4. Herramientas de programación.

4.1. Preliminares.

En la presente sección se da una introducción al software que será utilizado para la realización de las prácticas de control ya citadas anteriormente. Se describe que es y que hace cada software, se indica también la manera de instalar y configurar correctamente los programas Scilab y LabVIEW. En el caso de LabVIEW se describe donde se encuentran las herramientas de trabajo y que se se puede encontrar en cada una de ellas, así como los espacios de trabajo con los que cuenta. En el caso de Scilab se exponen los comandos necesarios para trabajar con álgebra lineal, gráficos e ingeniería de control, así como la manera de incorporarlo a LabVIEW.

4.2. Scilab.

Scilab es un software libre para cálculo científico y numérico que trabaja principalmente con el uso de matrices aunque también se pueden hacer cálculos simbólicos. Está disponible en versiones para 32 y 64 bits y para varios sistemas operativos tales como Unix, Linux (Madriva, Ubuntu), Windows(2000/XP/VISTA), Solaris, Alpha, MacOSX 10.5 entre otros.

Se puede integrar programas desarrollados en lenguajes de alto nivel como FORTRAN, Java, C y C++ mediante instrucciones propias de Scilab, y mediante una interfaz hecha por National Instruments puede trabajar en conjunto con LabVIEW.

Tiene herramientas para álgebra lineal, polinomios, ingeniería de control, gráficos 2D y 3D, además de tener un simulador por diagrama de bloques "Scicos".

Fue desarrollado por INRIA (Institut National de Recherche en Informatique et Automatique) y la ENPC (École Nationale des Ponts et Chaussées) desde 1990, aunque desde mayo del 2003, es mantenido y desarrollado por Scilab Consortium.

Los requerimientos mínimos para poder usar la version 5.1 (que es la que usaremos en este trabajo) son 247.5MB de espacio en disco duro y alguno de los sistemas operativos antes mencionados.

Existen herramientas llamadas "toolboxes" que son programas hechos por los usuarios como Rtool que es una herramienta para control en el que se pueden analizar las respuestas en el tiempo y en frecuencia de alguna planta con o sin controlador.

4.2.1. Instalación.

Para instalar Scilab en la plataforma windows hay que descargar la última versión del programa de la dirección

```
http://www.scilab.org/
```

ejecutarlo y seguir las instrucciones que proporciona el programa de instalación.

4.2.2. Conectando LabVIEW con Scilab.

Como se mencionó anteriormente, Scilab puede trabajar en conjunto con LabVIEW a través de un nodo de Scilab en LabVIEW, mediante el cual se puede incluir código matemático en instrumentos virtuales creados con LabVIEW como se muestra en la figura 21. Scilab LabVIEW Gateway es un archivo que crea este nodo y se encuentra en la página de National Instruments en la dirección:

http://zone.ni.com/devzone/cda/epd/p/id/657.

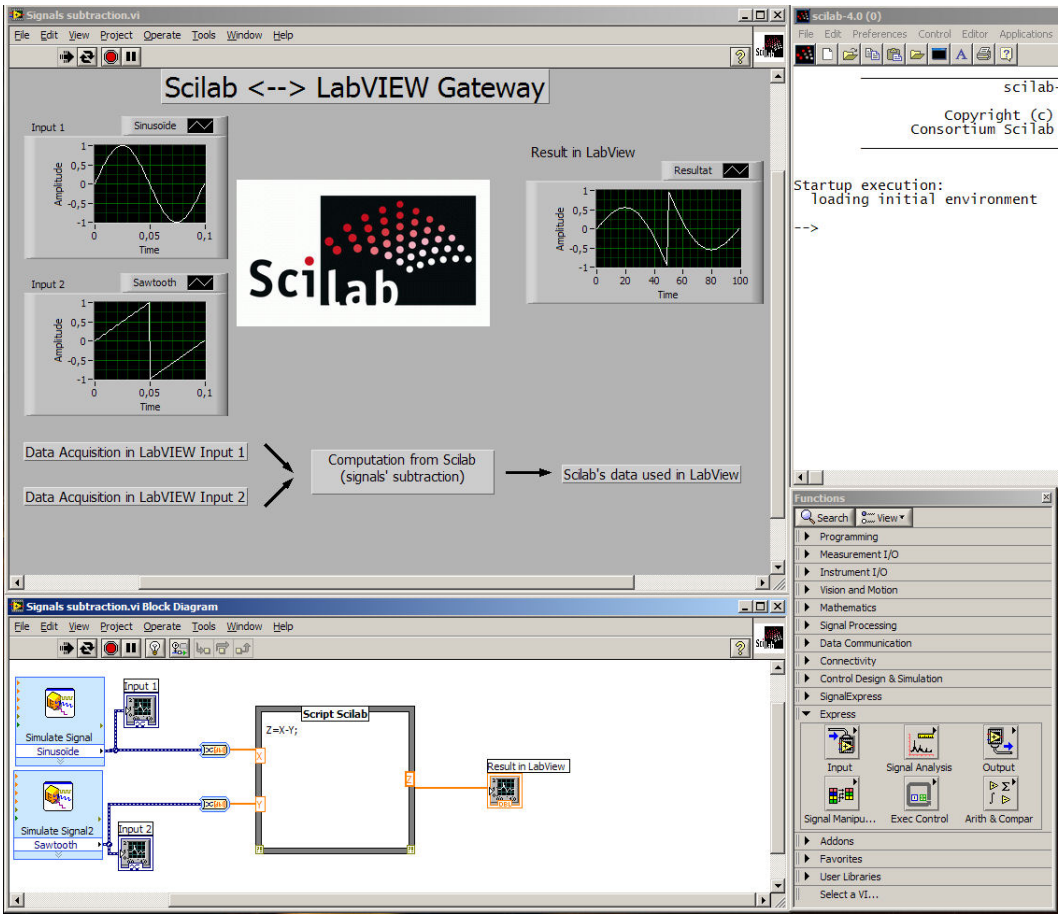

Figura 21: Scilab puede ser usado en instrumentos virtuales mediante LabVIEW Gateway

Los requerimientos de sistema para poder usar este archivo son los siguientes:

- Microsoft Windows Vista/XP/2000
- National Instruments LabVIEW (8.0, 8.2.x, or 8.5) completo, profesional, estudiantil, o ediciones de evaluación.
- INRIA Scilab (4.1.1 o posterior).

Para lograr el vínculo entre LabVIEW y Scilab se debe descargar de la dirección antes mencionada el archivo

`"labview_scilab_script_node"`

de extensión zip. Una vez descargado debe hacerse lo siguiente:

- Ir a la carpeta de LabVIEW.

- Descomprimir.

  "labview_scilab_script_node"

  en la carpeta raíz:

  C:\Archivos de programa\National Instruments\LabVIEW 8.5

- Si se dispone de la versión 8.5 de LabVIEW, que es nuestro caso, se debe cambiar

  C:\Archivos de programa\National Instruments\LabVIEW 8.5
  \menus\categories\Mathematics_scriptsAndFormulas
  \More Script Nodes

  por

  C:\Archivos de programa\National Instruments\LabVIEW 8.5
  \menus\categories\Mathematics_scriptsAndFormulas
  _More Script Nodes

- reiniciar LabVIEW.

Una vez hecho esto, podemos comprobar que Scilab ahora forma parte de los script Nodes de LabVIEW como se muestra en la figura 22, siguiendo la dirección:

Functions/Mathematics/Scripts & Formulas/Script Nodes/

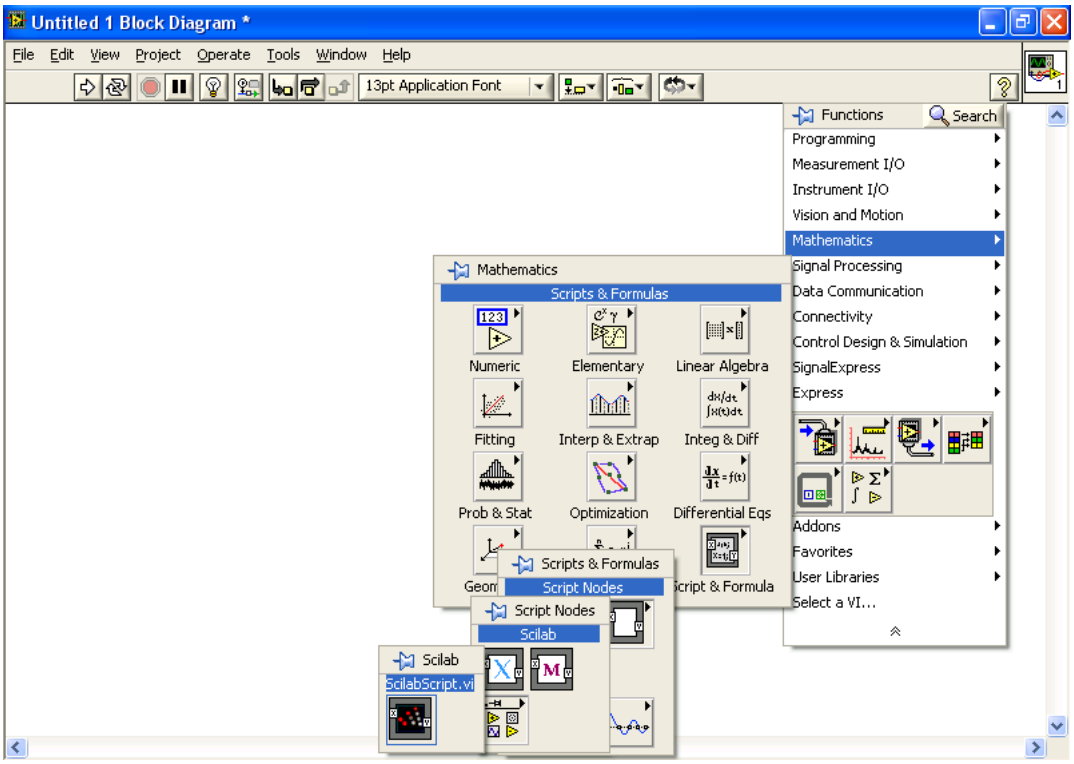

Figura 22: Ubicación del script node de Scilab.

Podemos arrastrar Script Node de Scilab hasta la ventana diagrama de bloques para trabajar con él. Se puede observar que aparece el cuadro de texto Scilab Script como se muestra en la figura 23, el cual sirve para escribir nuestro código matemático.

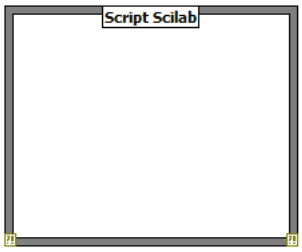

Figura 23: Cuadro de texto.

En la figura 24, se muestra como se pueden operar las entradas de nuestro instrumento virtual con el código matemático en Scilab al alambrar el nodo con estas.

Haciendo click derecho en el borde del nodo y haciendo una selección en el menú emergente se puede crear una variable de entrada o de salida (figura 24,1). Después, se escribe el nombre de la variable que se desea asociar a la nueva entrada (o salida) (figura 24, 2). Entonces se puede alambrar la variable de entrada o salida al nodo (figura 24, 3). Finalmente, en el cuadro de texto, se puede hacer uso de la variable alambrada al nodo escribiendo su nombre (figura 24, 4).

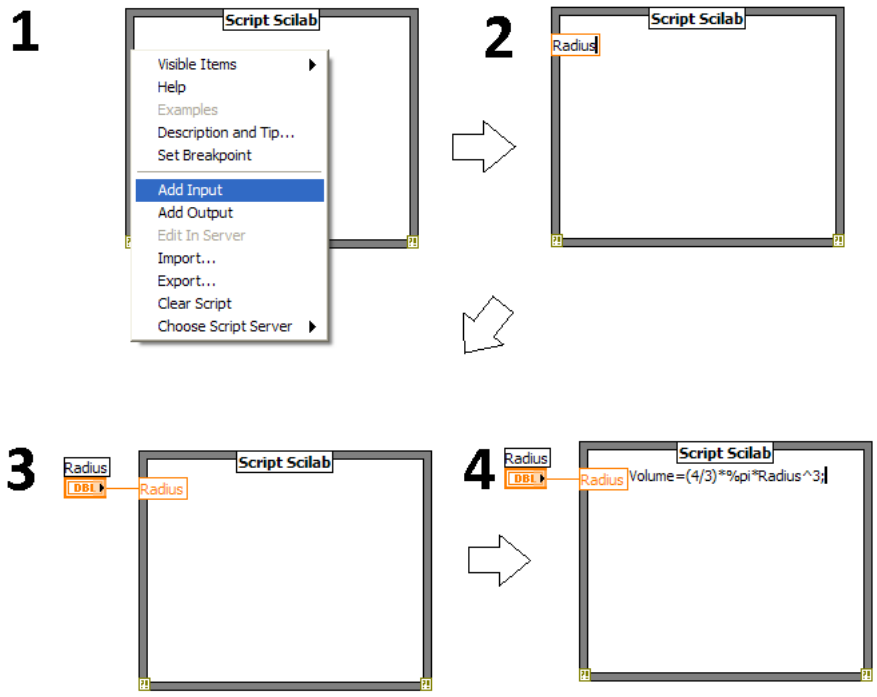

Figura 24: Conexión de variables de entrada o de salida de LabVIEW con el cuadro de texto de Scilab. Se puede trabajar estas variables con texto matemático de Scilab como se muestra en 4.

De esta manera es como podemos usar en LabVIEW el código matemático de Scilab.

4.2.3. Comandos en Scilab.

Scilab trabaja con comandos y una sintaxis muy parecidos a los que usa Matlab, gracias a esto, el aprendizaje de Scilab resulta muy sencillo y rápido. A continuación se presentarán varios comandos y sus aplicaciones en los temas que nos interesan en Fundamentos de Control, siendo estos álgebra lineal, manejo de polinomios, matrices, funciones de transferencia, análisis de respuesta en frecuencia, análisis en el tiempo y gráficos 2D.

Para invocar la ayuda de Scilab basta con escribir por ejemplo *help plot*, damos <enter> y se desplegará la ayuda del comando *plot*.

4.2.4. Constantes matemáticas.

Las constantes tales como π, e, i, etcétera se definen escribiendo el símbolo % antes de la variable.

■ Ejemplo **5** *Constantes matemáticas en Scilab de uso común:*

```
-->%pi  //constante pi
 %pi  =
```

```
    3.1415927

-->%e   //constante e
 e  =

    2.7182818

-->%i   //constante i (raiz cuadrada de -1)
 i  =

    i
```

4.2.5. Creación de un vector.

Para crear un vector basta con poner entre corchetes los elementos del vector, las columnas se separan con una coma o con un espacio en blanco y los renglones se separan con un punto y coma, tal como se hace en Matlab.

Ejemplo **6** *Para crear un vector se escribe:*

Si se quiere crear un vector renglón

$a = [1\ 2\ 3]$ *ó bien* $a = [1, 2, 3]$

Si se quiere crear un vector columna

$b = [1; 2; 3]$

Scilab nos arroja el siguiente resultado:

Vector renglón

```
a  = 1.    2.    3.
```

vector columna

```
b  =

          1.
          2.
          3.
```

Si queremos crear un vector teniendo como datos el valor inicial, el valor final y el espaciamiento entre cada valor utilizamos la siguiente sintaxis:

a = valor inicial : espaciamiento : valor final

■ Ejemplo **7** *Crear un vector t que inicie en 0 y termine en 1 con un espaciamiento de 0.1 entre cada dato.*

```
t=0:0.1:1
 t  =

    0.    0.1    0.2    0.3    0.4    0.5    0.6    0.7    0.8
0.9    1.
```

Si se quiere ocultar el resultado basta con escribir un punto y coma al final de la instrucción.

Para crear un vector columna con un valor inicial, un valor final y un espaciamiento se usa la misma instrucción pero encerrada entre paréntesis y con un apóstrofe al final de éste, el apóstrofe indica la transposición de un vector o una matriz.

■ Ejemplo **8** *Crear un vector transpuesto vector transpuesto t que inicie en 0 y termine en 1 con un espaciamiento de 0.1 entre cada dato.*

```
t=(0:0.1:1)'
 t  =

    0.
    0.1
    0.2
    0.3
    0.4
    0.5
    0.6
    0.7
    0.8
    0.9
    1.
```

4.2.6. Creación de una matriz.

Para crear una matriz se definen los renglones de esta entre corchetes separados por punto y coma. Los elementos de cada renglón se separan por un espacio en blanco o por una coma como se muestra en el ejemplo siguiente:

■ Ejemplo **9** *Crear la matriz* $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$

La matriz anterior se puede crear de dos maneras, escribiendo A=[1 2 3;4 5 6;7 8 9] o bien A=[1,2,3;4,5,6;7,8,9], el resultado que nos arroja Scilab es:

```
A  =

    1.    2.    3.
    4.    5.    6.
    7.    8.    9.
```

Definidas las matrices se pueden realizar las siguientes operaciones entre matrices:

```
 A+B   A*B   A-B
```

Para seleccionar una submatriz o algunos elementos de una matriz o columna existen las siguientes operaciones:

- a(2,5) denota el elemento de a en la fila 2 y en la columna 5.
- a(3,:) denota la tercera fila de la matriz a.
- a(:,4) denota la cuarta columna de la matriz a.
- a(1:2,2:5) denota la submatriz de tamaño 2*4, formado por los elementos que están en las filas 1, 2 y en las columnas 2 a 5.

4.2.7. Creación de polinomios.

Se pueden definir de dos formas:

- especificando sus raíces:
  $-->$p = poly([1,2],'x')

```
 p  =

              2
   2 - 3x + x
```

- especificando sus coeficientes:
  $-->$p=poly([1,2,3,4,5],'x','c')

```
 p  =

                2    3    4
   1 + 2x + 3x + 4x + 5x
```

Para encontrar las raíces del polinomio basta con usar el comando **roots()**.

Ejemplo **10** *Encontrar las raíces de un polinomio que tiene por coeficientes* $1, 2, 3, 4, 5$.

Calculamos el polinomio cuyos coeficientes son 1, 2, 3, 4 y 5.

```
-->p=poly([1,2,3,4,5],'x','c')
 p  =

              2    3    4
   1 + 2x + 3x + 4x + 5x
```

Obtenemos sus raíces con el comando **roots()**.

```
-->roots(p)
 ans  =

   0.1378323 + 0.6781544i
   0.1378323 - 0.6781544i
 - 0.5378323 + 0.3582847i
 - 0.5378323 - 0.3582847i
```

4.2.8. Creación de gráficos.

Para dibujar gráficos en 2D de alguna función y = f(t) se usa el comando **plot()** o bien **plot2d()**.

Ejemplo **11** *Dibujar la función seno con una amplitud de 31, una frecuencia de 1 Hz y un offset de 30.*

−− > t=(0:0.01:1);// Definición del rango de t.
*−− > ft = 21*sin(2*%pi*t)+30; // Definición de f(t).*
−− > plot2d(t, ft); // Ejecución del gráfico del tipo y = f(t).

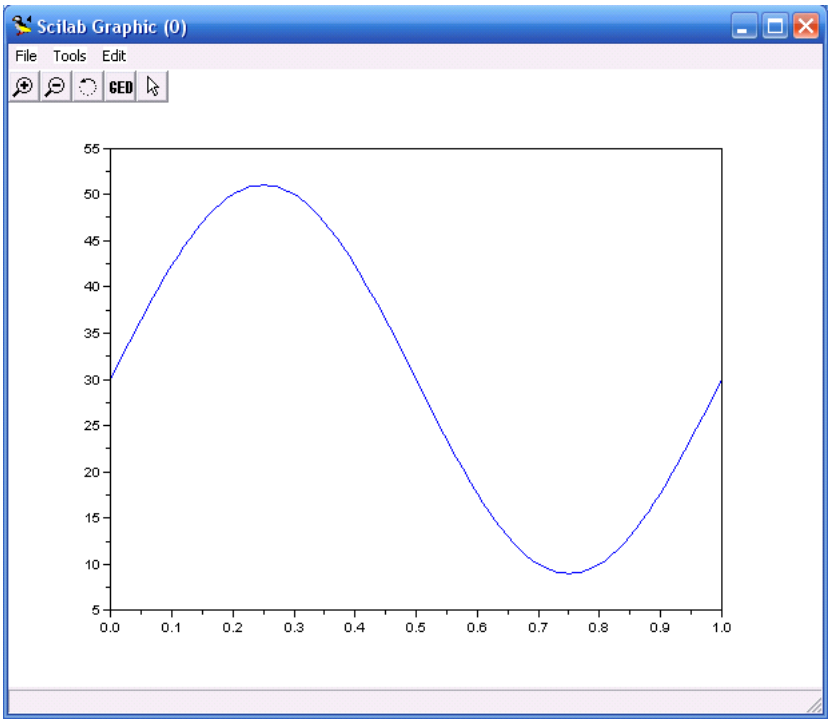

Figura 25: Uso del comando plot2d, el cual nos grafica una función.

La figura 25 muestra un ciclo de la función seno que se produjo con el código anterior.

Para dibujar gráficos paramétricos x = x(t), y = y(t) se usa el mismo comando **plot()**.

Ejemplo **12** *Dibujar un gráfico paramétrico de ecuaciones* $x(t) = (2 * cos(t) + 1). * cos(t)$ *y* $y(t) = (2 * cos(t) + 1). * sin(t)$.

−− > t=(-4:0.01:4); // Definición del rango de t.
*−− > xt = (2 * cos(t) +1) .* cos(t); // Definición de la función x(t).*
*−− > yt = (2 * cos(t) +1) .* sin(t); // Definición de la función y(t).*
−− > plot2d(xt, yt); // Ejecución del gráfico (x(t), y(t)).

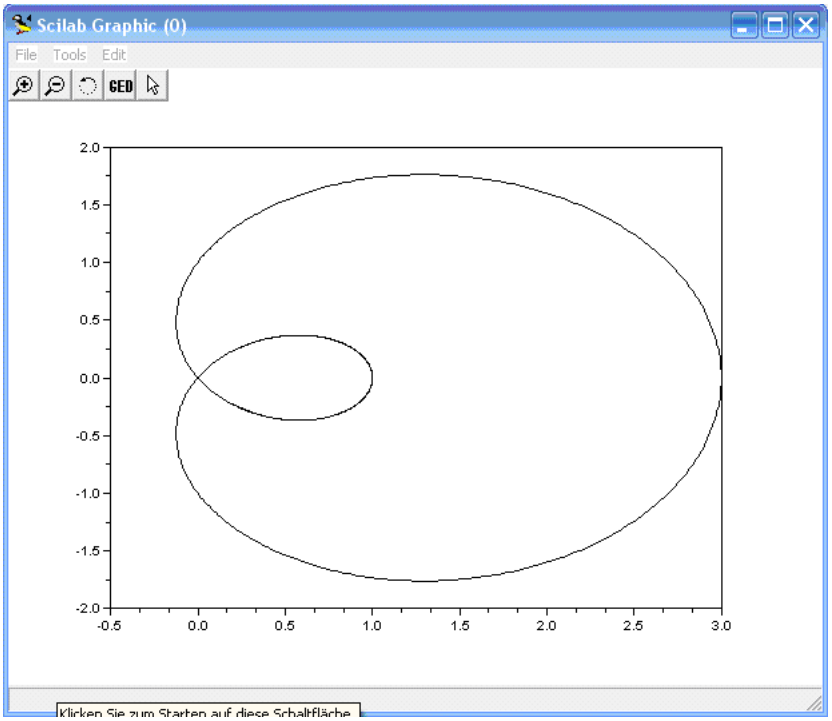

Figura 26: Uso del comando plot(), el cual nos grafica una función paramétrica.

En la figura 26 se muestra el resultado del código anterior, en el cual se grafica una función paramétrica conocida como cardioide.

4.2.9. Cálculo de la función de transferencia.

Para calcular una función de transferencia en Scilab, debemos crear un sistema de ecuaciones lineales y para eso ocupamos el comando **syslin**. Con este comando podemos convertir los polinomios del numerador y denominador en una función de transferencia.

Ejemplo **13** *Crear la función de transferencia:*

$$H(s) = \frac{Y(s)}{U(s)} = \frac{1}{s^2 + 0{,}25s + 1}$$

Procedemos a declarar la variable compleja **s**, *luego creamos los polinomios del numerador y el denominador usando la variable* **s** *y después creamos un sistema lineal:*

```
-->s=poly(0,'s') //definimos la variable compleja s

 s  =

    s

-->num=poly([1],'s','c')//creamos el polinomio del numerador
                         //a partir de sus coeficientes
 num  =

    1

-->den=poly([1 0.25 1],'s','c')//creamos el polinomio del
                               //denominador a partir de sus
                               //coeficientes
 den  =
                 2
    1 + 0.25s + s
-->FT=syslin('c',num,den) //creamos la función de transferencia,
                         //la 'c' significa
                         //que trabajamos en el dominio del
                         //tiempo continuo
 FT  =
          1
    -------------
                 2
    1 + 0.25s + s
```

Si queremos poner una realimentación negativa usamos el operador **/.[-1]**.

■ Ejemplo **14** *Aplicar una realimentación positiva a la función de transferencia en lazo abierto definida por* $\frac{1}{20+10s+s^2}$.

```
-->s=poly(0,'s') //definimos la variable compleja s

 s  =

    s

-->num=poly([1],'s','c')//creamos el polinomio del numerador
                         //a partir de sus coeficientes
 num  =

    1

-->den=poly([1 0.25 1],'s','c')//creamos el polinomio del
                               //denominador a partir de sus
                               //coeficientes
 den  =
                 2
    20 + 10s + s

-->g=syslin('c',num,den)       //funcion de transferencia
```

```
 g  =                          //en lazo abierto

         1
    -----------
                2
    20 + 10s + s

-->ft=g/.[1]                   //aplicando la realimentacion
 ft  =                         //positiva

         1
    -----------                //funcion de transferencia
                2              //en lazo cerrado con
    21 + 10s + s               //realimentación positiva
```

Si queremos multiplicar dos funciones de transferencia usamos el operador ***.

Ejemplo **15** *Multiplicar las funciones de transferencia* $G_1 = \dfrac{1}{10+s}$ *y* $G_2 = \dfrac{1}{1+s}$.

```
-->s=poly(0,'s');

-->G1=syslin('c',poly([1],'s','c'),poly([10 1],'s','c'))
 G1  =

      1
    -----
    10 + s

-->G2=syslin('c',poly([1],'s','c'),poly([1 1],'s','c'))
 G2  =

      1
    -----
    1 + s

-->w1=G1*G2
 w1  =

         1
    -----------
                2
    10 + 11s + s
```

4.2.10. Respuesta a escalón.

Para calcular la respuesta en el tiempo de un sistema usamos el comando **csim(función, vector de tiempo, sistema lineal)**.

Ejemplo **16** *Calcular la respuesta a una entrada escalón del sistema definido por la funcion de transferencia* $\dfrac{1}{1+0{,}25s+s^2}$.

```
-->s=poly(0,'s');
```

```
-->num=poly([1],'s','c')
 num  =

    1

-->den=poly([1 0.25 1],'s','c')
 den  =

                2
    1 + 0.25s + s

-->sistema6=syslin('c',num,den) //función de transferencia
 sistema6  =

          1
    -------------
                2
    1 + 0.25s + s

-->instante=0:0.05:45; //vector de tiempo

-->y=csim('step',instante,sistema6);

-->plot2d(instante',y')
```

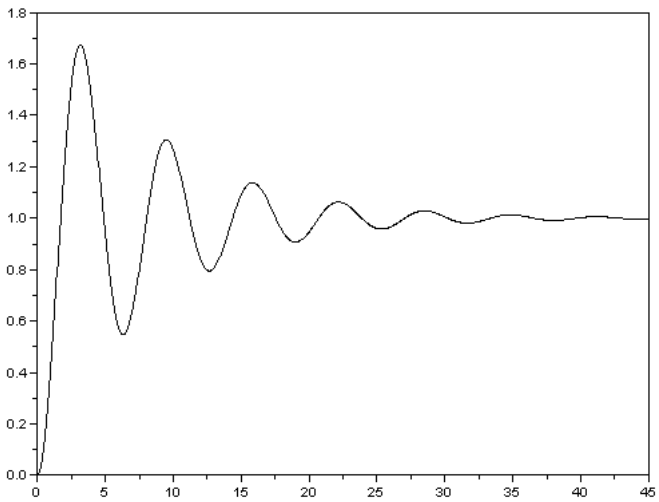

Figura 27: Respuesta a una entrada escalón.

La figura 27 muestra la respuesta de la función de transferencia anterior con una entrada escalón.

4.2.11. Trabajando en espacio de estados.

Para crear el espacio de estados a partir de sus matrices se utiliza el comando **Sss**, creamos las matrices A (matriz de estado), B (matriz de entrada), C (matriz de salida) y D (matriz de transmisión directa) y las ponemos como argumentos del comando.

Ejemplo **17** *Crear un sistema de ecuaciones de estado a partir de las matrices de estado, de entrada, de salida y de transmisión directa:*

```
-->A = [-5 -1;6 0];
-->B = [-1; 1];
-->C = [-1 0];
-->D =0;
-->Sss = syslin('c',A,B,C,D) //la 'c' indica que se trabaja en
                             //el dominio del tiempo continuo.
Sss  =


       Sss(1)   (state-space system:)

!lss  A  B  C  D  X0  dt  !

       Sss(2) = A matrix =         //matriz de estado

  - 5.  - 1.
    6.    0.

       Sss(3) = B matrix =         //matriz de entrada

  - 1.
    1.

       Sss(4) = C matrix =         //matriz de salida

  - 1.    0.

       Sss(5) = D matrix =         //matriz de transmisión directa

    0.

       Sss(6) = X0 (initial state) =  //estado inicial del sistema

    0.
    0.

       Sss(7) = Time domain =         //se trabaja en el dominio
                                      //del tiempo
 c
```

Para extraer las matrices de un espacio de estados se utiliza el comando **[A,B,C,D]=abcd(Sss)**.

4.2.12. Toolboxes.

Scilab tiene la posibilidad de trabajar con herramientas creadas por el usuario (toolboxes), en este caso trabajaremos con la herramienta rltool versión 1.7 para scilab 4.1, la cual puede descargarse del siguiente vínculo:

```
http://www.scilab.org/contrib/download.php?
 fileID=1019&attachFileName=rltool-1.7.tar.gz
```

Una vez descargado y descomprimido el archivo, hay que ejecutar el toolbox. Desde el menu principal de Scilab se ejecuta lo siguiente:

Archivo− >Exec− >loader.sce

Después ejecutar el comando **rlt()** y elegir "nuevo", aparecerá la pantalla siguiente:

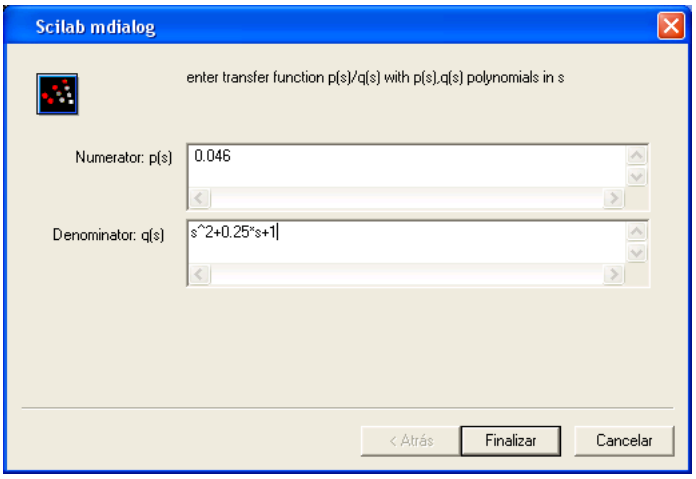

Figura 28: Cuadro de diálogo de rltool.

En los campos mostrados en la figura 28 debemos especificar el numerador y denominador de la función de transferencia, usaremos la siguiente función de transferencia:

$$H(s) = \frac{0{,}046}{s^2 + 0{,}25s + 1}$$

Después de proporcionarlos presionar <Finalizar>.

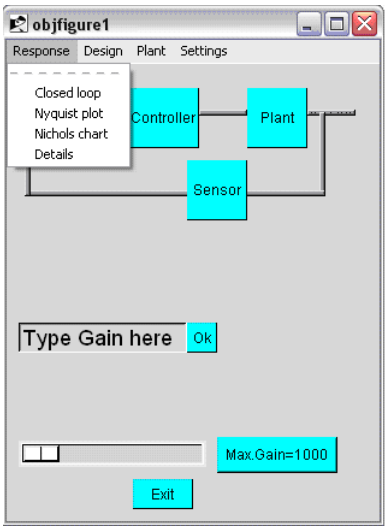

Figura 29: Pantalla principal.

La figura 29 muestra un diagrama de cómo queda el control realimentado con su ganancia, controlador y sensor. Por default muestra el diagrama de polos y ceros, en la opción **Response** del menú principal se pueden encontrar varias opciones según lo que queramos ver.

Cuando elegimos lazo cerrado nos aparecen las opciones que se muestran en la figura 30 las cuales brindan la respuesta a escalón, impulso, rampa etcétera del sistema.

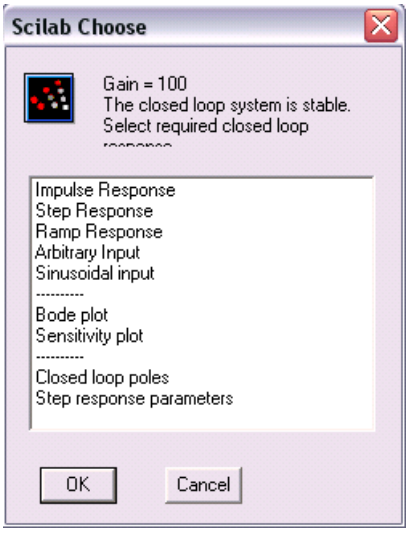

Figura 30: Diferentes opciones para el análisis de la función de transferencia.

También podemos usar únicamente el panel de control y proporcionarle una ganancia para ver su respuesta en el tiempo como lo muestra la figura 31.

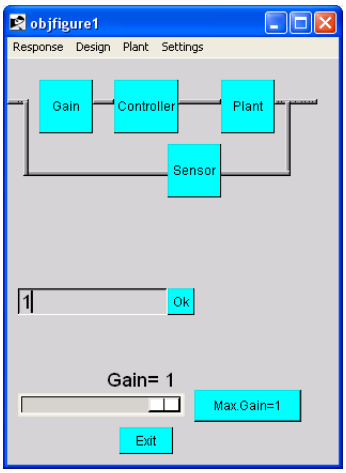

Figura 31: Panel de control, al proporcionar una ganancia nos mostrará su respuesta a escalón y diagramas de bode.

Al proporcionar una ganancia y dar click en OK nos muestra el diagrama de Bode, la respuesta al escalón unitario y la gráfica de sensibilidad, tal como se muestra en la figura 32.

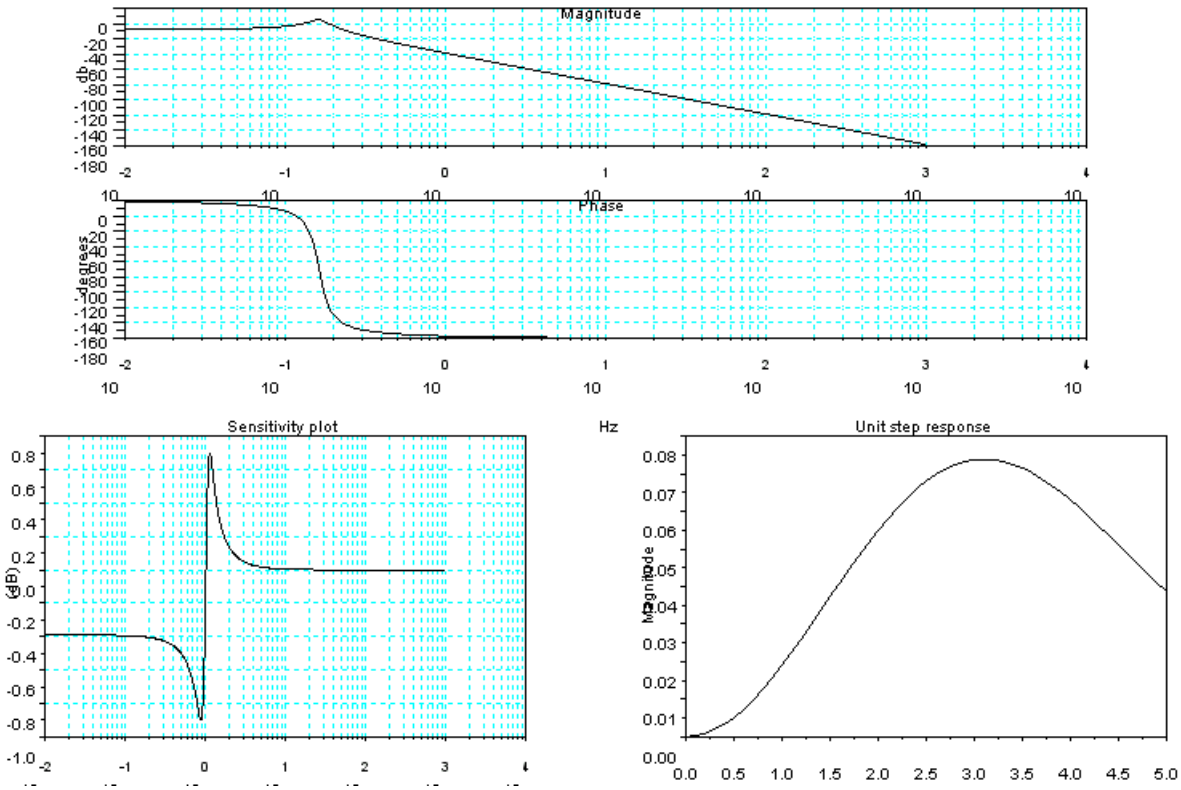

Figura 32: Diferentes gráficas que proporciona la herramienta rltool.

4.3. LabVIEW

LabVIEW tuvo inicio en 1986,cuando la compañía National Instruments empezó la instrumentación virtual, que permite a usuarios definir su propia solución usando software integrado a una computadora y una amplia variedad de hardware. Con las aplicaciones de medición y automatización de LabVIEW, se puede adquirir información de algún fenómeno o sistema físico, para procesarla y generar una respuesta deseada teniendo la posibilidad de presentar la información leída y la respuesta generada por medio de una interfáz gráfica, páginas Web o bien almacenarla para su uso posterior.

El software LabVIEW tiene funciones específicas para acelerar el desarrollo de aplicaciones de medida, control y automatización, nos proporciona herramientas poderosas para que el usuario pueda crear aplicaciones sin líneas de código (lenguaje C) por lo que en LabVIEW se crean programas basados en diagramas de bloques en un entorno de desarrollo gráfico con funciones integradas para realizar la adquisición de datos, control de instrumentos, análisis de medida y presentaciones de datos.

No se puede dejar de mencionar que LabVIEW se puede conectar con todo tipo de hardware incluyendo instrumentos de escritorio, tarjetas insertables, controladores de movimiento y controladores lógicos programables (PLCs).

Otra de las razones que explican las ventajas de este software es que como las necesidades de las aplicaciones van cambiando con el tiempo, los sistemas definidos y creados por el usuario de LabVIEW tienen la movilidad y la flexibilidad necesaria para adecuarse sin la necesidad de incorporar equipos nuevos.

4.4. Principales características

Hoy en día, científicos, ingenieros, técnicos y estudiantes utilizan LabVIEW para desarrollar soluciones que respondan a sus interrogantes más exigentes, es por ello que su principal característica es la facilidad de uso que posee. También es útil para personas con pocos conocimientos en programación, ya que pueden construir aplicaciones relativamente complejas, imposibles para ellos con los lenguajes tradicionales. LabVIEW posee facilidad de manejo para las siguientes interfaces de comunicación:

- Puerto Serie.
- Puerto Paralelo.
- GPIB.
- USB.
- Bluetooth.

entre otros.

LabVIEW posee la capacidad de interactuar con lenguajes y aplicaciones:

- DLL (Librerías de funciones), NET, ActiveX, MultiSim, Matlab/Simulink, Scilab, AutoCAD, entre otros.
- Herramientas gráficas y textuales para el proceso digital de señales.
- Visualización y manejo de gráficas con datos dinámicos.
- Adquisición y tratamiento de imágenes.
- Control de movimiento.
- Tiempo real.

- Programación de FPGA's (Field Programmable Gate Array) para control o validación.
- Sincronización entre dispositivos.

Se pueden realizar tareas como:

- Adquisición de datos.
- Automatización industrial.
- Diseño embebido.
- Control de instrumentos.
- Diseño de control.

entre otros.

4.4.1. Programación LabVIEW

Los programas desarrollados mediante LabVIEW se denominan Instrumentos Virtuales (VIs), porque su apariencia y funcionamiento imitan los de un instrumento real. Los VIs tienen una parte interactiva con el usuario y otra parte de código fuente, y aceptan parámetros procedentes de otros VIs.

El software puede iniciarse a través del acceso directo o del menú inicio con la siguiente ruta:

Inicio $->$ Programas $->$ National Instruments $->$ LabVIEW 8.5

4.4.2. Partes de un VI

Se da inicio en *Blank VI*= VI en blanco. Al hacer click en *Blank VI*, automáticamente se abren 2 ventanas, una llamada Panel Frontal (Front Panel, figura 33) y la otra llamada Diagrama de Bloques (Block Diagram), figura 34).

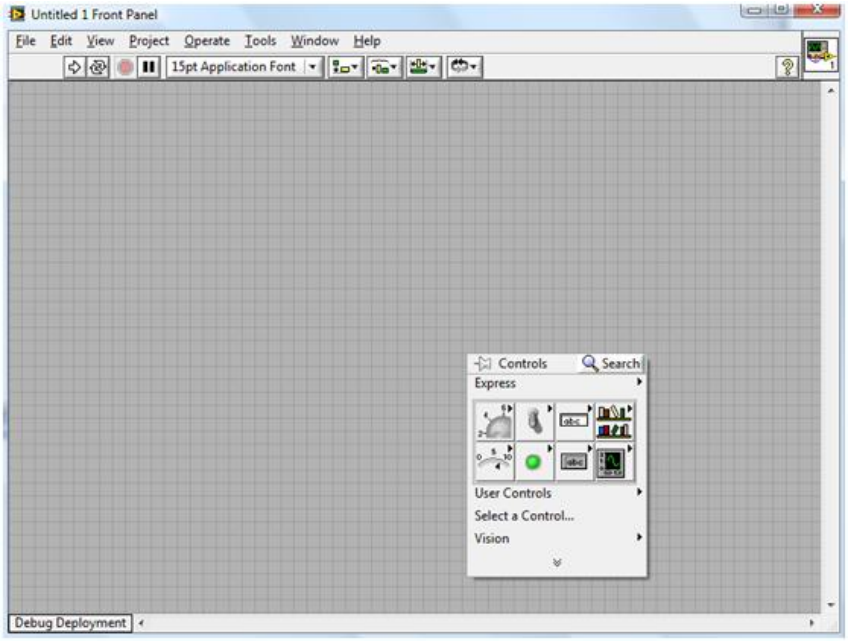

Figura 33: Panel frontal

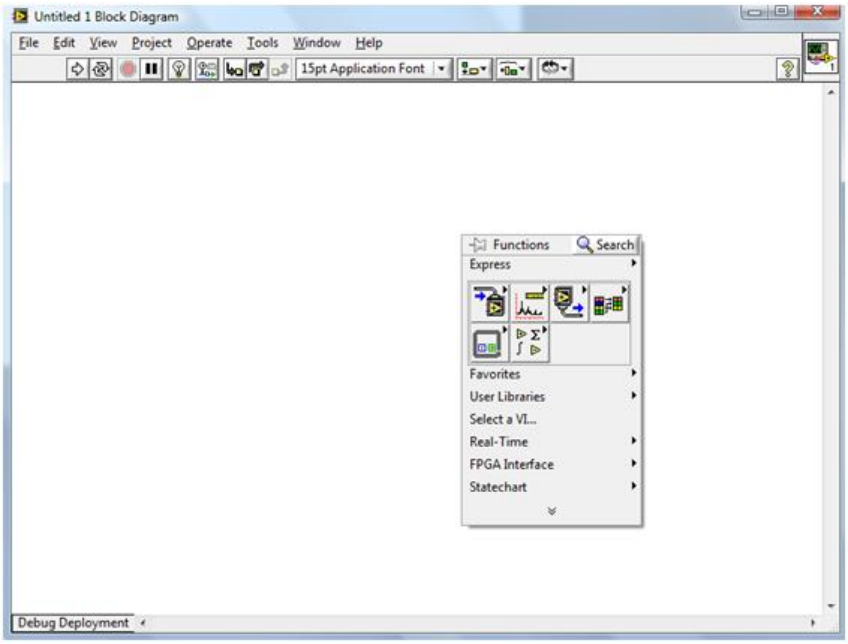

Figura 34: Diagrama de Bloques (Block Diagram)

4.4.3. Panel Frontal (Front Panel).

Es la interfaz gráfica del VI con el usuario, donde se puede controlar el programa, cambiar entradas y ver datos actualizados en tiempo real. Un panel frontal está formado por:

- Controles que sirven para introducir parámetros al VI. Estos pueden ser botones, botones de empuje y marcadores que simulan las entradas de equipos y suministran datos al Diagrama de Bloques (Block Diagram) del VI.
- Indicadores que se emplean para mostrar los resultados producidos, ya sean datos adquiridos o resultados de alguna operación; pueden ser: gráficas, luces y otros dispositivos que simulan salidas de instrumentos y suministran datos que el Diagrama de Bloques (Block Diagram) adquiere.

Las herramientas a usar en el Panel Frontal (Front Panel) se encuentran en la paleta de controles **(Controls)**. En la paleta de controles se ubican los controles e indicadores necesarios para poder crear un instrumento virtual. Para acceder a la paleta de controles se debe seguir la ruta Window $->$ Show Controls. en el menu principal del Panel Frontal (Front Panel). La paleta de controles se muestra en la figura 33.

4.4.4. Diagrama de Bloques (Block Diagram).

Contiene el código fuente gráfico. Los objetos del Panel Frontal (front panel) aparecen como terminales en el diagrama de bloque, que se construye conectando los distintos objetos entre si, como si se tratara de un circuito. Los cables unen terminales de entrada y salida con los objetos correspondientes, y por ellos fluyen los datos por lo que el código es el que controla el programa.

Se cuenta con una extensa biblioteca de funciones, entre ellas, aritméticas, comparaciones, conversiones, funciones de entrada/salida, de análisis, etc.

La paleta de funciones **Functions** nos permite construir un Diagrama de Bloques (Block Diagram), está disponible en el menu principal de la ventana Diagrama de Bloques (Block Diagram) siguiendo la ruta Window $->$ Show Functions Palette, o bien al hacer click derecho en el espacio de trabajo del diagrama de bloques. La paleta de funciones se muestra en la figura 34.

4.4.5. Paleta de herramientas (Tools palette).

Se emplea tanto en el Panel Frontal (front panel) como en el diagrama de bloques. Contiene las herramientas necesarias para editar y depurar los objetos en dichas ventanas, ver figura 35.

Figura 35: Paleta de herramientas.

Las opciones que presenta esta paleta son las siguientes:

- Operating tool - Cambia el valor de los controles.
- Positioning tool - Desplaza, cambia de tamaño y selecciona los objetos.
- Labeling tool - Edita texto y crea etiquetas.
- Wiring tool - Une los objetos en el diagrama de bloques.
- Object Pop-up Menu tool - Abre el menú desplegable de un objeto.
- Scroll tool - Desplaza la pantalla sin necesidad de emplear las barras de desplazamiento.
- Breakpoint tool - Fija puntos de interrupción de la ejecución del programa en VIs, funciones y estructuras.
- Probe tool - Crea puntos de prueba en los cables, en los que se puede visualizar el valor del dato que fluya por dicho cable en cada instante.
- Color Copy tool - Copia el color para después establecerlo mediante la siguiente herramienta.
- Color tool - Establece el color de fondo y el de los objetos

Para la identificación de las herramientas que contienen el Panel Frontal (front panel) y el Diagrama de Bloques (Block Diagram) se identifican las variables comunes de acuerdo al color de línea que tiene, ver figura 36.

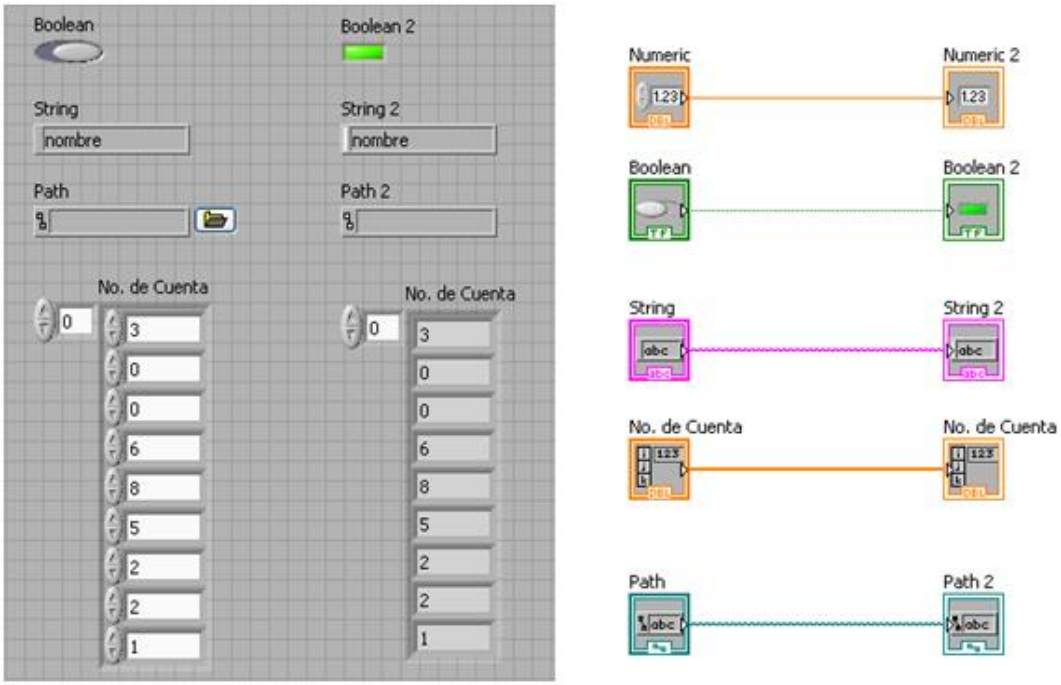

Figura 36: Dentro del Panel Frontal (front panel) y el Diagrama de Bloques (Block Diagram)s se encuentra la identificación de parámetros

El color naranja de los iconos indica que es un valor numérico, el color verde indica que es un valor booleano y el color rosa indica que es un texto.