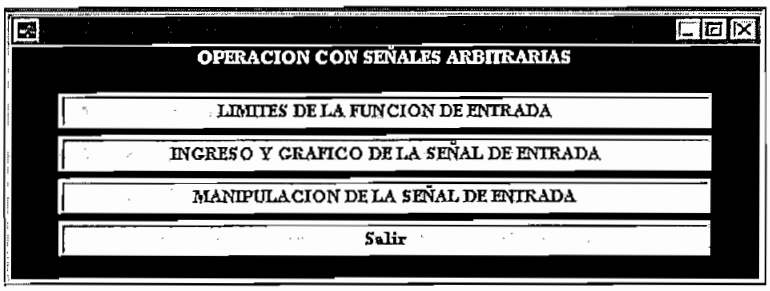


Figura A.7. Menú de operaciones con señales arbitrarias

Para realizar cualquier manipulación de la entrada se debe en primer lugar, ingresar los límites de la señal. Luego, definir la señal. Y por último, se pueden realizar las operaciones que se deseen (Ver figura A.8).

Figura A.8. Menú de manipulación de señales

Para realizar el desplazamiento de la señal se debe especificar si el desplazamiento es hacia la izquierda o a la derecha, y de cuántas unidades es el mismo.

El escalamiento puede ser en tiempo o en magnitud. La definición de este escalamiento se lo realiza siguiendo las instrucciones de la figura A.9.

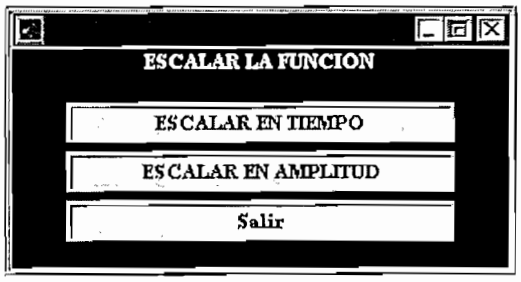


Figura A.9. Opciones para escalar la señal

a.3.2 Mediante ecuaciones diferenciales

Al ingresar a la opción de ecuaciones diferenciales se muestra el siguiente menú :

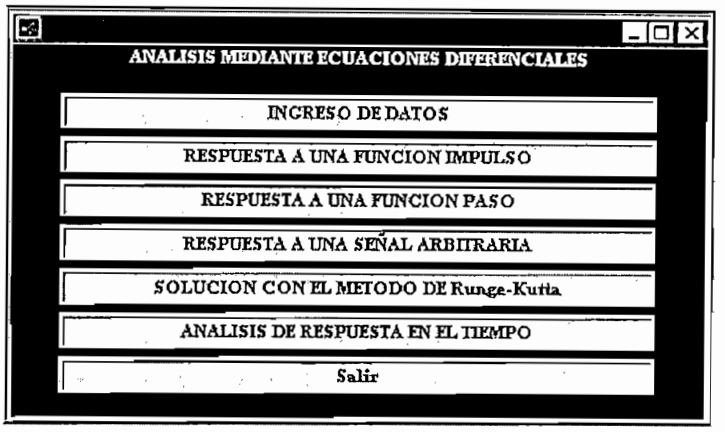


Figura A.10. Menú de análisis mediante ecuaciones diferenciales

En primer lugar se debe realizar el ingreso de datos con la opción del mismo nombre. Una vez ingresada la ecuación se puede obtener la respuesta del sistema a diferentes tipos de entradas. La señal arbitraria se ingresa de manera muy similar al caso del numeral A.3.1.

También se puede resolver la ecuación diferencial con el Método de Runge - Kutta como se muestra en la figura A.11.

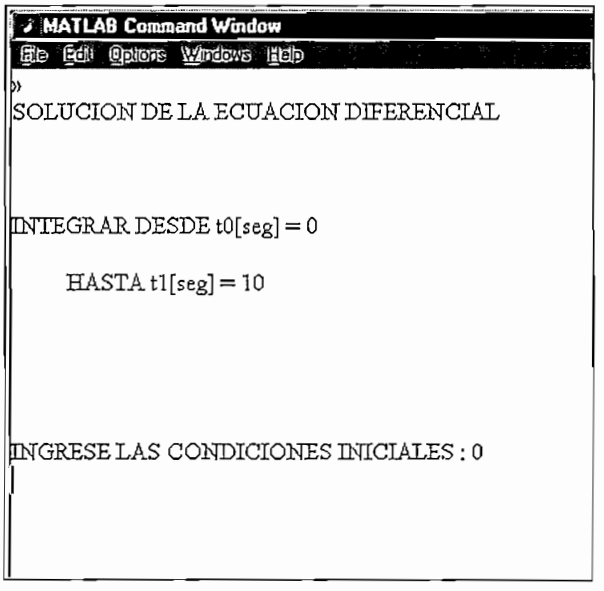


Figura A.11. Ingreso del intervalo de integración en el Método de Runge Kutta

a.3.3 Mediante ecuaciones en diferencias

Al ingresar a la opción de ecuaciones en diferencias se tiene el siguiente menú :

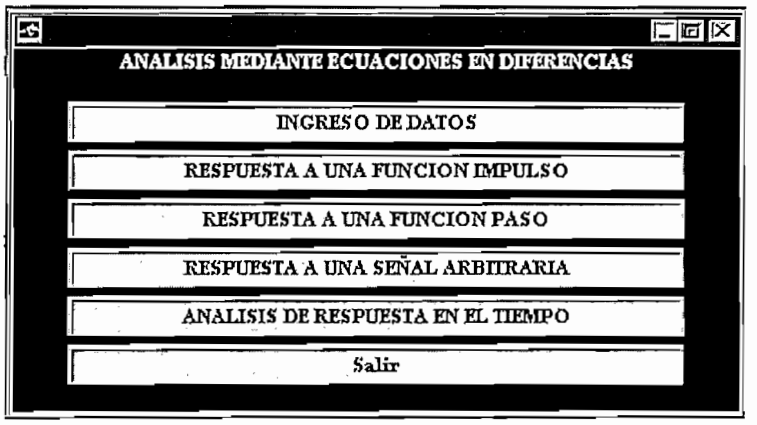


Figura A.12. Menú de análisis mediante ecuaciones en diferencias.

En primer lugar, se ingresan los coeficientes ecuación en diferencias. Luego, se pueden encontrar la respuesta en el tiempo a diferentes tipos de señales de entrada. Las señales arbitrarias se ingresan de forma similar al numeral A.3.1.

a.3.4 Mediante la integral de convolución

Para realizar el análisis utilizando la integral de convolución se utiliza el siguiente menú :

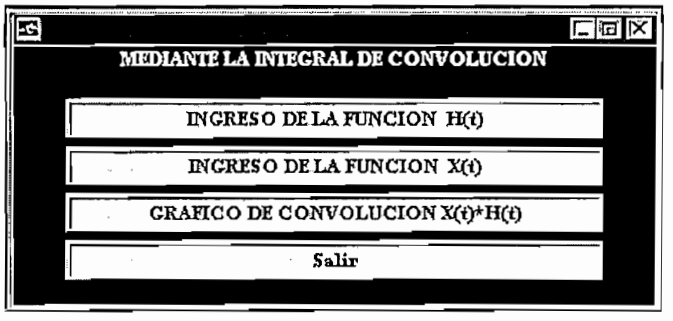


Figura A.13. Menú de análisis mediante la integral de convolución.

Se definen las funciones x(t) y h(t) en las dos primeras opciones del menú anterior, y luego, con la última se encuentra la respuesta de este análisis en el tiempo.

a.3.5 Mediante la sumatoria de convolución

Como se puede observar en la figura A.14, este menú es muy similar al anterior, con la diferencia de que se debe definir primero los límites extremos de las señales x[n] y h[n].

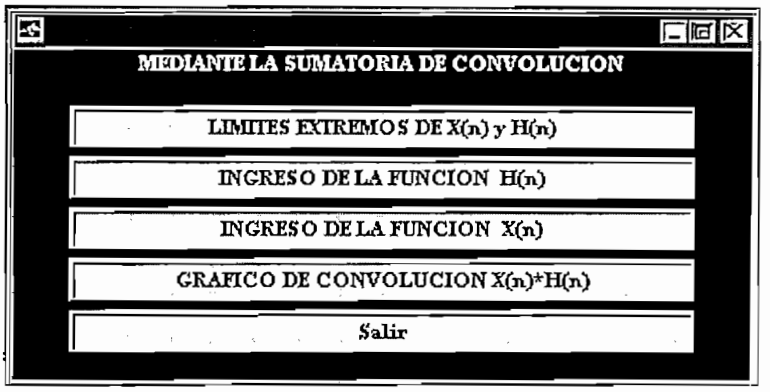


Figura A.14. Menú de análisis mediante la sumatoria de convolución.

a.3.6 Mediante el espacio de estado para sistemas continuos

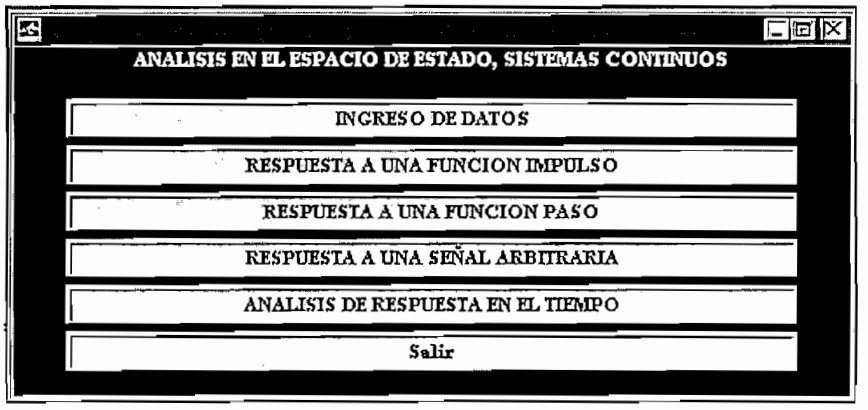


Figura A.15. Menú de análisis en el espacio de estado continuo.

Al igual que en los numerales anteriores, se debe ingresar primero el sistema motivo de análisis. En este caso, se ingresan las matrices A, B, C y D. Luego, se puede encontrar la respuesta en el tiempo a señales de entrada como la impulso, la paso o en general señales arbitrarias ingresadas por el usuario. Una vez definidos el sistema y la entrada se visualiza la respuesta gráficamente. Este menú, permite también, el analizar la respuesta en el tiempo del sistema.

a.3.7 Mediante el espacio de estado para sistemas discretos

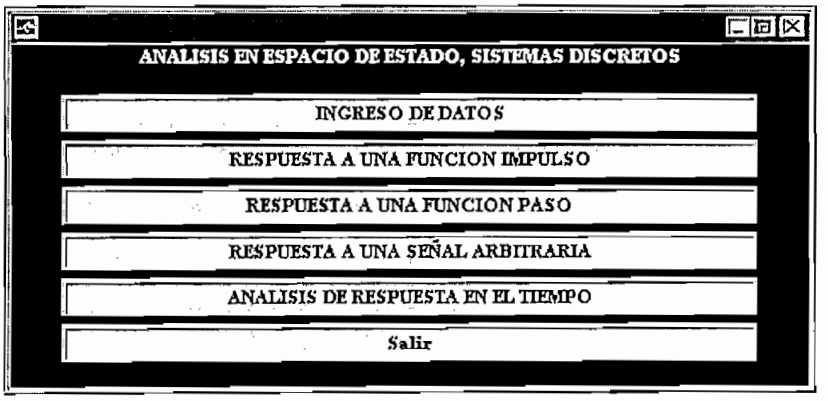


Figura A.16. Menú de análisis en el espacio de estado discreto.

Similar al menú del numeral anterior, se ingresan las matrices Ad, Bd, Cd y Dd. Los demás respuestas son las misma paro en el dominio discreto.

a.4 ANÁLISIS EN EL DOMINIO DE LA FRECUENCIA

Este menú al igual que en el numeral anterior permite tener un acceso muy sencillo a las diferentes opciones de análisis en el dominio de la frecuencia. La figura A.17 muestra un esquema general de la estructura de este menú de opciones.

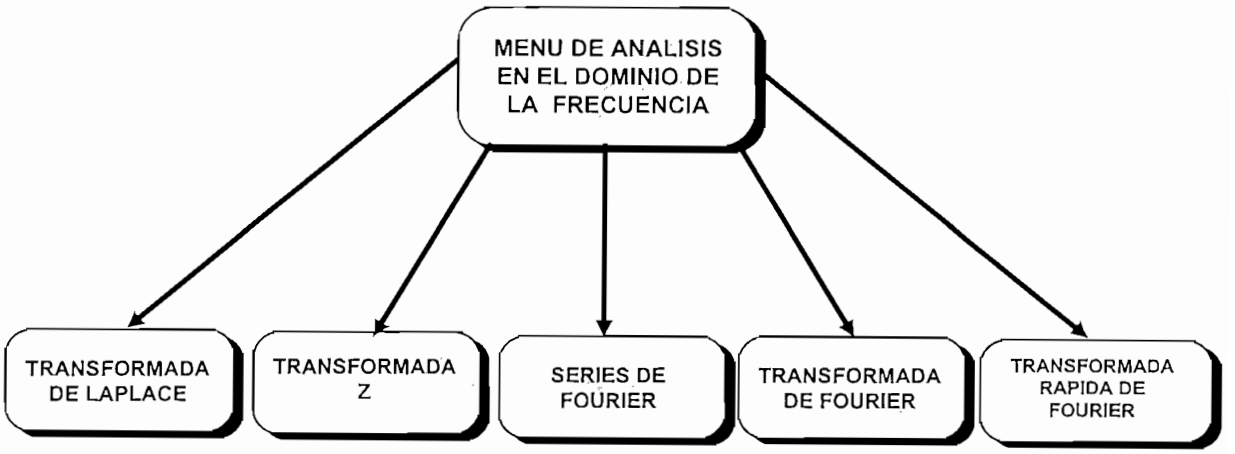


Figura A.17. Estructura del menú de análisis en el dominio de la frecuencia.

a.4.1 Mediante la transformada de Laplace

Figura A.18. Menú de análisis mediante la transformada de Laplace.

Se ingresan los coeficientes del numerador y del denominador de la función de transferencia del sistema. Con estos datos se puede encontrar la respuesta temporal del sistema a diferentes señales de entrada. También graficar el lugar geométrico de las raíces y el análisis en el plano s.

a.4.2 Mediante la transformada Z

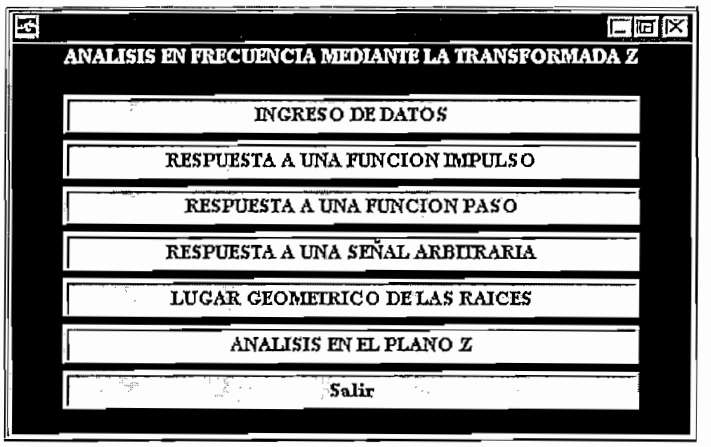


Figura A.19. Menú de análisis mediante la transformada Z.

El procedimiento para la utilización de esta opción de análisis es idéntico al del numeral anterior.

a.4.3 Mediante series de Fourier

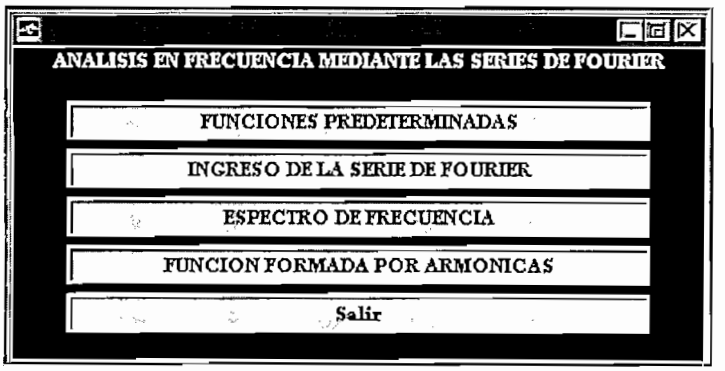


Figura A.20. Menú de análisis mediante series de Fourier.

Para el ingreso de la señal a ser analizada se puede escoger entre varias funciones predeterminadas, o ingresar la función por parte del usuario a través del editor de texto del D.O.S. modificando el archivo ***anf3.m***. Este análisis permite

observar el espectro de frecuencia de la función y la función conformada por armónicas.

a.4.4 Mediante la transformada de Fourier

Figura A.21. Menú de análisis en frecuencia mediante la transformada de Fourier.

En primer lugar, se debe ingresar la función a analizar. Esto se lo hace, ya sea escogiendo una función predeterminada o definiendo una función por parte del usuario modificando el archivo ***tff3.m*** con el editor de texto del D.O.S. Una vez ingresada la función se puede visualizar el espectro de frecuencia o analizar la respuesta de frecuencia de la función por medio de las opciones del menú que se muestra a continuación :

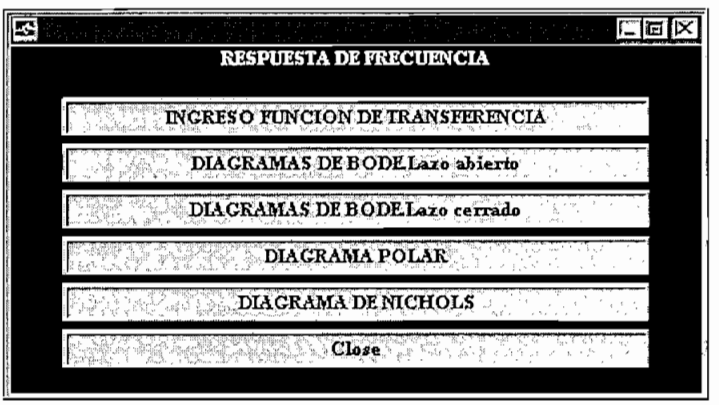


Figura A.22. Menú de análisis de la respuesta de frecuencia.

a.4.5 Mediante la transformada rápida de Fourier

Figura A.23. Menú de análisis en frecuencia utilizando transformada rápida de Fourier.

Al igual que en el numeral anterior, se puede escoger entre funciones predeterminadas o el ingreso por parte del usuario de una función arbitraria. Para la definición de esta función se modifica el archivo ***fftf5.m*** .Una vez que se ha ingresado la función para analizar se puede observar el espectro de frecuencia de la función utilizando la transformada rápida de Fourier.

a.5 APLICACIONES DE LOS SISTEMAS LINEALES

En el menú de las aplicaciones de los sistemas lineales se utilizan rutinas para el análisis, tanto en el dominio del tiempo, como en el dominio de la frecuencia. Estas son aplicadas para la resolución de los temas más característicos relacionados con los sistemas lineales. La figura A.24 muestra un esquema general de la estructura de este menú de opciones.

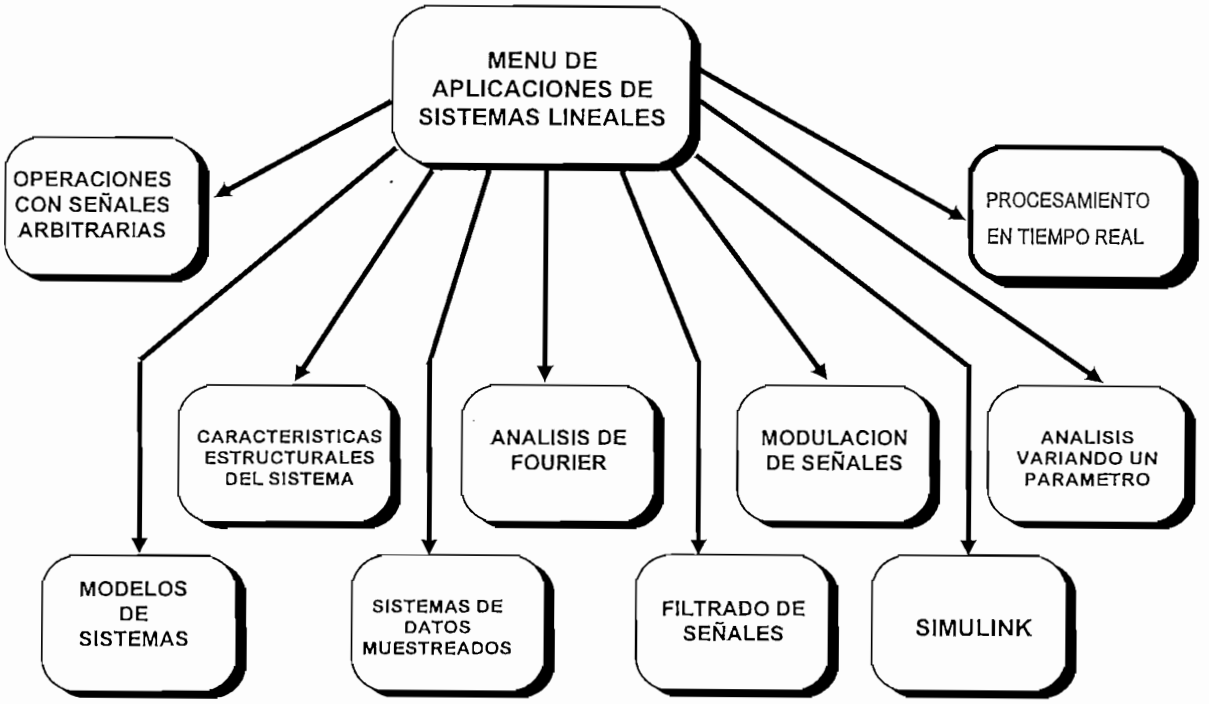


Figura A.24 Estructura del menú de aplicaciones de los sistemas lineales

a.5.1 Operaciones con señales arbitrarias

Esta aplicación es similar al la opción del numeral A.3.1. Difiere en que las operaciones son acumulativas. Debido a esto se pueden construir señales que son una combinación de todas las opciones disponibles.

a.5.2 Modelos de sistemas

La aplicación de modelos de sistemas tiene múltiples opciones para realizar la modelación de sistemas. Es posible modelar sistemas en el dominio de la frecuencia con funciones de transferencia, o en el espacio de estado. En la figura A.25 se encuentra una representación gráfica de las diferentes opciones disponibles.

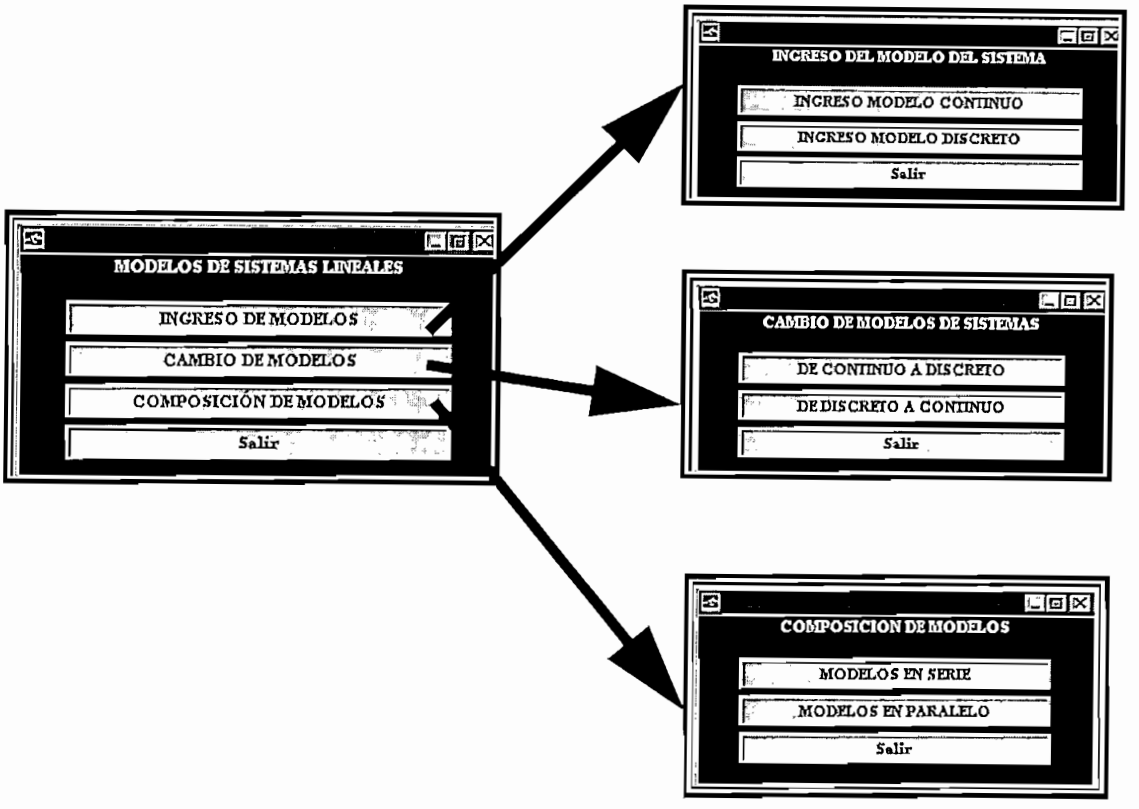


Figura A.25 Estructura del menú de la aplicación modelos de sistemas lineales

a.5.3 Caracteristicas estructurales del sistema

Luego de ingresar el sistema ya sea como función de transferencia o en el espacio de estado, se obtienen las diferentes características del sistema. Estas son : estabilidad, controlabilidad, observabilidad y la transformación de semejanza.

En la figura A.26 se observa el menú del análisis de estabilidad del sistema.

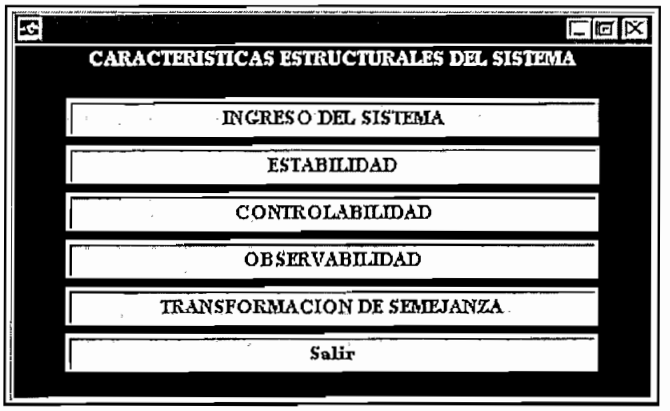


Figura A.26 Menú de la aplicación características estructurales del sistema.

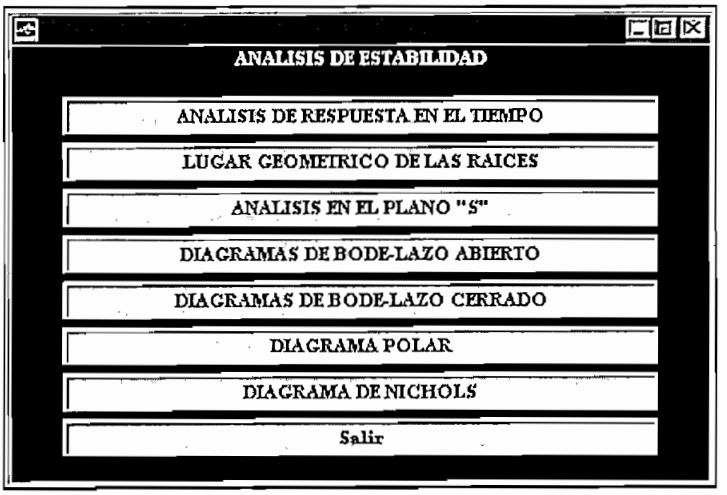


Figura A.27 Menú de análisis de estabilidad del sistema.

a.5.4 Sistemas de datos muestreados

La aplicación de sistemas de datos muestreados permite ingresar un sistema en tiempo continuo, discretizar el sistema, y encontrar la respuesta en el tiempo y las características estructurales del sistema. La estructura de los menúes de opciones se la puede observar en la figura A.28.

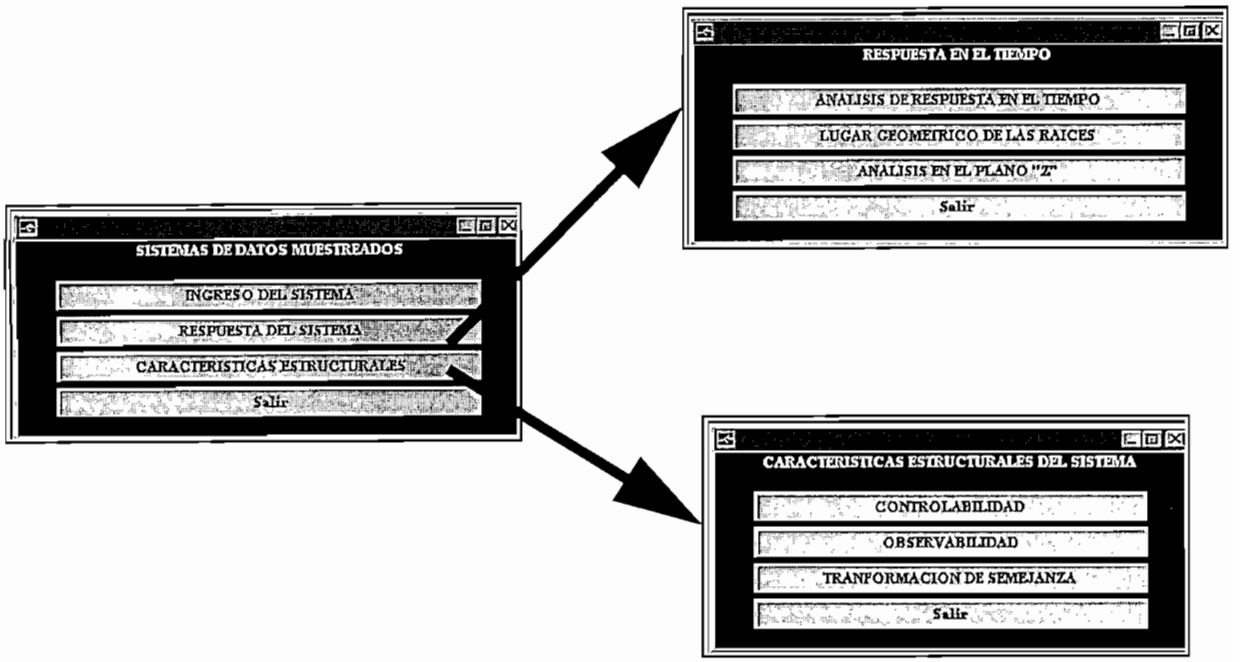


Figura A.28. Estructura del menú de la aplicación de sistemas de datos muestreados.

a.5.5 Análisis de Fourier

En primer lugar, se debe ingresar la función a analizar. Esto se lo hace definiendo una función por parte del usuario modificando el archivo ***ifunap.m*** con el editor de texto del D.O.S. Una vez ingresada la función se puede visualizar el espectro de frecuencia de la señal con y sin ruido y la señal con ruido.

Figura A.29. Menú de la aplicación de análisis de Fourier.

a.5.6 Filtrado de señales

En la aplicación de filtrado de señales, se ingresa primeramente una señal modificando el archivo ***ifilap.m*** de la misma manera que en el numeral anterior. Luego se procede a filtrar la señal con los diferentes tipos de filtros disponibles.

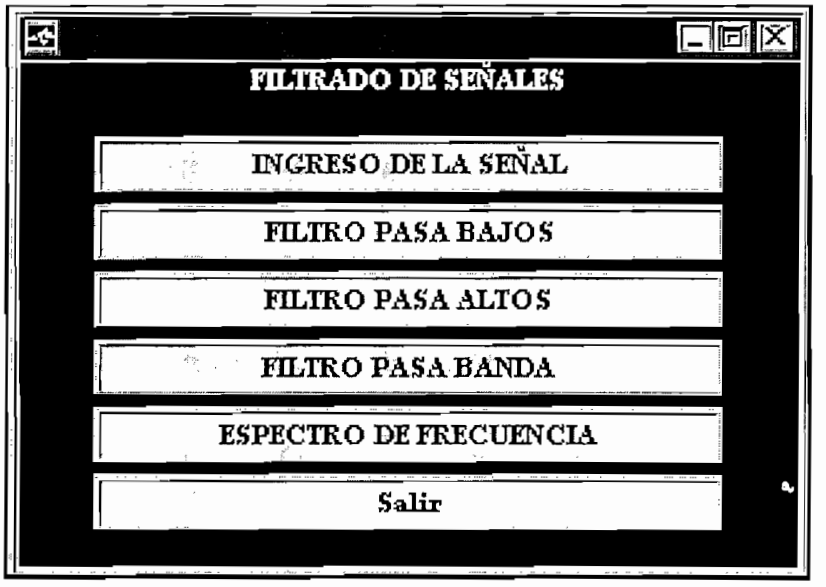


Figura A.30. Menú de la aplicación de filtrado de señales.

a.5.7 Modulación de señales

En esta aplicación, en primer lugar hay que escoger el tipo de modulación, luego se ingresan las señales en el tiempo a través de los archivos auxiliares. Para A.M. el archivo ***timap.m***, para F.M. el archivo ***tff3fp.m*** y para P.M. el archivo ***tff3ap.m.***

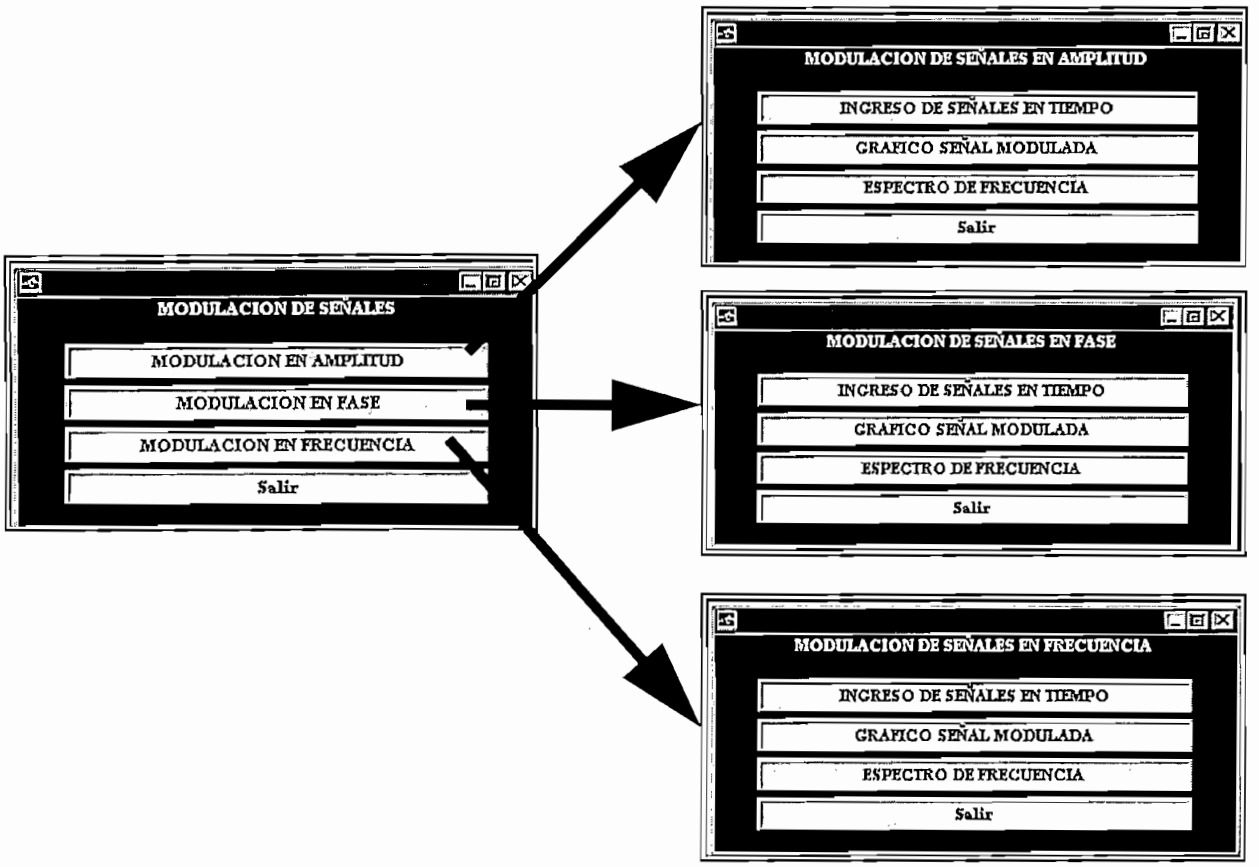


Figura A.31. Estructura del menú de opciones de la aplicación de modulación de señales.

a.5.8 Simulink

Esta aplicación invoca al programa Simulink Versión 1.3 para el ingreso del sistema. Por defecto abre el archivo auxiliar ***link1.m*** en el editor del Simulink, permitiendo hacer modificaciones al mismo.

En la visualización de la respuesta de la simulación existen barras de desplazamiento para ampliar o disminuir la máxima escala en eje del tiempo.

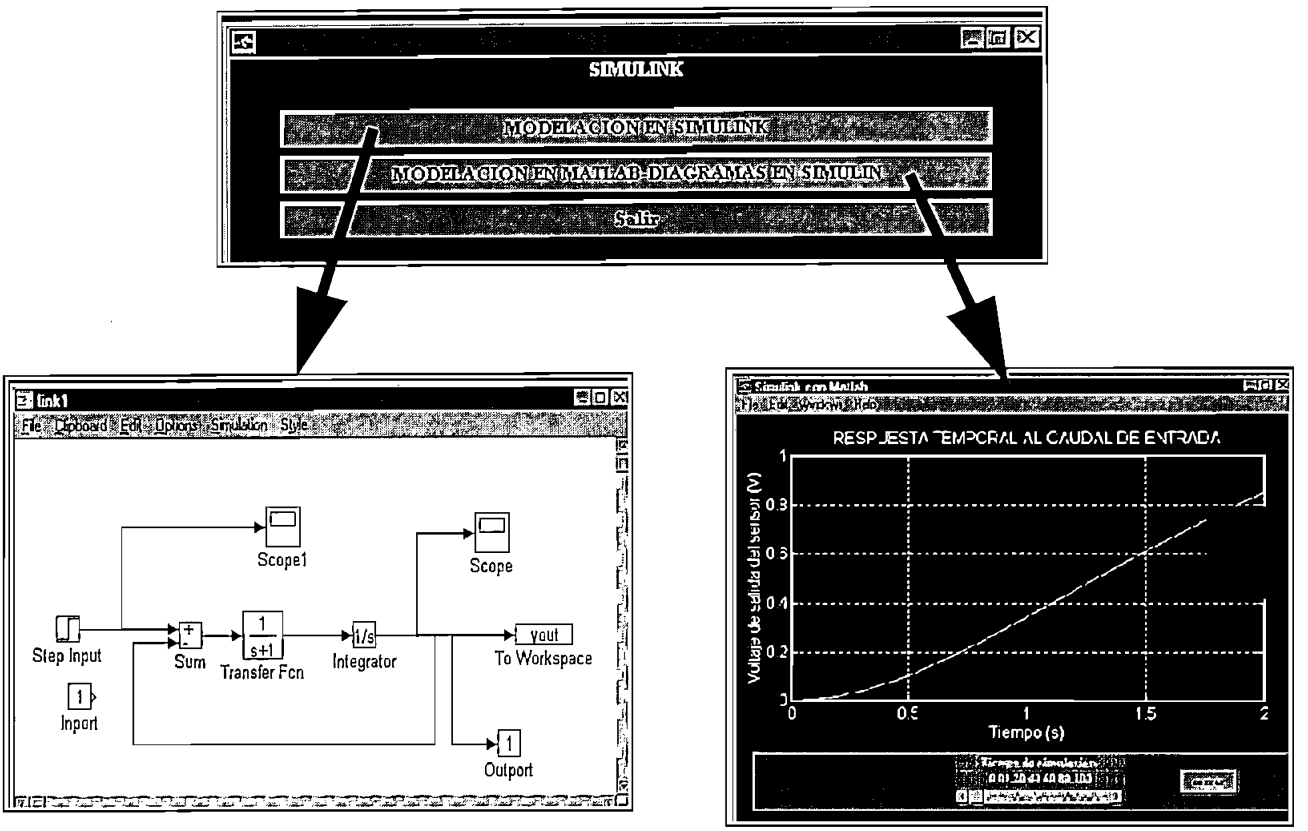


Figura A.32. Estructura del menú de opciones de la aplicación del Simulink.

a.5.9 Análisis variando un parámetro

Esta aplicación permite el analizar el comportamiento de un sistema lineal cuando uno de los coeficientes de la función de transferencia varia. Esta variación se la realiza utilizando barras de desplazamiento, las mismas que están incluidas en los gráficos de las respuestas.

Se realiza el análisis en el tiempo y el análisis en frecuencia de este sistema. Al variar el parámetro se recalcula la respuesta y se grafica la salida.

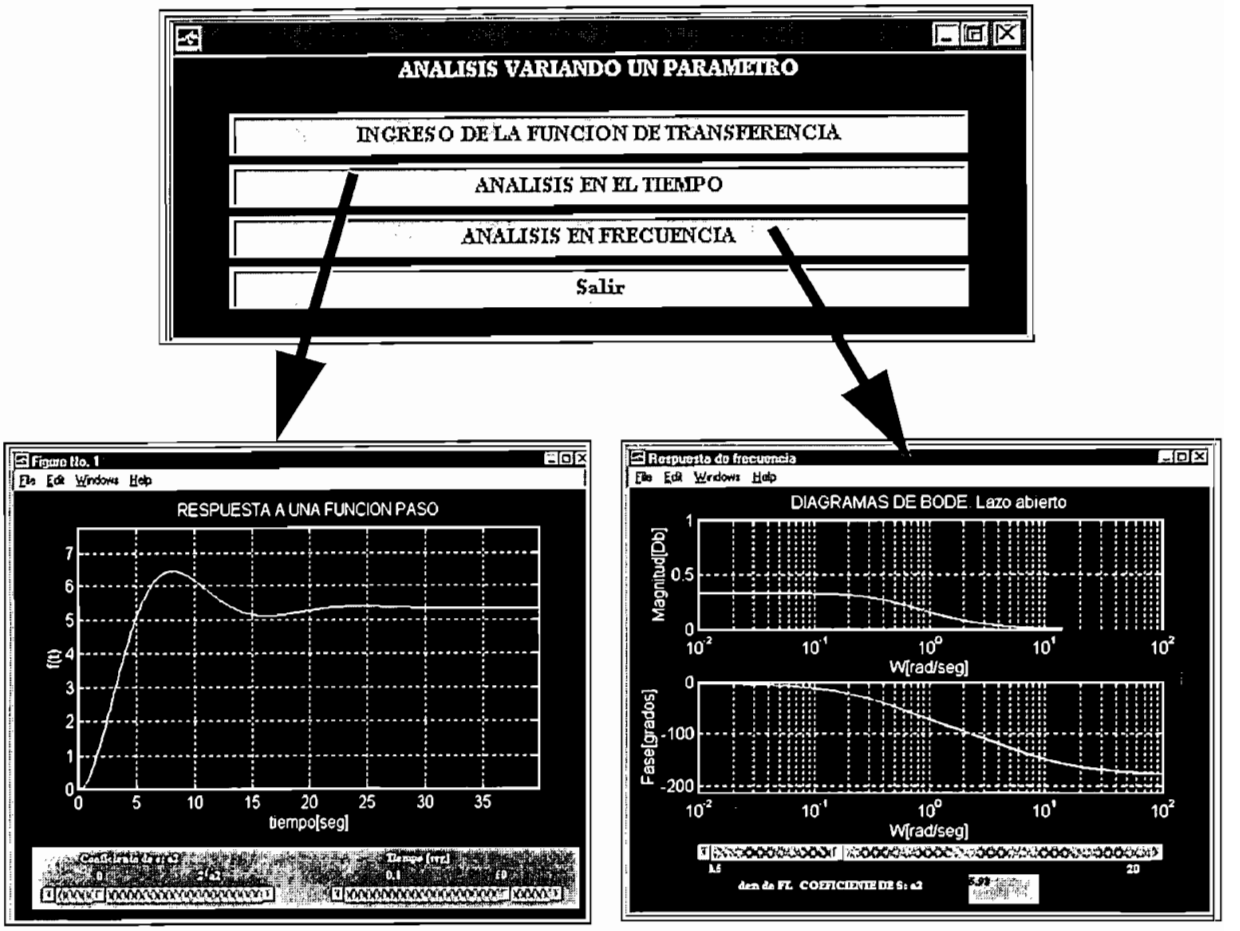


Figura A.33. Estructura del menú de aplicación de análisis variando un parámetro.

a.5.10. Procesamiento en tiempo real.

Esta aplicación permite realizar procesamiento en tiempo real de las señales muestreadas en un intervalo de tiempo. Para realizar la adquisición de datos se utiliza el programa ejecutable sislab.exe, que permite escoger la opción adquisición de datos y gráficos.

El menú de adquisición de datos y gráficos, permite adquirir datos de un sistema realimentado y/o de una señal arbitraria que es muestreada, como se observa en la figura A.34.

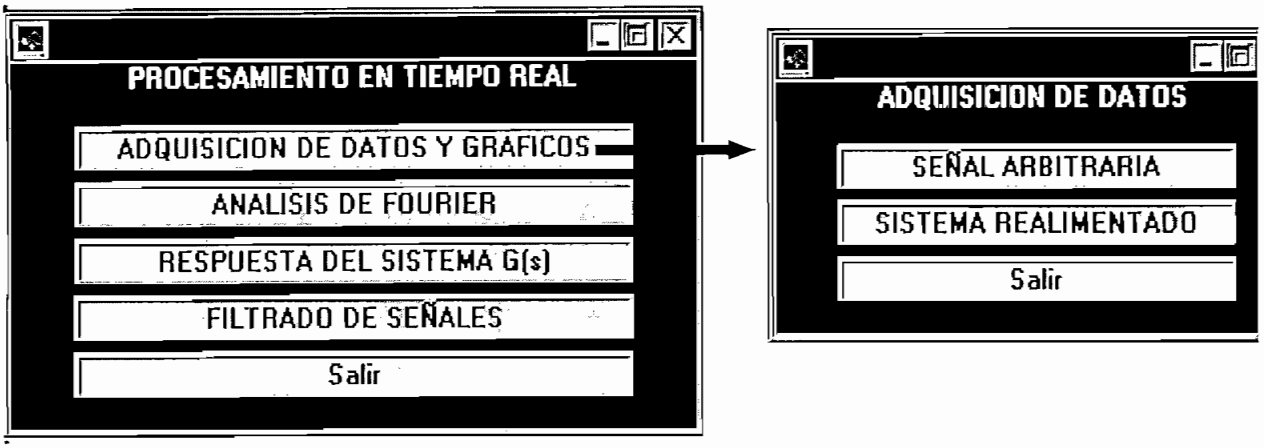


Figura. A.34. Estructura del menú procesamiento en tiempo real.

Para realizar la adquisición de datos de un sistema S1 realimentado se plantea el siguiente diagrama de bloques:

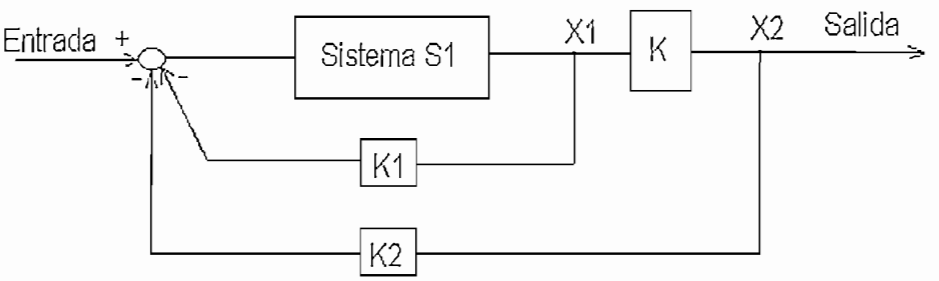


Figura. A.35. Disgrama de bloques de S1, realimentado.

En donde el sistema S1, es un sistema físico. K1 y K2 se manipulan a través del computador. La entrada del sistema S1, es una salida analógica manejada desde el computador.

En la figura A.36 se muestran los resultados obtenidos con la adquisición de datos.

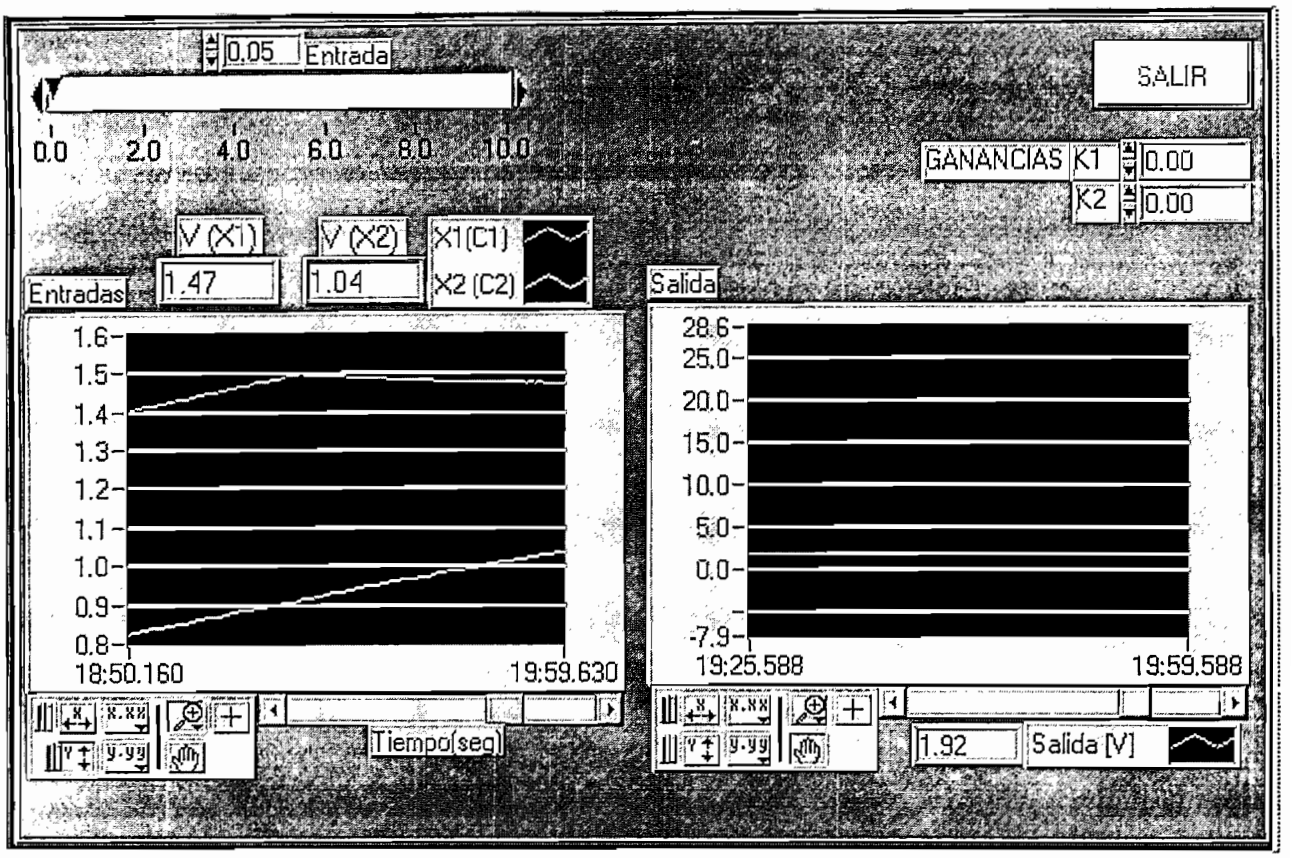


Figura. A.36. Adquisición de datos de un sistema realimentado

Las respuestas gráficas correspondientes a los otros submenúes, son similares a las obtenidas en los respectivos numerales de análisis de Fourier, respuesta a una señal arbitraria del sistema G(s) y filtrado de señales.

ANEXO B

LISTADO DE RUTINAS

B. ANÁLISIS Y APLICACIÓN DE SISTEMAS LINEALES UTILIZANDO MATLAB.

B.1. ANÁLISIS DE SISTEMAS LINEALES EN EL DOMINIO DEL TIEMPO.

OPERACIÓN CON SEÑALES ARBITRARIAS

ANÁLISIS MEDIANTE ECUACIONES DIFERENCIALES

ANÁLISIS MEDIANTE ECUACIONES EN DIFERENCIAS

ANÁLISIS MEDIANTE LA TRANSFORMADA Z.

ANÁLISIS MEDIANTE SERIES DE FOURIER

ANÁLISIS MEDIANTE LA TRANSFORMADA DE FOURIER

ANÁLISIS MEDIANTE LA TRANSFORMADA RÁPIDA DE FOURIER

ANEXO B.

LISTADO DE PROGRAMAS DE ANÁLISIS Y APLICACIÓN DE SISTEMAS LINEALES UTILIZANDO MATLAB (SISLIN)

Las rutinas implementadas, se presentan a continuación en el orden en el que se presentaron en los capítulos II, III y IV. En el anexo A.1, se muestran las pantallas de presentación de los menúes principales.

B. Análisis y Aplicación de Sistemas Lineales Utilizando Matlab.

La pantalla de presentación se implementa con la rutina menup.m

menup.m

```
function menup
close all
labels = str2mat(...
   'DOMINIO DEL TIEMPO', ...
   'DOMINIO DE LA FRECUENCIA', ...
   'APLICACIONES');
callbacks = [ ...
   'menu1        '
   'menu2        '
   'menu3        '];

choices1('menup','ANALISIS Y APLICACIÓN DE SISTEMAS LINEALES',
labels, callbacks);
```

Se utiliza la rutina choices1.m, para generar las pantallas interactivas con los botones que se definen con dicho comando. El botón de Salir se genera automáticamente con esta función. Función basada en choices.m de las funciones de Matlab 4.2.

choices1.m

```
function choices1(name,header,labels,callbacks,inter)
for
c = computer;
if ~isstr(name) | ~isstr(header) | ~isstr(labels) | ~isstr(callbacks)
  error('Requires string arguments.');
end
if nargin < 4
  error('Not enough input arguments.')
end
if nargin == 4
   inter = 0;
end
if inter
   yn = 'yes';
else
   yn = 'no';
end
uicok = strcmp(c(1:2),'PC') | strcmp(c(1:2),'MA');
if isunix | isvms | ~uicok
  uicok = strcmp(lower(get(0,'TerminalProtocol')),'x');
end
& VMS
if ~uicok
   labels = str2mat(labels,'Done');
   nl = size(labels,1);
  ss = deblank(labels(1,:));
   ss = ss(sort([1:length(ss) find(ss==' ')]));
   args = [' ',ss,' '];
   header = header(sort([1:length(header) find(header==' ')]));
   for i = 2:nl
      ss = deblank(labels(i,:));
      ss = ss(sort([1:length(ss) find(ss==' ')]));
```

```
      args = [args, ',''', ss,''''];
    end
    k = 1;
    while k > 0 & k < nl
      eval(['k = menu(''',header,''',', args,');']);
      if k == nl | k == 0
    return
      else
    ceval(callbacks(k,:));
      end
    end
    return
  end
  % can use uicontrols
  global CHOICELIST
  global CHOICEHANDLES
  name = deblank(name);
  if isempty(name)
   error('Requires non-blank string argument.')
  end
  figs = sort(get(0,'Children'));
  openfigs = size(figs);
  if ~isempty(figs)
    cf = gcf;
    if cf == 1
      cf = [];
    end
  else
    cf = [];
  end
  fig1 = 1;
  if isempty(figs)
   CHOICELIST = [];
```

```
  CHOICEHANDLES = [];
  figs = figure('visible','off');
  fig1 = 0;
end
if figs(1) ~= 1
  figs = [figure('visible','off'); figs];
  fig1 = 0;
end
matchn = 0;
for i = 1:size(CHOICELIST,1)
  if strcmp(name,deblank(CHOICELIST(i,:)))
      matchn = i;
      break;
  end
end
if ~matchn
  CHOICEHANDLES = [CHOICEHANDLES; 0];
  if isempty(CHOICELIST)
      CHOICELIST = name;
  else
      CHOICELIST = str2mat(CHOICELIST, name);
  end
  matchn = size(CHOICEHANDLES,1);
else
  matchh = 0;
  for i = 1:size(figs,1)
      if figs(i) == CHOICEHANDLES(matchn)
            matchh = i;
            break;
      end
  end
  if matchh
      figure(CHOICEHANDLES(matchn));
```

```
    return
  end
end
ss = get(0,'ScreenSize');
xedge = 30;
ybord = 30;
width = 30;
yedge = 35;
ha = axes('visible','off');
hh = text(.1,.1,str2mat(labels,'Salir'),'units','pixel');
maxwidth = 0;
height = 0;
for i = 1:length(hh),
  ext = get(hh(i),'extent');
  maxwidth = max(maxwidth,ext(3));
  height = max(height,ext(4));
end
delete(hh);delete(ha);
yedge = 1.5*height;
height = 6*yedge/7;
imax = 1;
twidth = maxwidth;
pixels
mwwidth = twidth + width + 2*xedge;
mwheight = (size(labels,1)+2.5)*yedge;
swidth = ss(3); sheight = ss(4);
left = 20;
bottom = sheight-mwheight-ybord;
rect = [left bottom mwwidth mwheight];
CHOICEHANDLES(matchn) = figure('Position',rect,'number','off', ...
    'name','','resize','off','colormap',[],'NextPlot','new',...
    'Menubar','none','visible','off');
fg = CHOICEHANDLES(matchn);
```

```
fgs = CHOICEHANDLES(CHOICEHANDLES ~= fg);
set(fgs,'visible','off')
set(gca,'Position',[0 0 1 1]); axis off;

% Place header

hdrpos = [.05 1-1/(size(labels,1)+1.6) .9 1/(size(labels,1)+1.6)];

hh=uicontrol(fg,'style','text','units','normal',...
      'position',hdrpos,'string',header,...
      'Horizontal','center');
set(hh,'backg',get(gcf,'color'),'foreg',[1 1 1]-get(gcf,'color'))
% set up pre-amble so figure 1 is available for rendering in
sb = ['figure(1),set(1,"visible","on");set(',int2str(fg),',"visible","off");'];
se                                    =                                    [';global
CHOICEHANDLES,set(CHOICEHANDLES(length(CHOICEHANDLES)),"vis
ible","on")'];
for ii=size(labels,1):-1:1
  i = size(labels,1) + 2 - ii;
  h1 = uicontrol(fg,'position',[xedge  (i-.5)*yedge width+twidth height],...
      'callback',[sb callbacks(ii,:) se],...
      'string',['  ',deblank(labels(ii,:)), '  '],...
      'HorizontalAlignment','left','interruptible',yn);
% left justify string inside button
end
% Create Salir button
uicontrol(fg,'position',[xedge .5*yedge width+twidth height],...
    'string','  Salir  ',...
    'callback',['choicex('',name,'')'],...
    'HorizontalAlignment','left');
set(fg,'visible','on')
if ~isempty(cf)
  figure(cf)
```

```
else
  if ~fig1
  close(1);
  end
end
```

B.1. ANÁLISIS DE SISTEMAS LINEALES EN EL DOMINIO DEL TIEMPO.

menu1.m

```
function menu1

labels = str2mat(...
  'OPERACION CON SEÑALES ARBITRARIAS', ...
  'MEDIANTE ECUACIONES DIFERENCIALES', ...
  'MEDIANTE ECUACIONES EN DIFERENCIAS', ...
  'MEDIANTE LA INTEGRAL DE CONVOLUCION', ...
  'MEDIANTE LA SUMATORIA DE CONVOLUCION', ...
  'ESPACIO DE ESTADO, SISTEMAS CONTINUOS', ...
  'ESPACIO DE ESTADO, SISTEMAS DISCRETOS');

callbacks = [ ...
   'entrada      '
   'diferel      '
   'diferen      '
   'iconvol      '
   'sconvol      '
   'econtin      '
   'ediscret     '];

choices1('DATA', 'ANALISIS EN EL DOMINIO DEL TIEMPO', labels,
callbacks);
```

OPERACIÓN CON SEÑALES ARBITRARIAS

entrada.m

```
function entrada
% SE REALIZA EL INGRESO Y GRAFICO DE LA SEÑAL DE ENTRADA
% SE REALIZA LA MANIPULACION DE LA SEÑAL DE ENTRADA
labels = str2mat(...
  'LIMITES DE LA FUNCION DE ENTRADA', ...
  'INGRESO Y GRAFICO DE LA SEÑAL DE ENTRADA', ...
  'MANIPULACION DE LA SEÑAL DE ENTRADA');
callbacks = [ ...
   'limita        '
   'arbit         '
   'manipula      '];

choices1('SEÑALES', 'OPERACION CON SEÑALES ARBITRARIAS',
labels, callbacks);
```

limita.m

```
%EL PROGRAMA VA A CONTENER:
%INGRESO DE LOS LIMITES PARA GRAFICAR LA FUNCION DE
ENTRADA
close
clc
disp(' ')
disp('LIMITES PARA MANEJAR LA FUNCION')
disp(' ')
disp(' ')
 li=input('LIMITE INFERIOR [seg] : ');
 disp(' ')
 ls=input('LIMITE SUPERIOR [seg] : ');
 t=-0.1+li:.1:ls+0.1;
end
```

arbit.m

```
%EL PROGRAMA VA A CONTENER:
%1.INGRESO DE LA FUNCION ARBITRARIA
%2.GRAFICO DE LA FUNCION ARBITRARIA

%1. PROGRAMA PARA GENERAR FUNCIONES ARBITRARIAS
close
disp(' INGRESE LA FUNCION  POR INTERVALOS')
xo=i;
clear f ft
for e=1:20,
disp(' ')
disp(' TIPO DE FUNCION A INGRESAR')
disp(' ')
disp(' 1. FUNCION PASO ')
disp(' 2. FUNCION RAMPA ')
disp(' 3. FUNCION SINUSOIDAL ')
disp(' 4. FUNCION EXPONENCIAL ')
disp(' 5. FINALIZAR')
in=input('REALIZAR: ');
if in==1
  disp(' FUNCION PASO')
  kp=input('INGRESE LA GANANCIA: ');
  to=input('INGRESE EL LIMITE INFERIOR DEL INTERVAL0 [seg]: ');
  f=kp*stepfun(t,to);
  else
  if in==2
          disp('FUNCION RAMPA')
          kr=input('INGRESE LA GANANCIA: ');
         tn=input('INGRESE EL LIMITE INFERIOR DEL INTERVAL0 [seg]: ');
          f=kr*(t-tn).*stepfun(t,tn);
          else
          if in==3
```

```
disp(' FUNCION SINUSOIDAL')
disp(' f(t)=k.sin(w.t) ')
ks=input('INGRESE k = ');
tos=input('INGRESE  w [rad/seg] = ');
disp(' ')
                tos=input('INGRESE EL  LIMITE INFERIOR DEL
                                                INTERVAL0 [seg] = ');
[nft,nct]=size(t);
t2=0:.1:nct;
f1=zeros(1,(tos-li)*10);
f2=ks*sin(t2);
[nft,nct]=size(t);
fx=[f1,f2];
f=fx(:,1:nct);
else
if in==4
        disp(' FUNCION EXPONENCIAL')
        disp(' f(t) = k.e^(-at) ')
        ke=input('INGRESE k: ');
        a=input('INGRESE a: ');
                toe=input('INGRESE EL LIMITE INFERIOR DEL
                                                INTERVAL0: ');
        [nft,nct]=size(t);
        t2=0:.1:nct;
        f1=zeros(1,(toe-li)*10);
        f2=ke*exp(-(a*t2));
        fx=[f1,f2];
        f=fx(:,1:nct);
        else
if in==5
e=20;
% FUNCION ARBITRARIA EN FORMA DE VECTOR t y rt
%2. GRAFICO DE LA FUNCION ARBITRARIA
```

```
                    rt=ft;
                    save t
                    save rt
                    plot(t,rt),title('FUNCION ARBITRARIA')
                        axis([1.2*(-0.5+min(t)) 1.2*(0.5+max(t)) 1.2*(-0.5+min(rt))
                                                                  1.2*(0.5+max(rt))])
                    xlabel('tiempo[seg]')
                    ylabel('X(t)')
                    grid
                    pause
                    break
                                else
                                disp('ERROR FUERA DE RANGO')
                    end
            end
      end
          end
    end
    if e==1
      ft=f;
    else
      if e<20
            ft=ft+f;
      end
    end
    end
```

manipula.m

```
    function manipula
    % SE REALIZA LA MANIPULACION DE LA SEÑAL
    %COMO DESPLAZAMIENTO, TRANSPOSICION
    labels = str2mat(...
      'DESPLAZAMIENTO', ...
```

```
  'TRANSPOSICION', ...
  'ESCALAMIENTO');
callbacks = [ ...
   'desplaz      '
   'transpon     '
   'escalar      '];
choices1('OPERACIONES', 'MANIPULACION DE SEÑALES', labels,
callbacks);
```

desplaz.m

```
%EL PROGRAMA VA A CONTENER:
%UNA FUNCION ARBITRARIA QUE SE VA A DESPLAZAR
%EN EL TIEMPO, HACIA LA DERECHA O HACIA LA IZQUIERDA
close
clc
disp('  DESPLAZAMIENTO DE SEÑALES ')
disp('  t-a: DESPLAZA a UNIDADES A LA DERECHA')
disp('  t+a: DESPLAZA a UNIDADES A LA IZQUIERDA')
disp(' ')
disp('   DESPALAZAR HACIA ? ')
disp('   1. LA IZQUIERDA ')
disp('   2. LA DERECHA ')
in=input('   REALIZAR:');
un1=input('CUANTAS UNIDADES ? ');
un=abs(un1);
if in==1
  d=-un;
  else
  if in==2
  d=un;
          else
                disp('ERROR FUERA DE RANGO')
                return
```

```
  end
end
[md,nd]=size(rt);
matriz=d*ones(1,nd);
t1=t+matriz;
  subplot(2,1,1)
  plot(t,rt),title('FUNCION ARBITRARIA * DESPLAZAMIENTO')
  axis([min(min(t),min(t1))  max(max(t),max(t1)) 1.2*min(rt) 1.2*max(rt)])
  xlabel('tiempo[seg]')
  ylabel('X(t)')
  grid
  subplot(2,1,2)
  plot(t1,rt)
  axis([min(min(t),min(t1))  max(max(t),max(t1)) 1.2*min(rt) 1.2*max(rt)])
  xlabel('tiempo[seg]')
  ylabel('Xdesplazada(t)')
  grid
  pause
end
```

transpon.m

```
%EL PROGRAMA VA A CONTENER:
%UNA FUNCION ARBITRARIA QUE SE VA A TRANSPONER EN EL
%TIEMPO
close
clc
t1=-t;
r1=(rot90(rt))';
  subplot(2,1,1)
  plot(t,rt),title('FUNCION ARBITRARIA * TRANSPOSICION')
  axis([min(min(t),min(t1))  max(max(t),max(t1)) 1.2*min(rt) 1.2*max(rt)])
  xlabel('tiempo[seg]')
  ylabel('X(t)')
```

```
  grid
  subplot(2,1,2)
  plot(t1,rt)
  axis([min(min(t),min(t1))  max(max(t),max(t1)) 1.2*min(rt) 1.2*max(rt)])
  xlabel('tiempo[seg]')
  ylabel('Xtranspuesta(t)')
  grid
  pause
end
```

escalar.m

```
function escalar
labels = str2mat(...
  'ESCALAR EN TIEMPO', ...
  'ESCALAR EN AMPLITUD');
callbacks = [ ...
   'escalart      '
   'escalara      '];
choices1('ESCALAMIENTO', 'ESCALAR LA FUNCION', labels, callbacks);
```

escalart.m

```
%EL PROGRAMA VA A CONTENER:
%UNA FUNCION ARBITRARIA QUE SE VA A ESCALAR EN TIEMPO
close
clc
disp('  ESCALAR LA FUNCION EN TIEMPO ')
disp(' ')
disp('  x(at)')
est1=input('CUANTAS UNIDADES a ? ');
est=abs(est1);
rt1=rt;
t1=t./est;
subplot(2,1,1)
```

```
plot(t,rt),title('FUNCION ARBITRARIA * ESCALAMIENTO')
axis([min(min(t),min(t1))      max(max(t),max(t1))    1.2*min(min(rt),min(rt1))
1.2*max(max(rt),max(rt1))])
xlabel('tiempo[seg]')
ylabel('X(t)')
grid
subplot(2,1,2)
plot(t1,rt1)
axis([min(min(t),min(t1))      max(max(t),max(t1))    1.2*min(min(rt),min(rt1))
1.2*max(max(rt),max(rt1))])
xlabel('tiempo[seg]')
ylabel('Xtiempo(t)')
grid
pause
end
```

escalara.m

```
%EL PROGRAMA VA A CONTENER:
%UNA FUNCION ARBITRARIA QUE SE VA A ESCALAR EN %AMPLITUD
close
clc
disp('  ESCALAR LA FUNCION EN AMPLITUD ')
disp(' ')
disp('  c.x(t)  ')
es1=input('CUANTAS UNIDADES c ? ');
es=abs(es1);
t1=t;
rt1=rt.*es;
subplot(2,1,1)
plot(t,rt),title('FUNCION ARBITRARIA * ESCALAMIENTO')
        axis([min(min(t),min(t1))  max(max(t),max(t1)) 1.2*min(min(rt),min(rt1))
                                                    1.2*max(max(rt),max(rt1))])
xlabel('tiempo[seg]')
```

```
ylabel('X(t)')
grid
subplot(2,1,2)
plot(t1,rt1)
        axis([min(min(t),min(t1))  max(max(t),max(t1)) 1.2*min(min(rt),min(rt1))
                                                        1.2*max(max(rt),max(rt1))])
xlabel('tiempo[seg]')
ylabel('Xamplitud(t)')
grid
pause
end
```

ANÁLISIS MEDIANTE ECUACIONES DIFERENCIALES

diferel.m

```
function diferel
labels = str2mat(...
  'INGRESO DE DATOS', ...
  'RESPUESTA A UNA FUNCION IMPULSO', ...
  'RESPUESTA A UNA FUNCION PASO', ...
  'RESPUESTA A UNA SEÑAL ARBITRARIA', ...
  'SOLUCION CON EL METODO DE Runge-Kutta', ...
  'ANALISIS DE RESPUESTA EN EL TIEMPO');
callbacks = [ ...
   'ingreso1      '
   'impulso1      '
   'paso1         '
   'arbit1        '
   'ode1          '
   'resp1         '];
choices1('EC.DIFERENCIALES', 'ANALISIS MEDIANTE ECUACIONES
DIFERENCIALES', labels, callbacks);
```

ingreso1.m

```
echo off
close
clc
disp(' ')
disp('    INGRESO DE DATOS ')
disp(' ')
disp('    TODOS LOS COEFICIENTES QUE INGRESE ')
disp('        SE HARAN EN ORDEN DESCENDENTE. Ej:  ')
disp(' ')
disp('        a2.Y´´(t)+a1.Y´(t)+a0.Y(t)=b2.X´´(t)+b1.X´(t)+b0.X(t)')
disp(' ')
disp('        A=[a2 a1 a0]')
disp(' ')
disp('    B=[b2 b1 b0]')
disp(' ')
disp('        CI=[Y´(0) Y(0)]')
disp(' ')
disp(' INGRESE LOS COEFICIENTES DE Y COMO EL VECTOR A')
p=input('        A = ');
disp(' ')
disp(' INGRESE LOS COEFICIENTES DE X COMO EL VECTOR B')
q=input('        B = ');
disp(' ')
disp('INGRESE LOS COEFICIENTES DE C.I. COMO EL VECTOR CI')
i=input('        CI = ');
disp(' ')
% LAS MATRICES a,b,c,d QUE SE OBTIENEN ESTAN EN LA %FORMA
CANONICA CONTROLABLE
[a,b,c,d] = tf2ss(q,p);
save a
save b
```

```
save c
save d
save i
save t
end
```

impulso1.m

```
%RESPUESTA A UNA FUNCION IMPULSO
clg
clc
close(gcf)
disp(' CUAL ES EL TIEMPO MAXIMO PARA VISUALIZAR ? ')
tii=input('t[seg] = ');
t=0:.1:tii;
if i==0
  y=impulse(a,b,c,d,1,t);
else
  y=initial(a,b,c,d,i,t);
end
plot(t,y),title('RESPUESTA A UNA FUNCION IMPULSO')
axis([min(t) max(t) 1.1*(-0.2+min(y)) 1.1*(0.2+max(y))])
xlabel('tiempo[seg]')
ylabel('f(t)')
grid
pause
end
```

paso1.m

```
% ESTE PROGRAMA PERMITE GRAFICAR LA RESPUESTA A UNA
%FUNCION PASO
clc
disp(' ')
close(gcf)
```

```
disp(' CUAL ES EL TIEMPO MAXIMO PARA VISUALIZAR ? ')
disp(' ')
tii=input('t[seg] = ');
t=0:.1:tii;
xo=i;
if i==0
  y=step(a,b,c,d,1,t);
else
  y=lsim(a,b,c,d,ones(1,length(t)),t,xo);
end
figure
hold on
plot(t,y),title('RESPUESTA A UNA FUNCION PASO')
axis([min(t) max(t) 1.1*(-0.2+min(0,min(y))) 1.1*(0.2+max(y))])
xlabel('tiempo[seg]')
ylabel('f(t)')
grid
pause
end
```

arbit1.m

```
%EL PROGRAMA VA A CONTENER:
%1.INGRESO DE LA FUNCION ARBITRARIA
%2.GRAFICO DE LA FUNCION ARBITRARIA
%3.RESPUESTA DE LA SEÑAL ARBITRARIA
close
%1. PROGRAMA PARA GENERAR FUNCIONES ARBITRARIAS
clc
disp(' ')
disp(' INGRESE LA FUNCION ARBITRARIA POR INTERVALOS')
disp(' ')
disp(' CUAL ES EL TIEMPO MAXIMO PARA VISUALIZAR ? ')
tm=input('t[seg] = ');
```

```
disp(' ')
xo=i;
clear f ft t y rt
for e=1:20,
disp(' TIPO DE FUNCION A INGRESAR')
disp(' ')
disp(' 1. FUNCION PASO ')
disp(' 2. FUNCION RAMPA ')
disp(' 3. FUNCION SINUSOIDAL ')
disp(' 4. FUNCION EXPONENCIAL ')
disp(' 5. FINALIZAR')
disp(' ')
in=input('REALIZAR: ');
disp(' ')
disp(' ')
if in==1
  disp(' ')
  disp(' FUNCION PASO')
  disp(' ')
  kp=input('INGRESE LA GANANCIA: ');
  disp(' ')
  to=input('INGRESE EL LIMITE INFERIOR DEL INTERVAL0 [seg]: ');
  t=0:.1:tm;
  if to==0
          to=0.0001;
   end
  f=kp*stepfun(t,to);
  else
  if in==2
  disp('FUNCION RAMPA')
  disp(' ')
  kr=input('INGRESE LA GANANCIA: ');
  disp(' ')
```

```
tn=input('INGRESE EL LIMITE INFERIOR DEL INTERVAL0 [seg]: ');
if tn==0
        tn=0.0001;
end
t=0:.1:tm;
f=kr*(t-tn).*stepfun(t,tn);
else
if in==3
        disp(' ')
        disp(' FUNCION SINUSOIDAL')
        disp(' ')
        disp(' f(t)=k.sin(w.t) ')
        disp(' ')
        ks=input('INGRESE k = ');
        disp(' ')
        tos=input('INGRESE  w [rad/seg] = ');
        disp(' ')
    tos=input('INGRESE EL  LIMITE INFERIOR DEL INTERVAL0 [seg] = ');
        if tos==0
        tos=0.0001;
        end
        [nft,nct]=size(t);
        t2=0:.1:nct;
        f1=zeros(1,(tos-li)*10);
        f2=ks*sin(t2);
        [nft,nct]=size(t);
        fx=[f1,f2];
        f=fx(:,1:nct);
                else
                if in==4
                disp(' FUNCION EXPONENCIAL')
                disp(' ')
                disp(' f(t) = k.e^(-at) ')
```

```
        disp(' ')
        disp(' ')
        ke=input('INGRESE k: ');
        disp(' ')
        a=input('INGRESE a: ');
        disp(' ')
    toe=input('INGRESE EL LIMITE INFERIOR DEL INTERVAL0: ');
        if toe==0
            toe=0.0001;
        end
        disp(' ')
        [nft,nct]=size(t);
        t2=0:.1:nct;
        f1=zeros(1,(toe-li)*10);
        f2=ke*exp(-(a*t2));
        fx=[f1,f2];
        f=fx(:,1:nct);

            else
            if in==5
                e=20;
        % FUNCION ARBITRARIA EN FORMA DE VECTOR t y rt
        %2. GRAFICO DE LA FUNCION ARBITRARIA
        rt=ft;
        subplot(121)
        plot(t,rt),title('FUNCION ARBITRARIA ')
        axis([min(t)        max(t)        1.1*(-0.2+min(0,min(rt)))
1.1*(0.2+max(rt))])
        xlabel('tiempo[seg]')
        ylabel('f(t)')
        grid

        %3. RESPUESTA A LA FUNCION ARBITRARIA
```

```
                    if xo==0
                            y=lsim(a,b,c,d,rt,t);
                    else
                            y=lsim(a,b,c,d,rt,t,xo);
                    end
                    subplot(122)
                    hold on
                    plot(t,y,'r'),title('RESPUESTA DE LA FUNCION ')
                    axis([min(t) max(t) -0.5+1.1*min(0,min(y)) 0.5+1.1*max(y)])
                    xlabel('X(t)')
                    ylabel('f(x)')
                    grid
                    pause
                    break
                            else
                            disp('ERROR FUERA DE RANGO')
            end
    end
  end
  end
  end
  if e==1
    ft=f;
    else
    if e<20
            ft=ft+f;
    end
  end
  end
  end
  end
```

ode1.m

```
%INTEGRACION DE ECUACIONES DIFERENCIALES CON EL METODO
%DE RUNGE KUTTA
echo off
global u
clc
!edit lotka1.m
clc
disp(' ')
disp('SOLUCION DE LA ECUACION DIFERENCIAL ')
disp(' ')
disp(' ')
close(gcf)
t0=input('INTEGRAR DESDE t0[seg] = ');
tf=input('        HASTA t1[seg] = ');
save t0 tf
disp(' ')
y0 =input('INGRESE LAS CONDICIONES INICIALES : ');
if y0==0
  y0=[0 0 ]';
end
[t,x] = ode23('lotka1',t0,tf,y0);
x1=x*[1;0];
x2=x*[0;1];
save x1
save x2
!edit lotka2.m
lotka2
load y
plot(t,y), title('SOLUCION DE LA ECUACION DIFERENCIAL')
axis([min(t) max(t) -0.01+min(min(y),0) 0.01+max(y)])
grid
xlabel('t[seg]')
```

```
ylabel('y(t)')
pause
end
```

lotka1.m

```
function xp = lotka1(t,x)
%La ecuación diferencial y''+3y'+2y=x''+2x'+x en variables de estado:
%       x1'=x2
%       x2' = -2x1-3x2+u
%     entrada escal¢n unitaria u=1, entrada rampa u=t
%      Se representa as¡:
%       u=1
%       xp(1)=x(2);
%       xp(2)=-2*x(1)-3*x(2)+u;
%
%       Las ecuaciones son:
u=1;
xp(1) = x(2);
xp(2) = -2*x(1)-3*x(2)+u;
%       PONGA GUARDAR Y SALIR
```

lotka2.m

```
function lotka2(t,y)
load x1
load x2
%ECUACION DE SALIDA EN VARIABLES DE ESTADO PARA EL
%SISTEMA
%       y=-x1-x2+u
%     u=entrada ; u=t rampa; u=1 escal¢n
%     La ecuaci¢n es:
u=1;
y=-x1-x2+u;
PONGA GUARDAR Y SALIR
```

```
save y
```

resp1.m

```
clc
close(gcf)
disp('ANALISIS DE LA RESPUESTA A UNA FUNCION PASO')
disp(' ')
disp(' ')
[num,den]=ss2tf(a,b,c,d);
%PARA EVALUAR EL POLINOMIO EN 0
%SE OBTIENE EL VALOR DE LA FUNCION EN ESTADO ESTABLE
%POLYVAL(F,S) EVALUA LA FUNCION F EN EL PUNTO S
disp('VALOR FINAL DE LA FUNCION')
  vf=polyval(num,0)/polyval(den,0)
%RESPUESTA A UNA FUNCION PASO
[y,x,t]=step(num,den);
%DEL VECTOR y QUE SE OBTIENE COMO RESPUESTA A LA
FUNCION %PASO Y ES EL VALOR MAXIMO DE y - k ES EL NUMERO
DE COLUMNA %EN EL QUE SE PRODUCE EL VALOR MAXIMO DE y
[Y,k]=max(y);
%PORCENTAJE DEL MAXIMO SOBREIMPULSO
disp('PORCENTAJE DE MAXIMO SOBREIMPULSO')
Mp=100*(Y-vf)/vf;
if Mp<0
  Mp=0
else
  Mp=Mp
%EL TIEMPO EN EL QUE SE PRODUCE EL MAXIMO SOBREIMPULSO
%ES EL DELVALOR K
  disp('TIEMPO PICO [seg]')
  tpico=t(k)
end
%n=1;
```

```
%        while y(n)<0.1*vf,n=n+1;
%        end
%m=1;
%        while y(m)<0.9*vf,m=m+1;
%        end
%disp('TIEMPO DE RESPUESTA DEL SISTEMA [seg]')
%tresp=t(m)-t(n)
l=length(t);
  while(y(l)>.98*vf)&(y(l)<1.02*vf);
  l=l-1;
  end
disp('TIEMPO DE ESTABLECIMIENTO DEL SISTEMA [seg]')
  ts=t(l)
pause
end
```

ANÁLISIS MEDIANTE ECUACIONES EN DIFERENCIAS

diferen.m

```
function diferen
labels = str2mat(...
   'INGRESO DE DATOS', ...
   'RESPUESTA A UNA FUNCION IMPULSO', ...
   'RESPUESTA A UNA FUNCION PASO', ...
   'RESPUESTA A UNA SEÑAL ARBITRARIA', ...
   'ANALISIS DE RESPUESTA EN EL TIEMPO');
callbacks = [ ...
    'ingreso2      '
    'impulso2      '
    'paso2         '
    'arbit2        '
    'resp2         '];
choices1('EC.EN-DIFERENCIAS', 'ANALISIS MEDIANTE ECUACIONES
EN DIFERENCIAS', labels, callbacks);
```

ingreso2.m

```
close
echo off
clear
clc
disp('      INGRESO DE DATOS ')
disp(' ')
disp('      ECUACIONES EN DIFERENCIAS')
disp('         METODO RECURSIVO. Ej  ')
disp(' ')
disp('         Y(n)+a1.Y(n-1)+a2.Y(n-2)=bo.X(n)+b1.X(n-1)')
disp(' ')
disp('         A=[a1 a2]  ')
disp(' ')
disp('         B=[bo b1]  ');
disp(' ')
disp(' ')
disp('        INGRESE LOS COEFICIENTES DE Y(n):')
A=input('         A=  ');
disp('      INGRESE LOS COEFICIENTES DE X(n):')
B=input('         B= ');
p=[1,A]
q=B
% LAS MATRICES a,b,c,d QUE SE OBTIENEN ESTAN EN LA FORMA
% CANONICA CONTROLABLE
[a,b,c,d] = tf2ss(q,p)
t=0:1:50;
end
```

impulso2.m

```
%RESPUESTA A UNA FUNCION IMPULSO
clc
clg
```

```
close(gcf)
disp(' ')
disp('RESPUESTA A UNA FUNCION IMPULSO')
disp(' ')
wn=input('MAXIMO VALOR DE n PARA VISUALIZAR ? ');
y=dimpulse(a,b,c,d,1,wn);
t=0:1:(wn-1);
yn=y(1:wn,:);
[ts,yns]=stairs(t,yn);
figure
plot(ts,yns),title('RESPUESTA A UNA FUNCION IMPULSO')
axis([min(ts) max(ts) min(yns) 1.2*max(yns)])
xlabel('[n]')
ylabel('f(n)')
grid
pause
end
```

paso2.m

```
% ESTE PROGRAMA PERMITE GRAFICAR LA RESPUESTA A UNA
%FUNCION PASO
clc
close(gcf)
disp('RESPUESTA A UNA FUNCION PASO')
disp(' ')
w=input('MAXIMO n PARA GRAFICAR ? ');
y=dstep(a,b,c,d,1,w);
t=0:1:w-1;
[ts,ynps]=stairs(t,y);
figure
plot(ts,ynps),title('RESPUESTA A UNA FUNCION PASO')
axis([min(ts) max(ts) min(ynps) 1.2*max(ynps)])
xlabel('[n]')
```

```
ylabel('f(n)')
grid
pause
end
```

arbit2.m

```
%EL PROGRAMA VA A CONTENER:
%1.INGRESO DE LA FUNCION ARBITRARIA
%2.GRAFICO DE LA FUNCION ARBITRARIA
%3.RESPUESTA DE LA SEÑAL ARBITRARIA
close(gcf)
%1. PROGRAMA PARA GENERAR FUNCIONES ARBITRARIAS
clc
disp(' ')
disp(' INGRESE LA FUNCION ARBITRARIA POR INTERVALOS')
disp(' ')
disp(' CUAL ES EL VALOR DE n MAXIMO PARA VISUALIZAR ? ')
tm1=input('n = ');
tm=tm1+1;
t=0:1:tm;
disp(' ')
xo=i;
clear f ft
for e=1:20,
disp(' TIPO DE FUNCION A INGRESAR')
disp(' ')
disp(' 1. FUNCION PASO ')
disp(' 2. FUNCION RAMPA ')
disp(' 3. FUNCION SINUSOIDAL ')
disp(' 4. FUNCION EXPONENCIAL ')
disp(' 5. FINALIZAR')
in=input('REALIZAR: ');
 if in==1
```

```
disp(' FUNCION PASO')
 disp(' ')
 kp=input('INGRESE LA GANANCIA: ');
 to=input('INGRESE EL LIMITE INFERIOR DEL INTERVAL0: ');
 t=0:1:tm;
 f=kp*stepfun(t,to);
 else
 if in==2
        disp('FUNCION RAMPA')
        kr=input('INGRESE LA GANANCIA: ');
        tn=input('INGRESE EL LIMITE INFERIOR DEL INTERVAL0: ');
        t=0:1:tm;
        f=kr*(t-tn).*stepfun(t,tn);
        else
                if in==3
                disp(' FUNCION SINUSOIDAL')
                disp(' ')
                ks=input('INGRESE LA GANANCIA: ');
        tos=input('INGRESE EL LIMITE INFERIOR DEL INTERVAL0: ');
                [nft,nct]=size(t);
                t2=0:1:nct;
                %t1=0:1:(tos-1)
                %*t2=tos:1:tm;
                f1=zeros(1,tos);
                f2=ks*sin(t2);
                fx=[f1,f2];
                f=fx(:,1:nct)
                else
                if in==4
                        disp(' FUNCION EXPONENCIAL')
                        disp(' ')
                        ke=input('INGRESE LA GANANCIA: ');
        toe=input('INGRESE EL LIMITE INFERIOR DEL INTERVAL0: ');
```

```
[nft,nct]=size(t);
t2=0:1:nct;
f1=zeros(1,toe);
f2=ke*exp(-t2);
fx=[f1,f2];
f=fx(:,1:nct)
else
if in==5
        e=20;
        [n,m]=size(ft);
        for k=1:n,
                for j=1:m,
                if k==1
                        r(k,j)=ft(k,j);
                else
                        r(k,j)=ft(k,j)+r((k-1),j);
                end
                end
        end
% FUNCION ARBITRARIA EN FORMA DE VECTOR t y rt
%2. GRAFICO DE LA FUNCION ARBITRARIA
        rt=r(n,:);
        subplot(121)
        [ts,rts]=stairs(t,rt);
plot(ts,rts),title('FUNCION ARBITRARIA')
axis([min(ts) max(ts) 1.2*min(0,min(rts)) 1.2*max(rts)])
xlabel('tiempo[seg]')
ylabel('f(t)')
grid
%3. RESPUESTA A LA FUNCION ARBITRARIA
y=dlsim(a,b,c,d,rt);
[w1,w2]=size(y);
if w1==1
```

```
                        w=w2;
                    else
                        w=w1;
                    end
                    for k=1:w-1,
                        yn(k)=y(k+1);
                    end
                    ynp=yn(1:w-1,:);
                    [wn1,wn2]=size(ynp);
                    if wn1==1
                        wn=wn2;
                    else
                        wn=wn1;
                    end
                    tnp=0:1:(w-2);
                    subplot(122)
                [tnps,ynps]=stairs(tnp,ynp);
                plot(tnps,ynps),title('RESPUESTA DE LA FUNCION ')
                axis([min(tnps) max(tnps) 1.2*min(ynps) 1.2*max(ynps)])
                xlabel('X(n)')
                ylabel('f(Xn)')
                grid
                pause
            else
                disp('ERROR FUERA DE RANGO')
                break
            end
        end
    end
    end
    end
    if e==1
        ft=f;
```

```
  else
  if e<20
        ft=[ft;f];
  end
end
end
end
end
```

resp2.m

```
clc
close(gcf)
n=input('CALCULAR HASTA n = ');
[num,den]=ss2tf(a,b,c,d);
y=dstep(a,b,c,d,1,n);
t=0:1:n;
disp(' ')
disp('ANALISIS DE LA RESPUESTA A UNA FUNCION PASO')
disp(' ')
%DEL VECTOR y QUE SE OBTIENE COMO RESPUESTA A LA
FUNCION %PASO Y ES EL VALOR MAXIMO DE y - k ES EL NUMERO
DE COLUMNA %EN EL QUE SE PRODUCE EL VALOR MAXIMO DE y
[Y,k]=max(y);
tpico=t(k);
%PORCENTAJE DEL MAXIMO SOBREIMPULSO
vf=abs(polyval(num,1)/polyval(den,1));
  Mp=100*(Y-vf)/vf;
if Mp<0
  Mp=0
  disp('VALOR FINAL DE LA FUNCION')
  vf=vf
else
  if Mp>100
```

```
  disp('ERROR.Mp demasiado grande REVISE LA FUNCION')
        else
                disp('VALOR FINAL DE LA FUNCION')
                vf=vf
                disp('PORCENTAJE DE MAXIMO SOBREIMPULSO')
                Mp=Mp
%EL TIEMPO EN EL QUE SE PRODUCE EL MAXIMO SOBREIMPULSO
%ES EL DEL VALOR K
                disp('TIEMPO PICO [seg] ')
                tpico=tpico
  end
end
%PARA ENCONTRAR EL TIEMPO DE ESTABLECIMIENTO
n=1;
  while y(n)<0.1*vf,n=n+1;
  end
m=1;
  while y(m)<0.9*vf,m=m+1;
  end
%disp('TIEMPO DE RESPUESTA DEL SISTEMA [seg]')
%tresp=t(m)-t(n)
disp(' ')
l=length(t);
while(y(l)>.98*vf)&(y(l)<1.02*vf);
l=l-1;
end
if Mp<100
  disp('TIEMPO DE ESTABLECIMIENTO DEL SISTEMA [seg]')
  ts=t(l)
end
pause
end
```

ANÁLISIS MEDIANTE LA INTEGRAL DE CONVOLUCIÓN

iconvol.m

```
function iconvol
% SE REALIZA EL INGRESO Y GRAFICO DE LAS SEÑALES X(t) y H(t)
% QUE SE VAN A CONVOLUCIONAR EN EL TIEMPO
labels = str2mat(...
   'INGRESO DE LA FUNCION  H(t)', ...
  'INGRESO DE LA FUNCION  X(t)', ...
  'GRAFICO DE CONVOLUCION X(t)*H(t)');
callbacks = [ ...
   'funhc        '
   'funxc        '
   'rescdc       '];
choices1('ICONVOL','MEDIANTE LA INTEGRAL DE CONVOLUCION',
labels, callbacks);
```

funhc.m

```
%EL PROGRAMA VA A CONTENER:
%INGRESO DE LA FUNCION H(t) QUE SE VA A CONVOLUCIONAR EN
EL TIEMPO

clc
close
disp('ELIGA LA FUNCION H(t)  ')
disp(' ')
disp(' ')
disp(' 1. FUNCION PASO: U(t)')
disp(' 2. FUNCION RAMPA: t.U(t)')
disp(' 3. FUNCION EXPONENCIAL: e^(at) ')
disp(' 4. FUNCION SINOSODAL: sen(wt) ')
disp(' 5. FUNCION COSENOIDAL: cos(wt) ')
disp(' 6. FUNCION EXPONENCIAL SENO: [e^(at)]*sen(wt)')
```

```
disp(' 7. FUNCION EXPONENCIAL COSENO: [e^(at)]*cos(wt)')
disp(' 8. INGRESE COMO FUNCION DE TRANSFERENCIA ')


in=input('REALIZAR: ');

 tcon=input('INGRESE EL LIMITE SUPERIOR PARA CONVOLUCIONAR
[seg]: ');
 t11=0:.1:tcon;
to0=0;
to1=tcon;
if in==1
 disp(' ')
 disp(' ')
 disp(' FUNCION PASO')
 disp(' ')
 numh=[0 1];
 denh=[0 1 0];
 else
 if in==2
        disp(' ')
        disp(' ')
        disp('FUNCION RAMPA')
        numh=[0 1];
        denh=[1 0 0];
        else
        if in==3
                disp(' ')
                disp(' ')
                disp(' FUNCION EXPONENCIAL')
                disp(' ')
                a=input('INGRESE a: ');
                numh=[0 1];
```

```
denh=[0 1 -a];
else
        if in==4
        disp(' ')
        disp(' ')
        disp(' FUNCION SINUSOIDAL sen(wt)')
        disp(' ')
        w=input('INGRESE W : ');
        numh=[0 w];
        denh=[1 0 w^2];
        else
        if in==5
                disp(' ')
                disp(' ')
                disp(' FUNCION COSENOIDAL cos(wt)')
                disp(' ')
                w=input('INGRESE W : ');
                numh=[1 0];
                denh=[1 0 w^2];
                else
                if in==6
                        disp(' ')
                        disp(' ')
                          disp('FUNCION EXPONENCIAL SENO
                                                [e^(at)]*sen(wt)')
                        a=input('INGRESE a : ');
                        w=input('INGRESE W : ');
                        numh=[0 1];
                        denh=[1 -2*a a^2+w^2];

                        else
                        if in==7
                                disp(' ')
```

```
                                        disp(' ')
                                          disp(' FUNCION EXPONENCIAL
                                                 COSENO [e^(at)]*cos(wt)')
                                        disp(' ')
                                        a=input('INGRESE a : ');
                                        w=input('INGRESE W : ');
                                        numh=[1 -a];
                                        denh=[1 -2*a a^2+w^2];
                                        else
                                        if in==8
                                                disp(' ')
                                                disp(' ')
                                                     disp(' INGRESO DE LA
                                        FUNCION DE TRANSFERENCIA')
                                  disp(' EN FORMA DE VECTORES DE ORDEN
                                                          DESCENDENTE')
                                num=input('INGRESE EL NUMERADOR : ');
                                den=input('INGRESE EL DENOMINADOR : ');
                                                numh=num;
                                                denh=den;
                                        else
                                   disp('ERROR FUERA DE RANGO')
                                        return
                                end
                        end
                end
            end
        end
    end
    end
    end
    end
    rt11=impulse(numh,denh,t11);
```

```
  plot(t11,rt11),title('CONVOLUCION DE FUNCIONES')
  axis([1.1*(to0-0.5) 1.1*(to1+0.5) 1.1*(min(rt11)-0.5) 1.1*(max(rt11)+0.5)])
  xlabel('tiempo[seg]')
  ylabel('H(t)')
  grid
  pause
end
end
```

funxc.m

```
%EL PROGRAMA VA A CONTENER:
%INGRESO DE LA FUNCION X(t) QUE SE VA A CONVOLUCIONAR EN
%EL TIEMPO
clc
close
disp('ELIGA LA FUNCION X(t)  ')
disp(' ')
disp(' ')
disp(' 1. FUNCION PASO: U(t)')
disp(' 2. FUNCION RAMPA: t.U(t)')
disp(' 3. FUNCION EXPONENCIAL: e^(at) ')
disp(' 4. FUNCION SINOSODAL: sen(wt) ')
disp(' 5. FUNCION COSENOIDAL: cos(wt) ')
disp(' 6. FUNCION EXPONENCIAL SENO: [e^(at)]*sen(wt)')
disp(' 7. FUNCION EXPONENCIAL COSENO: [e^(at)]*cos(wt)')
disp(' 8. INGRESO COMO FUNCION DE TRANSFERENCIA')
in=input('REALIZAR: ');
t0=0;
t1=tcon;
tr1=t11;
if in==1
  disp(' ')
  disp(' ')
```

```
disp(' FUNCION PASO')
disp(' ')
numx=[0 1];
denx=[0 1 0];
else
if in==2
        disp(' ')
        disp(' ')
        disp('FUNCION RAMPA')
        numx=[0 1];
        denx=[1 0 0];
        else
        if in==3
                disp(' ')
                disp(' ')
                disp(' FUNCION EXPONENCIAL')
                disp(' ')
                a=input('INGRESE a: ');
                numx=[0 1];
                denx=[0 1 -a];
                else
                        if in==4
                        disp(' ')
                        disp(' ')
                        disp(' FUNCION SINUSOIDAL sen(wt)')
                        disp(' ')
                        w=input('INGRESE W : ');
                        numx=[0 w];
                        denx=[1 0 w^2];
                        else
                        if in==5
                                disp(' ')
                                disp(' ')
```

```
                    disp(' FUNCION COSENOIDAL cos(wt)')
                    disp(' ')
                    w=input('INGRESE W : ');
                    numx=[1 0];
                    denx=[1 0 w^2];
                    else
                    if in==6
                          disp(' ')
        disp(' FUNCION EXPONENCIAL SENO [e^(at)]*sen(wt)')
                          disp(' ')
                          a=input('INGRESE a : ');
                          w=input('INGRESE W : ');
                          numx=[0 1];
                          denx=[1 -2*a a^2+w^2];
                          else
                          if in==7
        disp(' FUNCION EXPONENCIAL COSENO [e^(at)]*cos(wt)')
        disp(' ')
                                a=input('INGRESE a : ');
                                w=input('INGRESE W : ');
                                numx=[1 -a];
                                denx=[1 -2*a a^2+w^2];
                                else
                                if in==8
disp(' ')
disp(' ')
disp(' INGRESO DE LA FUNCION DE TRANSFERENCIA')
disp(' EN FORMA DE VECTORES DE ORDEN DESCENDENTE')
num=input('INGRESE EL NUMERADOR : ');
den=input('INGRESE EL DENOMINADOR : ');
                                      numx=num;
                                      denx=den;
                                else
```

```
                                        disp('ERROR FUERA DE RANGO')
                                                return
                                        end
                                end
                        end
                end
        end
    end
    end
    end
    end
    rt1=impulse(numx,denx,tr1);
    plot(tr1,rt1),title('CONVOLUCION DE FUNCIONES')
    hold on
    axis([1.1*(t0-0.5) 1.1*(t1+0.5) 1.1*(min(rt11)-0.5) 1.1*(max(rt1)+0.5)])
    xlabel('tiempo[seg]')
    ylabel('X(t)')
    grid
    pause
    end
    end
```

rescdc.m

```
%EL PROGRAMA VA A CONTENER:
%CONVOLUCION DE LAS FUNCIONES  X(t) Y H(t)
%Las funciones que se ingresaron y se graficaron se llaman:
% rt11,t11,numh,denh. Para H(t)
% rt1,tr1,numx,denx. Para X(t)
%Convolución en un intervalo común para las dos funciones
close
clc
  num=conv(numh,numx);
  den=conv(denh,denx);
```

```
tfin=t11;
tfinn=tfin;
yfin=impulse(num,den,tfin);
subplot(3,1,1)
rt11=impulse(numh,denh,t11);
  plot(t11,rt11),title('CONVOLUCION DE FUNCIONES: Y(t)=X(t)*H(t)')
  axis([1.2*min(tfinn) 1.2*max(tfinn) 1.1*(min(rt11)-1) 1.1*(max(rt11)+1)])
  xlabel('t[seg]')
  ylabel('H(t)')
  grid
subplot(3,1,2)
rt1=impulse(numx,denx,tr1);
  plot(tr1,rt1)
  axis([1.2*min(tfinn) 1.2*max(tfinn) 1.3*(min(rt1)-1) 1.3*(max(rt1)+1)])
  xlabel('t[seg]')
  ylabel('X(t)')
  grid
subplot(3,1,3)
  plot(tfin,yfin)
  axis([1.2*min(tfinn) 1.2*max(tfinn) min(1.1*min(yfin),0) 1.1*max(yfin)])
  xlabel('t[seg]')
  ylabel('y(t)')
  grid
pause
end
```

ANÁLISIS MEDIANTE LA SUMATORIA DE CONVOLUCIÓN

sconvol.m

```
function sconvol
% SE REALIZA EL INGRESO Y GRAFICO DE LAS SEÑALES X(n) y H(n)
% QUE SE VAN A CONVOLUCIONAR
labels = str2mat(...
```

```
    'LIMITES EXTREMOS DE X(n) y H(n)     ', ...
    'INGRESO DE LA FUNCION  H(n)', ...
    'INGRESO DE LA FUNCION  X(n)', ...
    'GRAFICO DE CONVOLUCION X(n)*H(n)');
  callbacks = [ ...
     'limitc        '
     'funh          '
     'funx          '
     'rescd         '];
  choices1('SCONVOL','MEDIANTE LA SUMATORIA DE CONVOLUCION',
  labels, callbacks);
```

limitc.m

```
  %EL PROGRAMA VA A CONTENER:
  %INGRESO DE LOS LIMITES DE LAS FUNCIONES X Y H
  %QUE SE VAN A CONVOLUCIONAR
  close
  clc
  disp(' ')
  disp('LIMITES PARA MANEJAR LAS FUNCIONES X(n) Y H(n)')
  disp(' ')
   li=input('LIMITE INFERIOR: ');
   disp(' ')
   ls=input('LIMITE SUPERIOR: ');
   N=round(li):1:round(ls);
  [fn,cn]=size(N);
  N1=round(li):1:(2*cn)-1;
  end
```

funh.m

```
  %EL PROGRAMA VA A CONTENER:
  %INGRESO DE LA FUNCION H(n) QUE SE A CONVOLUCIONAR
  clc
```

```
close
disp('FUNCION H(n) ')
disp(' ')
disp('INGRESE LA FUNCION CORRESPONDIENTE AL VECTOR N')
disp(' ')
N
disp(' ')
h=input('H(n) = ');
end
```

funx.m

```
%EL PROGRAMA VA A CONTENER:
%INGRESO DE LA FUNCION X(n) QUE SE A CONVOLUCIONAR
clc
close
disp('FUNCION X(n) ')
disp(' ')
disp('INGRESE LA FUNCION CORRESPONDIENTE AL VECTOR N')
disp(' ')
N
disp(' ')
x=input('X(n) = ');
end
```

rescd.m

```
%EL PROGRAMA VA A CONTENER:
%CONVOLUCION DE LAS FUNCIONES  X(n) Y H(n)
clc
close
disp(' ')
%fn ES EL NUMERO DE FILAS DE N (fn=1)
%cn ES EL NUMERO DE COLUMNAS DE N
[fn,cn]=size(N);
```

```
yr=zeros(1,2*cn);
for i=1:cn,
  if i==1
          y1=x(i).*h;
  else
          y1=x(i).*[zeros(1,(i-1)),h];
  end
l1=size(y1)*[0;1];
if l1==2*cn
  yt=y1;
  else
  if l1<2*cn
          yt=[y1,zeros(1,2*cn-l1)];
          else
          yt=y1(:,1:2*cn);
  end
end
yr=yr+yt;
end
N1=min(N):1:min(N)+size(yr)*[0;1]-1;
[fn1,cn1]=size(N);
[fn2,cn2]=size(N1);
xxx=cn1;
figure
for k=1:xxx,
          hold on
          discret=ones(1,xxx);
          nnn=N(k)*discret;
          matriz=zeros(1,xxx);
          matrizh=matriz;
          matriz(k)=x(k);
          funx=matriz;
          subplot(211)
```

```
        plot(nnn,funx)
        axis([min(N)-0.5 max(N)+0.5 min(1.2*min(x),0) 1.2*max(x)])
end
for k=1:xxx,
        hold on
        discret=ones(1,xxx);
        nnn=N(k)*discret;
        matrizh=zeros(1,xxx);
        subplot(212)
        matrizh(k)=h(k);
        funh=matrizh;
        plot(nnn,funh)
        axis([min(N)-0.5 max(N)+0.5 min(1.2*min(h),0) 1.2*max(h)])
end
subplot(211)
  plot(N,x,'oy'),title('CONVOLUCION DE Y(n)=X(n)*H(n)')
  xlabel('n')
  ylabel('X(n)')
  grid
subplot(212)
  plot(N,h,'oy')
  xlabel('n')
  ylabel('H(n)')
  grid
hold off
pause
xx=cn2;
figure
for k=1:xx,
        hold on
        discret=ones(1,xx);
        nn=N1(k)*discret;
        matrizy=zeros(1,xx);
```

```
            matrizy(k)=yr(k);
            funy=matrizy;
            plot(nn,funy)
            axis([min(N1)-0.5 max(N1)+0.5 min(1.2*min(yr),0) 1.2*max(yr)])
    end
    plot(N1,yr,'oy'),title('CONVOLUCION DE Y(n)=X(n)*H(n)')
    xlabel('n')
    ylabel('Y(n)=X(n)*H(n)')
    grid
    pause
    end
    end
```

ANÁLISIS MEDIANTE EL ESPACIO DE ESTADO PARA SISTEMAS CONTÍNUOS

econtin.m

```
    function econtin
    labels = str2mat(...
      'INGRESO DE DATOS', ...
      'RESPUESTA A UNA FUNCION IMPULSO', ...
      'RESPUESTA A UNA FUNCION PASO', ...
      'RESPUESTA A UNA SEÑAL ARBITRARIA', ...
      'ANALISIS DE RESPUESTA EN EL TIEMPO                    ');
    callbacks = [ ...
       'ingreso5      '
       'impulso1      '
       'paso1         '
       'arbit1        '
       'resp1         '];
    choices1('ESPACIO.ESTADO', 'ANALISIS EN EL ESPACIO DE ESTADO,
    SISTEMAS CONTINUOS', labels, callbacks);
```

ingreso5.m

```
echo off
close
clc
disp(' ')
disp('      INGRESO DE DATOS ')
disp(' ')
disp('    INGRESO DEL ESPACIO DE ESTADO ')
disp(' ')
disp('    X´ = A.X + B.u')
disp(' ')
disp('    Y  = C.X + D.u ')
disp(' ')
disp(' ')
disp('    A,B,C,D SON LAS MATRICES DE ESTADO ')
disp(' ')
disp(' ')
  a=input('Ingrese la matriz  A=');
  b=input('                   B=');
  c=input('                   C=');
  d=input('                  D=');
i=0;
end
```

Se utilizan las rutinas impulso1.m, paso1.m, arbit1.m y resp1.m, descritas en el tema: Análisis mediante ecuaciones diferenciales.

ANÁLISIS MEDIANTE EL ESPACIO DE ESTADO PARA SISTEMAS DISCRETOS

ediscret.m

```
function ediscret
labels = str2mat(...
```

```
  'INGRESO DE DATOS', ...
  'RESPUESTA A UNA FUNCION IMPULSO', ...
  'RESPUESTA A UNA FUNCION PASO', ...
  'RESPUESTA A UNA SEÑAL ARBITRARIA', ...
  'ANALISIS DE RESPUESTA EN EL TIEMPO              ');
 callbacks = [ ...
   'ingreso6      '
   'impulso2      '
   'paso2         '
   'arbit2        '
   'resp2         '];
 choices1('ESPACIO.ESTADO', 'ANALISIS EN ESPACIO DE ESTADO,
 SISTEMAS DISCRETOS', labels, callbacks);
```

ingreso6.m

```
 clear
 echo off
 close
 clc
 disp('      INGRESO DE DATOS ')
 disp(' ')
 disp('    ESPACIO DE ESTADO, SISTEMAS DISCRETOS ')
 disp('    Xn´ = A.Xn + B.u')
 disp('    Yn  = C.Xn + D.u ')
 disp('    A,B,C,D SON LAS MATRICES DE ESTADO ')
  a=input('Ingrese la matriz   A=');
  b=input('                    B=');
  c=input('                    C=');
  d=input('                    D=');
  i=0;
 end
```

Se utilizan las rutinas impulso2.m, paso2.m, arbit2.m y resp2.m, descritas en el tema: Análisis mediante ecuaciones en diferencias.

B.2. ANÁLISIS DE SISTEMAS LINEALES EN EL DOMINIO DE LA FRECUENCIA.

menu2.m

```
function menu2
labels = str2mat(...
  'MEDIANTE LA TRANSFORMADA DE LAPLACE', ...
  'MEDIANTE LA TRANSFORMADA Z', ...
  'MEDIANTE LAS SERIES DE FOURIER', ...
  'MEDIANTE LA TRANSFORMADA DE FOURIER', ...
  'CON LA TRANSFORMADA RAPIDA DE FOURIER');
callbacks = [ ...
   'tlaplace      '
   'tz            '
   'sfourier      '
   'tfourier      '
   'rfourier      '];
choices1('FRECUENCIA', 'ANALISIS EN EL DOMINIO DE LA
FRECUENCIA', labels, callbacks);
```

ANÁLISIS MEDIANTE LA TRANSFORMADA DE LAPLACE

tlaplace.m

```
function tlaplace
labels = str2mat(...
  'INGRESO DE DATOS                    ', ...
  'RESPUESTA A UNA FUNCION IMPULSO', ...
  'RESPUESTA A UNA FUNCION PASO', ...
  'RESPUESTA A UNA SEÑAL ARBITRARIA', ...
  'LUGAR GEOMETRICO DE LAS RAICES', ...
  'ANALISIS EN EL PLANO S');
callbacks = [ ...
   'ingresf1      '
```

```
    'impulso1      '
    'paso1         '
    'arbit1        '
    'lgrtl         '
    'respf1        '];
choices1('T.LAPLACE', 'ANALISIS EN FRECUENCIA MEDIANTE LA
TRANSFORMADA DE LAPLACE', labels, callbacks);
```

ingresf1.m

```
echo off
close
aplicacion=0;
clc
disp('ANALISIS MEDIANTE LA TRANFORMADA DE LAPLACE')
disp(' ')
disp('INGRESO DE LA FUNCION DE TRANSFERENCIA ')
disp('            H(s) = NUM(s) / DEN(s)')
disp('EL NUMERADOR NO PUEDE SER DE ORDEN MAYOR QUE EL
DENOMINADOR')
disp(' ')
disp('                    b2.S²+b1.S+b0')
disp('            H(s) = -----------------')
disp('                    a3.S³+a2.S²+a1.S+a0')
disp(' ')
disp('INGRESE DE LA SIGUIENTE MANERA:   num=[b2 b1 bo]')
disp('                          den=[a3 a2 a1 ao]')
disp(' ')
num=input('INGRESE SUS DATOS:    num= ');
den=input('                  den= ');
%PASANDO A LA FORMA CANONICA CONTROLABLE
[a,b,c,d]=tf2ss(num,den)
i=0;
end
```

lgrtl.m

```
clc
clf
rlocus(num,den)
title('L.G.R.  SISTEMAS CONTINUOS')
[gain,pol]=rlocfind(num,den)
pause
end
```

respf1.m

```
clc
close(gcf)
disp('ANALISIS EN  FRECUENCIA COMPLEJA "S"')
 [ceros,polos,ganancia]=tf2zp(num,den)
%ENCUENTRA Y DESPLIEGA LOS VALORES DE POLOS Y CEROS DE
%LA FUNCION
%z SON LOS CEROS DE LA FUNCION
%p SON LOS POLOS DE LA FUNCION
%A CONTINUACION SE VA A REALIZAR UN GRAFICO DE POLOS Y
%CEROS DE LA FUNCION
rceros=real(ceros);
iceros=imag(ceros);
rpolos=real(polos);
ipolos=imag(polos);
re1=max(rceros,rpolos);
re2=min(rceros,rpolos);
im1=max(iceros,ipolos);
im2=min(iceros,ipolos);
%PARA DETERMINAR ESTABILIDAD, TODAS LAS RAICES DEBEN
%ITUARSE EN LA PARTE DEL SEMIPLANO  IZQUIERDO, CASO
%CONTRARIO EL SISTEMA ES INESTABLE.
[m1,m2]=size(rpolos);
[n1,n2]=size(rceros);
```

```
for j=1:m1,
  if rpolos(j)<0
          estabilidad1=1;
  else
          estabilidad1=0;
          break
  end
end
  for k=1:n1,
          if rceros(j)<=0
                  estabilidad2=1;
          else
                  estabilidad2=0;
                  break
          end
  end
if n1==0
  estabilidad2=1;
end
if (estabilidad1==1 & estabilidad2==1)
  disp('EL SISTEMA ES ESTABLE')
  disp(' ')
  disp('    TODAS LAS RAICES SE ENCUENTRAN')
  disp('    EN EL SEMIPLANO IZQUIERDO ')
else
  disp('EL SISTEMA NO ES ESTABLE')
  disp(' ')
  disp('    TODAS LAS RAICES NO SE ENCUENTRAN')
  disp('    EN EL SEMIPLANO IZQUIERDO ')
end
disp('    PRESIONE UNA TECLA PARA CONTINUAR ')
pause
figure
```

```
hold on
plot(rceros,iceros,'og'),title('POLOS Y CEROS DE LA FUNCION DE TRANSFERENCIA')
plot(rpolos,ipolos,'xc')
xlabel('Eje Real')
ylabel('Eje Imaginario')
grid
pause
end
```

Se utilizan las rutinas impulso1.m, paso1.m y arbit1.m, descritas en el tema: Análisis mediante ecuaciones diferenciales.

ANÁLISIS MEDIANTE LA TRANSFORMADA Z.

tz.m

```
function tz
labels = str2mat(...
  'INGRESO DE DATOS', ...
  'RESPUESTA A UNA FUNCION IMPULSO', ...
  'RESPUESTA A UNA FUNCION PASO', ...
  'RESPUESTA A UNA SEÑAL ARBITRARIA', ...
  'LUGAR GEOMETRICO DE LAS RAICES', ...
  'ANALISIS EN EL PLANO Z      ');
callbacks = [ ...
   'ingresf2      '
   'impulso2      '
   'paso2         '
   'arbit2        '
   'lgrtz         '
   'respf2        ''];
choices1('T.Z.', 'ANALISIS EN FRECUENCIA MEDIANTE LA TRANSFORMADA Z', labels, callbacks);
```

Ingresf2.m

```
close
clc
disp('ANALISIS MEDIANTE LA TRANFORMADA Z')
disp(' ')
disp('INGRESO DE LA FUNCION X(z) ')
disp('            H(z) = NUM(z) / DEN(z)')
disp('EL NUMERADOR NO PUEDE SER DE ORDEN MAYOR QUE EL
DENOMINADOR')
disp(' ')
disp('                    b2.Z²+b1.Z+b0')
disp('            H(z) = ----------------')
disp('                    a3.Z³+a2.Z²+a1.Z+a0')
disp('INGRESE DE LA SIGUIENTE MANERA:   num=[b2 b1 bo]')
disp('                          den=[a3 a2 a1 ao]')
num=input('INGRESE SUS DATOS:    num= ');
den=input('                  den= ');
%PASANDO A LA FORMA CANONICA CONTROLABLE
[a,b,c,d]=tf2ss(num,den)
i=0;
end
```

Igrtz.m

```
clc
close(gcf)
disp('L.G.R. CON PROYECCION DEL CIRCULO DE RADIO UNITARIO ')
rlocus(num,den)
title('L.G.R.  SISTEMAS DISCRETOS')
axis('square')
zgrid
[gain,pol]=rlocfind(num,den)
pause
end
```

respf2.m

```
clc
close(gcf)
disp('ANALISIS EN  FRECUENCIA COMPLEJA "Z"')
disp(' ')
[ceros,polos,ganancia]=tf2zp(num,den)
%ENCUENTRA Y DESPLIEGA LOS VALORES DE POLOS Y CEROS DE
%LA FUNCION
%z SON LOS CEROS DE LA FUNCION
%p SON LOS CEROS DE LA FUNCION
%A CONTINUACION SE VA A REALIZAR UN GRAFICO DE POLOS Y
%CEROS DE LA FUNCION
rceros=real(ceros);
iceros=imag(ceros);
rpolos=real(polos);
ipolos=imag(polos);
re1=max(max(rceros),max(rpolos));
re2=min(min(rceros),min(rpolos));
im1=max(max(iceros),max(ipolos));
im2=min(min(iceros),min(ipolos));
%PARA DETERMINAR ESTABILIDAD , LOS POLOS DEL SISTEMA EN
EL %PLANO ZLOS POLOS DEBEN ESTAR DENTRO DEL CÍRCULO DE
%RADIO UNITARIO.
[m1,m2]=size(rpolos);
[n1,n2]=size(rceros);
[m3,m4]=size(ipolos);
[n3,n4]=size(iceros);
for j=1:m1,
 if (abs(rpolos(j)))<=1
        estabilidad1=1;
 else
        estabilidad1=0;
        break
```

```
  end
end
for k=1:n1,
  if (abs(rceros(k)))<=1
          estabilidad2=1;
  else
          estabilidad2=0;
          break
  end
end
for j=1:m3,
  if (abs(ipolos(j)))<=1
          estabilidad3=1;
  else
          estabilidad3=0;
          break
  end
end
for k=1:n3,
  if (abs(iceros(k)))<=1
          estabilidad4=1;
  else
          estabilidad4=0;
          break
  end
end
[fnum,cnum]=size(num); %Cuando no tiene ceros
if (fnum== 1 & cnum==1)
  estabilidad2=1;
  estabilidad4=1;
end
if ((estabilidad1==1 & estabilidad2==1)&(estabilidad3==1 &
estabilidad4==1))
```

```
  disp('EL SISTEMA ES ESTABLE')
  disp('    TODOS "LOS POLOS" SE ENCUENTRAN')
  disp('    DENTRO DEL CIRCULO DE RADIO UNITARIO ')
else
  disp('EL SISTEMA NO ES ESTABLE')
  disp('    TODOS "LOS POLOS" NO SE ENCUENTRAN')
  disp('    DENTRO DEL CIRCULO DE RADIO UNITARIO ')
end
pause
figure
hold on
plot(rceros,iceros,'og'),title('POLOS Y CEROS DE LA FUNCION DE
TRANSFERENCIA')
plot(rpolos,ipolos,'xc')
xlabel('Eje Real')
ylabel('Eje Imaginario')
grid
pause
end
```

Se utilizan las rutinas impulso2.m, paso2.m y arbit2.m, descritas en el tema: Análisis mediante ecuaciones en diferencias.

ANÁLISIS MEDIANTE SERIES DE FOURIER

sfourier.m

```
function sfourier
labels = str2mat(...
  'FUNCIONES PREDETERMINADAS', ...
  'INGRESO DE LA SERIE DE FOURIER', ...
  'ESPECTRO DE FRECUENCIA ', ...
  'FUNCION FORMADA POR ARMONICAS');
callbacks = [ ...
```

```
    'funcpf3        '
    'iserief3       '
    'espef3         '
    'farmonf3       '];
choices1('SERIES.FOURIER', 'ANALISIS EN FRECUENCIA MEDIANTE
LAS SERIES DE FOURIER', labels, callbacks);
```

funcpf3.m

```
%EL PROGRAMA VA A CONTENER:
%INGRESO DE LA FUNCION PREDETERMINADA Y GRAFICO
%GRAFICO DEL ESPECTRO DE FRECUENCIA DE LA FUNCION
%GRAFICO DE LA FUNCION PREDETERMINADA FORMADA POR n
%ARMONICAS
clc
clear t
clf
close(gcf)
disp('ANALISIS DE FOURIER PARA FUNCIONES PERIÓDICAS:   ')
disp(' ')
disp(' 1. FUNCION PULSO SIMETRICA')
disp(' 2. FUNCION TRIANGULAR')
disp(' 3. FUNCION DIENTE DE SIERRA')
disp(' 4. FUNCION SINOSODAL')
disp(' 5. FUNCION COSENOIDAL')
in=input('REALIZAR: ');
if in==1
  disp(' ')
  disp(' ')
  disp(' FUNCION PULSO SIMETRICA')
  disp(' ')
  disp('  an = 2 * sin(2*n*pi*t/T) / (pi*n)')
  disp(' ')
  %kp=input('INGRESE LA GANANCIA: ');
```

```
to1=input('INGRESE EL LIMITE SIMETRICO DEL INTERVAL0 t: ');
t=input('INGRESE EL PERIODO DE LA FUNCION T : ');
ti=-to1*1.5;
tf=to1*1.5;
global n
an='(2*sin(2*n*pi*to1/t))/(pi*n)';
    bn='0';
for k=1:1:101,
        n=-51+k;
        ann(k)=eval(an);
        bnn(k)=eval(bn);
end
ao=ann(51);
i=find(isnan(ao));
if i==1
        ao=(ann(50)+ann(52))/2
end
cnn=0;
else
if in==2
        disp(' ')
        disp(' ')
        disp('FUNCION TRIANGULAR ')
        disp(' ')
        disp('  bn=[4/(pi*pi*n*n))*sin(0.5*n*pi)-0.5*n*pi*cos(0.5*n*pi)] ')
        disp(' ')
        disp(' ')
        %kp=input('INGRESE LA GANANCIA: ');
        to1=input('INGRESE EL LIMITE SIMETRICO DEL INTERVAL0 t: ');
        t=input('INGRESE EL PERIODO DE LA FUNCION T : ');
        ti=-to1*1.5;
        tf=to1*1.5;
        global n
```

```
an='0';
bn='((4/(pi*pi*n*n))*sin(n*pi/2)-0.5*n*pi*cos(0.5*n*pi))';
for k=1:1:101,
        n=-51+k;
        ann(k)=eval(an);
        bnn(k)=eval(bn);
end
ao=ann(51);
i=find(isnan(ao));
if i==1
        ao=(ann(50)+ann(52))/2
end
cnn=0;

else
        if in==3
        disp(' ')
        disp(' ')
        disp(' DIENTE DE SIERRA')
        disp(' ')
        disp(' ')
        disp(' ')
        a=input('INGRESE a: ');
to0=input('INGRESE EL LIMITE INFERIOR DEL INTERVAL0: ');
to1=input('INGRESE EL LIMITE SUPERIOR DEL INTERVAL0: ');
        t11=to0:.1:to1
        else
                if in==4
                disp(' ')
                disp(' ')
                disp(' FUNCION SINUSOIDAL')
                disp(' ')
                w=input('INGRESE W : ');
```

```
        to0=input('INGRESE EL LIMITE INFERIOR DEL INTERVAL0: ');
        to1=input('INGRESE EL LIMITE SUPERIOR DEL INTERVAL0: ');
                t11=to0:.1:to1



                else
                if in==5
                        disp(' ')
                        disp(' ')
                        disp(' FUNCION COSENOIDAL ')
                        disp(' ')
                        w=input('INGRESE W : ');
        to0=input('INGRESE EL LIMITE INFERIOR DEL INTERVAL0: ');
        to1=input('INGRESE EL LIMITE SUPERIOR DEL INTERVAL0: ');
                        t11=to0:.1:to1
                        else
                        disp('ERROR FUERA DE RANGO')
                                return
            end
        end

  end
end
end
%FUNCION ESPEF3.M
% ESTE PROGRAMA PERMITE GRAFICAR EL ESPECTRO DE
FRECUENCIA
% DE UNA SERIE DE FOURIER
clc
disp(' ')
disp(' ')
disp('ESPECTRO DE FRECUENCIA DE LA FUNCION ')
disp(' ')
```

```
disp(' ')
fnn=input('Máximo valor de n para visualizar ? ');
 if cnn==0
        ann;
        bnn;
 else
        ann=conj(cnn)+cnn;
        bnn=cnn-conj(cnn);
 end
cnns=(ann.^2+bnn.^2).^(0.5);
%se va a graficar cnn que es el espectro de frecuencia
%como un sistema discreto
[fcnn,ccnn]=size(cnns);
if fnn>=50
 ccnn1=50;
else
 ccnn1=fnn;
end
figure
xxx=ccnn1*2+1;
hold on
if ccnn1<50
 for k=1:xxx,
        discret=ones(1,xxx);
        nnn=(-fnn+k-1)*discret;
        matriz=zeros(1,xxx);
        cnns1=cnns(:,50-ccnn1+1:51+fnn);
        matriz(k)=cnns1(k);
        fun=matriz;
        plot(nnn,fun)
 end
else
 for k=1:xxx,
```

```
        discret=ones(1,xxx);
        nnn=(-50+k-1)*discret;
        matriz=zeros(1,xxx);
        cnns1=cnns;
        matriz(k)=cnns1(k);
        fun=matriz;
        plot(nnn,fun)
  end
end

title('SERIES DE FOURIER')
xlabel('[n]')
ylabel('Espectro de frecuencia')
grid
pause


%FUNCION FARMONF3.M
% ESTE PROGRAMA PERMITE GRAFICAR LA FUNCION EXPRESADA
COMO
%UNA SERIE DE FOURIER, CON ARMONICAS DE FRECUENCIA Wo
if cnn==0
  [lf,lc] = size(ann);
  lcc=(lc-1)/2;
else
  [lf,lc] = size(cnn);
  lcc=(lc-1)/2;
end
clc
disp(' ')
disp(' ')
close(gcf)
disp('FUNCION PERIODICA FORMADA POR ARMONICAS ')
```

```
disp(' ')
disp(' ')
%tt es el periódo de la función
tt=t;
disp(' ')
disp(' ')
%ti=input('Límite inferior de la función [seg]?  ');
%tf=input('Límite superior de la función [seg]? ');
disp(' ')
disp(' ')
n=input('Ingrese el número de armónicas n ? ');
nwo=n*2*pi/tt;
wo=2*pi/tt;
if cnn==0
 i1=0;
 i2=0;
 i1=find(isnan(ann));
 if i1~=0
  [p1,q1]=size(i1);
   for x=1:q1,
        ann(i1(x))=(ann(i1(x)-1)+ann(i1(x)+1))/2;
   end
 end
 i2=find(isnan(bnn));
 if i2~=0
  [p2,q2]=size(i2);
   for xx=1:q2,
        bnn(i2(xx))=(bnn(i2(xx)-1)+bnn(i2(xx)+1))/2;
   end
 end

 ann;
 ann1=ann(:,lcc+2:lc);
```

```
 bnn;
 bnn1=bnn(:,lcc+2:lc);
 disp('                    inf')
 disp(' f(t) = ao + SUMATORIO  [an.cos(n.wo.t) + bn.sen(n.wo.t)]')
 disp('                    n=1 ')
 disp(' ')
 t=ti:.01:tf;
 ft1=0;
 for i=1:n,
        ft=ann1(i)*cos(i*wo*t)+bnn1(i)*sin(i*wo*t);
        ft1=ft+ft1;
 end
   if ao~=0
 fo=ao*ones(size(ft1));
 ftt1=fo+ft1;
   else
 ftt1=ft1;
   end
else
 i=0;
 i=find(isnan(cnn));
 if i~=0
  [p,q]=size(i);
   for x=1:q,
        cnn(i(x))=(cnn(i(x)-1)+cnn(i(x)+1))/2;
   end
 end
 cnn;
 cnn1=cnn(:,lcc+2:lc);
 disp('              inf')
 disp(' f(t) = SUMATORIO  [cn.e^(j.n.wo.t)]')
 disp('            - inf ')
 disp(' ')
```

```
%        ao=2\cnn(1,lcc+1)
  t=ti:.01:tf;
  ft1=0;
  ann1=cnn1+conj(cnn1);
  bnn1=cnn1-conj(cnn1);
  for i=1:n,
          ft=ann1(i)*cos(i*wo*t)+bnn1(i)*sin(i*wo*t);
          ft1=ft+ft1;
  end
    if ao~=0
  fo=ao*ones(size(ft1));
  ftt1=fo+ft1;
    else
  ftt1=ft1;
    end
end

figure
hold on
plot(t,ftt1)
axis([(ti+tf-tt*0.62) (ti+tf+tt*0.62) min(0,min(ftt1)) 1.2*max(ftt1)])
title('FUNCION COMPUESTA CON ARMONICAS')
xlabel('tiempo[seg]')
ylabel('X(t)')
grid
pause
end
```

iserief3.m

```
function iserief3
labels = str2mat(...
  'COMO SERIE TRIGONOMETRICA', ...
  'COMO SERIE EXPONENCIAL');
callbacks = [ ...
   'trigonf3      '
   'exponf3       '];
choices1('ISERIEF3', 'INGRESO DE LA SERIE DE f(t)', labels, callbacks);
```

trigonf3.m

```
%function trigonf3
%1. PROGRAMA PARA INGRESAR LA SERIE TRIGONOMETRICA DE
%OURIER
close
clc
disp(' FUNCION f(t) DESCRITA POR UNA SERIE TRIGONOMETRICA')
 disp(' ')
disp('                  inf')
disp(' f(t) = ao + SUMATORIO  [an.cos(n.wo.t) + bn.sen(n.wo.t)]')
disp('                  n=1 ')
disp(' ')
disp(' wo=fundamental')
disp(' n.wo=No. de armónicas')
disp(' ')
disp(' INGRESO DE DATOS:')
!edit anf3.m
anf3
load ann
load bnn
load cnn
end
```

anf3.m

```
function anf3
global n
%PROGRAMA PARA INGRESAR UNA FUNCION an y bn
%       Ejemplo
%       an='(n*n+5)/2'
%       bn='n*n + 11'
%       ao=0.6
%INGRESE UN VALOR DE an y bn EN LA SIGUIENTE LINEA

  an='(2*sin(0.6*pi*n))/(pi*n)'
  bn='0'                                                    ;
  ao=0.6
%PONGA GUARDAR Y SALIR DEL EDITOR
for k=1:1:101,
 n=-51+k;
        ann(k)=eval(an);
     bnn(k)=eval(bn);
end
ann(51)=ao;
cnn=0;
save ao
save ann
save bnn
save cnn
end
```

exponf3.m

```
function exponf3
%EL PROGRAMA VA A CONTENER:
%1.INGRESO DE LA SERIE EXPONENCIAL DE FOURIER
```

```
%1. PROGRAMA PARA INGRESAR LA SERIE EXPONENCIAL DE
FOURIER
close
clc
disp(' FUNCION f(t) DESCRITA POR UNA SERIE EXPONENCIAL')
disp('            +inf')
disp(' f(t) =SUMATORIO  [cn. e^(j.n.wo.t)]')
disp('            -inf ')
disp(' wo=fundamental')
disp(' n.wo=No. de armónicas')
disp(' INGRESO DE DATOS:')
!edit cnf3.m
cnf3
end
```

cnf3.m

```
function cnf3
global cn
%PROGRAMA PARA INGRESAR UNA FUNCION cn
%       Ejemplo
%       cn='n*n*n+1'
%       ao=0.6;
%INGRESE UN VALOR DE cn EN LA SIGUIENTE LINEA

      cn='2*(sin(0.6*pi*n))/(pi*n)'
      ao=0.6;

%PONGA GUARDAR Y SALIR DEL EDITOR
 for k=1:1:101,
         n=-51+k;
           cnn(k)=eval(cn);
 end
co=ao;
```

```
cnn(51)=co;
ann=0;
bnn=0;
save ao
save ann,save bnn,save cnn
end
```

espef3.m

```
% ESTE PROGRAMA PERMITE GRAFICAR EL ESPECTRO DE
%FRECUENCIA DE UNA SERIE DE FOURIER
load ann
load bnn
load cnn
clc
close(gcf)
fnn=input('Máximo valor de n para visualizar ? ');
 if cnn==0
        ann;
        bnn;
 else
        ann=conj(cnn)+cnn;
        bnn=cnn-conj(cnn);
 end
cnns=(ann.^2+bnn.^2).^(0.5);
%grafico de cnn que es el espectro de frecuencia como un sistema discreto
[fcnn,ccnn]=size(cnns);
if fnn>=50
 ccnn1=50;
else
 ccnn1=fnn;
end
figure
xxx=ccnn1*2+1;
```

```
hold on
if ccnn1<50
  for k=1:xxx,
        discret=ones(1,xxx);
        nnn=(-fnn+k-1)*discret;
        matriz=zeros(1,xxx);
        cnns1=cnns(:,50-ccnn1+1:51+fnn);
        matriz(k)=cnns1(k);
        fun=matriz;
        plot(nnn,fun)
  end
else
  for k=1:xxx,
        discret=ones(1,xxx);
        nnn=(-50+k-1)*discret;
        matriz=zeros(1,xxx);
        cnns1=cnns;
        matriz(k)=cnns1(k);
        fun=matriz;
        plot(nnn,fun)
  end
end
title('SERIES DE FOURIER')
xlabel('[n]')
ylabel('Espectro de frecuencia')
grid
pause
save ann bnn cnn
end
```

farmonf3.m

```
% ESTE PROGRAMA PERMITE GRAFICAR LA FUNCION EXPRESADA
%COMO UNA SERIE DE FOURIER, CON ARMONICAS DE FRECUENCIA
%Wo
```

```
load ao
load ann
load bnn
load cnn
if cnn==0
  [lf,lc] = size(ann);
  lcc=(lc-1)/2;
else
  [lf,lc] = size(cnn);
  lcc=(lc-1)/2;
end
clc
close(gcf)
disp('FUNCION PERIODICA FORMADA POR ARMONICAS ')
disp(' '
tt=input('Cuál es el periódo de la función To ? ')
disp(' '
ti=input('Límite inferior de la función [seg]?  ');
disp(' '
tf=input('Límite superior de la función [seg]? ');
disp(' '
n=input('Ingrese n ? ');
nwo=n*2*pi/tt;
wo=2*pi/tt;
if cnn==0
  i1=0;
  i2=0;
  i1=find(isnan(ann));
  if i1~=0
   [p1,q1]=size(i1);
     for x=1:q1,
            ann(i1(x))=(ann(i1(x)-1)+ann(i1(x)+1))/2;
     end
```

```
 end
 i2=find(isnan(bnn));
 if i2~=0
  [p2,q2]=size(i2);
   for xx=1:q2,
          bnn(i2(xx))=(bnn(i2(xx)-1)+bnn(i2(xx)+1))/2;
   end
 end
 ann;
 ann1=ann(:,lcc+2:lc);
 bnn;
 bnn1=bnn(:,lcc+2:lc);
 disp('                    inf')
 disp(' f(t) = ao + SUMATORIO  [an.cos(n.wo.t) + bn.sen(n.wo.t)]')
 disp('                    n=1 ')
 t=ti:.01:tf;
 ft1=0;
 for i=1:n,
          ft=ann1(i)*cos(i*wo*t)+bnn1(i)*sin(i*wo*t);
          ft1=ft+ft1;
 end
   if ao~=0
 fo=ao*ones(size(ft1));
 ftt1=fo+ft1;
   else
 ftt1=ft1;
   end
else
 i=0;
 i=find(isnan(cnn));
 if i~=0
  [p,q]=size(i);
   for x=1:q,
```

```
            cnn(i(x))=(cnn(i(x)-1)+cnn(i(x)+1))/2;
      end
    end
    cnn;
    cnn1=cnn(:,lcc+2:lc);
    disp('              inf')
    disp(' f(t) = SUMATORIO  [cn.e^(j.n.wo.t)]')
    disp('            - inf ')
    t=ti:.01:tf;
    ft1=0;
    ann1=cnn1+conj(cnn1);
    bnn1=cnn1-conj(cnn1);
    for i=1:n,
            ft=ann1(i)*cos(i*wo*t)+bnn1(i)*sin(i*wo*t);
            ft1=ft+ft1;
    end
      if ao~=0
    fo=ao*ones(size(ft1));
    ftt1=fo+ft1;
      else
    ftt1=ft1;
      end
  end
  figure
  hold on
  plot(t,ftt1)
  axis([(ti+tf-tt) (ti+tf+tt) min(0,min(ftt1)) 1.2*max(ftt1)])
  title('FUNCION COMPUESTA CON ARMONICAS')
  xlabel('tiempo[seg]')
  ylabel('X(t)')
  grid
  pause
  end
```

ANÁLISIS MEDIANTE LA TRANSFORMADA DE FOURIER

tfourier.m

```
function tfourier
labels = str2mat(...
  'FUNCIONES PREDETERMINADAS', ...
  'INGRESO DE LA TRANSFORMADA', ...
  'ESPECTRO DE FRECUENCIA', ...
  'RESPUESTA DE FRECUENCIA');
callbacks = [ ...
   'funcpf4      '
   'inftf4       '
   'espef4       '
   'respf4       ']';
choices1('T.FOURIER', 'ANALISIS EN FRECUENCIA MEDIANTE LA
TRANSFORMADA DE FOURIER', labels, callbacks);
```

funcpf4.m

```
%TRANSFORMADA DE FOURIER
%EL PROGRAMA VA A CONTENER:
%INGRESO DE LA FUNCION PREDETERMINADA Y GRAFICO
%GRAFICO DEL ESPECTRO DE FRECUENCIA DE LA FUNCION
close(gcf)
clc
disp('TRANSFORMADA DE FOURIER PARA FUNCIONES NO
PERIÓDICAS:  ')
disp(' 1. FUNCION PULSO SIMETRICA')
disp(' 2. FUNCION EXPONENCIAL')
disp(' 3. FUNCION TRIANGULAR SIMETRICA')
in=input('REALIZAR: ');
j=sqrt(-1);
if in==1
  disp(' FUNCION PULSO SIMETRICA')
```

```
disp(' ')
disp(' F(jw)=T*sin(0.5wT) / 0.5wT')
disp(' ')
F='(2*T)*(sin(w*T/2))/(w*T)';
T=input('INGRESE EL PERIODO T[seg]= ');
disp(' ')
ee=input('Máximo valor de w para visualizar ? ');
es=ee*10;
if es>=500
        es1=500;
else
        es1=es;
end
ta=-0.7*T:.01:-0.5*T-0.01;
xa=zeros(size(ta));
tb=-0.5*T:.01:0.5*T;
xb=ones(size(tb));
tc=0.5*T+0.01:0.01:0.7*T;
xc=zeros(size(tc));
tt=[ta,tb,tc];
xt=[xa,xb,xc];
else
if in==2
        disp('FUNCION EXPONENCIAL F(jw)=1/(jw+a)')
        F='1/(j*w+a)';
        a=input('INGRESE a = ');
        ee=input('Máximo valor de w para visualizar ? ');
        es=ee*10;
        if es>=500
                es1=500;
        else
                es1=es;
        end
```

```
        t=0:0.01:15;
        tt=t;
        xt=exp(-a*t);
        else
        if in==3
                disp(' FUNCION TRIANGULAR SIMETRICA')
                F='(8*((sin(0.25*w*T))^2)/(w^2*T))';
                T=input('INGRESE EL PERIODO T[seg]= ');
                ee=input('Máximo valor de w para visualizar ? ');
                es=ee*10;
                if es>=500
                        es1=500;
                else
                        es1=es;
                end
                tr=-0.5*T+0.01:0.01:0.5*T-0.01;
                xr=1-2*abs(tr)/T;
                ta=-0.7*T:0.01:-0.5*T;
                xa=zeros(size(ta));
                tc=0.5*T:0.01:0.7*T;
                xc=zeros(size(tc));
                xt=[xa,xr,xc];
                t=[ta,tr,tc];
                tt=t;
                        else
                        disp('ERROR FUERA DE RANGO ')
                        return
        end
  end
end
  ww=-50:.1:50;
  for k=1:1:1001,
        w=-50.1+0.1*k;
```

```
        ff(k)=eval(F);
  end
  ffo=ff(51);
% ESTE PROGRAMA PERMITE GRAFICAR EL ESPECTRO DE
%FRECUENCIA DE LA TRANSFORMADA DE FOURIER
ffr=real(ff);
ffi=imag(ff);
ff1=sqrt(ffr.^2+ffi.^2);
clc
es=ee*10;
if es>=500
  es1=500;
else
  es1=es;
end
ww1=ww(:,501-es1:501+es1);
ff11=ff1(:,501-es1:501+es1);
figure
hold on
subplot(211)
        plot(tt,xt)
        title('ESPECTRO DE FRECUENCIA CONTINUO')
        xlabel('t[seg]')
        ylabel('f(t)')
        grid
subplot(212)
        plot(ww1,ff11)
        %title('ESPECTRO DE FRECUENCIA CONTINUO')
        xlabel('[w]')
        ylabel('|F(jw)|')
        grid
        pause
end
```

inftf4.m

```
echo off
close
clc
disp('INGRESO DE LA TRANSFORMADA DE FOURIER ')
disp(' INGRESO DE DATOS:')
disp(' ')
j=sqrt(-1);
!edit tff3.m
tff3
load ff
load ww
end
```

tff3.m

```
function tff3
global w
%PROGRAMA PARA INGRESAR UNA FUNCION DE FOURIER F(jw)=F
%       Ejemplo
%       F='1/(2+w*j)'
%INGRESE F(jw) EN LA SIGUIENTE LINEA

  F='w\2*sin(3*w)'

%PONGA GUARDAR Y SALIR DEL EDITOR
ww=-50:.1:50;
for k=1:1:1001,
 w=-50.1+0.1*k;
        ff(k)=eval(F);
end
ffo=ff(51);
i=find(isnan(ff));
if i==1
```

```
  ff=(ff(50)+ff(52))/2
 end
 save ff ww
 end
 end
```

espef4.m

```
% ESTE PROGRAMA PERMITE GRAFICAR EL ESPECTRO DE
%FRECUENCIA DE LA TRANSFORMADA DE FOURIER
load ff
load ww
ffr=real(ff);
ffi=imag(ff);
ff1=sqrt(ffr.^2+ffi.^2);
clc
close(gcf)
ee=input('Máximo valor de w para visualizar ? ');
es=ee*10;
if es>=500
  es1=500;
else
  es1=es;
end
figure
ww1=ww(:,501-es1:501+es1);
ff11=ff1(:,501-es1:501+es1);
          plot(ww1,ff11)
title('ESPECTRO DE FRECUENCIA CONTINUO')
xlabel('[w]')
ylabel('|F(jw)|')
grid
pause
end
```

respf4.m

```
function respf4
labels = str2mat(...
   'INGRESO FUNCION DE TRANSFERENCIA', ...
   'DIAGRAMAS DE BODE.Lazo abierto', ...
   'DIAGRAMAS DE BODE.Lazo cerrado', ...
   'DIAGRAMA POLAR', ...
   'DIAGRAMA DE NICHOLS');
callbacks = [ ...
   'ftf4        '
   'bodef4       '
   'resonf4      '
   'polarf4      '
   'nicolf4      '];
choices('respf4', 'RESPUESTA DE FRECUENCIA', labels, callbacks);
```

ftf4.m

```
global num den
echo off
close
clear
clc
disp('ANALISIS DE RESPUESTA MEDIANTE LA TRANFORMADA DE FOURIER')
disp('INGRESO DE LA FUNCION DE TRANSFERENCIA ')
disp(' ')
disp('            G(s) = NUM(s) / DEN(s)')
disp(' ')
disp('EL NUMERADOR NO PUEDE SER DE ORDEN MAYOR QUE EL DENOMINADOR')
disp('                    b2.S²+b1.S+b0')
disp('            G(s) = -----------------')
disp('                    a3.S³+a2.S²+a1.S+a0')
```

```
disp(' ')
disp('INGRESE DE LA SIGUIENTE MANERA:   num=[b2 b1 bo]')
disp('                                  den=[a3 a2 a1 ao]')
disp(' ')
num=input('INGRESE SUS DATOS:    num= ');
disp(' ')
den=input('                      den= ');
%PASANDO A LA FORMA CANONICA CONTROLABLE
[a,b,c,d]=tf2ss(num,den)
i=0;
save num,save den
end
```

bodef4.m

```
clc
disp('DIAGRAMAS DE BODE. Lazo abierto')
bode(num,den)
 [gm,pm]=margin(num,den);
disp('MARGEN DE GANANCIA [dB]: ')
  mg=20*log10(gm)
disp('MARGEN DE FASE [grados]: ')
  mf=pm
pause
end
```

resonf4.m

```
clc
clf
hold on
disp('DIAGRAMAS DE BODE. Lazo cerrado')
bode(num,den);
[mag,fase,w]=bode(num,den);
faserad=fase*2*pi/360;
```

```
i=sqrt(-1);
gcl=mag.*exp(i*faserad)./...
(1+mag.*exp(i*faserad));
magc1=abs((gcl));
temfase=angle(gcl);
fasec1=unwrap(temfase)*360/(2*pi);
clf
hold on
subplot(211)
semilogx(w,20*log10(magc1))
grid
title('DIAGRAMA DE BODE.Lazo cerrado ')
xlabel('w[rad/seg]')
ylabel('Ganancia [dB]')
subplot(212)
semilogx(w,fasec1)
grid
xlabel('w [rad/seg]')
ylabel('Angulo [grados]')
%diagrama de bode en lazo cerrado para ver el margen de
%resonancia y el ancho de banda de la función.
%El valor de máxima resonancia Mr es un indicativo de la estabilidad relativa,
%se obtiene una buena respuesta transitoria si 0db<Mr<3db. (igual 1<Mr<1.4)
%Para encontrar el valor de máxima resonancia Mr, k es el
%k-ésimo elemento en el que se produce Mr.
mrdb=20*log10(magc1);
[Mrs,k]=max(mrdb);
%El valor de w en el que se produjo Mrs, que es el máximo valor
disp('MARGEN DE RESONANCIA [dB]')
Mrodb=20*log10(magc1(1));
Mr1=Mrs-Mrodb;
```

```
if Mr1<0
  Mr=0
else
    Mr=Mr1
  disp(' ')
  disp('FRECUENCIA DE RESONANCIA [rad/seg] ')
  wr=w(k)
  disp(' ')
end
pause
end
```

polarf4.m

```
%DIAGRAMAS DE NYQUIST
clf
clc
%num,den define la función de transferencia en lazo abierto
nyquist(num,den)
title('DIAGRAMA POLAR DE G(s)')
pause
%para determinar la estabilidad del sistema, se debe interpretar el gráfico
del %diagrama polar, si contiene el punto -1+j0. Este punto debe ser
rodeado o %contenido tantas veces como sea el número de polos de
G(s)H(s) que hayan %en el  seemiplano derecho del plano s, caso contrario
el sistema %es %inestable.Los sistemas analizados pueden ser demasiado
diferentes, %resultaría demasiado complejo tener un análisis
predeterminado con el %programa sislin.
end
```

nicolf4.m

```
%disp('DIAGRAMAS DE NICHOLS')
clc
%num,den define la función de transferencia en lazo abierto
```

```
ngrid('new')
nichols(num,den)
title('DIAGRAMA NICHOLS DE G(s)')
pause
end
```

ANÁLISIS MEDIANTE LA TRANSFORMADA RÁPIDA DE FOURIER

rfourier.m

```
function rfourier
labels = str2mat(...
  'FUNCIONES PREDETERMINADAS', ...
  'INGRESO DE FUNCIONES', ...
  'ESPECTRO DE FRECUENCIA ');
callbacks = [ ...
   'funcpf5       '
   'ifftf5        '
   'espef5        '];
choices1('FFT', 'ANALISIS EN FRECUENCIA CON LA TRANSFORMADA
RAPIDA DE FOURIER', labels, callbacks);
```

funcpf5.m

```
%TRANSFORMADA RAPIDA DE FOURIER
%EL PROGRAMA VA A CONTENER:
%INGRESO DE LA FUNCION PREDETERMINADA Y GRAFICO
%DE LA TRANSFORMADA RAPIDA DE FOURIER Y DEL ESPECTRO DE
%RECUENCIA
clc
close
disp('TRANSFORMADA RAPIDA DE FOURIER ')
disp('  ANALISIS DE:')
disp(' 1. FUNCION PULSO')
disp(' 2. FUNCION EXPONENCIAL ')
```

```
disp(' 3. FUNCION TRIANGULAR')
disp('')
in=input('REALIZAR: ');
if in==1
 disp('FUNCION PULSO SIMETRICA')
 disp('F(jw) = 2.sin(0.5*w*T)/w ')
 T1=input('INGRESE EL PERIODO  T[seg] = ');
 ta=-0.7*T1:0.01:-0.5*T1-0.01;
 xa=zeros(size(ta));
 tb=-0.5*T1:.01:0.5*T1;
 xb=ones(size(tb));
 tc=0.5*T1+0.01:0.01:0.7*T1;
 xc=zeros(size(tc));
 x=[xa,xb,xc];
 t=[ta,tb,tc];
 else
 if in==2
        disp(' FUNCION EXPONENCIAL')
        disp(' f(t)=e^(-at)')
        t=0:.001:10;
        a=input('CUÁL ES EL VALOR DE a ? ');
        x=exp(-a*t);
        else
        if in==3
                        disp(' FUNCION TRIANGULAR SIMETRICA')
                        disp(' ')
                        disp('        8*[sin(0.25*wT)]^2 ')
                        disp(' f(t) =----------------------')
                        disp('            (T.w^2)')
                        disp(' ')
                        T1=input('INGRESE EL PERIODO  T[seg] = ');
                        ta=-0.7*T1:0.01:-0.5*T1-0.01;
                        xa=zeros(size(ta));
```

```
                    tb=-0.5*T1:.01:0.5*T1;
                    xb=1-2*abs(tb)/T1;
                    tc=0.5*T1+0.01:0.01:0.7*T1;
                    xc=zeros(size(tc));
                    x=[xa,xb,xc];
                    t=[ta,tb,tc];

                    else
                        disp('ERROR FUERA DE RANGO')
                    return
        end
  end
end
[ft,ct]=size(t);
delta=(max(t)-min(t))/ct;
clc
val=0;
for n=1:100,
  val=n
  poten=ct/(2^n);
  if poten<1
        break
  end
end
ex=2^(val);
s=fft(x,2*ex);
subplot(211)
plot(t,x)
title('TRANSFORMADA RAPIDA DE FOURIER')
xlabel('t[seg]')
ylabel('f(t)')
grid
w=(1/2*delta)*[0:ex-1]/ex;
```

```
subplot(212)
plot(w,abs(s(1:ex)))
xlabel('[w]')
ylabel('Espectro de frecuencia fft(f)')
grid
pause
end
```

ifftf5.m

```
function ifftf5
%EL PROGRAMA VA A CONTENER:
%1.INGRESO DE UNA FUNCION PARA CALCULAR LA
TRANSFORMADA
%RAPIDA DE FOURIER
close
clc
disp(' INGRESO DE LA FUNCION f(t) ')
!edit fftf5.m
fftf5
end
```

fftf5.m

```
function fftf5
global t
%PROGRAMA PARA INGRESAR UNA FUNCION f(t)
%       Ejemplo
%       f ='sin(2*pi*5*t)'
%         t=0:0.001:5;
%INGRESE LA FUNCION f EN LA SIGUIENTE LINEA
      f='sin(2*pi*25*t)'

%DEFINA EL VECTOR t
      t=0:0.01:1;
```

```
%PONGA GUARDAR Y SALIR DEL EDITOR
t1=t;
[ft,ct]=size(t1);
for k=1:1:ct,
  t=t1(k);
  ff(k)=eval(f);
end
i=find(isnan(ff));
  if i~=0
   [p,q]=size(i);
     for x=1:q,
          ff(i(x))=(ff(i(x)-1)+ff(i(x)+1))/2;
     end
  end
save t1
save ff
end
```

espef5.m

```
% ESTE PROGRAMA PERMITE GRAFICAR EL ESPECTRO DE
%RECUENCIA DE LA TRANSFORMADA RAPIDA DE FOURIER
load ff
load t1
[ft1,ct1]=size(t1);
delta=(max(t1)-min(t1))/ct1;
clc
close(gcf)
 [ft1,ct1]=size(t1);
val=0;
for n=1:100,
  val=n;
  poten=ct1/(2^n)
```

```
  if poten<1
        break
  end
end
ex=2^val
s=fft(ff,2*ex);
plot(t1,ff)
title('TRANSFORMADA RAPIDA DE FOURIER')
xlabel('t[seg]')
ylabel('f(t)')
grid
pause
figure
w=(1/(2*delta))*(0:ex-1)/ex;
plot(w,abs([s(1:ex)']))
title('TRANSFORMADA RAPIDA DE FOURIER')
xlabel('[w]')
ylabel('Espectro de frecuencia fft(f)')
grid
pause
```

B.3. APLICACIONES DE SISTEMAS LINEALES

menu3.m

```
function menu3
labels = str2mat(...
  'OPERACION DE SEÑALES EN EL TIEMPO', ...
  'MODELOS DE SISTEMAS ', ...
  'CARACTERISTICAS ESTRUCTURALES DEL SISTEMA', ...
  'SISTEMAS DE DATOS MUESTREADOS', ...
  'ANALISIS DE FOURIER', ...
  'FILTRADO DE SEÑALES', ...
  'MODULACION DE SEÑALES', ...
```

```
    'SIMULINK', ...
    'ANALISIS VARIANDO UN PARAMETRO', ...
    'PROCESAMIENTO EN TIEMPO REAL');
  callbacks = [ ...
    'operap        '
    'modeloap      '
    'caracap       '
    'datamap       '
    'fourirap      '
    'filtroap      '
    'modulaap      '
    'simulap       '
    'variarap      '
    'timerap       '];
  choices1('APLICACIONES', 'APLICACIONES DE SISTEMAS LINEALES',
  labels, callbacks);
```

OPERACIÓN DE SEÑALES EN EL TIEMPO.

operap.m

```
  function operap
  % SE REALIZA EL INGRESO Y GRAFICO DE LA SEÑAL DE ENTRADA
  % SE REALIZA LA MANIPULACION DE LA SEÑAL DE ENTRADA
  %CONSECUTIVAMENTE
  labels = str2mat(...
    'LIMITES DE LA FUNCION DE ENTRADA', ...
    'INGRESO Y GRAFICO DE LA SEÑAL DE ENTRADA', ...
    'OPERACIONES CON LA SEÑAL DE ENTRADA');
  callbacks = [ ...
    'limitap       '
    'arbitap       '
    'manipap       '];
  choices1('OPERAP', 'OPERACION CON SEÑALES ARBITRARIAS', labels,
  callbacks);
```

limitap.m

```
%EL PROGRAMA VA A CONTENER:
%INGRESO DE LOS LIMITES PARA GRAFICAR LA FUNCION DE
%ENTRADA EN APLICACIONES.
close
clc
disp('LIMITES PARA MANEJAR LA FUNCION')
 li=input('INGRESE EL LIMITE INFERIOR: ');
 ls=input('INGRESE EL LIMITE SUPERIOR: ');
 t=li:.1:ls;
end
```

arbitap.m

```
%EL PROGRAMA VA A CONTENER:
%1.INGRESO DE LA FUNCION ARBITRARIA
%2.GRAFICO DE LA FUNCION ARBITRARIA
%1. PROGRAMA PARA GENERAR FUNCIONES ARBITRARIAS
close
disp(' INGRESE LA FUNCION  POR INTERVALOS')
xo=i;
clear f ft
for e=1:20,
disp(' TIPO DE FUNCION A INGRESAR')
disp(' ')
disp(' 1. FUNCION PASO ')
disp(' 2. FUNCION RAMPA ')
disp(' 3. FUNCION SINUSOIDAL ')
disp(' 4. FUNCION EXPONENCIAL ')
disp(' 5. FINALIZAR')
in=input('REALIZAR: ');
if in==1
 disp(' FUNCION PASO')
 kp=input('INGRESE LA GANANCIA: ');
```

```
to=input('INGRESE EL LIMITE INFERIOR DEL INTERVAL0: ');
f=kp*stepfun(t,to);
else
if in==2
        disp('FUNCION RAMPA')
        kr=input('INGRESE LA GANANCIA: ');
        tn=input('INGRESE EL LIMITE INFERIOR DEL INTERVAL0: ');
        f=kr*(t-tn).*stepfun(t,tn);
        else
        if in==3
                disp(' FUNCION SINUSOIDAL')
                ks=input('INGRESE LA GANANCIA: ');
        tos=input('INGRESE EL LIMITE INFERIOR DEL INTERVAL0: ');
                [nft,nct]=size(t);
                t2=0:.1:nct;
                f1=zeros(1,(tos-li)*10);
                f2=ks*sin(t2);
                [nft,nct]=size(t);
                fx=[f1,f2];
                f=fx(:,1:nct);
                else
                if in==4
                        disp(' FUNCION EXPONENCIAL')
                        ke=input('INGRESE LA GANANCIA: ');
        toe=input('INGRESE EL LIMITE INFERIOR DEL INTERVAL0: ');
                        [nft,nct]=size(t);
                        t2=0:.1:nct;
                        f1=zeros(1,(toe-li)*10);
                        f2=ke*exp(-t2);
                        fx=[f1,f2];
                        f=fx(:,1:nct);
                        else
                        if in==5
```

```
                    e=20;
                % FUNCION ARBITRARIA EN FORMA DE VECTOR t y rt
                %2. GRAFICO DE LA FUNCION ARBITRARIA
                rtap=ftap;
                plot(t,rtap),title('FUNCION ARBITRARIA')
                  axis([1.2*(-0.5+min(t)) 1.2*(0.3+max(t)) 1.2*(-0.5+min(rtap))
                                                    1.2*(0.5+max(rtap))])
                xlabel('tiempo[seg]')
                ylabel('X(t)')
                grid
                pause
                break
                else
                      disp('ERROR FUERA DE RANGO')
                end
                end
            end
      end
end
    if e==1
      ft=f;
    else
      if e<20
            ft=ft+f;
      end
    end
    ftap=ft;
    rtap=ft;
    end
```

manipap.m

```
    function manipap
```

```
% SE REALIZA LA MANIPULACION DE LA SEÑAL
CONSECUTIVAMENTE
%COMO DESPLAZAMIENTO, TRANSPOSICION
labels = str2mat(...
  'DESPLAZAMIENTO', ...
  'TRANSPOSICION', ...
  'ESCALAMIENTO');
callbacks = [ ...
   'desplaap     '
   'transpap     '
   'escalaap     '];
choices1('MANIPULAR', 'OPERACIONES EN EL TIEMPO CON SEÑALES',
labels, callbacks);
```

desplaap.m

```
%EL PROGRAMA VA A CONTENER:
%UNA FUNCION ARBITRARIA QUE SE VA A DESPLAZAR
%EN EL TIEMPO, HACIA LA DERECHA O HACIA LA IZQUIERDA
close
clc
disp('  DESPLAZAMIENTO DE SEÑALES ')
disp('  t-a: DESPLAZA a UNIDADES A LA DERECHA')
disp('  t+a: DESPLAZA a UNIDADES A LA IZQUIERDA')
disp('   DESPALAZAR HACIA ? ')
disp('   1. LA IZQUIERDA ')
disp('   2. LA DERECHA ')
in=input('   REALIZAR:');
if in==1
  disp(' ')
  else
  if in==2
          disp(' ')
          else
```

```
                disp('ERROR FUERA DE RANGO')
                break
        end
end
disp(' ')
un1=input('CUANTAS UNIDADES ? ');
un=abs(un1);
if in==1
 d=-un;
 else
 if in==2
 d=un;
        else
                disp('ERROR FUERA DE RANGO')
                break
 end
end
[md,nd]=size(rtap);
matriz=d*ones(1,nd);
t1=t+matriz;
 subplot(2,1,1)
 plot(t,rtap),title('FUNCION ARBITRARIA * DESPLAZAMIENTO')
        axis([1.2*(-0.5+(min(min(t),min(t1))))  1.2*(0.5+(max(max(t),max(t1))))
                                1.2*(-0.5+(min(rtap))) 1.2*(0.5+(max(rtap)))])
 xlabel('tiempo[seg]')
 ylabel('X(t)')
 grid
 subplot(2,1,2)
 plot(t1,rtap)
        axis([1.2*(-0.5+(min(min(t),min(t1))))  1.2*(0.5+(max(max(t),max(t1))))
                                1.2*(-0.5+(min(rtap))) 1.2*((0.5+max(rtap)))])
 xlabel('tiempo[seg]')
 ylabel('Xdesplazada(t)')
```

```
  grid
  pause
t=t1;
end
```

transpap.m

```
%EL PROGRAMA VA A CONTENER:
%UNA FUNCION ARBITRARIA QUE SE VA A TRANSPONER EN EL
TIEMPO
close
clc
rt=rtap;
t1=-t;
r1=(rot90(rt))';
  subplot(2,1,1)
  plot(t,rt),title('FUNCION ARBITRARIA * TRANSPOSICION')
     axis([1.2*(-0.3+min(min(t),min(t1)))  1.2*(0.5+max(max(t),max(t1))) 1.2*(-
                                                0.3+min(rt)) 1.2*(0.3+max(rt))])
  xlabel('tiempo[seg]')
  ylabel('X(t)')
  grid
  subplot(2,1,2)
  plot(t1,rt)
     axis([1.2*(-0.3+min(min(t),min(t1)))  1.2*(0.3+max(max(t),max(t1))) 1.2*(-
                                                0.3+min(rt)) 1.2*(0.3+max(rt))])
  xlabel('tiempo[seg]')
  ylabel('Xtranspuesta(t)')
  grid
  pause
t=t1;
rtap=rt;
end
```

escalaap.m

```
function escalaap
labels = str2mat(...
  'ESCALAR EN TIEMPO', ...
  'ESCALAR EN AMPLITUD');
callbacks = [ ...
   'escalapt      '
   'escalapa      '];
choices1('ESCALAR', 'ESCALAR LA FUNCION', labels, callbacks);
```

escalapt.m

```
%EL PROGRAMA VA A CONTENER:
%UNA FUNCION ARBITRARIA QUE SE VA A ESCALAR EN TIEMPO
close
clc
disp('  ESCALAR LA FUNCION EN TIEMPO ')
disp('x(a.t) ')
est1=input('CUANTAS UNIDADES a ? ');
rt=rtap;
est=abs(est1);
rt1=rt;
t1=t./est;
subplot(2,1,1)
plot(t,rt),title('FUNCION ARBITRARIA * ESCALAMIENTO')
    axis([1.1*(-0.3+min(min(t),min(t1)))  1.1*(0.3+max(max(t),max(t1))) 1.2*(-
                0.3+min(min(rt),min(rt1))) 1.2*(0.3+max(max(rt),max(rt1)))])
xlabel('tiempo[seg]')
ylabel('X(t)')
grid
subplot(2,1,2)
plot(t1,rt1)
    axis([1.1*(-0.3+min(min(t),min(t1)))  1.1*(0.3+max(max(t),max(t1))) 1.2*(-
                0.3+min(min(rt),min(rt1))) 1.2*(0.3+max(max(rt),max(rt1)))])
```

```
xlabel('tiempo[seg]')
ylabel('Xamplitud(t)')
grid
pause
t1=t;
rtap=rt1;
end
```

escalapa.m

```
%EL PROGRAMA VA A CONTENER:
%UNA FUNCION ARBITRARIA QUE SE VA A ESCALAR EN AMPLITUD
close
clc
disp('  ESCALAR LA FUNCION EN AMPLITUD ')
disp(' c.x(t()')
es1=input('CUANTAS UNIDADES c? ');
rt=rtap;
es=abs(es1);
t1=t;
rt1=rt.*es;
subplot(2,1,1)
plot(t,rt),title('FUNCION ARBITRARIA * ESCALAMIENTO')
    axis([1.1*(-0.3+min(min(t),min(t1)))  1.1*(0.3+max(max(t),max(t1))) 1.2*(-
                0.3+min(min(rt),min(rt1))) 1.2*(0.3+max(max(rt),max(rt1)))])
xlabel('tiempo[seg]')
ylabel('X(t)')
grid
subplot(2,1,2)
plot(t1,rt1)
    axis([1.1*(-0.3+min(min(t),min(t1)))  1.1*(0.3+max(max(t),max(t1))) 1.2*(-
                0.3+min(min(rt),min(rt1))) 1.2*(0.3+max(max(rt),max(rt1)))])
xlabel('tiempo[seg]')
ylabel('Xamplitud(t)')
```

```
grid
pause
t=t1;
rtap=rt1;
end
```

MODELOS DE SISTEMAS.

modeloap.m

```
function modeloap
labels = str2mat(...
   'INGRESO DE MODELOS', ...
   'CAMBIO DE MODELOS', ...
   'COMPOSICIÓN DE MODELOS');
callbacks = [ ...
   'inmodap      '
   'cammodap     '
   'commodap     '];
choices1('modeloap','MODELOS DE SISTEMAS
LINEALES',labels,callbacks);
```

inmodap.m

```
function inmodap
labels = str2mat(...
   'INGRESO MODELO CONTINUO', ...
   'INGRESO MODELO DISCRETO');
callbacks = [ ...
   'inmodcap     '
   'inmoddp      '];
choices1('inmodap','INGRESO DEL MODELO DEL
SISTEMA',labels,callbacks);
```

inmodcap.m

```
echo off
close
clc
disp('INGRESO DEL MODELO CONTINUO DEL SISTEMA')
disp('1. COMO FUNCION DE TRANSFERENCIA ')
disp('2. COMO VARIABLES DE ESTADO ')
in=input('REALIZAR: ');
if in==1
clc
disp('1. INGRESO DE LA FUNCION DE TRANSFERENCIA ')
disp('            H(s) = NUM(s) / DEN(s)')
disp('EL NUMERADOR NO PUEDE SER DE ORDEN MAYOR QUE EL
DENOMINADOR')
disp('                    b2.S²+b1.S+b0')
disp('            H(s) = -----------------')
disp('                    a3.S³+a2.S²+a1.S+a0')
disp('INGRESE DE LA SIGUIENTE MANERA:   num=[b2 b1 bo]')
disp('                          den=[a3 a2 a1 ao]')
numc=input('INGRESE SUS DATOS:    num= ');
denc=input('                  den= ');
disp(' ')
%PASANDO A LA FORMA CANONICA CONTROLABLE
[ac,bc,cc,dc]=tf2ss(numc,denc);
disp(' ')
disp('SISTEMA EXPRESADO MEDIANTE MATRICES DE ESTADO SON: ')
disp(' ')
A=ac
B=bc
C=cc
D=dc
i=0;
  else
```

```
if in==2
clc
disp('    INGRESO DE LAS MATRICES DE ESTADO ')
disp('    X` = A.X + B.u ')
disp(' ')
disp('    Y  = C.X + D.u ')
disp('    A,B,C,D SON LAS MATRICES DE ESTADO ')
        ac=input('ingrese la matriz   A=');
        bc=input('                    B=');
        cc=input('                    C=');
        dc=input('                    D=');
        i=0;
        [fb,cbc]=size(bc)
        [numc,denc]=ss2tf(ac,bc,cc,dc,cbc);
disp('SISTEMA EXPRESADO COMO FUNCION DE TRANSFERENCIA')
        NUMERADOR=numc
        DENOMINADOR=denc
                else
                disp('ERROR FUERA DE RANGO')
                break
end
end
a=ac;,b=bc;,c=cc;,d=dc;
num=numc;,den=denc;
end
```

inmoddp.m

```
echo off
close
clc
disp('INGRESO DEL MODELO DISCRETO DEL SISTEMA')
disp('1. COMO FUNCION DE TRANSFERENCIA ')
disp('2. COMO VARIABLES DE ESTADO ')
```

```
in=input('REALIZAR: ');
if in==1
clc
disp('1. INGRESO DE LA FUNCION DE TRANSFERENCIA ')
disp(' ')
disp('            H(z) = NUM(z) / DEN(z)')
disp(' ')
disp('EL NUMERADOR NO PUEDE SER DE ORDEN MAYOR QUE EL
DENOMINADOR')
disp(' ')
disp('                    b2.z²+b1.z+b0')
disp('            H(z) = -----------------')
disp('                    a3.z³+a2.z²+a1.z+a0')
disp('INGRESE DE LA SIGUIENTE MANERA:   num=[b2 b1 bo]')
disp('                          den=[a3 a2 a1 ao]')
disp(' ')
numd=input('INGRESE SUS DATOS:    num= ');
dend=input('                  den= ');
%PASANDO A LA FORMA CANONICA CONTROLABLE
[ad,bd,cd,dd]=tf2ss(numd,dend);
disp('SISTEMA EXPRESADO MEDIANTE MATRICES DE ESTADO SON: ')
Ad=ad
Bd=bd
Cd=cd
Dd=dd
i=0;
 else
 if in==2
 clc
 disp('    INGRESO DE LAS MATRICES DE ESTADO ')
 disp('    X` = Ad.X + Bd.u ')
 disp('    Y  = Cd.X + Dd.u ')
 disp('    A,B,C,D SON LAS MATRICES DE ESTADO ')
```

```
            ad=input('ingrese la matriz   Ad=');
            bd=input('                    Bd=');
            cd=input('                    Cd=');
            dd=input('                    Dd=');
            i=0;
            [fbd,cbd]=size(bd)
            [numd,dend]=ss2tf(ad,bd,cd,dd,cbd);
      disp('SISTEMA EXPRESADO COMO FUNCION DE TRANSFERENCIA')
            NUMERADOR=numd
            DENOMINADOR=dend
                  else
                  disp('ERROR FUERA DE RANGO')
                  break
      end
    end
    end
```

cammodap.m

```
    function cammodap
    labels = str2mat(...
       'DE CONTINUO A DISCRETO', ...
       'DE DISCRETO A CONTINUO');
    callbacks = [ ...
       'camcdap       '
       'camdcap       '];
    choices1('cammodap','CAMBIO          DE          MODELOS          DE
    SISTEMAS',labels,callbacks);
```

camcdap.m

```
    echo off
    close
    clc
    disp('   CAMBIO DE MODELO CONTINUO A DISCRETO')
```

```
disp('        METODO DE DISCRETIZACION ZOH')
ts=input(' TIEMPO DE DISCRETIZACÓN  [seg] ?  ');
[ad,bd,cd,dd] = c2dm(ac,bc,cc,dc,ts,'zoh');
[numd,dend] = c2dm(numc,denc,ts,'zoh');
clc
disp('    MODELO DISCRETO DEL SISTEMA EXPRESADO COMO:')
disp(' FUNCION DE TRANSFERENCIA: ')
NUMERADOR=numd
DENOMINADOR=dend
disp('PRESIONE UN TECLA PARA CONTINUAR ')
pause
clc
disp(' VARIABLES DE ESTADO: ')
disp(' ')
Ad=ad,Bd=bd,Cd=cd,Dd=dd
pause
```

camdcap.m

```
echo off
close
clc
disp('    CAMBIO DE MODELO DISCRETO A CONTINUO')
disp('        METODO DE CONVERSION ZOH')
ts=input(' TIEMPO DE CONVERSION  [seg] ?  ');
[ac,bc,cc,dc] = d2cm(ad,bd,cd,dd,ts,'zoh');
[numc,denc] = d2cm(numd,dend,ts,'zoh');
disp(' ')
clc
disp('    MODELO CONTINUO DEL SISTEMA EXPRESADO COMO:')
disp(' FUNCION DE TRANSFERENCIA: ')
NUMERADOR=numc
DENOMINADOR=denc
disp('PRESIONE UN TECLA PARA CONTINUAR ')
```

```
pause
clc
disp(' VARIABLES DE ESTADO: ')
disp(' ')
A=ac,B=bc,C=cc,D=dc
pause
end
```

commodap.m

```
function commodap
labels = str2mat(...
   'MODELOS EN SERIE', ...
   'MODELOS EN PARALELO');
callbacks = [ ...
   'serieap      '
   'cascadap     '];
choices1('commodap','COMPOSICION DE MODELOS',labels,callbacks);
```

serieap.m

```
function serieap
labels = str2mat(...
   'MODELO SIN REALIMENTACION EN SERIE', ...
   'MODELO CON REALIMENTACION EN SERIE');
callbacks = [ ...
   'nserieap     '
   'rserieap     '];
choices1('serieap','MODELOS DE SISTEMAS EN SERIE',labels,callbacks);
```

nserieap.m

```
echo off
close
clc
disp('              MODELO SERIE SIN REALIMENTACION')
```

```
disp(' ')
disp('      u --->[Sistema 1]--->[Sistema 2]---->....--->[Sistema n]> y ')
disp(' ')
numero=input('CUANTOS SISTEMAS ESTAN EN SERIE ? ');
if numero==1
        disp('ERROR NO SE PUEDE TENER SOLO UN SISTEMA')
        return
else
 for i=1:numero-1
    if i==1
        disp('INGRESE LOS SISTEMAS COMO FUNCION DE
        TRANSFERENCIA ')
        fprintf('SISTEMA N. %g',i)
        num1=input('Numerador = ');
        den1=input('Denominador = ');
        fprintf('SISTEMA N. %g',i+1)
        num2=input('Numerador = ');
        den2=input('Denominador = ');
        [num,den]=series(num1,den1,num2,den2);
    else
        fprintf('SISTEMA N. %g',i+1)
        numa=input('Numerador = ');
        dena=input('Denominador = ');
        [num,den]=series(num,den,numa,dena)
 end
end
end
 disp('SISTEMA TOTAL ')
 NUMERADOR=num
 DENOMINADOR=den
pause
end
```

rserieap.m

```
echo off
close
clc
disp('          MODELO SERIE CON REALIMENTACION NEGATIVA')
disp(' ')
disp(' u --->o--->[Sistema 1]--->[Sistema 2]---->....--->[Sistema n]---+---> y ')
disp('     |                                                    |')
disp('     +------------<--------------[H]----------<----------------+')
disp(' ')
disp('INGRESE H COMO FUNCION DE TRANSFERENCIA ')
disp(' ')
numh=input('num(H) = ');
denh=input('den(H) = ');
disp(' ')
numero=input('CUANTOS SISTEMAS ESTAN EN SERIE ? ');
if numero==1
        disp('ERROR NO SE PUEDE TENER SOLO UN SISTEMA')
        return
else
 for i=1:numero-1
    if i==1
   disp('INGRESE LOS SISTEMAS COMO FUNCION DE
   TRANSFERENCIA ')
        fprintf('SISTEMA N. %g',i)
        num1=input('Numerador = ');
        den1=input('Denominador = ');
        fprintf('SISTEMA N. %g',i+1)
        num2=input('Numerador = ');
        den2=input('Denominador = ');
        [num,den]=series(num1,den1,num2,den2);
    else
        fprintf('SISTEMA N. %g',i+1)
```

```
            numa=input('Numerador = ');
            dena=input('Denominador = ');
            disp(' ')
            [num,den]=series(num,den,numa,dena);
      end
    end
    end
    [num,den]=feedback(num,den,numh,denh,-1);
      disp('SISTEMA TOTAL REALIMENTADO ')
      NUMERADOR=num
      DENOMINADOR=den
    pause
    end
```

cascadap.m

```
    function cascadap
    labels = str2mat(...
       'MODELO SIN REALIMENTACION EN PARALELO', ...
       'MODELO CON REALIMENTACION EN PARALELO');
    callbacks = [ ...
       'ncascap      '
       'rcascap      '];
    choices1('cascadap','MODELOS DE SISTEMAS EN
    PARALELO',labels,callbacks);
```

ncascap.m

```
    echo off
    close
    clc
    disp(' ')
    disp('          MODELOS EN PARALELO SIN REALIMENTACION')
    disp(' ')
    disp('          +---->[Sistema 1]----+')
```

```
disp('        u----->+                    o------>y')
disp('              +---->[Sistema 2]----+')
disp('              +---->[Sistema 3]----+')
disp('              +....................+  ')
disp('              +---->[Sistema n]----+')
numero=input('CUANTOS SISTEMAS ESTAN EN PARALELO ? ');
if numero==1
                  disp('ERROR NO SE PUEDE TENER SOLO UN SISTEMA')
          return
else
 for i=1:numero-1
    if i==1
  disp('INGRESE     LOS     SISTEMAS     COMO     FUNCION     DE
  TRANSFERENCIA ')
 fprintf('SISTEMA N. %g',i)
 disp(' ')
 num1=input('Numerador = ');
 den1=input('Denominador = ');
 disp(' ')
 disp(' ')
 fprintf('SISTEMA N. %g',i+1)
 disp(' ')
 num2=input('Numerador = ');
 den2=input('Denominador = ');
 disp(' ')
 [num,den]=parallel(num1,den1,num2,den2);
   else
 disp(' ')
 fprintf('SISTEMA N. %g',i+1)
 disp(' ')
 numa=input('Numerador = ');
 dena=input('Denominador = ');
 disp(' ')
```

```
  [num,den]=parallel(num,den,numa,dena)
  end
end
end
  disp('SISTEMA TOTAL ')
  NUMERADOR=num
  DENOMINADOR=den
pause
end
```

rcascap.m

```
echo off
close
clc
        disp('        MODELOS EN PARALELO CON REALIMENTACION
                                                        NEGATIVA')
disp(' ')
disp('            +---->[Sistema 1]----+')
disp('   u---o------->+              o-------------+------>y')
disp('      |      +---->[Sistema 2]----+          |')
disp('      |      +---->[Sistema 3]----+          |')
disp('      |      +....................+          |')
disp('      |      +---->[Sistema n]----+          |')
disp('      |                                      |')
disp('      +----------<--------[H]--------<------------+')
disp(' ')
disp('INGRESE H COMO FUNCION DE TRANSFERENCIA ')
disp(' ')
numh=input('num(H) = ');
denh=input('den(H) = ');
numero=input('CUANTOS SISTEMAS ESTAN EN PARALELO ? ');
if numero==1
```

```
            disp(' ')
            disp('ERROR NO SE PUEDE TENER SOLO UN SISTEMA')
            return
else
  for i=1:numero-1
     if i==1
   disp('INGRESE LOS SISTEMAS COMO FUNCION DE
   TRANSFERENCIA ')
            disp(' ')
            fprintf('SISTEMA N. %g',i)
            disp(' ')
            num1=input('Numerador = ');
            den1=input('Denominador = ');
            fprintf('SISTEMA N. %g',i+1)
            num2=input('Numerador = ');
            den2=input('Denominador = ');
            [num,den]=series(num1,den1,num2,den2);
     else
            fprintf('SISTEMA N. %g',i+1)
            numa=input('Numerador = ');
            dena=input('Denominador = ');
            [num,den]=parallel(num,den,numa,dena);
  end
end
end
[num,den]=feedback(num,den,numh,denh,-1);
  disp('SISTEMA TOTAL REALIMENTADO ')
disp(' ')
  NUMERADOR=num
  DENOMINADOR=den
pause
end
```

CARACTERÍSTICAS ESTRUCTURALES DEL SISTEMA

caracap.m

```
function caracap
labels = str2mat(...
   'INGRESO DEL SISTEMA', ...
   'ESTABILIDAD', ...
   'CONTROLABILIDAD', ...
   'OBSERVABILIDAD', ...
   'TRANSFORMACION DE SEMEJANZA');
callbacks = [ ...
   'inmodcap      '
   'estabap       '
   'contrbap      '
   'obserap       '
   'tsemejap      '];
choices1('caracap','CARACTERISTICAS ESTRUCTURALES DEL
SISTEMA',labels,callbacks);
```

inmodcap.m

```
echo off
close
clc
disp('INGRESO DEL MODELO CONTINUO DEL SISTEMA')
disp('1. COMO FUNCION DE TRANSFERENCIA ')
disp('2. COMO VARIABLES DE ESTADO ')
in=input('REALIZAR: ');
if in==1
clc
disp('1. INGRESO DE LA FUNCION DE TRANSFERENCIA ')
disp(' ')
disp('          H(s) = NUM(s) / DEN(s)')
```

```
disp(' ')
disp('EL NUMERADOR NO PUEDE SER DE ORDEN MAYOR QUE EL
DENOMINADOR')
disp('                    b2.S²+b1.S+b0')
disp('            H(s) = -----------------')
disp('                    a3.S³+a2.S²+a1.S+a0')
disp(' ')
disp('INGRESE DE LA SIGUIENTE MANERA:   num=[b2 b1 bo]')
disp('                          den=[a3 a2 a1 ao]')
numc=input('INGRESE SUS DATOS:    num= ');
denc=input('                  den= ');
disp(' ')
%PASANDO A LA FORMA CANONICA CONTROLABLE
[ac,bc,cc,dc]=tf2ss(numc,denc);
disp('SISTEMA EXPRESADO MEDIANTE MATRICES DE ESTADO SON: ')
disp(' ')
A=ac,B=bc,C=cc,D=dc,i=0;
 else
 if in==2
 clc
 disp('    INGRESO DE LAS MATRICES DE ESTADO ')
 disp('    X` = A.X + B.u ')
 disp(' ')
 disp('    Y  = C.X + D.u ')
 disp('    A,B,C,D SON LAS MATRICES DE ESTADO ')
        ac=input('ingrese la matriz   A=');
        bc=input('                B=');
        cc=input('                C=');
        dc=input('                D=');
        i=0;
        [fb,cbc]=size(bc)
        [numc,denc]=ss2tf(ac,bc,cc,dc,cbc);
 disp('SISTEMA EXPRESADO COMO FUNCION DE TRANSFERENCIA')
```

```
            NUMERADOR=numc
            DENOMINADOR=denc
                else
                disp('ERROR FUERA DE RANGO')
                break
  end
end
a=ac;,b=bc;,c=cc;,d=dc;,num=numc;,den=denc;
save ac ,save bc,save cc,save dc,se numc,save denc,save a ,save b ,save
c save d ,save num,save den
end
```

estabap.m

```
function estabap
load num
load den
labels = str2mat(...
  'ANALISIS DE RESPUESTA EN EL TIEMPO', ...
  'LUGAR GEOMETRICO DE LAS RAICES ', ...
  'ANALISIS EN EL PLANO "S" ', ...
  'DIAGRAMAS DE BODE-LAZO ABIERTO', ...
  'DIAGRAMAS DE BODE-LAZO CERRADO', ...
  'DIAGRAMA POLAR', ...
  'DIAGRAMA DE NICHOLS');


callbacks = [ ...
  'resp1ap      '
  'lgrtl        '
  'respf1       '
  'bodef4       '
  'resonf4      '
  'polarf4      '
```

```
    'nicolf4        '];
choices1('ESTABILIDAD', 'ANALISIS DE ESTABILIDAD', labels, callbacks);
```

resp1ap.m

```
clc
close(gcf)
disp('ANALISIS DE LA RESPUESTA A UNA FUNCION PASO')
%PARA EVALUAR EL POLINOMIO EN 0
%SE OBTIENE EL VALOR DE LA FUNCION EN ESTADO ESTABLE
%POLYVAL(F,S) EVALUA LA FUNCION F EN EL PUNTO S
disp('VALOR FINAL DE LA FUNCION')
  vf=polyval(num,0)/polyval(den,0)
%RESPUESTA A UNA FUNCION PASO
[y,x,t]=step(num,den);
%DEL VECTOR y QUE SE OBTIENE COMO RESPUESTA A LA FUNION
%PASO Y ES EL VALOR MAXIMO DE y - k ES EL NUMERO DE
COLUMNA %EN EL QUE SE PRODUCE EL VALOR MAXIMO DE y
[Y,k]=max(y);
%PORCENTAJE DEL MAXIMO SOBREIMPULSO
disp('PORCENTAJE DE MAXIMO SOBREIMPULSO')
Mp=100*(Y-vf)/vf;
if Mp<0
  Mp=0
else
  Mp=Mp
  %EL TIEMPO EN EL QUE SE PRODUCE EL MAXIMO SOBREIMPULSO
%ES EL DEL VALOR K
  disp('TIEMPO PICO [seg]')
  tpico=t(k)
end
%PARA ENCONTRAR EL TIEMPO DE ESTABLECIMIENTO
n=1;
  while y(n)<0.1*vf,n=n+1;
```

```
  end
 m=1;
  while y(m)<0.9*vf,m=m+1;
  end
 %disp('TIEMPO DE RESPUESTA DEL SISTEMA [seg]')
 %tresp=t(m)-t(n)
 l=length(t);
  while(y(l)>.98*vf)&(y(l)<1.02*vf);
  l=l-1;
  end
 disp('TIEMPO DE ESTABLECIMIENTO DEL SISTEMA [seg]')
  ts=t(l)
 figure
 hold on
 step(num,den)
 title('RESPUESTA A UNA FUNCION PASO')
 grid
 pause
 end
```

Se utilizan las rutinas lgrtl.m, respf1.m, bodef4.m, resonf4.m, polarf4.m y nicolf4.m descritas en el tema: Análisis de sistemas lineales en el dominio del tiempo.

contrbap.m

```
 echo off
 close(gcf)
 clc
 [ab,bb,cb,db]=canon(ac,bc,cc,dc,'companion');
 disp('SISTEMA EN FORMA CANONICA CONTROLABLE')
 A=ab'
 B=cb'
 disp('PRESIONE UNA TECLA PARA CONTINUAR ')
```

```
pause
clc
C=bb'
D=db'
disp('MATRIZ DE CONTROLABILIDAD ')
disp(' ')
con=ctrb(ab,bb);
Co=con
con1=det(con);
if con1==0
  disp('EL SISTEMA NO ES CONTOLABLE')
else
  disp('EL SISTEMA ES CONTROLABLE')
end
pause
```

obserap.m

```
echo off
close(gcf)
clc
[ab,bb,cb,db,T]=canon(ac,bc,cc,dc, 'companion');
disp('SISTEMA EN FORMA CANONICA OBSERVABLE')
A=ab,B=bb
disp('PRESIONE UNA TECLA PARA CONTINUAR')
pause
clc
C=cb,D=db
obs=obsv(ab,cb);
obs1=det(obs);
disp('MATRIZ DE OBSERVABILIDAD ')
Ob=obs
if obs1==0
  disp('EL SISTEMA NO ES OBSERVABLE')
```

```
else
  disp('EL SISTEMA ES OBSERVABLE')
end
pause
end
```

tsemejap.m

```
echo off
close
clc
[ab,bb,cb,db,T]=canon(ac,bc,cc,dc,'modal');
disp('SISTEMA EN FORMA MODAL')
A=ab',B=cb'
disp(' ')
disp(' ')
disp('PRESIONE UNA TECLA PARA CONTINUAR')
pause
clc
C=bb',D=db'
  disp('LA MATRIZ DE TRANSFORMACION DE SEMEJANZA ES:')
T
pause
end
```

SISTEMAS DE DATOS MUESTREADOS

datamap.m

```
function datamap
labels = str2mat(...
   'INGRESO DEL SISTEMA', ...
   'RESPUESTA DEL SISTEMA', ...
   'CARACTERISTICAS ESTRUCTURALES');
```

```
callbacks = [ ...
   'idatamap      '
   'rdatamap      '
   'cdatamap      '];
choices1('datamap','SISTEMAS DE DATOS
MUESTREADOS',labels,callbacks);
```

idatamap.m

```
echo off
close
clc
disp('INGRESO DEL MODELO CONTINUO DEL SISTEMA')
disp('1. COMO FUNCION DE TRANSFERENCIA ')
disp('2. COMO VARIABLES DE ESTADO ')
in=input('REALIZAR: ');
if in==1
clc
disp('1. INGRESO DE LA FUNCION DE TRANSFERENCIA ')
disp('          H(s) = NUM(s) / DEN(s)')
disp('EL NUMERADOR NO PUEDE SER DE ORDEN MAYOR QUE EL
DENOMINADOR')
disp(' ')
disp('                    b2.S²+b1.S+b0')
disp('          H(s) = -----------------')
disp('                   a3.S³+a2.S²+a1.S+a0')
disp(' ')
disp('INGRESE DE LA SIGUIENTE MANERA:   num=[b2 b1 bo]')
disp('                          den=[a3 a2 a1 ao]')
disp(' ')
numc=input('INGRESE SUS DATOS:    num= ');
disp(' ')
denc=input('                  den= ');
disp(' ')
```

```
%PASANDO A LA FORMA CANONICA CONTROLABLE
[ac,bc,cc,dc]=tf2ss(numc,denc);
disp('SISTEMA EXPRESADO MEDIANTE MATRICES DE ESTADO SON: ')
disp(' ')
A=ac,B=bc,C=cc,D=dc,i=0;
 else
 if in==2
 clc
 disp('    INGRESO DE LAS MATRICES DE ESTADO ')
 disp('    X` = A.X + B.u ')
 disp('    Y  = C.X + D.u ')
 disp('    A,B,C,D SON LAS MATRICES DE ESTADO ')
        ac=input('ingrese la matriz   A=');
        bc=input('                B=');
        cc=input('                C=');
        dc=input('                D=');
        i=0;
        [fb,cbc]=size(bc)
        [numc,denc]=ss2tf(ac,bc,cc,dc,cbc);
 disp('SISTEMA EXPRESADO COMO FUNCION DE TRANSFERENCIA')
        NUMERADOR=numc
        DENOMINADOR=denc
                else
                disp('ERROR FUERA DE RANGO')
                break
 end
end
clc
disp('   CAMBIO DE MODELO CONTINUO A DISCRETO')
disp('      METODO DE DISCRETIZACION ZOH')
ts=input(' TIEMPO DE DISCRETIZACÓN  [seg] ?  ');
[ad,bd,cd,dd] = c2dm(ac,bc,cc,dc,ts,'zoh');
[numd,dend] = c2dm(numc,denc,ts,'zoh');
```

```
clc
disp('    MODELO DISCRETO DEL SISTEMA EXPRESADO COMO:')
disp(' FUNCION DE TRANSFERENCIA: ')
NUMERADOR=numd
DENOMINADOR=dend
disp('PRESIONE UN TECLA PARA CONTINUAR ')
pause
clc
disp(' VARIABLES DE ESTADO: ')
disp(' ')
Ad=ad,Bd=bd,Cd=cd,Dd=dd
a=ad;,b=bd;,c=cd;,d=dd;,save ac,save bc,save cc,save dc,save a
save b,save c,save d,save num,save den,pause
end
```

rdatamap.m

```
function rdatamap
load a ,load b,load c,load d,load num,load den
labels = str2mat(...
  'ANALISIS DE RESPUESTA EN EL TIEMPO', ...
  'LUGAR GEOMETRICO DE LAS RAICES ', ...
  'ANALISIS EN EL PLANO "Z" ');
callbacks = [ ...
   'resp2ap       '
   'lgrtz         '
   'respf2        '];
choices1('RDATAMAP', 'RESPUESTA EN EL TIEMPO', labels, callbacks);
```

resp2ap.m

```
clc
close(gcf)
n=input('CALCULAR HASTA n = ');
[num,den]=ss2tf(a,b,c,d);
```

```
  end
end
%PARA ENCONTRAR EL TIEMPO DE ESTABLECIMIENTO
n=1;
```

```
    'OBSERVABILIDAD', ...
    'TRANFORMACION DE SEMEJANZA');
  callbacks = [ ...
    'contrbap'    '
    'obserap      '
    'tsemejap     '];
  choices1('cdatamap','CARACTERISTICAS ESTRUCTURALES DEL
  SISTEMA',labels,callbacks);
```

Se utilizan las rutinas contrbap.m, obserap.m y tsemejap.m, decritas en el tema: Aplicaciones de sistemas lineales (Características estructurales del sistema).

ANÁLISIS DE FOURIER

fourirap.m

```
  function fourirap
  labels = str2mat(...
    'INGRESO DE LA SEÑAL Y GRAFICO', ...
    'ESPECTRO DE POTENCIA SIN RUIDO', ...
    'GRAFICO DE LA SEÑAL CON RUIDO', ...
    'ESPECTRO DE POTENCIA CON RUIDO');
  callbacks = [ ...
    'ifourap      '
    'espotnap     '
    'ruidfap      '
    'espotrap     '];
  choices1('fourirap','ANALISIS DE FOURIER',labels,callbacks);
```

ifourap.m

```
  function ifourap
  !edit ifunap.m
  ifunap
  load t ,load ft

      i=0;
   i=find(isnan(ft));
```

```
xlabel('t[seg]')
ylabel('F(T)+RUIDO')
grid
pause
end
```

espotrap.m

```
% ESTE PROGRAMA PERMITE GRAFICAR EL ESPECTRO DE
%POTENCIA DE UNA FUNCION CON RUIDO
load t ,load ft,load f,load nur
ft1=ft+2*(rand(t)-0.5);
y=fft(ft1);
Pyy=y.*conj(y);
clc
plot(f(1:nur),Pyy(1:nur))
title('ANALISIS DE FOURIER')
xlabel('t[seg]')
ylabel('Espectro de Potencia sin ruído')
grid
pause
end
```

FILTRADO DE SEÑALES

filtroap.m

```
function filtroap
labels = str2mat(...
   'INGRESO DE LA SEÑAL', ...
   'FILTRO PASA BAJOS', ...
   'FILTRO PASA ALTOS', ...
   'FILTRO PASA BANDA', ...
   'ESPECTRO DE FRECUENCIA');
```

```
callbacks = [ ...
    'infilap      '
    'pbajoap      '
    'paltoap      '
    'pbandaap     '
    'espefap      '];
choices1('filtroap','FILTRADO DE SEÑALES',labels,callbacks);
```

infilap.m

```
%function infilap
%1. PROGRAMA PARA INGRESAR UNA SEÑAL PARA SER FILTRADA
close
clc
disp(' INGRESO DE LA SEÑAL f(t) PARA FILTRAR')
!edit ifilap.m
ifilap
load ff,load t1,load fil
figure
hold on
plot(t1,ff)
title('SEÑAL PARA FILTRAR')
xlabel('t[seg]')
ylabel('f(t)')
grid
pause
end
```

ifilap.m

```
function ifilap
global t
%PROGRAMA PARA INGRESAR UNA FUNCION f(t)
%       Ejemplo
%       f(t)='sin(2*pi*25*t)'
```

```
%        t=0:0.01:1;
%INGRESE f(t) EN LA SIGUIENTE LINEA:

      f='8*sin(2*pi*25*t)+8*sin(2*pi*5*t)'

%DEFINA EL INTERVALO DE TIEMPO

  t=0:0.01:1;

%PONGA GUARDAR Y SALIR DEL EDITOR

t1=t;
[ft,ct]=size(t1);
for k=1:1:ct,
  t=t1(k);
  ff(k)=eval(f);
end
i=find(isnan(ff));
  if i~=0
   [p,q]=size(i);
     for x=1:q,
           ff(i(x))=(ff(i(x)-1)+ff(i(x)+1))/2;
     end
  end
fil=0;
save fil,save t1,save ff
end
```

pbajoap.m

```
clc
echo off
close(gcf)
load ff,load t1
```

```
disp(' FILTRADO DE SEÑALES')
disp(' ')
disp(' FILTRO PASA BAJOS')
disp(' INGRESE:')
  wn1=input('FRECUENCIA DE ELIMINACION [Hz] = ');
  [b,a]=ellip(4,0.1,40,wn1*0.02);
  [h,w]=freqz(b,a,512);
%        figure
%        hold on
%        plot(w*100/(2*pi),abs(h),'m')  %grafica el filtro pasa bajos
%        axis([ 0 14 1.2*min(abs(h)) 1.2*max(abs(h))])
%        grid
%        title('FILTRO PASA BAJOS')
%        pause
sf=filter(b,a,ff);
clear fil
fil=sf;
save fil
plot(t1,sf)
title('SEÑAL FILTRADA')
ylabel(' f(t) Filtrada ')
xlabel(' t[seg] ')
grid
pause
end
```

paltoap.m

```
clc
echo off
close(gcf)
load ff
load t1
disp(' FILTRADO DE SEÑALES')
```

```
disp(' ')
disp(' FILTRO PASA ALTOS')
disp(' INGRESE:')
  wn1=input('FRECUENCIA DE ELIMINACION [Hz] = ');
  [b,a]=ellip(4,0.1,40,wn1*0.02,'high');
%        figure
%        [h,w]=freqz(b,a,512);
%        plot(w*100/(2*pi),abs(h),'m')  %grafica el filtro pasa bajos
%        grid
%        axis([0 20 min(abs(h))*1.2 max(abs(h))*1.2])
%        title('FILTRO')
%        pause
sf=filter(b,a,ff);
clear fil
fil=sf;
save fil
plot(t1,sf)
title('SEÑAL FILTRADA')
ylabel(' f(t) Filtrada ')
xlabel(' t[seg] ')
grid
pause
end
```

pbandaap.m

```
clc
echo off
close(gcf)
load ff
load t1
disp(' FILTRADO DE SEÑALES')
disp(' FILTRO PASA BANDA')
disp(' ')
```

```
disp(' LIMITES DE FRECUENCIA PARA FILTRAR LA SEÑAL:')
  wn1=input('FRECUENCIA INFERIOR [Hz] = ');
  disp(' ')
  wn2=input('FRECUENCIA SUPERIOR [Hz] = ');
  [b,a]=ellip(4,0.1,40,[wn1 wn2]*0.02);
%       figure
%       [h,w]=freqz(b,a,512);
%       plot(w*100/(2*pi),abs(h),'m')  %grafica el filtro pasa bajos
%       axis([0 20 0 1.4])
%       grid
%       title('FILTRO')
%       pause
figure
sf=filter(b,a,ff);
clear fil
fil=sf;
save fil
plot(t1,sf)
title('SEÑAL FILTRADA')
ylabel(' f(t) Filtrada ')
xlabel(' t[seg] ')
grid
pause
end
```

espefap.m

```
% ESTE PROGRAMA PERMITE GRAFICAR EL ESPECTRO DE
%FRECUENCIA CON LA TRANSFORMADA RAPIDA DE FOURIER
load ff,load t1
[ft1,ct1]=size(t1);
delta=(max(t1)-min(t1))/ct1;
clc
close(gcf)
```

```
[ft1,ct1]=size(t1);
val=0;
for n=1:100,
  val=n;
  poten=ct1/(2^n);
  if poten<1
          break
  end
end
ex=2^val;
s=fft(ff,2*ex);
w=(1/(2*delta))*(0:ex-1)/ex;
load fil
if fil==0
  plot(w,abs([s(1:ex)']))
  title('TRANSFORMADA RAPIDA DE FOURIER')
  xlabel('[w]')
  ylabel('Espectro de frecuencia fft(f)')
  grid
  pause
  clear fil
else
  w=(1/(2*delta))*(0:ex-1)/ex;
  fil1=fft(fil,2*ex);
  plot(w,abs([fil1(1:ex)']))
  title('TRANSFORMADA RAPIDA DE FOURIER-SEÑAL FILTRADA')
  xlabel('[w]')
  ylabel('Espectro de frecuencia fft(f)')
  grid
  pause
  clear fil
end
```

MODULACION DE SEÑALES

modulaap.m

```
function modulaap
labels = str2mat(...
   'MODULACION EN AMPLITUD', ...
   'MODULACION EN FASE', ...
   'MODULACION EN FRECUENCIA');
callbacks = [ ...
   'modultap      '
   'modulfap      '
   'modulffp      '];
choices1('modulaap','MODULACION DE SEÑALES',labels,callbacks);
```

modultap.m

```
function modultap
labels = str2mat(...
   'INGRESO DE SEÑALES EN TIEMPO', ...
   'GRAFICO SEÑAL MODULADA',...
   'ESPECTRO DE FRECUENCIA');
callbacks = [ ...
   'istimcap      '
   'tamplap       '
   'espefapm      '];
choices1('modultap','MODULACION DE SEÑALES EN
AMPLITUD',labels,callbacks);
```

istimcap.m

```
echo off
close
clc
disp('MODULACION EN AMPLITUD ')
disp(' ')
```

```
disp('INGRESO DE LA FUNCION ')
disp(' INGRESO DE LA FUNCION A MODULAR:')
j=sqrt(-1);
!edit timap.m
timap
load t,load ff,load xs1,load t1
[ft1,ct1]=size(t1);
delta=(max(t1)-min(t1))/ct1;
[ft1,ct1]=size(t1);
val=0;
for n=1:100,
  val=n;
  poten=ct1/(2^n);
  if poten<1
          break
  end
end
ex=2^val;
s=fft(xs1,2*ex);
w=(1/(2*delta))*(0:ex-1)/ex;
  w=(1/(2*delta))*(0:ex-1)/ex;
  fil1=fft(xs1,2*ex);
clc
disp('GRAFICO DE LA FUNCIÓN QUE INGRESÓ PARA MODULAR')
disp('Y SU ESPECTRO DE FRECUENCIA ')
subplot(211)
  plot(t,xs1)
  title('SEÑAL PARA MODULAR')
  xlabel('t[seg]')
  ylabel('xs(t)')
  grid
subplot(212)
      ef=abs([fil1(1:ex)']);
```

```
  [Mef,k]=max(ef);
  kef=min(5*k,ex);
  plot(w(1:kef),ef(1:kef))
  xlabel('f[Hz]')
  ylabel('Espectro de frecuencia fft(f)')
  grid
  pause
end
```

timap.m

```
function timap
%FUNCION PARA MODULAR  x(t)=xc
%Ejemplo
%       xs='cos(2*pi*fs*t)'
%       fs=3
%INGRESE x(t) EN LA SIGUIENTE LINEA

      xs='cos(2*pi*fs*t)'
      fs=3;

%CUAL ES EL VALOR DE fs[Hz] PARA MODULAR?
% fs<fc
%Ejemplo:     fc=11.5

            fc=11.5;

%PONGA GUARDAR Y SALIR DEL EDITOR
global t
t=0:0.001*fs/fc:4/fs;
xs1=eval(xs);
xc=sin(2*pi*fc*t);
xx=xc.*xs1;
save xs1,save xs,save xc,save xx,save t
```

```
ff=xs1;
t1=t;
save ff,save t1
end
```

tamplap.m

```
%GRAFICA LA FUNCION QUE SE VA A MODULAR Y LA MODULADA
load xs1,load xc,load xx,load t
clc
close(gcf)
ee=input('Máximo valor de t para visualizar ? ');
es=ee*1000*fc/fs
if es>=1000*4*fc/(fs*fs);
  es1=1000*4*fc/(fs*fs)
else
  es1=es
end
figure
t1=t(:,1:es1);
ff=xx(:,1:es1);
hold on
  plot(t1,ff)
  title('MODULACION EN AMPLITUD')
  xlabel('t [seg]')
  ylabel('Señal modulada xc(t)')
  grid
fil=ff;
save fil
ffs=xs1(:,1:es1);
  plot(t1,ffs,'m')
  plot(t1,-ffs,'m')
pause
end
```

espefapm.m

```
% ESTE PROGRAMA PERMITE GRAFICAR EL ESPECTRO DE
%FRECUENCIA CON LA TRANSFORMADA RAPIDA DE FOURIER
load xx
load t1
[ft1,ct1]=size(t1);
delta=(max(t1)-min(t1))/ct1;
clc
close(gcf)
disp(' ')
[ft1,ct1]=size(t1);
val=0;
for n=1:100,
  val=n;
  poten=ct1/(2^n);
  if poten<1
          break
  end
end
ex=2^val;
s=fft(xx,2*ex);
w=(1/(2*delta))*(0:ex-1)/ex;
fil=xx;
  w=(1/(2*delta))*(0:ex-1)/ex;
  fil1=fft(fil,2*ex);
  plot(w,abs([fil1(1:ex)']))
  title('TRANSFORMADA RAPIDA DE FOURIER-SEÑAL MODULADA')
  xlabel('[w]')
  ylabel('Espectro de frecuencia fft(f)')
  grid
  pause
end
```

modulfap.m

```
function modulfap
labels = str2mat(...
   'INGRESO DE SEÑALES EN TIEMPO', ...
   'GRAFICO SEÑAL MODULADA',...
   'ESPECTRO DE FRECUENCIA');
callbacks = [ ...
   'isfrecap      '
   'mamplap       '
   'espefapf      '];
choices1('modulfap','MODULACION DE SEÑALES EN
FASE',labels,callbacks);
```

isfrecap.m

```
echo off
clear
clc
disp('MODULACION EN FASE')
disp(' ')
disp('INGRESO DE SEÑALES EN EL TIEMPO ')
disp(' INGRESO DE LA FUNCION A MODULAR:')
j=sqrt(-1);
!edit tff3ap.m
tff3ap
load x1,load y,load t
t1=t;
[ft1,ct1]=size(t1);
delta=(max(t1)-min(t1))/ct1;
[ft1,ct1]=size(t1);
val=0;
for n=1:100,
  val=n;
  poten=ct1/(2^n);
```

```
  if poten<1
          break
  end
end
ex=2^val;
s=fft(x1,2*ex);
w=(1/(2*delta))*(0:ex-1)/ex;
  w=(1/(2*delta))*(0:ex-1)/ex;
  fil1=fft(x1,2*ex);
clc
disp('GRAFICO DE LA FUNCIÓN QUE INGRESÓ PARA MODULAR')
disp('Y SU ESPECTRO DE FRECUENCIA ')
subplot(211)
  plot(t,x1)
  title('SEÑAL PARA MODULAR')
  xlabel('t[seg]')
  ylabel('x(t)')
  grid
subplot(212)
      ef=abs([fil1(1:ex)']);
  [Mef,k]=max(ef);
  kef=min(5*k,ex);
  plot(w(1:kef),ef(1:kef))
  xlabel('f[Hz]')
  ylabel('Espectro de frecuencia fft(f)')
  grid
  pause
end
```

tff3ap.m

```
function tff3ap
%FUNCION PARA MODULAR EN FASE x(t)
%Ejemplo
```

```
%       x='sin(2*pi*f1*t)'
%         f1=800
%INGRESE x(t) EN LA SIGUIENTE LINEA. f1[Hz]

      x='3*sin(2*pi*f1*t)'
      f1=800;

%CUAL ES EL VALOR DE fc[Hz] PARA MODULAR?
%Ejemplo:
%         fc=2000

      fc=2000;

%PONGA GUARDAR Y SALIR DEL EDITOR
global t
clc
fs=25000;
t=0:1/fs:4/f1;
%t=0:1/fs:fs/fc;
x1=eval(x);
kp=pi/2/max(max(abs(x1)));
y=modulate(x1,fc,fs,'pm',kp);
%y=modulate(x1,fc,fs,'pm',pi);
save x1,save y,save t
end
```

mamplap.m

```
%GRAFICA LA FUNCION QUE SE VA A MODULAR Y LA MODULADA
load x1
load y
load t
clc
close(gcf)
```

```
ee=input('Máximo valor de t para visualizar ? ');
es=ee*25000;
if es>=4/f1*25000
  es1=max(size(t));
else
  es1=es;
end
figure
t1=t(:,1:es1);
ff=x1(:,1:es1);
hold on
ffs=y(:,1:es1);
  plot(t1,ffs)
  title('MODULACION EN FASE')
  xlabel('t [seg]')
  ylabel('x(t)Modulada')
  grid
pause
end
```

espefapf.m

```
% ESTE PROGRAMA PERMITE GRAFICAR EL ESPECTRO DE FRECUENCIA
% CON LA TRANSFORMADA RAPIDA DE FOURIER. Modulación en fase
load y
load t
t1=t;
[ft1,ct1]=size(t1);
delta=(max(t1)-min(t1))/ct1;
clc
close(gcf)
 [ft1,ct1]=size(t1);
val=0;
```

```
for n=1:100,
  val=n;
  poten=ct1/(2^n);
  if poten<1
          break
  end
end
ex=2^val;
w=(1/(2*delta))*(0:ex-1)/ex;
fil1=fft(y,2*ex);
  ef=abs([fil1(1:ex)']);
  [Mef,k]=max(ef);
  kef=min(5*k,ex);
  plot(w(1:kef),ef(1:kef))
  title('TRANSFORMADA RAPIDA DE FOURIER-SEÑAL MODULADA')
  xlabel('[w]')
  ylabel('Espectro de frecuencia fft(f)')
  grid
  pause
```

modulffp.m

```
function modulffp
labels = str2mat(...
   'INGRESO DE SEÑALES EN TIEMPO', ...
   'GRAFICO SEÑAL MODULADA',...
   'ESPECTRO DE FRECUENCIA');
callbacks = [ ...
   'isfrecfp      '
   'mamplfp       '
   'espefaff      '];
choices1('modulffp','MODULACION DE SEÑALES EN
FRECUENCIA',labels,callbacks);
```

isfrecfp.m

```
echo off
close
clear
clc
disp('MODULACION EN FRECUENCIA')
disp(' ')
disp('INGRESO DE SEÑALES EN EL TIEMPO ')
disp(' INGRESO DE LA FUNCION A MODULAR:')
disp(' ')
j=sqrt(-1);
!edit tff3fp.m
tff3fp
load x1
load y
load t
t1=t;
clc
figure
t1=t;
ff=x1;
[ft1,ct1]=size(t1);
delta=(max(t1)-min(t1))/ct1;
[ft1,ct1]=size(t1);
val=0;
for n=1:100,
  val=n;
  poten=ct1/(2^n);
  if poten<1
        break
  end
end
ex=2^val;
```

```
s=fft(x1,2*ex);
w=(1/(2*delta))*(0:ex-1)/ex;
  w=(1/(2*delta))*(0:ex-1)/ex;
  fil1=fft(x1,2*ex);
clc
disp('GRAFICO DE LA FUNCIÓN QUE INGRESÓ PARA MODULAR')
disp('Y SU ESPECTRO DE FRECUENCIA ')
subplot(211)
  plot(t1,ff)
  title('SEÑAL PARA MODULAR')
  xlabel('t [seg]')
  ylabel('x(t)')
  grid
subplot(212)
  ef=abs([fil1(1:ex)']);
  [Mef,k]=max(ef);
  kef=min(10*k,ex);
  plot(w(1:kef),ef(1:kef))
  xlabel('f[Hz]')
  ylabel('Espectro de frecuencia fft(f)')
  grid
  pause
end
end
```

tff3fp.m

```
function tff3fp
%FUNCION PARA MODULAR EN FRECUENCIA x(t)
%Ejemplo
%       x='sin(2*pi*f1*t)'
%       f1=800

%INGRESE x(t) EN LA SIGUIENTE LINEA. fs[Hz]
```

```
        x='4*sin(2*pi*f1*t)'
        f1=800;

%CUAL ES EL VALOR DE fc[Hz] PARA MODULAR?
%Ejemplo:
%       fc=2000
        fc=2000;
%PONGA GUARDAR Y SALIR DEL EDITOR
global t
fs=25000;
t=0:1/fs:4/f1;
x1=eval(x);
%y=modulate(x1,fc,fs,'fm',pi);
kf=(fc/fs)*pi/2;
y=modulate(x1,fc,fs,'fm',kf);
save x1
save y
save t
end
```

mamplfp.m

```
%GRAFICA LA FUNCION QUE SE VA A MODULAR Y LA MODULADA
load x1
load y
load t
clc
close(gcf)
ee=input('Máximo valor de t para visualizar ? ');
es=ee*25000;
if es>=4/f1*25000
  es1=max(size(t));
```

```
else
  es1=es;
end
figure
t1=t(:,1:es1);
ff=x1(:,1:es1);
hold on
ffs=y(:,1:es1);
  plot(t1,ffs)
  xlabel('t [seg]')
  ylabel('x(t)Modulada')
  grid
pause
end
```

espefaff.m

```
% ESTE PROGRAMA PERMITE GRAFICAR EL ESPECTRO DE
%FRECUENCIA CON LA TRANSFORMADA RAPIDA DE FOURIER
load y
load x1
load t
t1=t;
[ft1,ct1]=size(t1);
delta=(max(t1)-min(t1))/ct1;
clc
close(gcf)
 [ft1,ct1]=size(t1);
val=0;
for n=1:100,
  val=n;
  poten=ct1/(2^n);
  if poten<1
          break
```

```
  end
 end
 ex=2^val;
 w=(1/(2*delta))*(0:ex-1)/ex;
 w=(1/(2*delta))*(0:ex-1)/ex;
 fil1=fft(y,2*ex);
  ef=abs([fil1(1:ex)']);
  [Mef,k]=max(ef);
  kef=min(5*k,ex);
  plot(w(1:kef),ef(1:kef))
  xlabel('f[Hz]')
  ylabel('X(t) modulada')
  grid
  pause
 end
```

SIMULINK

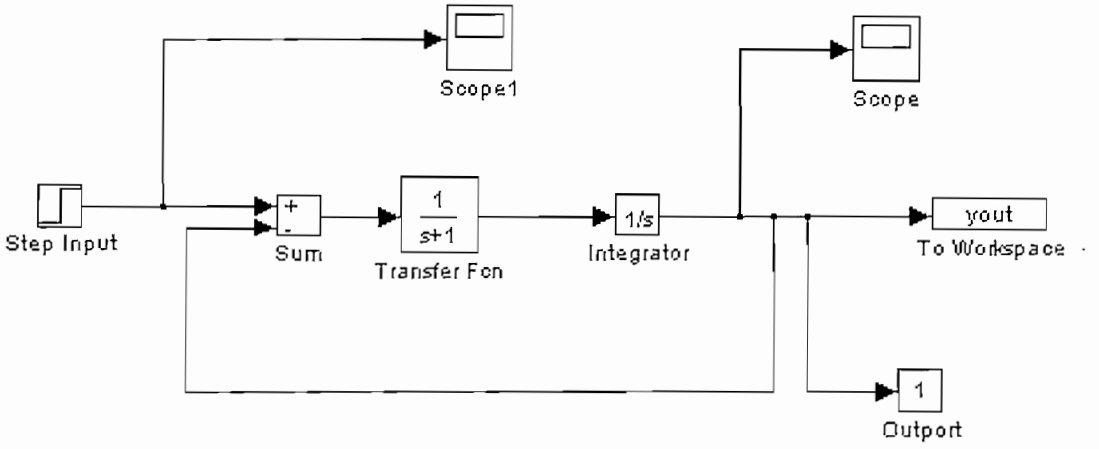
simulap.m

```
function simulap
labels = str2mat(...
   'MODELACION EN SIMULINK', ...
   'MODELACION EN MATLAB-DIAGRAMAS EN SIMULIN');
callbacks = [ ...
   'modsimap      '
   'matsimap      '];
choices1('simulap','SIMULINK',labels,callbacks);
```

modsimap.m

```
%RUTINA PARA INGRESAR AL GRAFICO DE SIMULINK
link1
end
```

link1.m



matsimap.m

```
clc
% Graphics initialization
   oldFigNumber = watchon;
   figNumber = figure;
   set(gcf, ...
      'NumberTitle','off', ...
      'Name','Simulink con Matlab', ...
  'backingstore','off',...
  'Units','normalized');
%========
ts=2
axes( ...
      'Units','normalized', ...
      'Position',[0.1 0.3 0.85 0.6], ...
      'Visible','on');
sim1
darkgray=[1/3 1/3 1/3];
cerrar='close;';
tsc = uicontrol('style','slider','units','normal','pos',[.40 .05 .3 .035], ...
```

```
        'min',0.1,'max',100,'val',ts, ...
        'call','ts = get(tsc,"value"); sim1');
    tsct = uicontrol('style','text','units','normal','pos',[.40 .13 .3 .04], ...
        'string','Tiempo de simulación','fore','white','back',darkgray);
    tsct = uicontrol('style','text','units','normal','pos',[.40 .09 .30 .04], ...
        'string',' 0.01 20 40 60 80 100','fore','white','back', darkgray);
    cerrarh = uicontrol('style','push','units','normal','pos',[.8 .075 .1 .06], ...
          'string','cerrar','call',cerrar);
```

sim1.m

```
    cla reset
    t=[0:.1:99.9]';
     vt=[t,ones(size(t))];
     [T,X,Y]=linsim('link1',ts);
    plot(T,Y)
    title('RESPUESTA DEL SISTEMA')
    grid
    xlabel('t [seg]')
    ylabel('y(t)')
    % Marco de fondo
       frmPos=[0.03 0.03 1 0.15];
       h=uicontrol( ...
          'Style','frame', ...
          'Units','normalized', ...
          'Position',frmPos, ...
     'BackgroundColor',[0.50 0.50 0.50]);
```

ANALISIS VARIANDO UN PARAMETRO

variarap.m

```
    function variarap
    labels = str2mat(...
       'INGRESO DE LA FUNCION DE TRANSFERENCIA', ...
```

```
    'ANALISIS EN EL TIEMPO', ...
    'ANALISIS EN FRECUENCIA');
callbacks = [ ...
    'ftf4          '
    'pasoap        '
    'bodeap        '];
choices1('variarap','ANALISIS                    VARIANDO                    UN
PARAMETRO',labels,callbacks);
```

Se utiliza la rutina ftf4.m en el tema: Análisis de sistemas lineales en el dominio de la frecuencia (Transformada de Fourier)

pasoap.m

```
global num den
load num,load den
[fden,cden]=size(den)
co=den(cden-1);  %coeficiente de s
ts=25;
den(cden-1)=co;
axes( ...
      'Units','normalized', ...
      'Position',[0.11 0.31 0.8 0.6], ...
      'Visible','on');
paso1ap
darkgray=[0.5 0.5 0.5];
coef = uicontrol('style','slider','units','normal','pos',[.05 .05 .40 .035], ...
      'min',0,'max',2*co,'val',den(cden-1), ...
      'call','den(cden-1) = get(coef,"value"); paso1ap');
coefc = uicontrol('style','text','units','normal','pos',[.05 .13 .30 .04], ...
     'string','Coeficiente de s: a2','fore','white','back', darkgray);
coefc = uicontrol('style','text','units','normal','pos',[.05 .09 .4 .04], ...
     'string','0                          2*a2','fore','white','back', darkgray);
tsc = uicontrol('style','slider','units','normal','pos',[.55 .05 .4 .035], ...
```

```
    'min',0.1,'max',50,'val',ts, ...
    'call','ts = get(tsc,"value"); paso1ap');
tsct = uicontrol('style','text','units','normal','pos',[.50 .13 .4 .04], ...
    'string','Tiempo [seg]','fore','white','back',darkgray);
tsct = uicontrol('style','text','units','normal','pos',[.55 .09 .4 .04], ...
    'string','0.1                    50','fore','white','back', darkgray);
end
```

paso1ap

```
hold on
cla reset
global t
global y
if ts==0
  t=0:.1:50;
else
  t=0:.1:ts;
end
xo=i;
[a,b,c,d] = tf2ss(num,den);
if xo==0
  y=step(a,b,c,d,1,t);
else
  y=lsim(a,b,c,d,ones(1,length(t)),t,xo);
end
save t,save y
plot(t,y),title('RESPUESTA A UNA FUNCION PASO')
axis([min(t) max(t) min(0,min(y)) 1.2*max(y)])
xlabel('tiempo[seg]')
ylabel('f(t)')
grid
% Marco de fondo
   frmPos=[0.03 0.03 0.95 0.15];
```

```
    h=uicontrol( ...
        'Style','frame', ...
        'Units','normalized', ...
        'Position',frmPos, ...
  'BackgroundColor',[0.50 0.50 0.50]);
end
```

bodeap.m

```
function bodeap(action,in1,in2);
global num,global den,load num,load den
if nargin<1,
   action='start';
end;
global BODEAP_DAT
if strcmp(action,'start'),
   %=================================
   % Graphics initialization
   oldFigNumber = watchon;
   figNumber = figure;
   set(gcf, ...
      'NumberTitle','off', ...
      'Name','Respuesta de frecuencia', ...
  'backingstore','off',...
  'Units','normalized');
   %=================================
   % Information for all buttons
   top=0.95;
   bottom=0.05;
   left=0.82;
   yInitLabelPos=0.90;
   btnWid = 0.13;
   btnHt=0.08;
   % Spacing between the label and the button for the same command
```

```
   btnOffset=0.02;
   % Spacing between the button and the next command's label
   spacing=0.02;
   %bottom=bottom+spacing;
num=1;
den=[1 2 3];
[fden,cden]=size(den);
% freq Valores iniciales y extremos de a2.
   min_freq =0.5;
   max_freq =10*den(cden-1);
   freq=den(cden-1);    % hertz
 fmin=0
 fmax=1
   [t,f,w,F]=tfunc(freq);
%COLOCACION DEL TEXTO DEL VALOR VARIABLE DE LA BARRA
   freq_text=uicontrol('Style','text','Position',[0.1 .01 .5 .07],...
      'Units','normalized','BackgroundColor','black',...
      'ForegroundColor','white','String','den de FT.  COEFICIENTE DE S:
a2');
%COLOCACION VALOR MINIMO DE LA BARRA
   uicontrol('style','text','pos',[.14 .07 .02 .05],...
      'Units','normalized','BackgroundColor','black',...
      'ForegroundColor','white','String',num2str(min_freq));
%COLOCACIÓN VALOR MÁXIMO DE LA BARRA
   uicontrol('style','text','pos',[.74 .07 .3 .05 ],...
      'Units','normalized','BackgroundColor','black',...
      'ForegroundColor','white','String',num2str(max_freq));
%COLOCACION POSICION DEL CUADRO CON EL VALOR
   freq_field=uicontrol('Style','edit','Position',[.59 .02 .12 .07],...
      'Units','normalized','String',num2str(freq),...
      'CallBack','bodeap("setfreq",2); bodeap("redraw");');
%DIMENSIONES DE LA BARRA DE DESPLAZAMIENTO
   freq_slider=uicontrol('Style','slider','Position',[.12 .12 .8 .04],...
```

```
    'Units','normalized','Value',freq,'Max',max_freq,'Min',min_freq,...
    'Callback','bodeap("setfreq",1); bodeap("redraw");');
  % Gráfico de la fase en grados
                    ax_freq=axes('Position',[.12 .28 .8 .26],'XLim',[min(w)
                                max(w)],'YLim',[1.2*min(F) 1.2*max(F)]);
  freq_line=semilogx(w,F,'EraseMode','xor');
  axis([min(w) max(w) 1.2*min(F) 1.2*max(F)])
  grid
  ylabel('Fase[grados]')
  xlabel('W[rad/seg]')
  % Gráfico de la magnitud en db
   ax_time=axes('Position',[.12 .66 .8 .26],'XLim',[min(t) max(t)],'YLim',[fmin
                                                                    fmax]);
  time_line=semilogx(t,f,'EraseMode','xor');
  axis([min(t) max(t) fmin fmax])
  grid
  ylabel('Magnitud[Db]')
  xlabel('W[rad/seg]')
  title('DIAGRAMAS DE BODE. Lazo abierto')
  BODEAP_DAT = [freq;time_line; freq_line; freq_field; freq_slider;
ax_time];
 elseif strcmp(action,'redraw'),
  freq=BODEAP_DAT(1);
  set(BODEAP_DAT(4),'string',num2str(freq));
  set(BODEAP_DAT(5),'value',freq);
  [t,f,w,F]=tfunc(freq);
  set(BODEAP_DAT(2),'YData',f);
  set(BODEAP_DAT(3),'YData',F);
  drawnow;
elseif strcmp(action,'setfreq'),
  ax_time=BODEAP_DAT(6);
  BODEAP_DAT(1)=get(BODEAP_DAT(5),'value');
  drawnow;
```

```
end
end
```

tfunc.m

```
function [t,f,w,F] = tffunc(freq)
num=1;
den=[1 freq 3];
w = 0:.01:100;
[mag,phase,w]=bode(num,den,w);
   t = 0:.01:100;
   f = mag;
   w = 0:.01:100;
   F = phase;
```

PROCESAMIENTO EN TIEMPO REAL

timerap.m

```
function timerap
labels = str2mat(...
   'ADQUISICION DE DATOS Y GRAFICOS', ...
   'ANALISIS DE FOURIER',...
   'RESPUESTA DEL SISTEMA G(s)', ...
   'FILTRADO DE SEÑALES');
callbacks = [ ...
   'adki        '
   'four1ap      '
   'rarbitap     '
   'filterap     '];

choices1('timerap','PROCESAMIENTO EN TIEMPO REAL',
labels,callbacks);
```

adki.m

```
function adki
labels = str2mat(...
   'SEÑAL ARBITRARIA', ...
   'SISTEMA REALIMENTADO');
callbacks = [ ...
   'adkisap      '
   'adkisap1     '];
choices1('adki','ADQUISICION DE DATOS',labels,callbacks);
```

adkisap.m

```
clc
close(gcf)
sislab.exe
load c:\matlab\bin\tempvar
load c:\matlab\bin\timevar
t=timevar;
x=tempvar;
end
```

adkisap1.m

```
clc
close(gcf)
!sislab1.exe
load c:\matlab\bin\tempvar
load c:\matlab\bin\timevar
load c:\matlab\bin\c1
load c:\matlab\bin\c2
t=timevar;
x=tempvar;
xc1=c1;
xc2=c2;
end
```

four1ap.m

```
% ESTE PROGRAMA PERMITE GRAFICAR EL ESPECTRO DE
%POTENCIA DE LA FUNCIÓN ADQUIRIDA EN TIEMPO REAL
close(gcf)
clc
disp(' ')
disp('      GRAFICO DEL ESPECTRO DE POTENCIA DE:')
disp(' ')
disp('      1. Vx1 ')
disp('      2. Vx2 ')
disp('      3. Vsalida ')
disp('      4. Señal arbitraria ')
in=input('REALIZAR: ');
disp(' ')
if in==1
  vf=xc1;
  if in==2
        vf=xc2;
        if (in==3 | in==4)
              vf=x;
        else
              disp('ERROR. FUERA DE RANGO')
              return
        end
  end
end
[fil,col]=size(t);
for i=1:20,
      if (2^i<=col & 2^(i+1)>col)
  nur=2^i;
      end
end
delta=(max(t)-min(t))/col;
```

```
f=(0:nur-1)/(2*nur*delta);
y=fft(xf);
Pyy=y.*conj(y);
clc
plot(f(1:nur),Pyy(1:nur))
title('ANALISIS DE FOURIER')
xlabel('t[seg]')
ylabel('Espectro de Potencia')
grid
pause
end
```

rarbitap.m

```
%EL PROGRAMA VA A CONTENER:
%RESPUESTA A LA SEÑAL ARBITRARIA QUE SE ADQUIRIÓ EN TIEMPO REAL
close
%La señal x(t) es la señal adquirida en tiempo real
% RESPUESTA A LA FUNCION ARBITRARIA x(t)
echo off
close(gcf)
clc
disp('INGRESO DE LA FUNCION DE TRANSFERENCIA ')
disp('            H(s) = NUM(s) / DEN(s)')
disp('EL NUMERADOR NO PUEDE SER DE ORDEN MAYOR QUE EL DENOMINADOR')
disp(' ')
disp('                    b2.S²+b1.S+b0')
disp('            H(s) = ----------------')
disp('                   a3.S³+a2.S²+a1.S+a0')
disp('INGRESE DE LA SIGUIENTE MANERA:   num=[b2 b1 bo]')
disp('                          den=[a3 a2 a1 ao]')
disp(' ')
```

```
num=input('INGRESE SUS DATOS:    num= ');
disp(' ')
den=input('                    den= ');
clc
disp('     GRAFICO DEL ESPECTRO DE POTENCIA DE:')
disp(' ')
disp('     1. Vx1 ')
disp('     2. Vx2 ')
disp('     3. Vsalida  ')
disp('     4. Señal arbitraria ')
disp(' ')
in=input('REALIZAR: ');
disp(' ')
if in==1
  vf=xc1;
  if in==2
        vf=xc2;
        if (in==3 | in ==4)
              vf=x;
        else
              disp('ERROR. FUERA DE RANGO')
              return
        end
  end
end
%PASANDO A LA FORMA CANONICA CONTROLABLE
[a,b,c,d]=tf2ss(num,den)
y=lsim(a,b,c,d,vf,t);
hold on
plot(t,y,'r'),title('RESPUESTA DE LA FUNCION ')
axis([min(t) max(t) -0.01+1.05*min(0,min(y)) 0.01+1.05*max(y)])
xlabel('t[seg]')
ylabel('y(t)')
```

```
grid
pause
end
```

filterap.m

```
function filterap
close(gcf)
clc
disp('      SEÑAL PARA FILTRAR')
disp(' ')
disp('      1. Señal Vx1')
disp('      2. Señal Vx2 ')
disp('      3. Señal de salida ')
disp('      4. Señal arbitraria ')
disp(' ')
in=input('REALIZAR: ');
if in==1
  ff=xc1;
  if in==2
        ff=xc2;
        if (in==3 | in==4)
              ff=x;
        else
              disp('ERROR. FUERA DE RANGO')
              return
        end
  end
end
t1=t;
labels = str2mat(...
   'FILTRO PASA BAJOS', ...
   'FILTRO PASA ALTOS', ...
   'FILTRO PASA BANDA', ...
```

```
    'ESPECTRO DE FRECUENCIA');
callbacks = [ ...
    'pbajoap        '
    'paltoap        '
    'pbandaap       '
    'espefap        '];
choices1('filter','FILTRADO DE SEÑALES',labels,callbacks);
```

Las rutinas pbajoap.m, paltoap.m, pbandaap.m y espefap.m se implementaron en el numeral correspondiente a filtrado de señales.

SISLAB1.EXE

Pantalla.vi

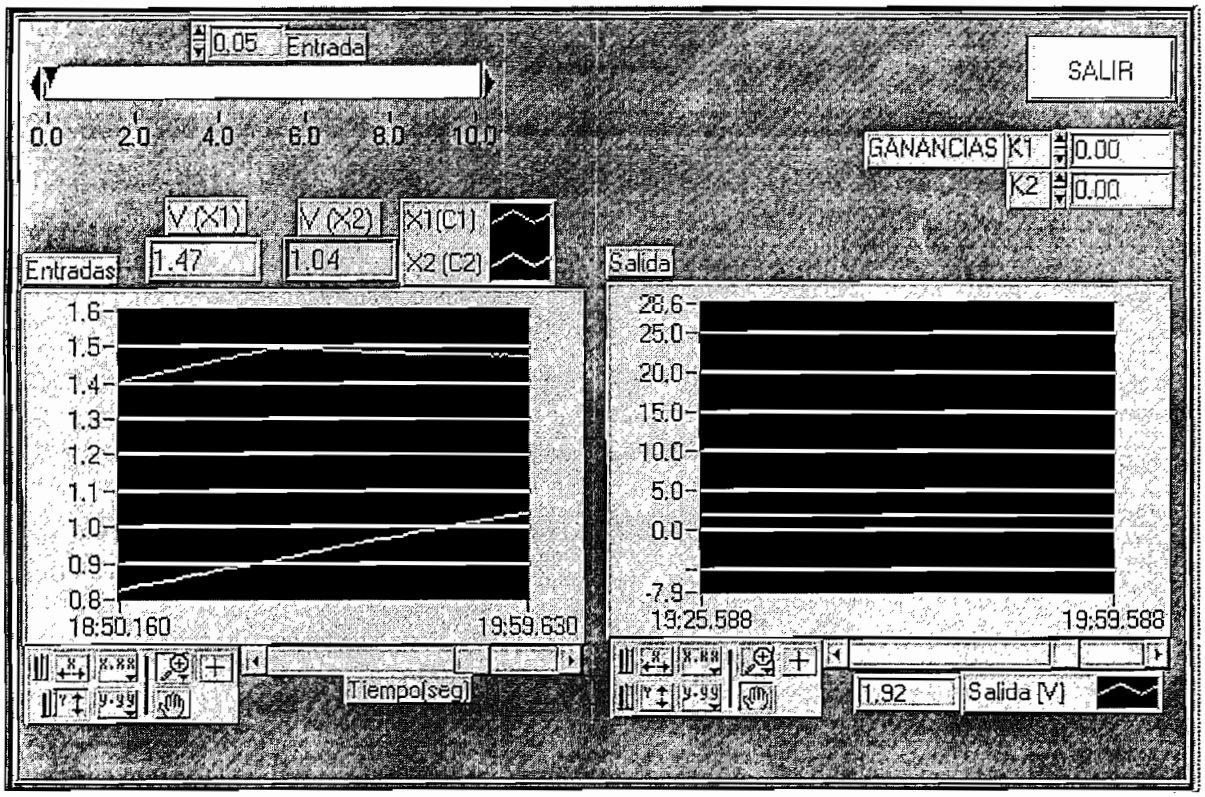
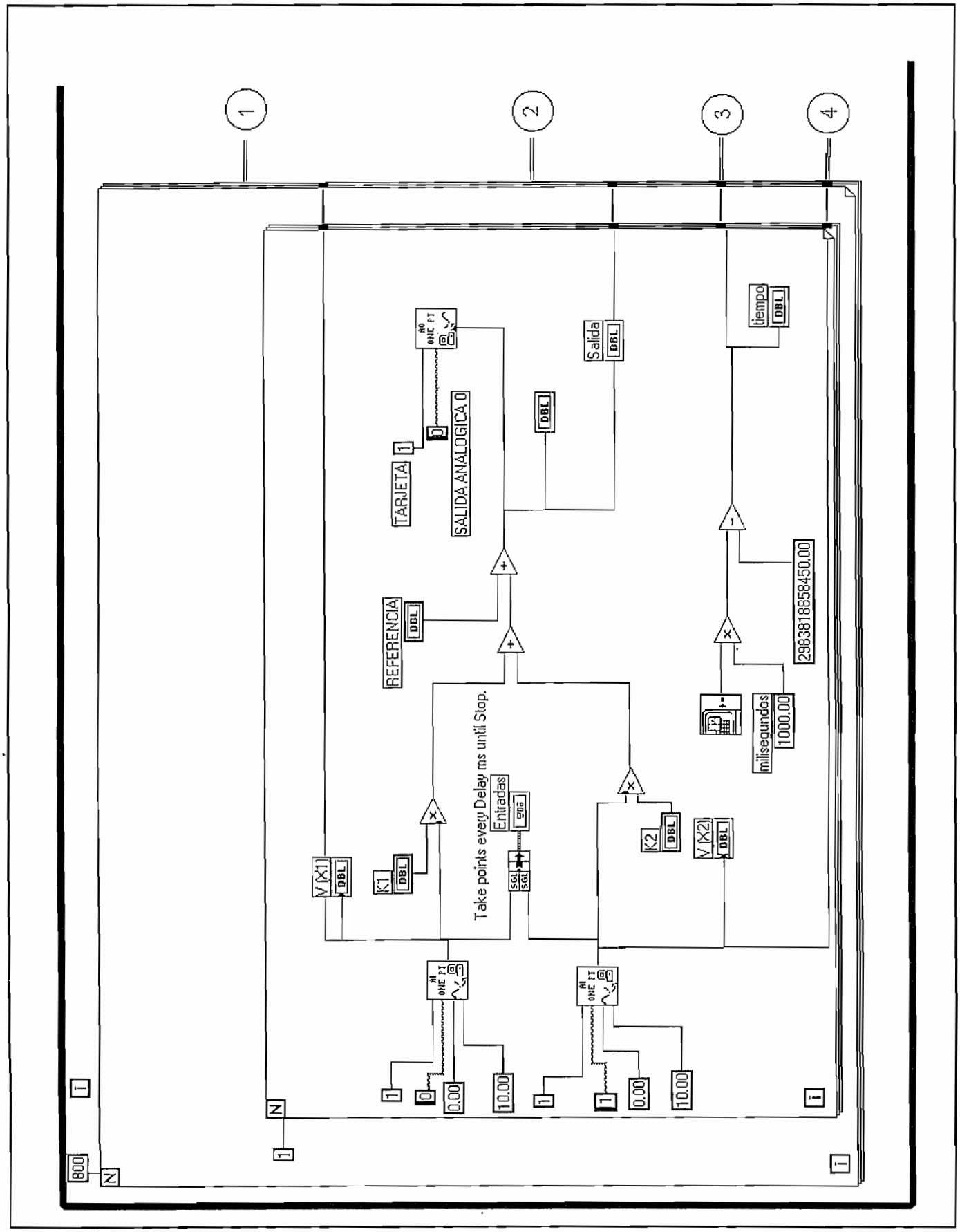
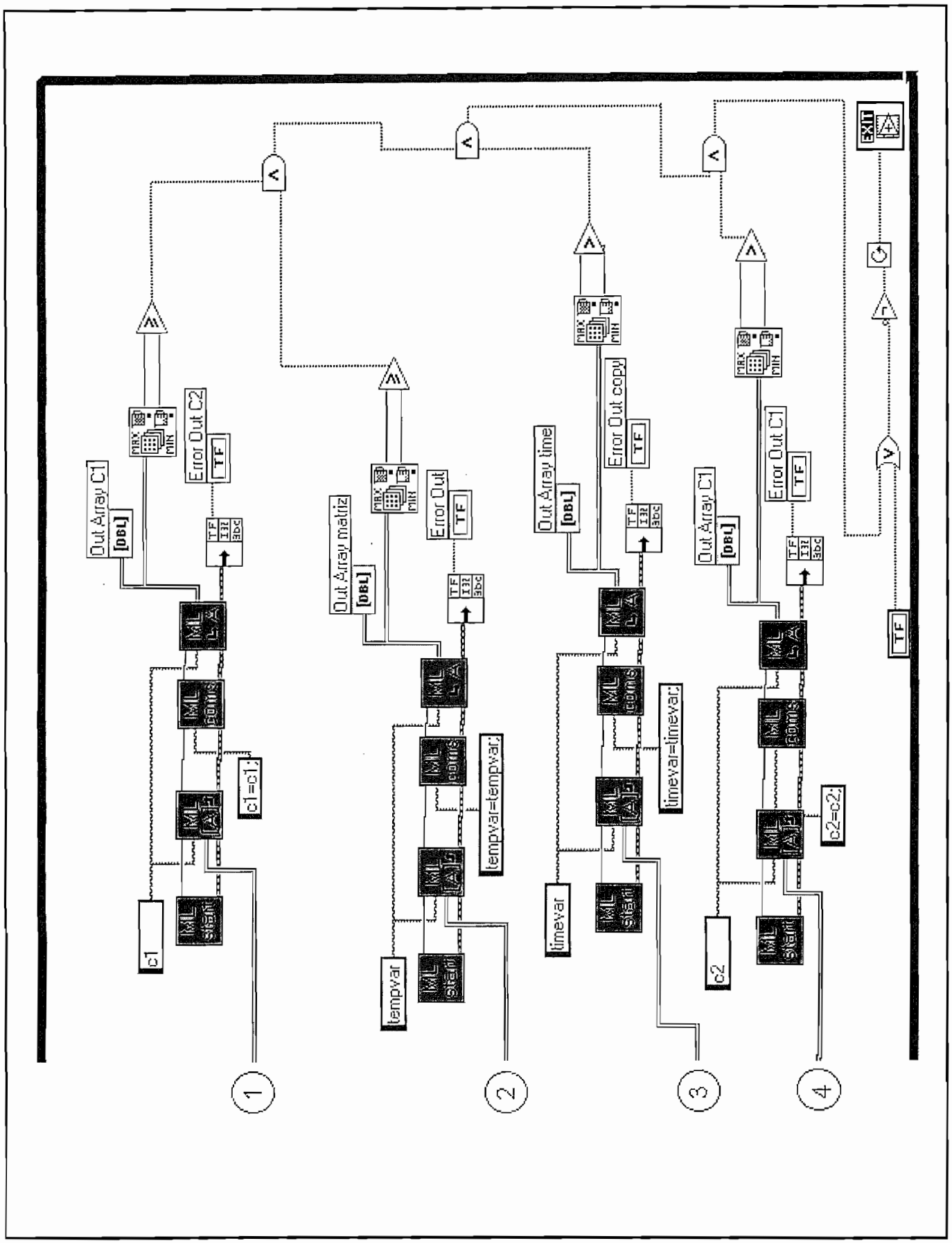


Figura 1. Programa para adquisición de datos en Labview


1
2
3
4
AO ONE PT
TARJETA
1
SALIDA ANALOGICA 0
DBL
Salida
DBL
tiempo
DBL
REFERENCIA
DBL
2983818658450.00
milisegundos
1000.00
Take points every Delay ms until Stop.
Entradas
SGL
V (X1)
DBL
K1
DBL
K2
DBL
V (X2)
DBL
AI ONE PT
0
0.00
10.00
AI ONE PT
1
0.00
10.00
800
N
i


Out Array C1
[DBL]
Error Out C2
TF
c1
c1=c1;
Out Array matriz
[DBL]
Error Out
TF
tempvar
tempvar=tempvar;
Out Array time
[DBL]
Error Out copy
TF
timevar
timevar=timevar;
Out Array C1
[DBL]
Error Out C1
TF
c2
c2=c2;
EXIT
TF
1
2
3
4

Las figuras anteriores corresponden al programa sislab.vi.

Para que los programas sislab.exe y sislab1.exe se ejecuten, es necesario que dentro del toolbox sislin se encuentren los archivos device.dll, serpdrv, daqdrv y gpibdrv.

CONEXIONES DE LA TARJETA LAB-PC1200.

La tarjeta debe configurarse de las siguiente manera:

Entradas analógicas:

Canal 0. Entrada1(1). Entrada X1 del sistema
Canal 0. Entrada 2(2). Entrada X2 del sistema
Punto común GND(11)

Definiéndose entrada1 y entrada 2, como entrada a la tarjeta de adquisición de datos. Se observa en la figura A.35 del anexo A, los puntos X1 y X2 que corresponden a un sistema realimentado para el ejemplo.

Si se ingresa la **señal arbitraria**, configurar la entrada1 del canal 0 e ingresar la señal de entrada en el pin 1, se tiene como referencia el pin 11.

Salida analógica:

Canal 0. Salida (11) . Entrada X1 del sistema
Punto común GND(11)

(##) Es el número de pin que le corresponde a cada entrada o salida en la tarjeta de adquisición de datos.

La tarjeta debe configurarse previamente dependiendo del rango de voltaje al que se va a someter.

B. LISTADO DE RUTINAS DE ANÁLISIS Y APLICACIÓN DE SISTEMAS LINEALES UTILIZANDO MATLAB.