UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DETECCIÓN DE FUGAS EN DUCTOS POR SOFTWARE

TESIS PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO MECÁNICO
ÁREA MECATRÓNICA

PRESENTA:

JORGE RAMÍREZ GUERRA

Director de tesis:

Dra. Cristina Verde Rodarte

CIUDAD UNIVERSITARIA, MÉXICO, D.F. 2008

Índice

Capítulo I

Introducción

El transporte de fluidos por redes de tubería es una función fundamental en muchos procesos cotidianos como distribución de agua potable o hidrocarburos. Para garantizar dicha función, las tuberías son diseñadas y protegidas contra diferentes fenómenos que pudiesen dañarlas. Una forma particular de protección contra fugas es la vigilancia o monitoreo del sistema, el cual puede ser periódico o continúo dependiendo de las necesidades impuestas por la naturaleza del fluido, tal es el caso del desarrollo y prueba de un método analítico para detección de fugas en ductos de hidrocarburos en fase líquida del Instituto Mexicano del Petróleo en colaboración con la UNAM, del cual forma parte esta tesis. La tendencia general es la de incorporar una mayor seguridad en los procesos sin perder de vista el aspecto económico; dadas las actuales técnicas y equipos, existe una creciente preferencia por sistemas que son monitoreados de manera continua.

Tal vigilancia puede realizarse mediante instrumentos junto con sistemas de procesamiento de la información proveniente de la planta, lo que hace posible detectar cuando los datos indican una desviación de los valores normales de operación, sin embargo, la utilización solamente de instrumentos puede resultar costosa en comparación con el uso de algoritmos computacionales.

En los últimos años se han desarrollado diversos esquemas de detección y aislamiento de fallas (*Fault Detection and Isolation* (FDI)) basados en el modelo matemático del fluido. Por ejemplo, (Verde, 2001) valida con datos experimentales un método basado en observadores con entradas desconocidas considerando un modelo lineal del fluido capaz de detectar dos fugas simultáneas, suponiendo mediciones únicamente de gastos y presiones en los extremos del ducto, sin embargo, la posición de las fugas es estimada con poca precisión ante comportamientos no lineales severos del fluido.

Un procedimiento de monitoreo que se usa para detectar, aislar fallas y evaluar su significado y gravedad en un sistema es llamado sistema de diagnostico de fallas (Patton et al., 2000). Tal sistema normalmente consta de las siguientes tareas:

- Detección de la falla: tomar una decisión binaria, ya sea que todo esté bien o no.
- Aislamiento de la falla: determinar la localización de la falla, es decir, qué sensor, actuador o componente ha fallado.
- Identificación de la falla: estimar el tamaño y tipo o naturaleza de la falla.

Dentro de la comunidad de FDI, una falla se entiende como una desviación no permitida de al menos una propiedad característica de una variable con un comportamiento inadecuado (Isermann,1997). Estos malos funcionamientos pueden ocurrir ya sea en los sensores, en los actuadores o en los componentes del proceso. Entre las técnicas más estudiadas en el área de FDI se encuentran los métodos de redundancia analítica. Esta metodología implica que se utiliza la redundancia inherente contenida en las relaciones dinámicas y estáticas entre las entradas y las salidas del sistema, es decir, se hace uso de un modelo matemático del sistema.

La idea básica de redundancia analítica es verificar que el comportamiento del sistema real tenga consistencia con el modelo nominal.

Una de las ventajas del enfoque de redundancia analítica es el hecho de que la redundancia existente se puede evaluar mediante un proceso de información bajo condiciones de operación bien caracterizadas sin necesidad de instrumentación física adicional en la planta.

En (Verde & Visairo, 2001), el problema de detección de fallas se resuelve utilizando un modelo matemático no lineal del fluido con posiciones fijas, sin embargo, este método no es robusto con respecto a incertidumbres en la posición y sólo puede ser aplicado para detectar y localizar fugas en casos muy limitados. Ya que en una aplicación real como el transporte de hidrocarburos la localización de las fugas tiene una alta prioridad, se formula el problema de localización con dos fugas. La posición de la fuga se determina minimizando el efecto de una y maximizando el de la otra.

Este trabajo toma como base un algoritmo de detección de fallas del tipo mencionado, un método que permite detectar 2 fugas de líquido en un ducto sin tomas laterales, considerando conocidas sus posiciones en el análisis del modelo del ducto utilizado, y las presiones y gastos en los extremos del ducto en la operación.

Debe mencionarse que en cualquier sistema real existen incertidumbres y perturbaciones externas en el sistema de medición, por lo que en casos prácticos, los mediciones pueden desviarse de su valor real aún ante la ausencia de fuga moderadamente.

Es así que esta tesis tiene como objetivo principal el desarrollo de un sistema de software para adquisición de datos y control en tiempo real alojado en una PC como el mostrado en la fig. 1.1, para registro de gastos y presiones a la entrada y salida en un ducto, desde el cual se realice la operación del sistema de bombeo en la tubería y la apertura y cierre de una servo-

válvula para la generación de fugas de manera controlada, además del muestreo y almacenamiento de lecturas de las variables que nos interesan. La integración de este sistema con el algoritmo de detección así como validar el detector de fugas mediante pruebas y valores experimentales, sin más equipo que la PC y los sensores de flujo volumétrico y presión del ducto completan esta investigación.

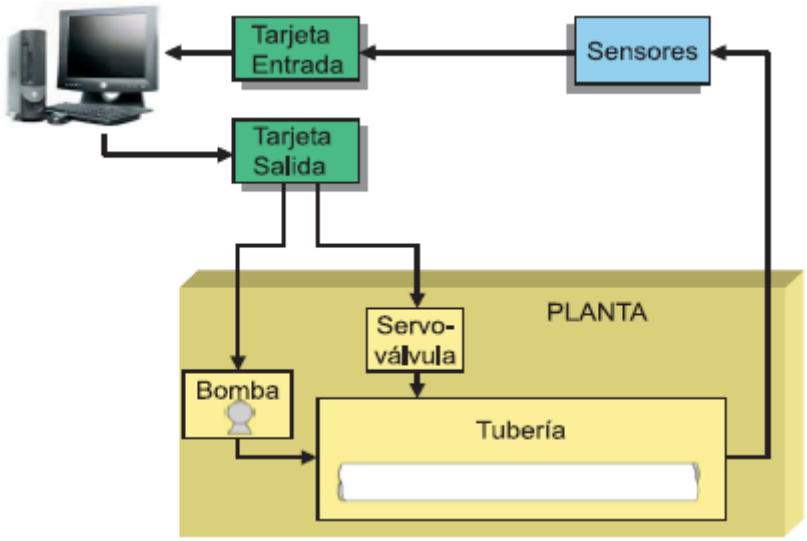

Fig. 1.1 Realización del algoritmo de detección de dos fugas

El Capítulo II de esta tesis es una descripción de todos los elementos del sistema físico de nuestro estudio, desde las instalaciones de trabajo hasta el equipo electrónico comercial y especializado con el que se contó para la realización de las pruebas experimentales y obtención de datos. El diseño del sistema de adquisición de datos se encuentra en el Capítulo III, presentando la filosofía de la teoría de control que sirvió para el desarrollo de este sistema, introduciendo al lector a la tecnología de software empleada en el proceso de diseño que se describe en esta sección. El Capítulo IV contiene el marco teórico del modelo del fluido y el algoritmo de detección que se pretenden validar. En el Capítulo V se exponen los resultados experimentales obtenidos al trabajar con el algoritmo de detección y los datos obtenidos en campo. Las conclusiones son presentadas en el Capítulo VI, sin perder de vista los objetivos planteados al inicio de este trabajo. Por último se incluye una lista de las referencias consultadas durante el desarrollo de este trabajo, así como apéndices con información específica sobre el proyecto.

Capítulo II

Descripción de Equipo e Instrumentación

Este capítulo se dedica a describir el diseño del equipo utilizado en las pruebas realizadas en el desarrollo de esta tesis, mostrando las características del circuito hidráulico y el equipo de monitoreo construidos para la detección y localización de fugas en ductos. El sistema de tubería proporciona un flujo de agua controlado así como una o dos fugas provocadas a la tubería mientras se monitorean y registran por computadora una o más variables físicas de interés.

II.1 Sistema de Tubería

La instalación se encuentra en el Laboratorio de Ingeniería Hidromecánica del Instituto de Ingeniería de la UNAM en Ciudad Universitaria y es descrita a continuación.

El circuito hidráulico de la fig. 2.1 y fig.2.2 consiste en una tubería de fundición de hierro de 132 m. de longitud y 0.10 m. de diámetro que nos permitió simular una instalación industrial de transporte de fluidos. La arquitectura de la tubería nos facilita generar fugas mediante válvulas colocadas en cuatro diferentes puntos a lo largo de ésta, siendo la más recurrente a 68 m. de los sensores aguas arriba donde se cuenta con una servo-válvula controlada por software obteniéndose exactitud en el tiempo y magnitud de apertura.

En la fig. 2.3, se puede ver el depósito de agua con aproximadamente 5000 lt. de capacidad, que sirve de alimentación al circuito de la tubería, recorriendo su trayectoria y regresando al depósito, constituyendo por esto un circuito cerrado de flujo. Esta configuración puede presentar ciertos inconvenientes, tales como turbulencia en el fluido, aire mezclado en el agua y el riesgo de que al obstruirse la salida del ducto por el mismo líquido, se alteren los parámetros del modelo.

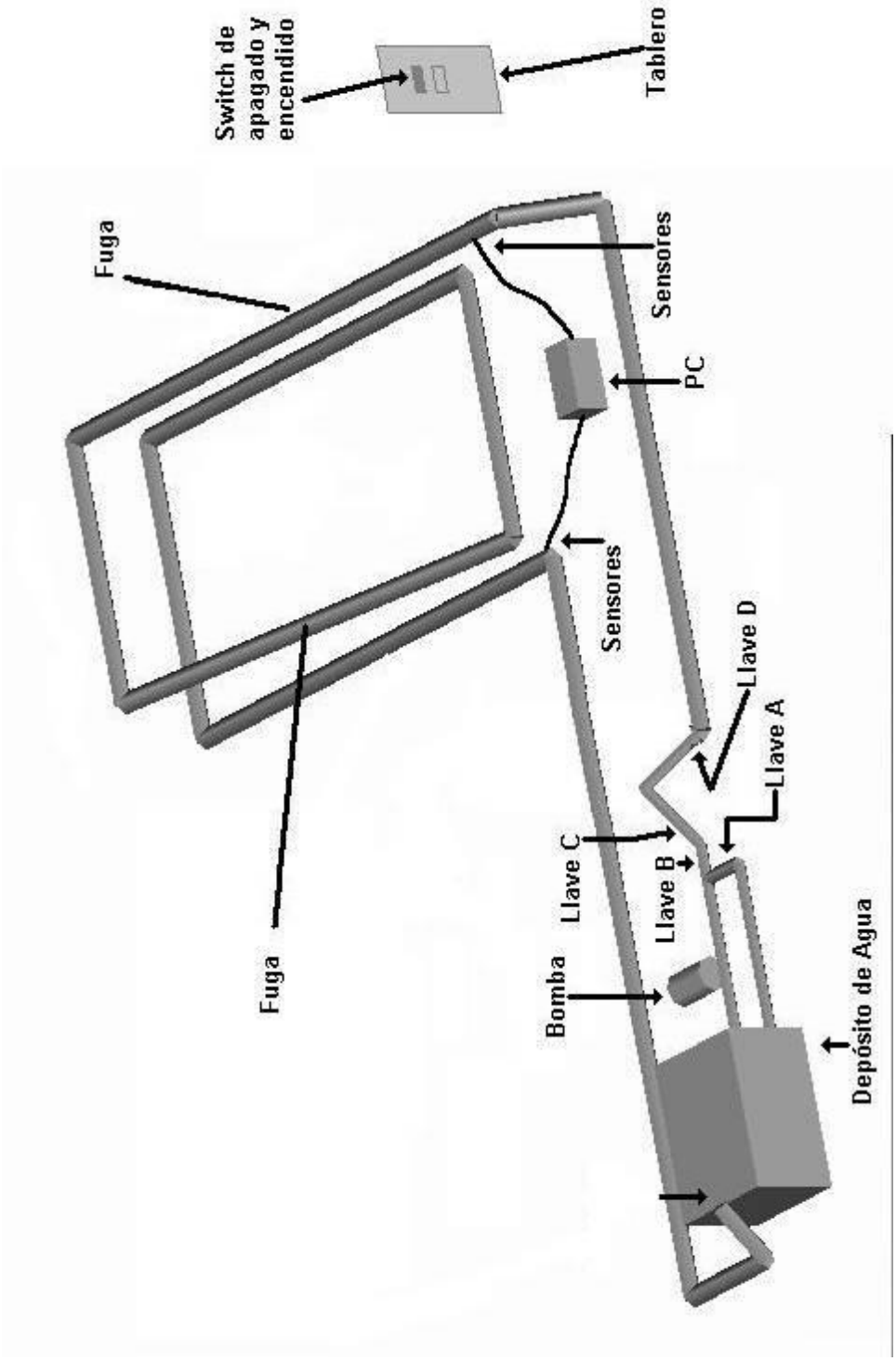

Fig. 2.1 Esquema de la instalación experimental

Figura 2.2 Vista del laboratorio

Fig. 2.3 Depósito de agua

Es importante que es necesario buscar condiciones de operación en la tubería que den un mayor grado de repetitibilidad a las pruebas, tales como mantener limpia el agua y el tanque, verificar los niveles del depósito y mantener la salida del ducto libre de obstrucciones.

II.2 Excitación del sistema

El elemento de control de la bomba utilizado fue un Inversor de Frecuencia Mitsubishi modelo A500 encargado del control del motor eléctrico trifásico Siemens de 5 HP de potencia instalado a un costado del tanque, encargado de hacer circular el agua por el ducto.

Fig. 2.4 Inversor de frecuencia

Este inversor mostrado en la fig. 2.4, puede ser operado en diferentes modos: externo, local y combinado, siendo de nuestro interés el primero de ellos y que fue seleccionado por medio de la programación de parámetros (Mitsubishi Electric).

El inversor se manejó bajo el control de señales externas a manera de variaciones de corriente en el rango de 4 a 20 *mA* suministrada por una tarjetas E/S de la sección de interfaz mediante una entrada analógica conectada a terminales del inversor, este proceso se describe en la sección II.5.1.

El inversor transforma la entrada de corriente en salida de frecuencia de voltaje de operación del motor eléctrico síncrono, lo que varía en forma directa y proporcional su velocidad de giro, dándonos la posibilidad de controlar el flujo en la tubería. Los rangos de operación para la realización de pruebas experimentales estuvieron entre 45 y 65 *Hz*, como medida de seguridad para el inversor y el motor, obteniéndose rangos para la presión y el gasto entre 12 a 15 *psi* y 15 a 20 *l/s* respectivamente.

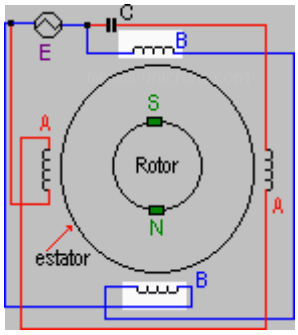

Fig. 2.5 Funcionamiento del motor AC

Podemos ver la relación existente entre la frecuencia del voltaje y la velocidad de rotación del motor fig. 2.5 por la ecuación que lo describe: $Ns = 60 \times f / p$, Donde:

- Ns = velocidad del motor en rpm (revoluciones por minuto)
- f = frecuencia de la alimentación en Hertz (Hz)
- p = número de pares de polos del motor.

II.3 Servo-válvula

La servo-válvula utilizada para generar fugas de manera controlada es un actuador capaz de abrir mecánicamente una válvula esfera hacia un depósito temporal de donde se reenvía el agua al tanque principal.

Fig. 2.6 Servo-válvula instalada a 68 m aguas abajo del tanque

Básicamente consta de un motor de 115 *VAC* y un controlador *Peaktronics* que posiciona la flecha del motor proporcionalmente a la señal de entrada de corriente en un rango de 4 a 20 *mA*, permitiendo así el paso de agua por la válvula. Como puede verse en la fig. 2.6, este dispositivo cuenta con una manija indicadora del sentido de giro que además es útil si se pretende girar manualmente el motor.

II.4 Sensores

Durante el proceso de validación, se realizaron gran número de pruebas en laboratorio, en las cuales se adquirieron lecturas de las variables de interés descritas en el Capítulo IV, gastos y presiones en los extremos de la tubería, utilizando los sensores descritos a continuación.

II.4.1 Sensor de flujo ultrasónicos

El sensor de flujo digital *Panametrics* no invasivo, consta de dos transductores colocados en la superficie del ducto fig. 2.7, un emisor y un receptor de la señal de ultrasonido que viaja a través del fluido en su dirección de flujo, y que es enviada desde la interfaz electrónica que monitorea las mediciones, las convierte a unidades útiles para almacenarlas en su memoria interna y a voltaje para ser enviadas a la PC por sus salidas analógicas (*Panametrics*).

Las mediciones se llevaron a cabo colocando un par de transductores asignados al canal 1 del sensor a la entrada de la tubería, y otro par asignado al canal 2 a la salida de la misma. Ambos canales del sensor se caracterizaron de manera individual, tomando la lectura de flujo desplegada en su pantalla y el voltaje correspondiente por medio del software diseñado en *Matlab* descrito en el siguiente capítulo; para cada tiempo se tomaron mediciones variando la frecuencia de operación del inversor de modo local obteniéndose los datos de la tabla 2.1:

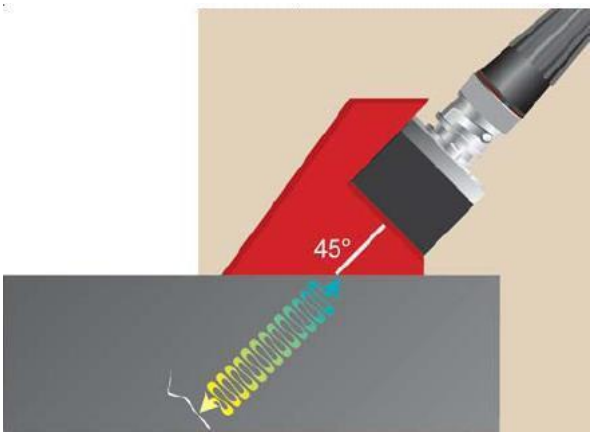

Fig. 2.7 Sensor ultrasónico de flujo

Tabla 2.1. Lecturas de gasto y su correspondiente voltaje para el sensor Panametrics

Frecuencia [hz]	Canal 1			Canal 2		
	Qe [l/s]	Tiempo [s]	Voltaje [V]	Qs [l/s]	Tiempo [s]	Voltaje [V]
45	15.37	50	4.0698	15.38	60	4.0552
	15.36	70	4.0601	15.34	80	4.0332
	15.36	90	4.0552	15.29	100	4.0039
	15.36	125	4.0625	15.22	140	3.9624
	15.38	150	4.0723	15.2	160	3.9478
47.5	16.14	10	4.5759	16.3	20	4.6094
	16.11	30	4.5264	16.28	40	4.6045
	16.14	50	4.5386	16.29	60	4.6167
	16.16	70	4.5563	16.27	80	4.6045
	16.18	90	4.5776	16.29	100	4.6094
50	17.26	15	5.2271	17.04	30	5.0684
	17.28	40	5.2393	17.01	75	5.0586
	17.25	80	5.2271	17	90	5.0562
	17.25	100	5.2271	17.01	110	5.0732
	17.25	120	5.2295	16.99	130	5.0513
52.5	18.37	10	5.9277	17.97	20	5.6396
	18.49	30	5.9595	17.96	40	5.6348
	18.82	50	6.2329	17.99	70	5.6592
	18.79	80	6.1719	18	90	5.6616
55	19.71	10	6.7358	19.98	20	6.2622
	19.73	30	6.7731	18.96	40	6.2524
	19.78	50	6.7798	18.92	60	6.2256
	19.85	70	6.8066	18.9	90	6.2231
	19.82	100	6.8066	18.87	110	6.2012
57.5	20.7	10	7.3462	19.63	20	6.6577
	20.71	30	7.3511	19.69	40	6.6687
	20.69	50	7.3462	19.72	60	6.6992
	20.7	70	7.3511	19.74	80	6.7188
	20.66	90	7.3193	19.74	100	6.7236
60	20.55	20	7.2217	20.62	30	7.2681
	20.82	40	7.2417	20.6	90	7.2461
	20.43	60	7.1802	20.57	70	7.2314
	20.5	80	7.2217	20.55	90	7.2241
	20.52	100	7.2314	20.56	110	7.2388
62.5	21.6	10	7.9004	21.21	20	7.6221
	21.57	30	7.8833	21.23	40	7.6392
	21.5	50	7.8832	21.26	60	7.6538
	21.52	70	7.8516	21.24	80	7.6392
	21.56	90	7.8418	21.24	100	7.6416
65	22.4	10	8.4229	21.88	20	8.0347
	22.44	30	8.4033	21.89	45	8.0396
	22.44	55	8.4155	21.98	70	8.0884
	22.37	80	8.3276	21.93	90	8.0688
	22.34	100	8.3446	21.97	110	8.0933

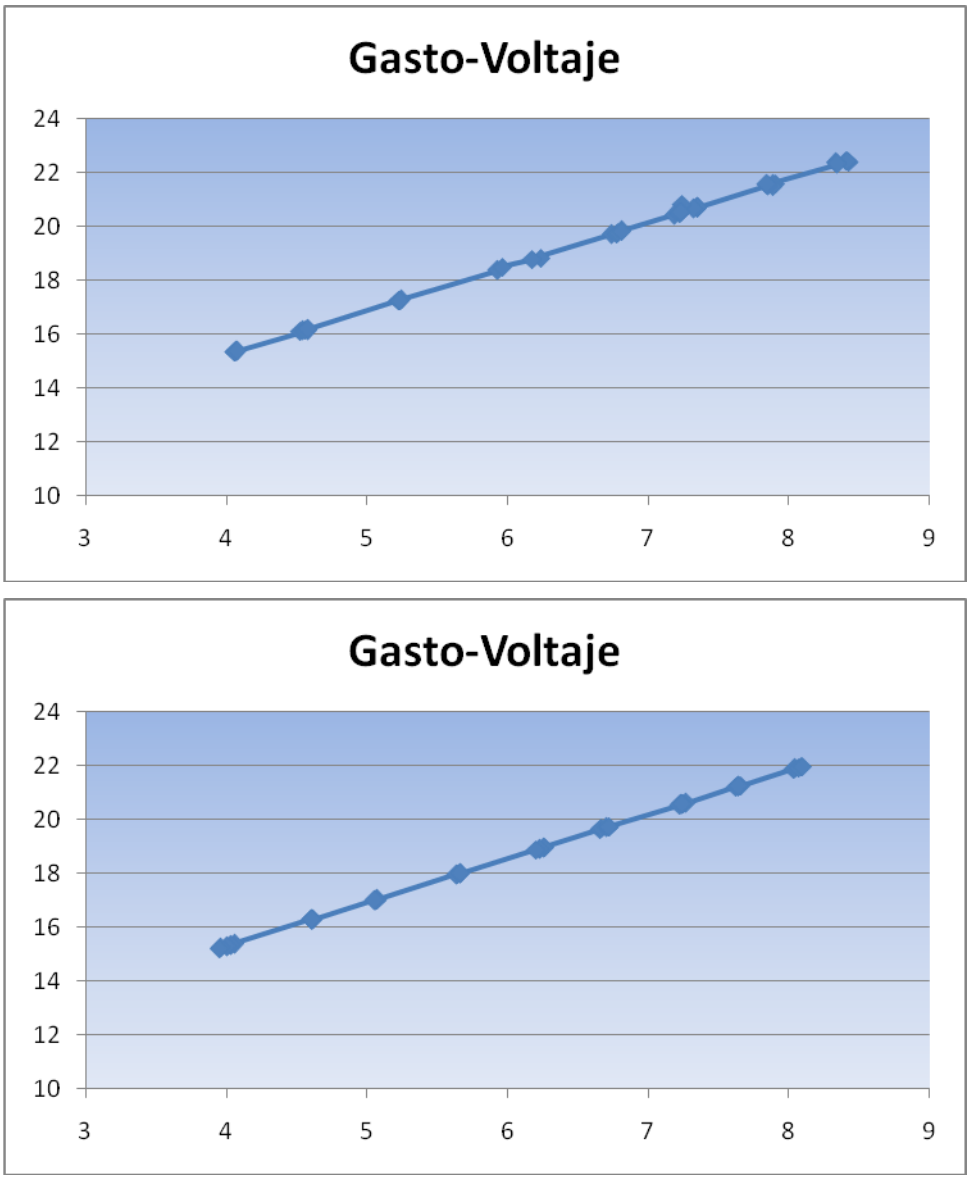

Fig. 2.8 Datos de Sensor Panametrics en canales de entrada y salida

Con estos datos es posible obtener la expresión que relaciona el voltaje recibido del sensor de flujo (eje X) con valores en unidades de gasto volumétrico (eje Y) mostrados en la fig. 2.8:

$$Q_e = 1.6295V + 8.7358 \ \ (l/s) \qquad (2.1)$$

$$Q_s = 1.6326V + 8.7600 \ \ (l/s)$$

II.4.2 Sensores de flujo de propela

Durante varios meses se realizaron las pruebas de detección de fugas en la tubería, tiempo en el cual fue posible detectar deficiencias en la respuesta y comportamiento de los sensores digitales de flujo, entre estas un periodo de refrescamiento muy grande (4.5 seg.) y lenta respuesta a los cambios de lecturas que no es útil al análisis que se pretendía, por lo que se

decidió utilizar los sensores analógicos invasivos de propela que se venían usando anteriormente alimentados a 24 *VDC* con una salida en lazo de corriente de 4 a 20 *mA*.

El principio de funcionamiento de estos sensores es el siguiente: un sistema mecánico rotatorio de propela en un extremo del sensor se sumerge en el ducto y es accionado por el movimiento del fluido, siendo proporcional su velocidad lineal a la velocidad angular de la propela, señal que es transformada por el transductor *+GF+SIGNET* 8512 mostrado en la fig. 2.9 y enviada a la tarjeta de conversión corriente voltaje descrita en la sección II.5.3.

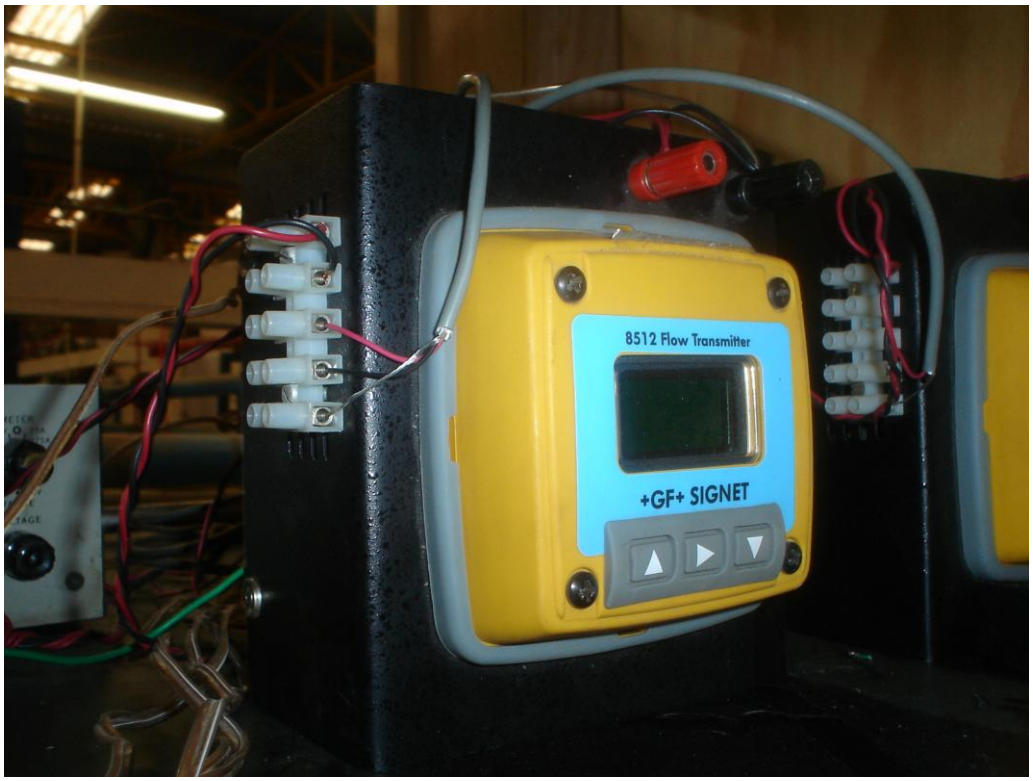

Fig. 2.9 Sensores de flujo analógicos

La configuración de este equipo se realiza de acuerdo al manual de usuario donde se establecen las unidades de despliegue en pantalla, en nuestro caso *l/s,* así como sus valores mínimo y máximo de salida de corriente.

En la fig. 2.10 se muestra la parte exterior del sensor instalado, donde se observan las varillas y turcas usadas para la correcta ubicación de la propela en el centro de la tubería donde es más estable el fluido. Estos sensores presentan un tiempo de respuesta muy corto, de unas cuantas décimas de segundo, lo que les da mayor precisión que a los digitales permitiéndonos detectar variaciones en el fluido.

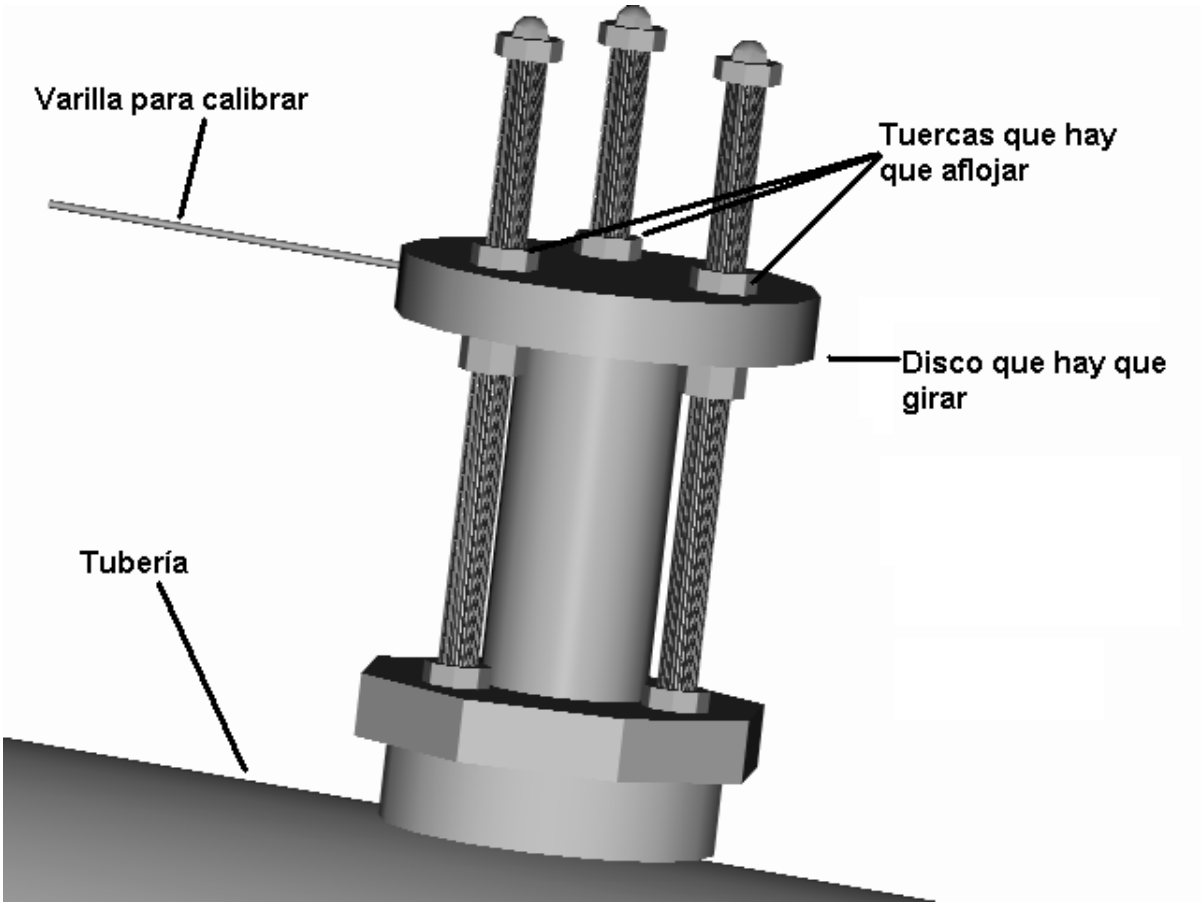

Fig. 2.10 Cuerpo de los sensores de flujo volumétrico de propela

Tabla 2.2 Caracterización de sensores analógicos de flujo volumétrico

Frecuencia [*Hz*]	Sensor 1		Sensor 2	
	Voltaje [*V*]	Gasto [*l/s*]	Voltaje [*V*]	Gasto [*l/s*]
45	6.8921	14.8	6.8091	14.54
	6.8115	14.56	6.8164	14.66
	6.8726	14.62	6.8799	14.76
	6.8701	14.62	6.8164	14.65
	6.9198	14.9	6.7944	14.44
50	7.3682	16.39	7.4243	16.41
	7.3657	16.24	7.4146	16.46
	7.4609	16.52	7.3828	16.27
	7.4219	16.39	7.4414	16.59
	7.3853	16.23	7.3803	16.43
55	8.0908	18.5	7.9272	17.95
	7.9712	18.17	7.9565	18.11
	8.03	18.45	7.959	18.09
	8.0713	18.58	7.937	17.92
	8.0249	18.25	7.8711	17.83
60	8.3521	19.69	8.3032	19.91
	8.3472	19.32	8.2982	19.82
	8.4277	20	8.2813	19.76
	8.374	19.74	8.2642	19.67
	8.3667	19.68	8.2873	19.8

De igual manera se caracterizaron estos sensores de flujo tomando los valores desplegados en su display y su correspondiente en tiempo de conversión de salida en corriente a voltaje, mostrados en la Tabla 2.2:

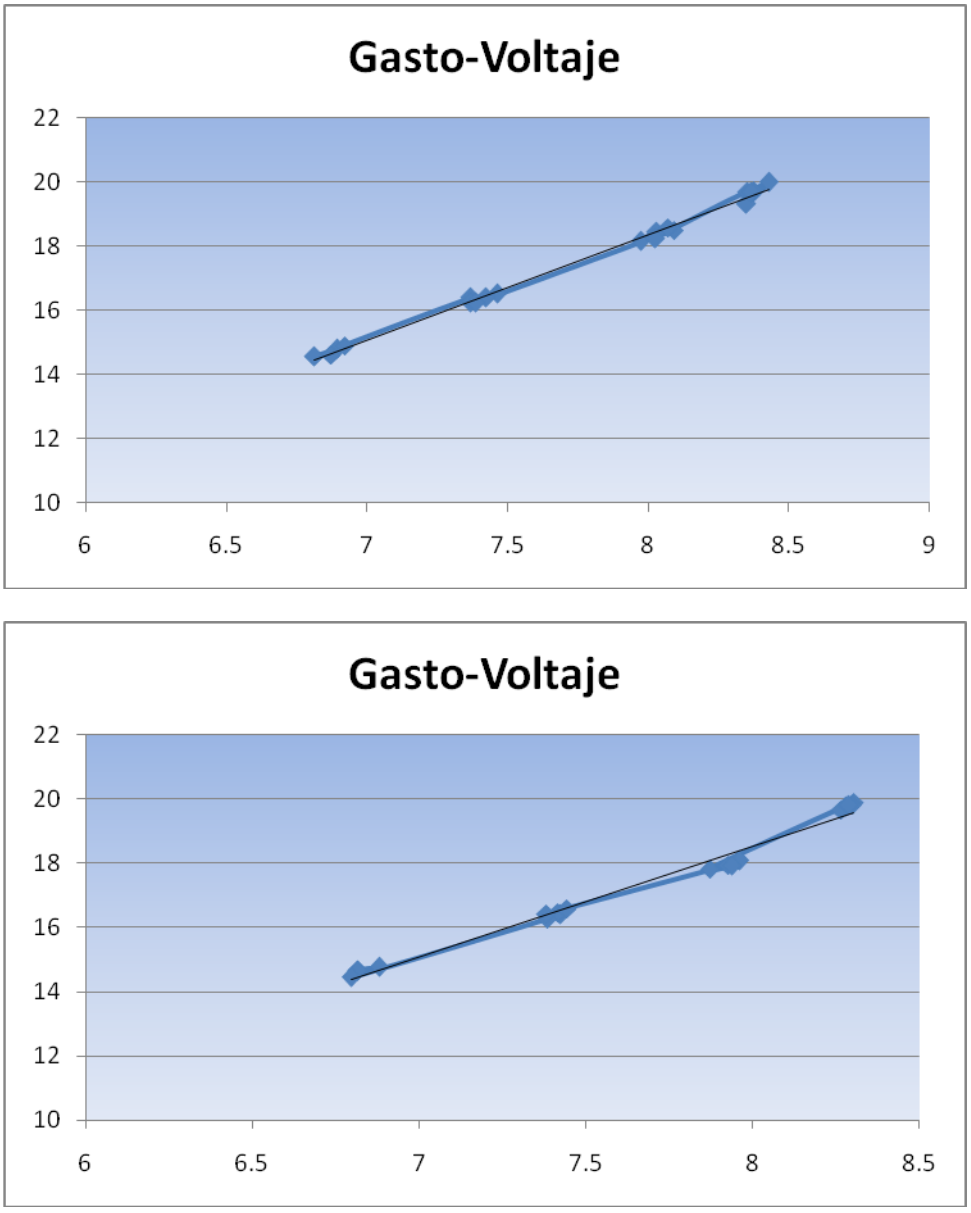

Fig. 2.11 Datos de Sensor de propela en canales de entrada y salida

Con estos datos es posible obtener la expresión que relaciona el voltaje recibido del sensor de flujo (eje X)con valores en unidades de gasto volumétrico (eje Y) mostrados en la fig. 2.11:

$$Q_e = 3.294V - 7.992 \quad (l/s) \tag{2.2}$$

$$Q_s = 3.439V - 8.975 \quad (l/s)$$

II.4.3 Sensores de Presión

La presión manométrica en el fluido es otra variable de interés, que se involucra en el modelo del fluido, siendo de gran importancia su medición. En los sensores electrónicos en general, la presión actúa sobre una membrana elástica, midiéndose la flexión. Los sensores de presión analógicos WIKA piezoeléctricos utilizados fueron de tipo invasivo alimentados a 24 VDC, constan de un transductor protegido contra vibraciones y temperatura, insertado en boquillas al inicio y final de la tubería, encargados de convertir la presión en una señal de corriente continua que es recogida por la interfaz, y después convertida a voltaje para cálculos.

Se utilizó una báscula de pesos muertos para obtener la relación presión-voltaje de los sensores de entrada y salida de la tubería. A continuación se muéstrala Tabla 2.3 con las lecturas obtenidas y el modelo linealizado con unidades *psi*:

Tabla 2.3 Caracterización de sensores de presión

Presión (*psi*)	Voltaje [*V*]	Voltaje [*V*]
5	2.61	2.84
10	3.29	3.5
15	3.97	4.17
20	4.65	4.84
25	5.32	5.51
30	6.06	6.19
35	6.66	6.85
40	7.35	7.51
35	6.68	6.84
30	6	6.18
25	5.32	5.51
20	4.65	4.84
15	3.96	4.17
10	3.3	6.5
5	2.62	2.84

Con estos datos es posible obtener la expresión que relaciona el voltaje recibido del sensor (eje X) con valores en unidades de presión (eje Y) mostrados en la fig. 2.12:

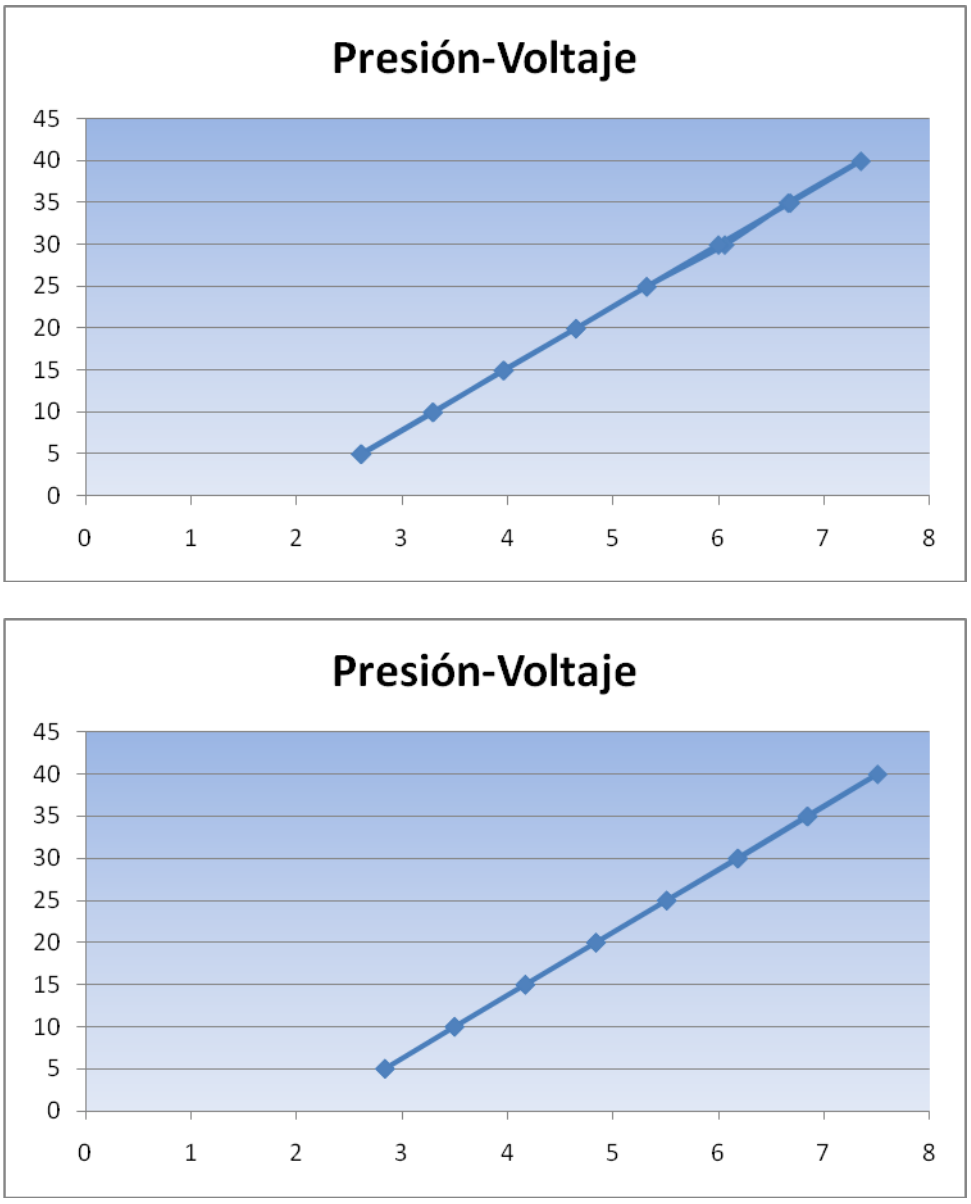

Fig. 2.12 Datos de Sensor de Presión en canales de entrada y salida

$$P_e = 7.3783V - 14.299 \ \ (psi) \qquad (2.3)$$

$$P_s = 7.4803V - 16.213 \ \ (psi)$$

II.5 Interfaz

La interfaz con la que se contó, esta formada por dos tarjetas de adquisición de datos y la circuitería requerida para la conversión de corriente a voltaje.

Estas tarjetas reciben su nombre del estándar *PCI* (Interconexión de Componentes Periféricos), que consiste en un bus de computadora para conectar dispositivos periféricos directamente a su tarjeta madre. Estos dispositivos pueden ser circuitos integrados ajustados

en ésta como la fig. 2.13 o tarjetas de expansión que se ajustan en conectores. Es común en PCs, donde ha desplazado al *ISA* como bus estándar, pero también se emplea en otro tipo de computadoras, permite la configuración dinámica de un dispositivo periférico. En el tiempo de arranque del sistema, las tarjetas *PCI* y el *BIOS* interactúan y negocian los recursos solicitados por la tarjeta *PCI*. Esto permite asignación de direcciones del puerto por medio de un proceso dinámico diferente del bus *ISA*, donde tienen que ser configuradas manualmente usando jumpers externos. Aparte de esto, el bus *PCI* proporciona una descripción detallada de todos los dispositivos de su tipo conectados.

El bajo costo de un sistema basado en estas tarjetas las hacen ideales para trabajo de laboratorio en ambientes industrial y académico. Pueden usarse los canales de salida analógica para generar excitaciones experimentales para controlar máquinas y procesos, y para generar funciones analógicas. Los canales de entrada sirven para registrar señales de voltaje de dispositivos analógicos. Sus líneas digitales *I/O* pueden usarse como interruptores externos para transistores y leer el estado de circuitos digitales.

Fig. 2.13 Tarjeta PCI típica

Las tarjetas de adquisición de datos utilizadas en el proyecto son dispositivos analógicos-digitales para PC con circuitería para salidas de corriente descritas en la siguiente sección. Cuentan con líneas digitales de 16 bits (*Nacional Instruments*, 1996), en conjunción con la computadora, son una plataforma versátil para pruebas de laboratorio, medición y control. Ventajas adicionales son su tamaño pequeño, poco peso y bajo consumo de energía. Su terminal de conexiones se muestra en la fig. 2.14:

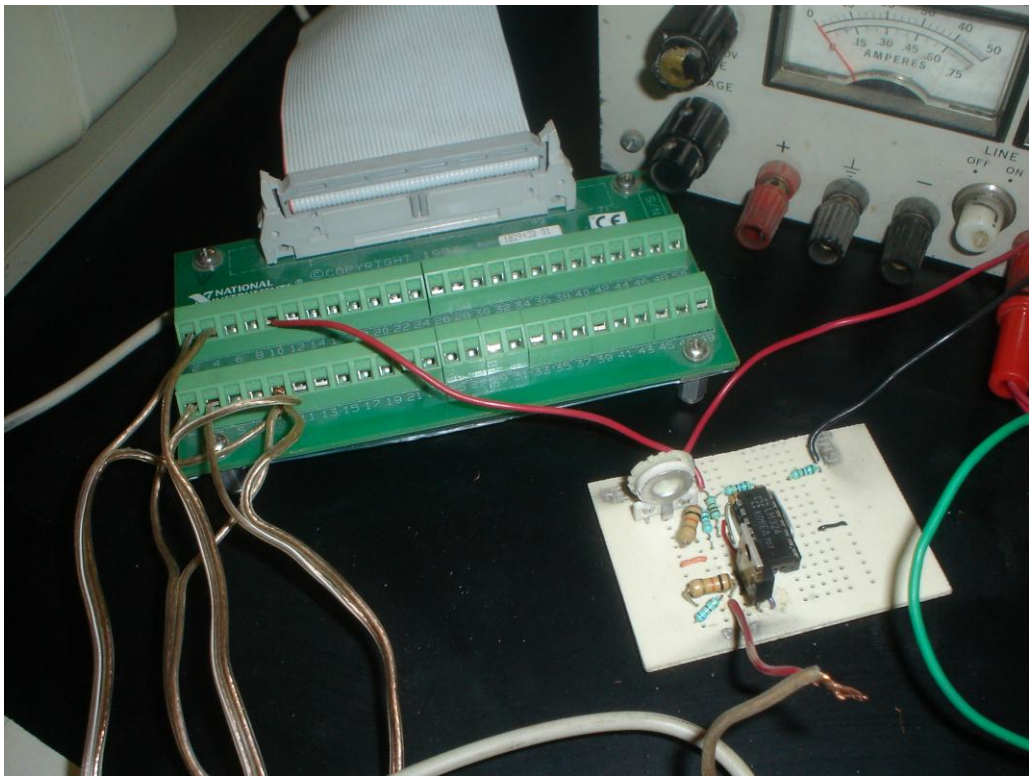

Fig. 2.14 Terminal de Tarjetas DAC

II.5.1 Escritura de datos

La tarjeta *PCA-O2-DC* de *National Instruments* consiste de los siguientes circuitos generales:

- Canales Entrada Salida (E/S)
- Salida Digital
- Salida Analógica

Mediante las salidas analógicas de corriente de la tarjeta se controlaron el inversor de frecuencia y la servo-válvula. Cada canal puede proveer voltaje polar o bipolar configurable por software (*Nacional Instruments*, 1996).

La fig. 215 presenta el diagrama interno para esta salida, se observa que la tarjeta cuenta con salidas analógicas de voltaje y una fuente de corriente por canal, además de E/S digitales, todas controladas desde la unidad *PCI*. La comunicación con el exterior se realiza por medio del canal como bus de datos, a través del cual fluye la información hacia y desde la tarjeta. Debe mencionarse que *Matlab* no cuenta con un controlador disponible para el manejo de esta tarjeta, por lo que fue necesario diseñar uno. El manejo y configuración de la tarjeta se trata en el Capítulo III, mientras el proceso de diseño del controlador en el apéndice A.

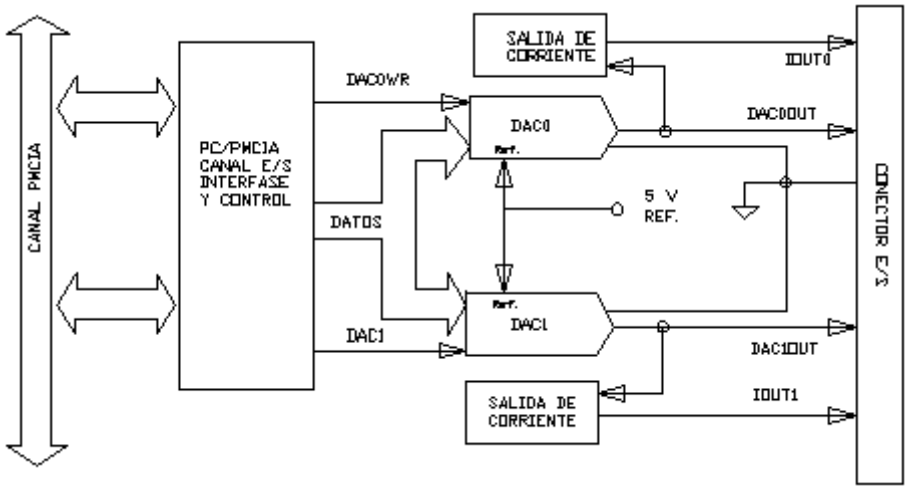

Fig. 2.15 Funcionamiento de Tarjeta *PCAO2DC*

II.5.2 Lectura de datos

El proceso de lectura de datos se realizó usando la tarjeta *Lab-PC1200* de *National Instruments* de alto rendimiento analógico y digital (*Nacional Instruments*, 1996).

La tarjeta consta de los siguientes circuitos principales:

- Temporalización
- Entrada Analógica
- E/S Digital
- Calibración

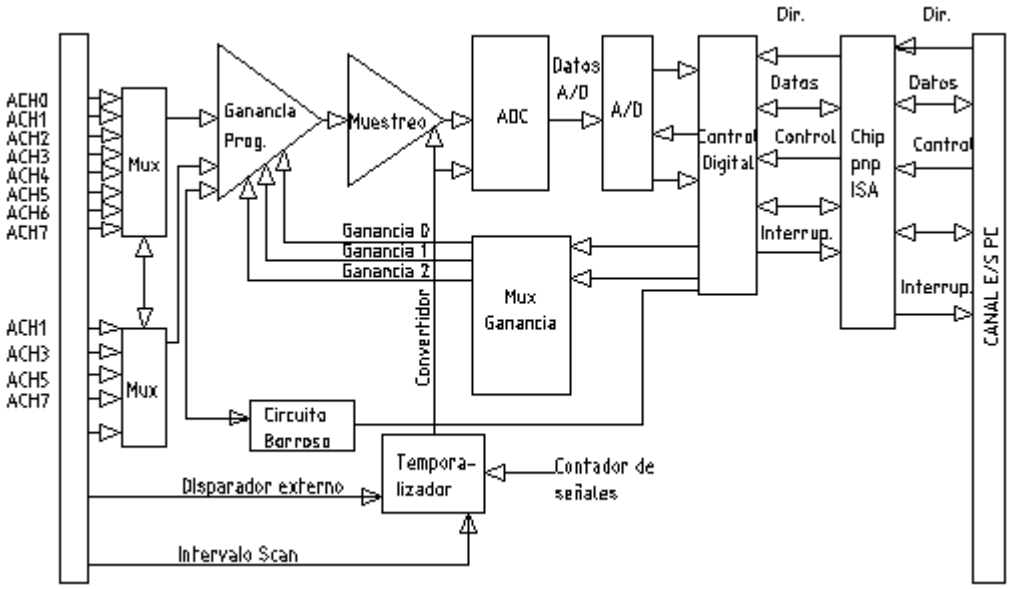

Fig. 2.16 Entrada analógica *Lab PC 1200*

II.6 Conversión corriente- voltaje

Los sensores utilizados en las pruebas experimentales, tienen la característica común de enviar su señal de salida en un lazo de corriente en un rango de 4 a 20 *mA* de acuerdo a su calibración. Por otro lado, la tarjeta de adquisición de datos descrita en la sección anterior, cuenta únicamente con canales de entrada para señales en voltaje, por lo que fue necesario construir un circuito eléctrico diseñado para realizar la conversión de corriente a voltaje utilizando la Ley de Ohm:

$$V = RI \tag{2.4}$$

donde

V = voltaje [V]
R = resistencia [$Ohms$]
I = intensidad de corriente [A]

el circuito construido se muestra en la fig. 2.17, donde se observa del lado derecho los cables con las señales provenientes de los sensores en corriente y salientes del lado izquierdo en voltaje:

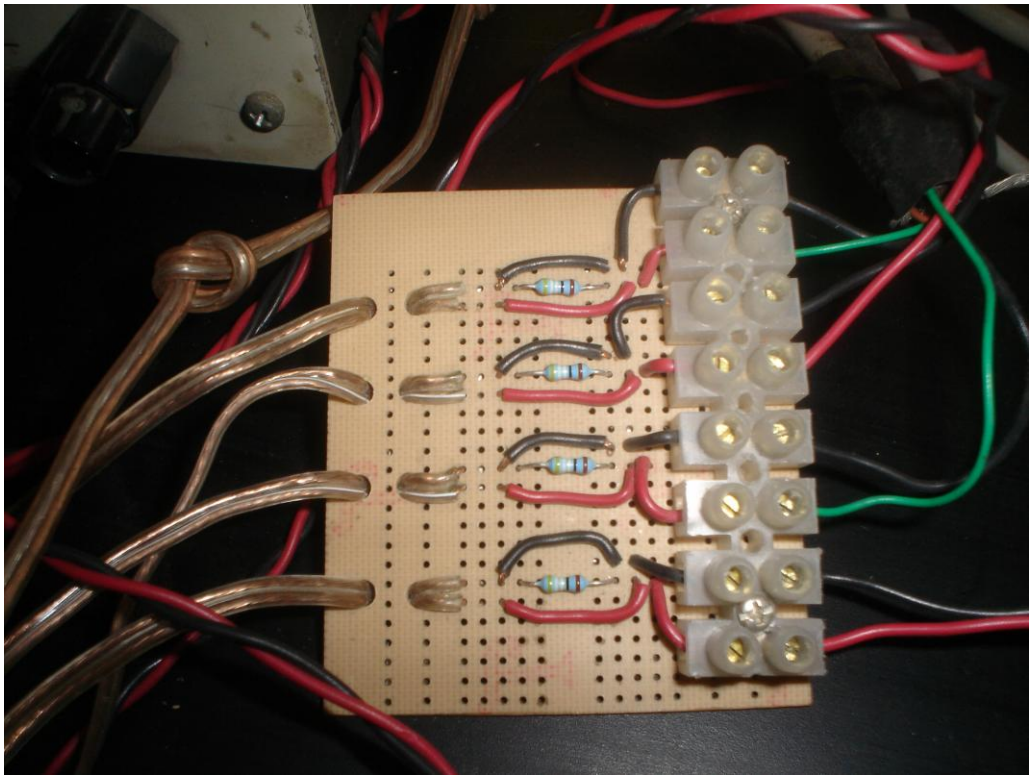

Fig. 2.17 Circuito para conversión corriente-voltaje

de este modo fue posible leer las señales de los sensores en un rango programable por software de 0 a 10 *V* en la tarjeta, utilizando el valor de resistencia de precisión de 498 *Ohms*.

Capítulo III

Diseño del sistema SCADA e Integración

El objetivo principal de este trabajo de tesis y en particular de este capítulo, es el diseño, adaptación y puesta en marcha del Sistema de Adquisición de Datos y Control para el sistema de Tubería del Laboratorio de Ingeniería Hidromecánica del Instituto de Ingeniería de la UNAM. Este sistema fue desarrollado utilizando el software *Matlab*, descrito en la sección III.2.1, además de otros elementos utilizados.

III.1 Enfoque utilizado

La ingeniería en general hoy en día no puede separarse de la Informática. La Ingeniería de Control sigue esta pauta por lo que se auxilia de manera más frecuente de este tipo de herramientas para la instrumentación y monitoreo de procesos.

Como muestra de lo anterior, se encuentra la tecnología SCADA (*Supervisory Control And Data Acquisition*), aplicada durante el desarrollo de este trabajo y que se define a continuación:

> *Un SCADA es un sistema basado en el uso de computadoras que permite supervisar y controlar a distancia una instalación de cualquier tipo. A diferencia de los Sistemas de Control Distribuido, el lazo de control es generalmente cerrado por el operador. Los Sistemas de Control Distribuido se caracterizan por realizar las acciones de control en forma automática. Hoy en día es fácil hallar un sistema SCADA realizando labores de control automático en cualquiera de sus niveles, aunque su labor principal sea de supervisión y control por parte del operador.*

El flujo de la información dentro de la planta y su sistema SCADA es el siguiente: el fenómeno físico lo constituye la variable que se debe medir. Dependiendo del proceso, la naturaleza del

fenómeno es muy diversa, por ejemplo presión, temperatura, flujo, potencia, intensidad de corriente, voltaje, ph, densidad, etc. Este fenómeno debe traducirse a una variable que sea inteligible para el sistema SCADA, es decir, en una variable eléctrica. Para ello, se utilizan sensores o transductores.

Los sensores o transductores convierten las variaciones del fenómeno físico en variaciones proporcionales de una variable eléctrica. Las variables eléctricas más utilizadas son: voltaje, corriente, carga, resistencia o capacitancia.

Sin embargo, esta variedad de tipos de señales eléctricas debe ser procesada para ser comprendida por la computadora. Para ello se utilizan acondicionadores de señal, cuya función es la de referenciar estos cambios eléctricos a una misma escala de corriente o voltaje. Además, provee aislamiento eléctrico y filtraje de la señal con el objeto de proteger el sistema de transitorios y ruidos originados en campo.

Una vez acondicionada la señal, la misma se convierte en un valor digital equivalente en el bloque de conversión de datos. Generalmente, esta función es llevada a cabo por un circuito de conversión analógico/digital. La computadora almacena esta información, la cual es utilizada para su análisis. Simultáneamente, se muestra la información al usuario del sistema, en tiempo real.

Basado en la información, el operador del sistema puede decidir realizar una acción de control sobre el proceso, de manera digital donde la computadora hace la conversión a una señal eléctrica. Esta señal es procesada por una salida de control, la cual funciona como un acondicionador de señal, y la escala para manejar un dispositivo. Para este caso específico, las señales eléctricas accionarán la servo válvula y la bomba, por medio de *Matlab* a través de la tarjeta *PCAO2DC* que hace la función de acondicionador de señal.

III.1.1 Necesidad de un sistema SCADA

Para evaluar si un sistema SCADA es necesario para manejar una instalación dada, el proceso a controlar debe cumplir las siguientes características (Ogata, 2001):

- El número de variables del proceso que se necesita monitorear es alto, haciendo difícil la toma de mediciones por el operario.

- El proceso está geográficamente distribuido. Esta condición no es limitativa, ya que puede instalarse un SCADA para la supervisión y control de un proceso concentrado en una localidad, en nuestro caso el laboratorio de Hidrodinámica del Instituto.
- La información del proceso se necesita en el momento en que los cambios se producen, es decir, la información se requiere en tiempo real.
- La necesidad de optimizar y facilitar las operaciones de la planta, así como la toma de decisiones, tanto gerenciales como operativas.
- Los beneficios obtenidos en el proceso justifican la inversión en un sistema *SCADA*. Estos beneficios pueden reflejarse como aumento de la efectividad de la producción, confiabilidad, seguridad, etc.
- La complejidad y velocidad del proceso permiten que la mayoría de las acciones de control sean iniciadas por un operador. En caso contrario, se requerirá de un Sistema de Control Automático, el cual lo puede constituir un Sistema de Control Distribuido, PLC's, Controladores a Lazo Cerrado o una combinación de ellos (automatas.org).

III.1.2 Funciones del sistema SCADA

Dentro de las funciones de un Sistema SCADA están las siguientes:

- Recabar, almacenar y mostrar información, en forma continua y confiable, correspondiente a la señalización de campo: estados de dispositivos, mediciones, alarmas, etc.
- Ejecutar acciones de control iniciadas por el operador, tales como: abrir o cerrar válvulas, arrancar o parar bombas, etc.
- Alertar al operador de cambios detectados en la planta, tanto aquellos que no se consideren normales (alarmas) como cambios que se produzcan en la operación diaria de la planta (eventos). Estos cambios pueden ser almacenados en el sistema para su posterior análisis.
- Aplicaciones en general, basadas en la información obtenida por el sistema, tales como: reportes, gráficos de tendencia, historia de variables, cálculos, predicciones, detección de fallas, etc.

Todo esto hace a este tipo de sistemas tener ventajas como economía, flexibilidad y fácil operación.

III.2 Plataforma de desarrollo

En la actualidad se cuenta con el apoyo de herramientas computacionales en el diseño e implementación de los Sistemas de Adquisición de Datos y Control. Tal es el caso del software llamado *Matlab* y sus subsistemas auxiliares *Simulink* y *Real Time Windows Target*, cada uno de los cuales es presentado en esta sección.

III.2.1 Introducción a *Matlab*

Matlab es un entorno de computación y desarrollo de aplicaciones totalmente integrado orientado para llevar a cabo proyectos en donde se encuentren implicados elevados cálculos matemáticos y la visualización gráfica de los mismos. *Matlab* integra análisis numérico, cálculo matricial, proceso de señal y visualización gráfica en un entorno completo donde los problemas y sus soluciones son expresados del mismo modo en que se escribirían, sin necesidad de hacer uso de la programación tradicional (Matlab, Math Works).

Este software dispone también en la actualidad de un amplio abanico de programas de apoyo especializado, denominado *Toolboxes*, que extienden significativamente el número de funciones incorporadas en el programa principal. Estos *Toolboxes* cubren en la actualidad prácticamente casi todas las áreas principales en el mundo de la ingeniería y la simulación, destacando entre ellos el *toolbox* de procesamiento de imágenes y de señales, control robusto, estadística, análisis financiero, matemáticas simbólicas, análisis de redes neuronales, lógica borrosa, identificación de sistemas, simulación de sistemas dinámicos, etc.

Matlab nace como una solución a la necesidad de contar con mejores herramientas de cálculo para resolver problemas complejos en el ámbito académico en los que es necesario aprovechar las amplias capacidades de procesamiento de datos de las más grandes computadoras.

El nombre *Matlab* viene de "matrix laboratory" (laboratorio matricial), fue originalmente escrito para proveer acceso fácil al software matricial desarrollado por los proyectos *LINPACK* y *EISPACK*. Hoy *Matlab* es usado en una variedad de áreas de aplicación incluyendo procesamiento de señales e imágenes, diseño de sistemas de control, ingeniería financiera e investigación médica.

Integra la computación matemática con funciones de visualización y un lenguaje de gran eficacia a fin de ofrecer un entorno flexible para la realización de computación técnica. La arquitectura abierta facilita el uso de este software y de los productos acompañantes para

explorar datos, crear algoritmos y herramientas a medida que permitan obtener de manera rápida la información.

Los usuarios han encontrado en la combinación de la interfaz intuitiva, el lenguaje y las funciones gráficas y matemáticas incorporadas de *Matlab,* la plataforma ideal para el cálculo técnico, sobre todo en comparación con *C*, *Fortran* y otros lenguajes y aplicaciones.

Matlab dispone de herramientas para:

- Adquisición de datos
- Exploración y análisis de datos
- Visualización y procesamiento de imágenes
- Construcción de prototipos y desarrollo de algoritmos
- Modelado y simulación
- Programación y desarrollo de aplicaciones

Características principales

- Computación numérica para obtener rápidamente resultados precisos
- Gráficos para visualizar y analizar los datos
- Lenguaje y entorno de programación interactivos
- Herramientas para construir GUI (Interfases Gráficas para Usuario) a la medida
- Integración con aplicaciones externas como *C*, C++, Fortran, Java, componentes COM o Excel
- Posibilidad de importar datos desde archivos y dispositivos externos y de usar archivos E/S de bajo nivel (además de acceso a bases de datos y hardware adicional a través de productos añadidos)
- Conversión de aplicaciones *Matlab* a *C* y *C++* mediante *Compiler*

Este amplio conjunto de prestaciones hacen de *Matlab* la base principal ideal para el desarrollo de soluciones a los problemas técnicos.

III.2.2 Introducción a *Simulink*

Simulink es una herramienta interactiva para el modelado, simulación y análisis de sistemas dinámicos multidominio. Permite crear diagramas de bloques, simular el comportamiento del sistema, evaluar su rendimiento y ajustar el diseño según convenga. *Simulink* se integra a la

perfección con *Matlab*, lo cual proporciona acceso inmediato a un amplio elenco de herramientas de diseño y análisis. Todas estas ventajas hacen de *Simulink* la herramienta idónea para el diseño de sistemas de control, diseño de sistemas de comunicación y otras aplicaciones de simulación (Simulink, *Using Simulink*. Math Works).

Creación del modelo

- Amplia biblioteca de bloques predefinidos
- Posibilidad de agrupar los modelos en jerarquías para crear una vista simplificada de los componentes o los subsistemas
- Model Browser para la navegación por las jerarquías de los modelos
- Library Browser para una selección adecuada de los bloques
- Herramienta Finder para la búsqueda de modelos y bibliotecas
- Conexión y enrutación automáticas para simplificar la edición
- Bloques personalizables para permitir la incorporación de código *Matlab*, *C, Fortran* y *Ada* ya existente
- Sistemas lineales, no lineales, de tiempo continuo, de tiempo discreto, multitasa, ejecutados condicionalmente e híbridos
- Soporte para operaciones con señales de matrices, señales basadas en marcos y complejas
- Tipos de datos soportados: de 32 y 64 bits de punto flotante, de punto fijo (requieren Fixed-Point Blockset) y enteros
- Model Discretizer, que convierte de forma sencilla diseños continuos en equivalentes
- *Simulink Data Explorer GUI* para la visualización y edición de los datos

Ejecución del modelo

- Depurador gráfico para buscar y corregir los errores de los diseño
- Posibilidad de ejecutar las simulaciones desde la línea de comandos de *Matlab*, ya sea de forma interactiva o en modo batch.
- Visualizador de diagnóstico para facilitar la identificación y corrección precisas de los errores de modelado

- Bloques de verificación de modelos para comprobar los elementos del diseño durante su ejecución
- Compatibilidad intrínseca con punto fijo para permitir cambiar con facilidad entre representaciones de punto fijo y punto flotante.

Herramientas de creación de modelos

Simulink brinda una colección completa de herramientas de modelado que permiten desarrollar con rapidez diagramas de bloques detallados de los sistemas. Las características tales como las bibliotecas de bloques, el modelado jerárquico, el etiquetado de señales y la personalización de los subsistemas conforman una colección de prestaciones para crear, modificar y mantener diagramas de bloques. Gracias a estas características de modelado y a la completa colección de bloques predefinidos que incluye *Simulink*, la creación de representaciones precisas de un sistema resulta de lo más sencillo, independientemente de la complejidad que revista.

Por otra parte, *Simulink* le brinda funciones para crear bloques y bibliotecas de bloques personalizados. Así, podemos personalizar no sólo la funcionalidad de un bloque, sino también su interfaz de usuario mediante iconos y cuadros de diálogo. Además, los cambios que efectúe en la versión de biblioteca de un bloque se propagarán a todos los modelos que empleen dicho bloque.

Integración de código personalizado

Gracias a las funciones *S-Functions* (funciones del sistema) el usuario puede incorporar código nuevo o existente (*Matlab*, *C*, *Fortran* o *Ada*) en sus diagramas de bloques de *Simulink*. Las funciones *S-Function* son módulos de código propio que definen el comportamiento de un bloque de *Simulink*. Se cuenta con *S-Function Builder* para funciones del sistema de código *C*, una herramienta con la que se puedan crear funciones de sistema propias e incluirlas en los modelos que vaya a desarrollar (Simulink, *Writing* S-*Functions*. Math Works).

Especificación de tipos de datos

Simulink admite señales complejas y de tipos de datos. Muchos de los bloques de *Simulink* admiten varios tipos de datos. La posibilidad de especificar los tipos de datos de la señal de un modelo y los parámetros del bloque resulta especialmente útil para aplicaciones en tiempo real

tales como microcontroladores. Gracias a esta capacidad se podrá especificar los tipos de datos óptimos para la representación de las señales, los parámetros de los bloques y las funciones matemáticas, exactamente como se representan en estos dispositivos. Además, la selección de los tipos de datos apropiados para las señales y los parámetros del modelo contribuye a mejorar considerablemente el rendimiento y a reducir los requisitos de memoria del código generado por el modelo. Los tipos de datos compatibles actualmente son de punto flotante de doble precisión, punto flotante de simple precisión, enteros de 8, 16 y 32 bits con y sin signo, y booleanos.

III.2.3 Real Time Windows Target

Real Time Windows Target es una solución para la prueba de prototipos y de sistemas en tiempo real. Es un ambiente donde se utiliza una computadora como objetivo "stand alone" y alojado en sí mismo para ejecutar modelos *Simulink* de manera interactiva y en tiempo real. *Real-Time Windows Target* soporta E/S directas, proporcionando interacción en tiempo real con su elaboración del modelo, para la construcción de prototipos rápida y simulación Hardware-in-the-loop (Real Time Windows Target, *User´s Guide*. Math Works).

Después de crear un modelo y simularlo con *Simulink* en modo normal, puede generarse un código ejecutable con el *Real-Time Workshop* y un compilador de *C*. Entonces es posible correr las aplicaciones en tiempo real en el modo externo de Simulink.

La integración entre el modo externo de *Simulink* y el *Real Time Windows Target* permite utilizar a *Simulink* como una interfaz gráfica; en nuestro caso, para el control en tiempo real y la instrumentación de sistemas de control. Para esto, es necesario instalar en *Matlab* lo que se denomina *Real Time Kernel* para asegurar que las aplicaciones corran en tiempo real, esto se hace con la instrucción en *Matlab*

rtwintgt –setup

El *kernel* es la interfaz y la comunicación con el hardware de entrada/salida(I/O) usando bloques controladores I/O .

El modo externo de Simulink es la comunicación entre *Simulink* y la aplicación en tiempo real. Este módulo trata directamente con el *kernel* y es usado para empezar la aplicación en tiempo real, cambiar parámetros y recuperar datos de un bloque *Scope* (osciloscopio) donde se visualizan los datos adquiridos como el mostrado en la fig. 3.1:

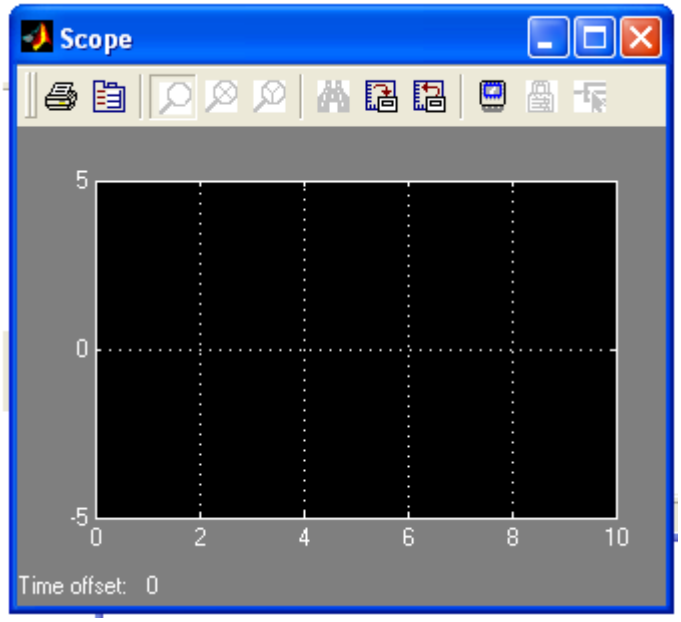

Fig. 3.1 Bloque *Scope* de *Simulink*

La aplicación en tiempo real corre en la PC y tiene las siguientes características:

- Compilación de código. Creado desde el compilador de *C*, en nuestro caso particular con auxilio del compilador *Watcom C/C++*.
- Relación con el modelo en *Simulink*. El ejecutable contiene una forma binaria de todos los componentes del modelo, conexiones entre los bloques, las dependencias del tiempo y variables en los bloques.
- Relación con el *kernel*. El ejecutable debe ser cargado y ejecutado directamente por el *Real Time Windows Target kernel*. No puede ser ejecutado sin el *kernel*.

Si se hacen cambios significativos al modelo de Simulink deberá reconstruirse el ejecutable antes de intentar conectarlo.

Para la utilización del *Real Time*, es necesario contar con el hardware necesario, una PC o procesador así como dispositivos de entrada/salida de apoyo.

Real Time utiliza una gran variedad de Tarjetas I/O para PC. Cuando corren los modelos en tiempo real, el *Real Time Windows Target* captura los datos muestreados de uno o más canales de las tarjetas, usando los datos como entradas a los bloques del modelo, inmediatamente procesa el dato, mandándolo de regreso hacia afuera a través de un canal de salida de la tarjeta I/O.

El *Real Time* soporta una gran variedad de tarjetas I/O, incluyendo tarjetas *ISA*, *PCI* y *PCMCIA*. Esto incluye conversiones analógico-digital (A/D), digital-analógico (D/A), entradas y salidas digitales y entradas de encoders.

III.3 Sistema de adquisición de datos y control

El sistema *SCADA* con el que se trabajó se presenta en esta sección, consiste en una interfaz gráfica diseñada en *Simulink*, que facilita la tarea de monitoreo del operario al presentarse en un ambiente gráfico.

Las señales pueden ser adquiridas, desplegadas y salvadas usando el bloque de *Simulink Scope* y el modo externo de *Matlab*. Esto permite observar el comportamiento del modelo durante la simulación de la aplicación en tiempo real.

El bajo costo de un sistema basado en estas tarjetas las hacen ideales para trabajo de laboratorio en ambientes industrial y académico. Pueden usarse los canales de salida analógica para generar excitaciones experimentales para controlar máquinas y procesos, y para generar funciones analógicas. Sus líneas digitales I/O pueden usarse como interruptores externos para transistores y leer el estado de circuitos digitales.

Este sistema cuenta con dos subsistemas principales, ambos diseñados como parte del proyecto para funcionar en tiempo real y descrito en la siguiente sección:

- Adquisición de datos
- Instrucciones de mando

Como parte del diseño de modelos en Simulink, es necesario establecer ciertos parámetros antes de iniciar operaciones como se muestra en la fig. 3.2, entre los que destacan:

- El periodo de muestreo
- El método numérico con el que se realizarán las operaciones (diferencial, integral).
- El tiempo que durará la prueba
- El target, necesario para la comunicación entre *Matlab* y *Real Time*.

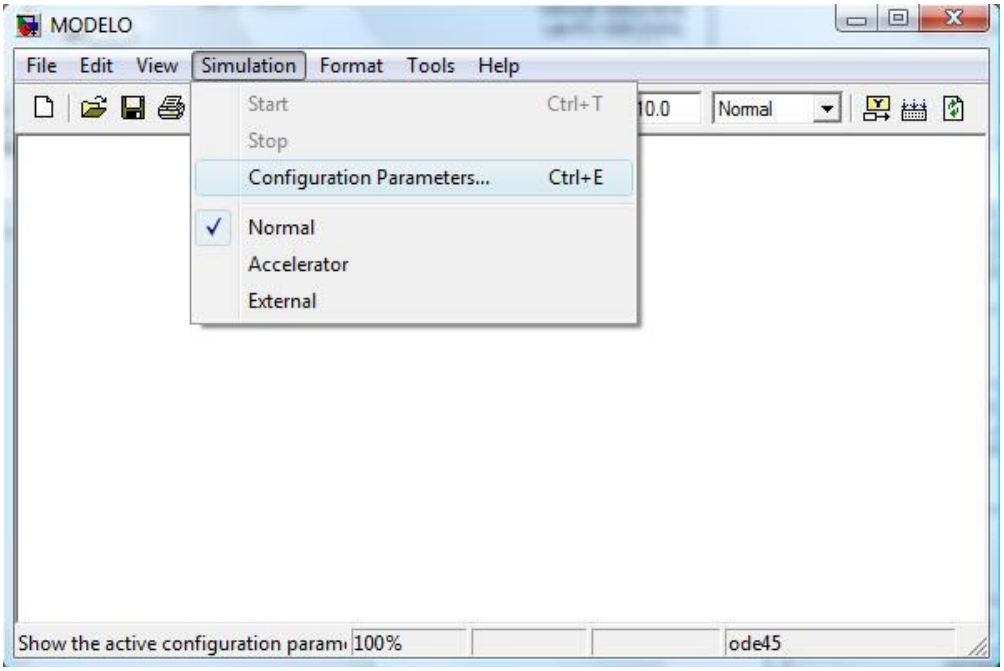

Fig. 3.2 Configuración de parámetros de operación

III.3.1 Sistema de adquisición de datos

La adquisición de datos se realiza utilizando la Tarjeta *Lab-PC 1200* descrita en el Capítulo II, utilizando 5 de sus 8 canales analógicos para la lectura de las señales de los sensores de gasto y presión en los extremos de la tubería (National Instruments, 1996).

Matlab cuenta con el controlador para esta tarjeta, lo que permite comunicarse directamente con ella, sin necesidad de ninguna interfaz desde el ambiente de este software mediante su bloque correspondiente, donde se introducen parámetros tales como: canales activos, tipo de entradas y salidas (rango y polaridad) y frecuencia de muestreo que deberá ser igual al periodo del modelo en *Simulink*.

La asignación de canales de entrada fue la siguiente, evitando el uso de canales continuos por problema de ruido presentados entre los canales por la tierra común:

- Canal 1 Flujo de Entrada
- Canal 3 Flujo de Salida
- Canal 5 Presión de Entrada
- Canal 7 Presión de Salida

La fig. 3.3 muestra la parte del modelo correspondiente a la lectura de datos del sistema, a la izquierda se tiene el bloque que representa a la Tarjeta *Lab-PC 1200* adquiriendo señales en 5 canales, las que se separan con un demultiplexor para operar cada una de las señales por separado con factores correspondientes a la caracterización de los sensores usados para obtener las lecturas en unidades de nuestro interés aplicando las ecuaciones (2.1), (2.2) y (2.3).

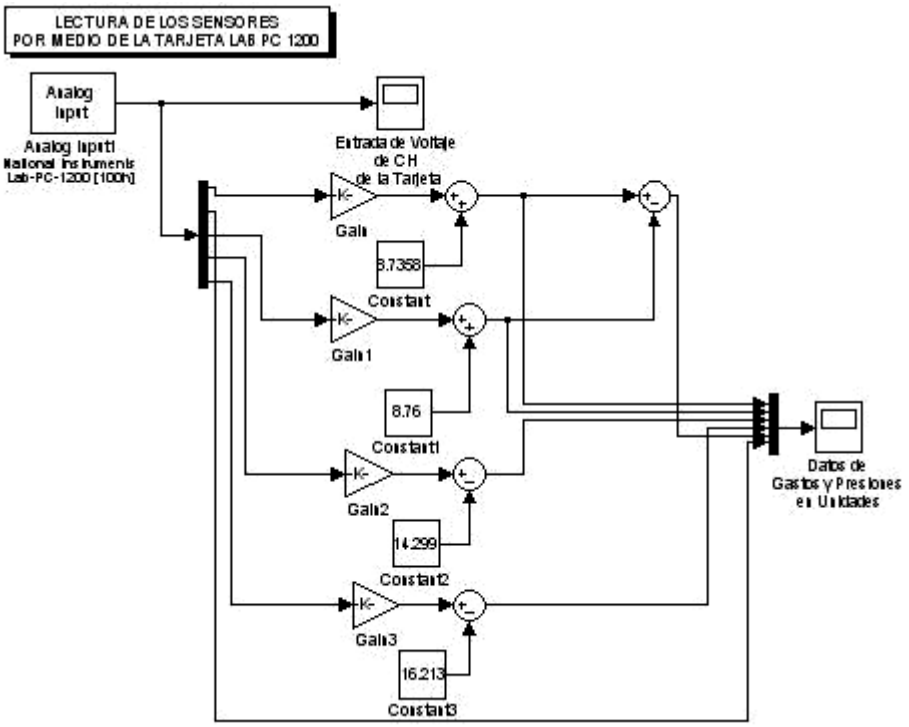

Fig. 3.3 Sistema de Adquisición de datos en *Simulink*

En la fig. 3.4 se muestra el bloque de configuración para la entrada analógica a través de la tarjeta que se desee instalar, así como alta de canales a usar y rango de los valores de entrada en Volts.

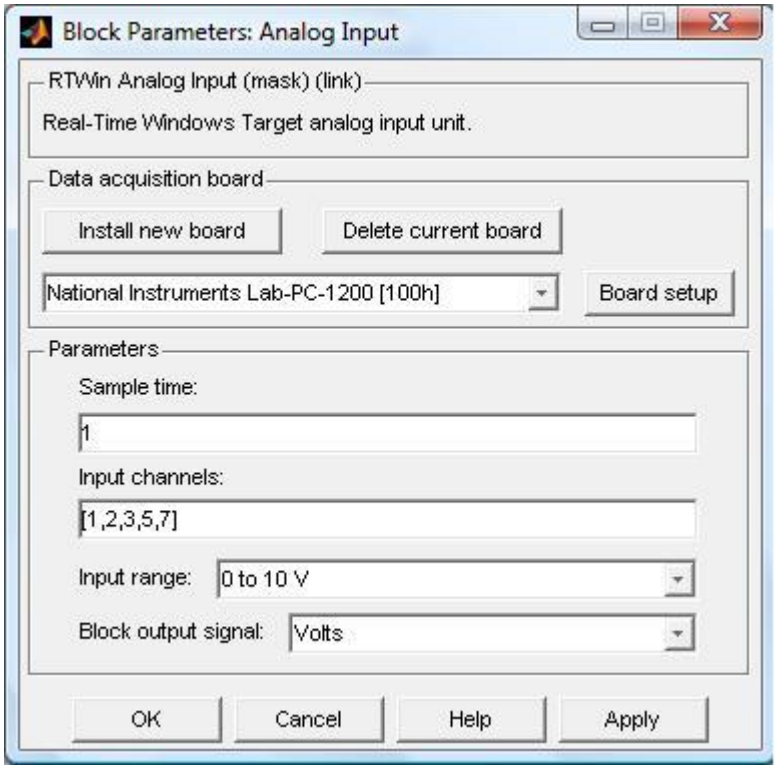

Fig.3.4 Bloque de configuración de la entrada analógica para Tarjeta *Lab-PC-1200*

Posteriormente se envían las señales a través de un multiplexor al bloque *Scope* para su observación mientras se realiza el experimento y posterior almacenamiento de los datos en *Matlab* como archivo .mat. Para esto es necesario configurar el *Scope* con los valores para los dos ejes cartesianos que presenta: en las abscisas se tiene el tiempo que durará el experimento, el cual debe ser igual al tiempo introducido en *Simulink*; las ordenadas son el valor de la variable leída por los sensores, aquí es necesario identificar cada tipo de señal, pues se presentan 5 curvas, algunas de ellas en diferentes unidades.

III.3.2 Instrucciones de accionamiento

El sistema de accionamiento de los dispositivos se instrumentó usando la tarjeta *PC-AO-2DC* descrita en el Capítulo II como salida analógica de corriente por sus 2 canales. El sistema se encarga de la operación de los siguientes dispositivos:

- Bomba Eléctrica a través de un inversor digital que convierte la señal de la tarjeta a una frecuencia de operación para generar el flujo del líquido por el ducto.

- Servo Válvula para generar con precisión en tiempo y tamaño de las fugas. Su rango de apertura corresponde al rango de su entrada, 4 a 20 *mA*, siendo 4 *mA* cierre y 20 *mA* apertura total.

La señal de salida utilizada de la tarjeta fue de corriente en un rango de 4 a 20 *mA*, aunque también cuanta con salida en voltaje de 0 a 10 *V*.

El modelo en *Simulink* para el control de los dispositivos es el siguiente:

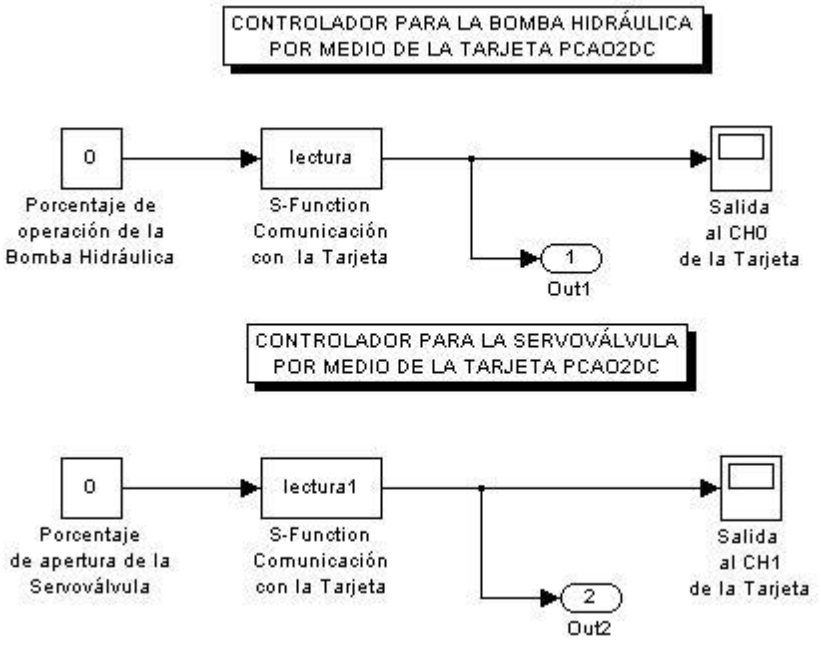

Fig. 3.2 Sistema de Control en *Simulink*

Los modelos "*Lectura*" y "*Lectura 1*" de la fig. 3.2 corresponden al controlador de la tarjeta *PC-AO-2DC* en *Matlab*. Puede verse que se requieren unas entradas para el bloque del controlador, siendo la operación de los dispositivos en proporción al porcentaje de la corriente máxima de la tarjeta; en los bloques *Scope* puede verse el valor de la señal de salida. Estas entradas están en el rango de 0 a 100% del valor nominal de la salida.

Cabe mencionar en esta sección, que la tarjeta *PC-AO-2DC* no se encuentra soportada de origen por *Matlab*, por lo fue necesario diseñar el controlador utilizado en el modelo para las funciones de control, proceso paralelo al desarrollo de esta tesis y que se describe en el Apéndice A.

III.4 Observaciones

Se hacen notar algunas observaciones hechas durante el proceso de prueba y diseño del sistema en cuanto a la detección de algunas limitaciones de *Matlab* para realizar experimentos, tales como la imposibilidad de trabajar con algunas funciones que realizan interrupciones con el sistema operativo en tiempo real para almacenamiento de datos (*To File, To Workspace*) y que obligó a salvar dichos datos de manera indirecta en archivos .mat, además de no tener una satisfactoria visualización de los datos en el bloque S*cope* por no guardar datos en memoria RAM, esto sumado a problemas que aparecieron sin hasta ahora tener explicación mas que ser errores internos de *Matlab*.

Capítulo IV

Algoritmo de detección de fugas en ductos

Los algoritmos de diagnóstico que utilizan redundancia analítica constan de dos bloques: generación de residuos y evaluación de residuos, como se muestra en la fig. 4.1, donde u(t) y y(t) representan la medición (entradas y salidas) y r(t) los residuos (Gertler, 1991). A su vez, el problema de detección se puede dividir en dos tareas: a) la generación de subsistemas desacoplados tanto de incertidumbres como de fallas consideradas como perturbaciones usando una inyección de la salida y, estrictamente hablando, b) la tarea de diseño del generador de residuos.

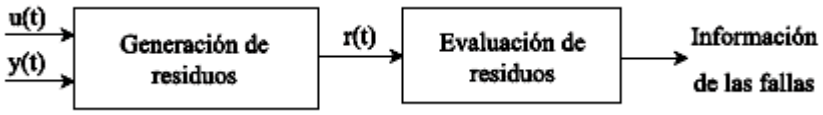

Fig. 4.1 Algoritmo de diagnóstico

Los residuos son variables que representan la inconsistencia entre las variables de la planta real y las variables del modelo matemático. Los residuos se calculan a partir de las observaciones de la planta que constan de las salidas y las entradas medidas. El enfoque de redundancia analítica requiere que el generador de residuos cumpla con cierta clase de validación de relaciones del sistema usando la entrada real y salida real. Los residuos resultantes se usan para formar funciones de decisión como normas o funciones de probabilidad. Luego se evalúan en la lógica de decisión de fallas para monitorear el tiempo de ocurrencia y la localización de la falla.

Muchos de los enfoques desarrollados para diseñar el generador de residuos, ya sea con herramientas algebraicas o geométricas, formulan el problema de FDI considerando un residuo dependiente solamente a una falla e independiente al resto de ellas y a las incertidumbres, haciendo trivial el problema de aislamiento.

El punto crucial en cualquier esquema FDI basado en el modelo matemático es la influencia de las perturbaciones no modeladas como, por ejemplo, parámetros ambiguos, cambios en los parámetros del sistema, y ruidos del sistema y de medición. Todas estas influencias se pueden resumir como entradas desconocidas actuando sobre el sistema. Debido a esto, existe siempre un error entre el proceso actual y su modelo matemático aún cuando no ocurran fallas en el proceso. El efecto de las entradas desconocidas degrada el desempeño de FDI y actúa como una fuente de falsas alarmas.

De acuerdo con lo anterior, se tienen dos requerimientos de importancia para los sistemas FDI:

a) Distinguir fallas diferentes.

b) Robustez con respecto a las entradas desconocidas (incluyendo los errores del modelo).

IV.1 Desacoplamiento de perturbaciones

Una herramienta útil para diagnóstico de fallas es el concepto de desacoplamiento de perturbaciones; aquí los efectos de las perturbaciones y algunas fallas sobre los residuos se desacoplan a través de un diseño elaborado de un generador de residuos; de modo que el diagnóstico se pueda lograr. Una forma es haciendo uso de una transformación de estado.

Considere el modelo

$$\dot{x} = g(x,u) + K(x)d + E(x)f \tag{4.1}$$

$$y = h(x) \tag{4.2}$$

donde x es el vector de estado, u el vector de entrada, y el vector de salida, d el vector de perturbaciones o entradas desconocidas y f el vector de fallas, $K(x)$ y $E(x)$ son matrices conocidas.

La transformación

$$z = T(x) \tag{4.3}$$

llamada transformación de desacoplamiento a perturbaciones y sensible a fallas, genera un subsistema con inyección de la salida de la forma

$$\dot{z} = \bar{g}(z,u,y) + \bar{E}(x)f \tag{4.4}$$

$$y_z = \bar{h}(z) \tag{4.5}$$

sí y sólo sí se cumplen la condición de desacoplo y la condición de sensibilidad

$$\frac{\partial T(x)}{\partial x}K(x)=0 \quad \text{y} \quad rank\left(\frac{\partial T(x)}{\partial x}E(x)\right)=rank\left(E(x)\right) \tag{4.6}$$

De esta forma, el sistema en coordenadas z está totalmente desacoplado de las perturbaciones sin perder el efecto de las fallas. Este resultado de desacoplamiento a perturbaciones se puede utilizar para resolver tanto el problema de detección de fallas como el de aislamiento de fallas. El aislamiento de fallas es posible cuando se consideran fallas modeladas como perturbaciones. De este modo, el aislamiento de fallas requiere del desacoplo de los efectos de diferentes fallas sobre el residuo lo que permite decidir qué falla o fallas han ocurrido.

Otro de los resultados importantes en la teoría de FDI está relacionado con las condiciones necesarias para detectar un conjunto de fallas usando un generador de residuos. Se sabe que con q salidas sólo es posible a lo más tener $q-1$ conjuntos de fallas que pueden ser aislados (Massoumnia et al., 1989). Si se definen conjuntos de fallas con un solo elemento, el problema de detección y aislamiento global podrá tener solución si el número de fallas es menor o igual al número de salidas. Esta consideración simplifica la tarea de FDI ya que cada residuo se puede diseñar sensible solamente a una falla y robusto al resto y así el problema de aislamiento resulta trivial.

IV.2 Modelo del fluido

Asumiendo que los cambios de velocidad por turbulencia y la compresibilidad son despreciables y la densidad del fluido y el área de la sección transversal del ducto son constantes, las ecuaciones diferenciales parciales de la dinámica y continuidad del flujo están dadas por (Chaudry, 1979)

$$\frac{\partial Q}{\partial t}+gA\frac{\partial H}{\partial z}+\frac{f}{2DA}Q|Q|=0 \tag{4.7}$$

$$b^2\frac{\partial Q}{\partial z}+gA\frac{\partial H}{\partial t}=0 \tag{4.8}$$

En las dos ecuaciones anteriores hay dos variables independientes, t coordenada de tiempo (s) y z coordenada de longitud (m) y dos variables dependientes, Q el gasto (m^3/s) y H la presión (m). La velocidad de onda b se considera constante (m/s), f es el coeficiente de fricción adimensional, A es el área de la sección transversal del ducto (m^2), D es el diámetro del ducto (m) y g la gravedad (m/s^2).

Se puede describir una fuga (orificio) en un punto z_f como (Thomas, 1999)

$$Q|_{z_f} = \lambda\sqrt{H|_{z_f}} \tag{4.9}$$

con $\lambda > 0$. La presencia de una fuga produce una discontinuidad en las ecuaciones (4.6) y (4.7), por lo cual el ducto con una fuga se ve como dos ductos o secciones con una condición de frontera entre

$$Q^a|_{z_f} = Q^d|_{z_f} + Q|_{z_f} \tag{4.10}$$

donde Q^a *y* Q^d son los flujos en las secciones antes y después de la fuga, respectivamente. De esta forma, si se asumen $n - 1$ fugas, el comportamiento del fluido estaría descrito por n pares de ecuaciones de la forma (4.6) y (4.7) con $n - 1$ condiciones de frontera.

Asumiendo un ducto de longitud L y fugas distribuidas uniformemente a lo largo del ducto, el espacio z se puede dividir en n secciones de tamaño $\Delta z = L/n$.

Considerando condiciones de frontera conocidas en los extremos del ducto caracterizadas por la presión en el inicio y final del ducto H_e y H_s y el tamaño de sección Δz, las ecuaciones diferenciales parciales con respecto a la coordenada z en (4.6) y (4.7) se pueden aproximar por

$$\frac{\partial H}{\partial z} \simeq \frac{H_{i+1} - H_i}{\Delta z} \qquad \forall i = 1, .., n \tag{4.11}$$

$$\frac{\partial Q}{\partial z} \simeq \frac{Q_i - Q_{i-1}}{\Delta z} \qquad \forall i = 2, .., n \tag{4.12}$$

donde el subíndice i está asociado a las variables en el comienzo de la sección i (figura 4.2) y la condición de frontera para cada sección queda descrita por

$$Q_{i+1} = \lambda_i\sqrt{H_{i+1}} \quad \forall i = 1, ..., n-1 \tag{4.13}$$

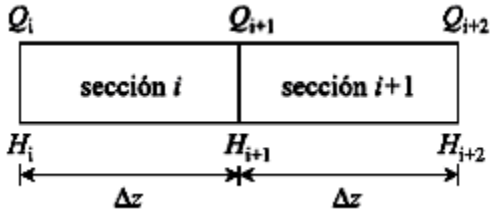

Fig. 4.2 Distribución de variables en el ducto

Así, sustituyendo en (4.6) y (4.7) las aproximaciones de las parciales dadas en (4.10) y (4.11), el modelo del fluido se puede escribir como n conjuntos de pares de ecuaciones diferenciales no lineales acopladas dadas por

$$\dot{Q}_i = a_1 (H_i - H_{i+1}) - \mu Q_i |Q_i| \qquad \forall i = 1, \ldots, n \tag{4.14}$$

$$\dot{H}_i = a_2 \left(Q_{i-1} - Q_i - \lambda_{i-1} \sqrt{H_i} \right) \qquad \forall i = 2, \ldots, n \tag{4.15}$$

con $H_1 = H_e$ y $Hn+1 = Hs$ como entradas y $Q1 = Qe$ y $Qn = Qs$ como salidas del sistema, de manera análoga a los modelos de parámetros y de cuadripolos de circuitos eléctricos (Kuo, 1966), y constantes relacionados con los parámetros físicos del ducto y el tamaño de las secciones:

$$a_1 = \frac{gA}{\Delta z}, \qquad a_2 = \frac{b^2}{\Delta z g A}, \qquad \mu = \frac{f}{2DA} \tag{4.16}$$

El problema de detección se divide en tantos problemas de detección particulares como fugas se consideren en el modelo.

IV.3 Análisis de detección

Se muestra el análisis de detección del ducto con dos fugas simultáneas planteado como objetivo de esta tesis. El objetivo de esta sección es mostrar en qué casos es posible resolver el problema de detección de dos fugas para el modelo (4.14) y (4.15).

El análisis inicia con las dos fugas distribuidas como se muestra en la figura 4.3. Esto es, las fugas se encuentran en el límite de las dos últimas secciones de los extremos, en los puntos λ_1 y λ_2 de la figura.

Fig. 4.3 Distribución de variables en el ducto con dos fugas en los límites de las secciones de los extremos

Se considera el modelo parametrizado del fluido dado por (4.14) y (4.15) con $\lambda_i = \lambda_i (Hi+1)^{1/2}$

$$
\begin{aligned}
\dot{Q}_1 &= -\mu Q_1^2 + a_1(H_1 - H_2) \\
\dot{H}_2 &= a_2(Q_1 - Q_2 - \bar{\lambda}_1) \\
\dot{Q}_2 &= -\mu Q_2^2 + a_1(H_2 - H_3) \\
\dot{H}_3 &= a_2(Q_2 - Q_3) \\
&\vdots \\
\dot{H}_{n-1} &= a_2(Q_{n-2} - Q_{n-1}) \\
\dot{Q}_{n-1} &= -\mu Q_{n-1}^2 + a_1(H_{n-1} - H_n) \\
\dot{H}_n &= a_2(Q_{n-1} - Q_n - \bar{\lambda}_2) \\
\dot{Q}_n &= -\mu Q_n^2 + a_1(H_n - H_{n+1})
\end{aligned}
\qquad 4.17
$$

El modelo se puede rescribir de la siguiente forma

$$
\begin{aligned}
\dot{x} &= f(x) + Bu + E_1\bar{\lambda}_1 + E_2\bar{\lambda}_2 \\
y &= \begin{bmatrix} x_1 & x_{\tilde{n}} \end{bmatrix}^T
\end{aligned}
\qquad 4.18
$$

donde x $\in R^{\tilde{n}}$ son los gastos y las presiones en cada sección del ducto y $\tilde{n} = 2n - 1$, $u \in R^2$ son las presiones conocidas en los extremos del ducto, H_1 y H_{n+1}, $\lambda_i \in R$ para i = 1,2 las señales de falla. Como salida el gasto en los extremos, Q_1 y Q_n, y las matrices *f(x), B, E_1 y E_2* teniendo los parámetros relacionados con los parámetros físicos del ducto.

De (4.17) se puede ver que existen dos señales de falla, λ_1 y λ_2. Por lo tanto, se formulan dos problemas particulares. Para cada caso se considera una fuga como la perturbación o falla de no interés ω_1 y la otra como la falla m_1.

Sea λ_1 la falla de no interés o perturbación ω_1 y $m1 = \lambda_2$, entonces (4.2) se puede rescribir como

$$
\dot{x} = f(x) + Bu + p_1\omega_1 + l_1 m_1 \qquad 4.19
$$

donde $p_1 = E_1$ y $l_2 = E_2$.

Estos resultados muestran que es posible generar residuos robustos a una fuga y sensibles a la otra. Por lo tanto, el problema de detección de fallas para un ducto con dos fugas simultáneas distribuidas como se muestra en la figura 4.1 se puede resolver. Esto trae como consecuencia la existencia de subsistemas desacoplados de una fuga y sensibles a la otra.

En conclusión, para el caso de dos fugas en el ducto con sólo mediciones de gasto y presión en sus extremos, el problema de detección tiene solución sólo si la presión en los puntos de fuga se encuentra explícitamente relacionada con la salida del sistema.

IV.4 Generación de residuos

En este trabajo el problema de detección de fallas se divide en dos tareas: la primera consiste en generar subsistemas desacoplados de la fuga de no interés ω_1 y sensible a la fuga de interés m_1, y la segunda en diseñar el generador de residuos mediante observadores no lineales.

Se considera el modelo no lineal con dos fugas de la forma

$$\begin{aligned}
\dot{Q}_1 &= -\mu Q_1^2 + \alpha_1(H_1 - H_2) \\
\dot{H}_2 &= \alpha_2(Q_1 - Q_2 - \bar{\lambda}_1) \\
\dot{Q}_2 &= -\mu Q_2^2 + \alpha_1(H_2 - H_3) \\
\dot{H}_3 &= \alpha_2(Q_2 - Q_3 - \bar{\lambda}_2) \\
\dot{Q}_3 &= -\mu Q_3^2 + \alpha_1(H_3 - H_4)
\end{aligned} \qquad 4.20$$

donde x ϵ R^5 es el vector de estado, H_1 y H_4 las entradas, λ_1 y λ_2 las señales de fuga y Q_1 y Q_3 las salidas y parámetros $\alpha_1 = gA/\Delta z$, $\alpha_2 = b2/gA\Delta z$ y $\mu = f/2DA$.

Para generar subsistemas desacoplados de cada fuga se propone una transformación lineal con sentido físico e inyección de la salida. La transformación está asociada con la suma y resta de los gastos relacionados con la fuga a desacoplar y tiene la ventaja de obtener un subespacio con variables de estado físicas. Se puede observar, por ejemplo, que para desacoplar $\lambda 1$ del sistema, la suma Q_1+Q_2 desacopla completamente la fuga ya que la variable de estado H_2 desaparece (Verde & Visairo, 2002).

Para aislar la fuga $\lambda 1$ en (4.19) se utiliza la siguiente transformación:

$$z = \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} x = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ - & - & - & - & - \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} x$$

que genera el subsistema sensible a λ_2

$$\begin{aligned}
\dot{z}_1 &= -\mu z_1^2 + 2\mu y_1 z_1 + \alpha_1(u_1 - z_2) - 2\mu y_1^2 \\
\dot{z}_2 &= \alpha_2(z_1 - y_1) - \alpha_2 z_3 - \alpha_2 \bar{\lambda}_2 \\
\dot{z}_3 &= -\mu z_3^2 + \alpha_1(z_2 - u_2)
\end{aligned} \qquad 4.21$$

con (u_1, u_2, y_1) como entradas y salida y_a = z_3.

Similarmente se obtiene el subsistema desacoplado de la fuga λ_2 usando la transformación

$$w = \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} x = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ - & - & - & - & - \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} x$$

resultando

$$\begin{aligned} \dot{w}_1 &= -\mu w_1^2 + 2\mu y_2 w_1 + \alpha_1(w_2 - u_2) - 2\mu y_2^2 \\ \dot{w}_2 &= \alpha_2(w_3 + y_2) - \alpha_2 w_1 - \alpha_2 \bar{\lambda}_1 \\ \dot{w}_3 &= -\mu w_3^2 + \alpha_1(u_1 - w_2) \end{aligned} \qquad 4.22$$

con (u_1,u_2,y_2) como entradas y y_a = w_3 como salida.

En el enfoque basado en observadores la idea básica es reconstruir las salidas del sistema a partir de las mediciones disponibles del proceso con la ayuda de observadores usando el error de estimación como un residuo para la detección y aislamiento de las fallas.

Existen varios diseños de observadores para los subsistemas (4.20) y (4.21) para la generación de los residuos, siguiéndose el dado en (Schreier 1998) presentado a continuación:

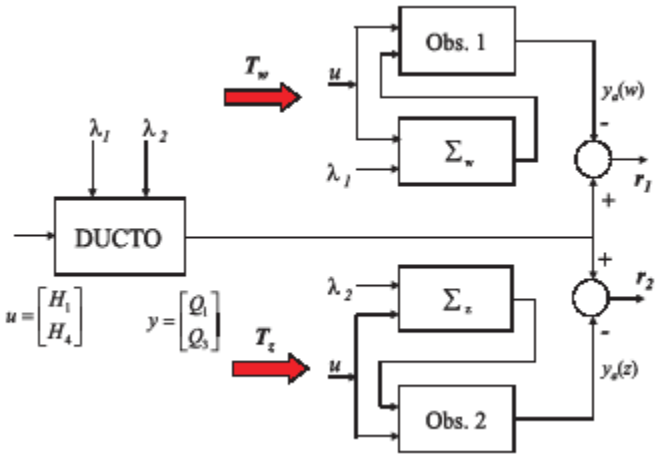

Fig. 4.4 Diagrama de bloques del generador de residuos

El método basado en observadores es apropiado si las fallas están asociadas con cambios en actuadores, sensores o variables de estado no medibles; esto es apropiado especialmente para detectar y aislar fallas aditivas. Se requiere de un modelo matemático detallado de la planta preferentemente derivado de los estados tal que las ecuaciones de estado tengan una interpretación física.

Nuestro algoritmo de detección cumple con el análisis descrito anteriormente, con señales de entrada conocidas provenientes del modelo del ducto, desacoplamiento en dos subsistemas independientes con uso de observadores y la generación de residuos robustos entre sí, permitiendo la identificación de dos fugas independientes entre sí.

Es importante notar que cuando se utiliza un observador para propósitos de FDI, no es necesaria la estimación del estado completo, sino solo de la salida. Para minimizar la ocurrencia de falsas alarmas se debe diseñar un observador que sea robusto a las entradas desconocidas.

La representación en *Simulink* del modelo físico del ducto es el siguiente:

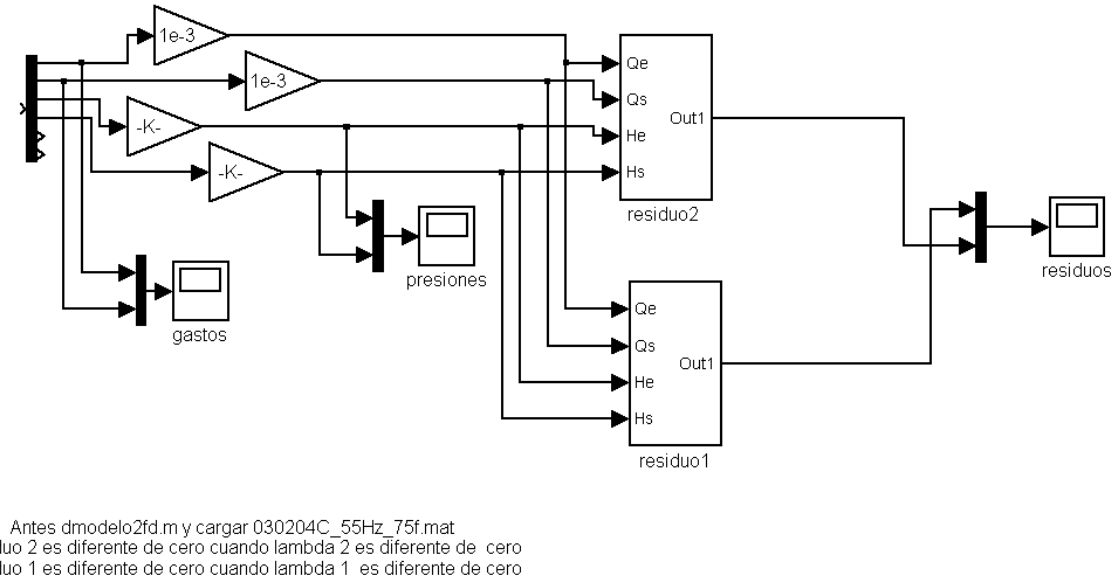

Fig. 4.3 Modelo del generador de residuos en *Simulink*

Los datos de los sensores son recuperados del modelo de lectura, se convierten entonces en las entradas a los bloques "*residuo 1*" y "*residuo 2*" que representan al observador de la tubería.

Estos últimos bloques constituyen lo que se conoce como subsistema en *Simulink*, es decir que contienen una colección de nuevas operaciones sobre sus entradas y entregando el valor del *residuo* correspondiente a su salida fig. 4.4 y 4.5, reduciendo ramificaciones y simplificando con esto la presentación del modelo al usuario. Al ingresar al primer subsistema "*residuo 1*" puede observarse el tratamiento que se hace de los datos en el siguiente modelo:

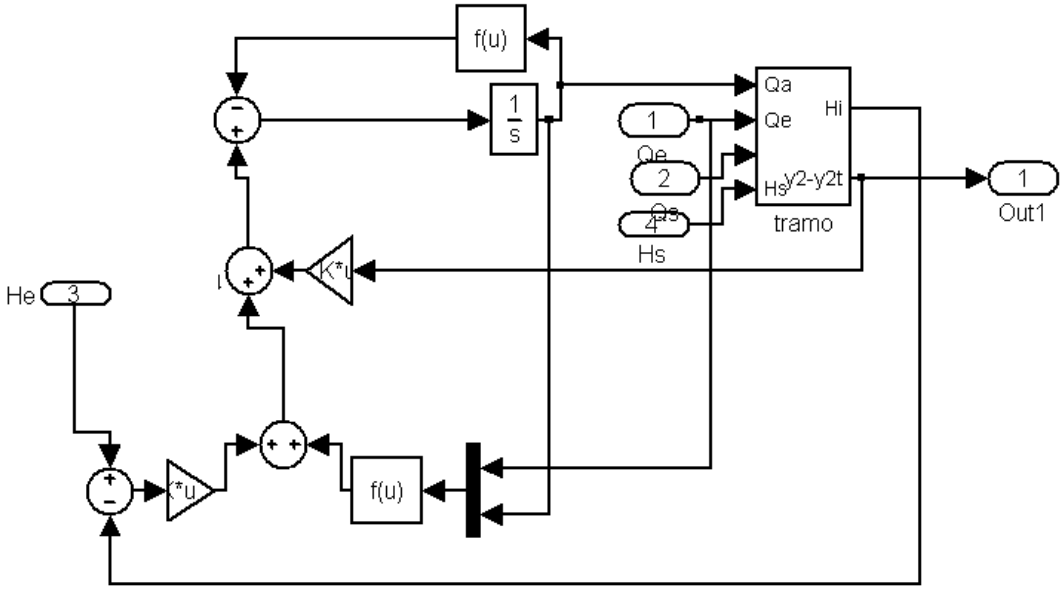

Fig. 4.4 Subsistema *residuo*

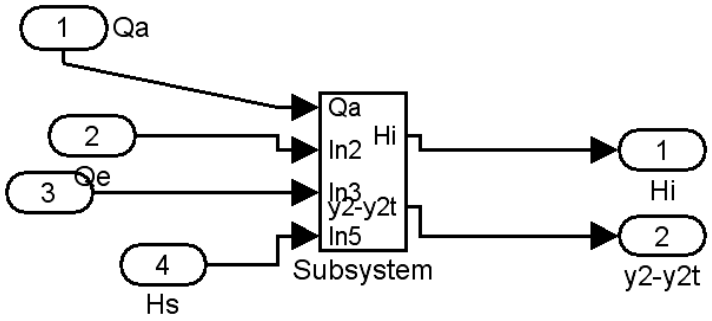

Fig. 4.5 Subsistema *tramo*

Estos subsistemas representan el sistema de ecuaciones (4.21) ó (4.22) donde desacoplamos cada una de las señales de falla λ_1 ó λ_2, logrando visualizar a la salida de estos bloques, el comportamiento de los residuos en tiempo real y de manera independiente.

Capítulo V

Resultados en simulación y experimentales

V.1 Resultados en simulación

Uno de los objetivos planteados al inicio de este trabajo de tesis, fue la validación en línea y fuera de línea del algoritmo de detección de fugas descrito anteriormente. Es en este capítulo donde se presentan los resultados obtenidos después de numerosas pruebas en laboratorio, además de las simulaciones que sirvieron para ajustar los parámetros del ducto descrito.

Para analizar el desempeño bajo condiciones reales, en los resultados de simulación se utiliza como entrada una señal senoidal que desvía la presión aguas arriba un 10% de su valor nominal 6.81 m. La presión aguas abajo es de 1.49 m. Los coeficientes de fuga λ_1 y λ_2 son 0.001. La fuga 1 se presenta a los 400 s y la fuga 2 a los 800 s. En la figura 5.1 se pueden observar las señales conocidas de gasto y presión en los extremos de la tubería.

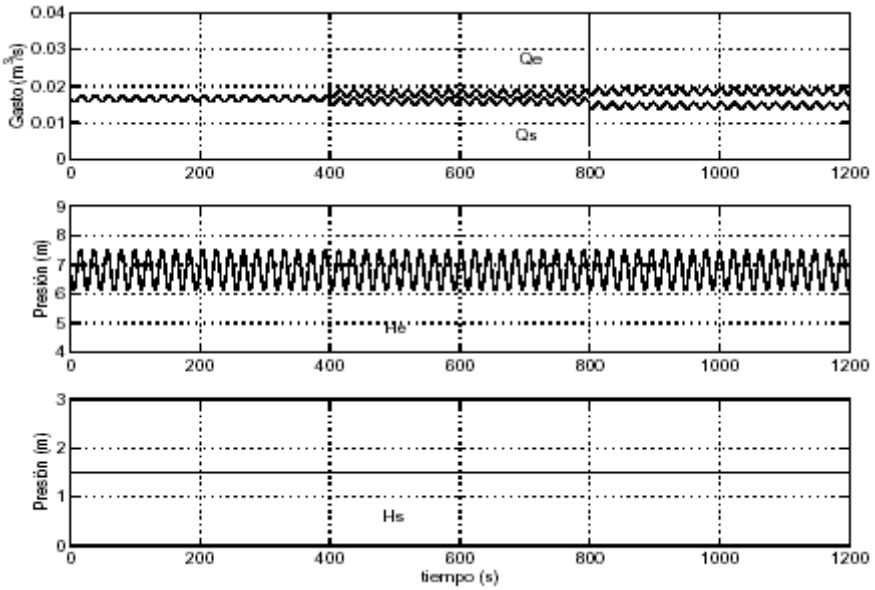

Fig. 5.1 Señales conocidas de gasto y presión en los extremos de la tubería con dos fugas distribuidas uniformemente

La evolución *r1* y *r2* de los residuos se muestra en la fig. 5.2. En esta figura se puede observar que cada residuo es sensible a su fuga asociada y esta totalmente desacoplado de la otra. Esto se puede ver cuando se presenta la primera fuga, el residuo *r1* se hace diferente de cero y *r2* permanece en cero. Cuando aparece la segunda fuga a los 800 s el residuo *r1* permanece en su valor y *r2* se desvía de cero; de modo que cuando las dos fugas están presentes, los dos residuos son diferentes de cero.

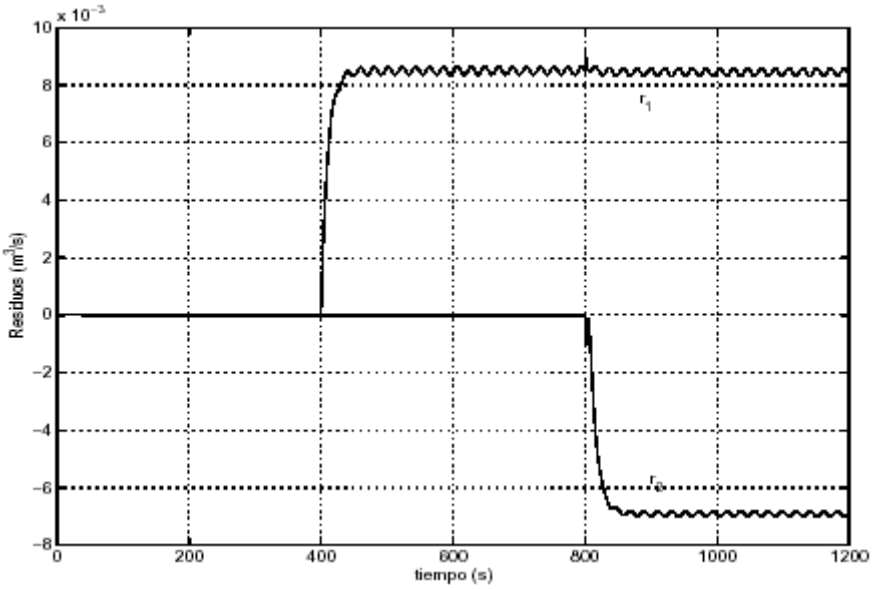

Fig. 5.2 Evolución de los residuos con 2 fugas localizadas uniformemente a lo largo del ducto

La ventaja que presenta un generador de residuos considerando las no linealidades del sistema en su diseño, es la robustez ante variaciones del punto de operación. Para probar la robustez del generador de residuos ante estas variaciones, se simula un cambio de punto de operación del 9.7% del gasto nominal a los 600 s mostrado en la fig. 5.3.

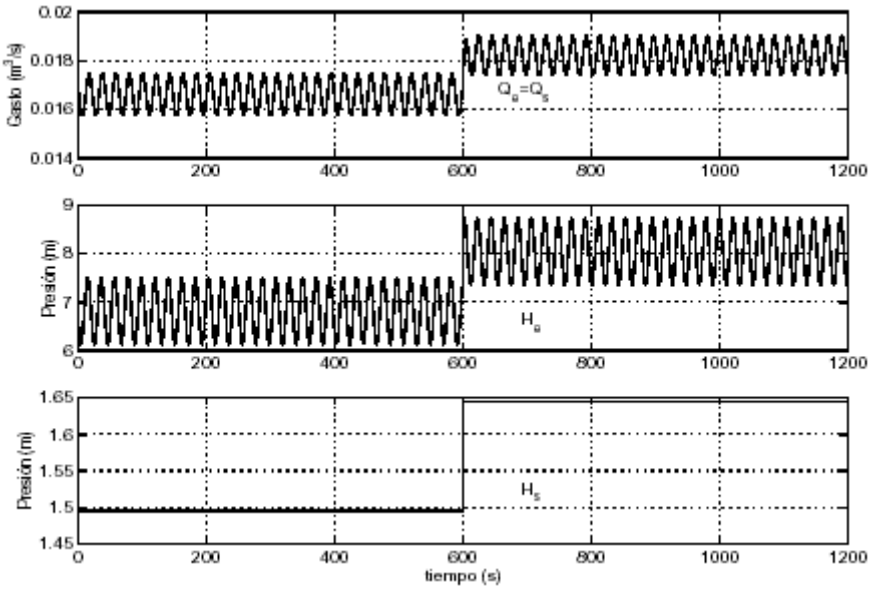

Fig. 5.3 Señales de gasto y presión con un cambio de punto de operación del 9.7% del gasto nominal a los 600 s

En la figura 5.4 se muestra la evolución de los residuos. Los residuos en ausencia de fugas son cero. Se puede observar que los residuos se mantienen en cero cuando ocurre el cambio de punto de operación; es decir, son robustos ante estas variaciones.

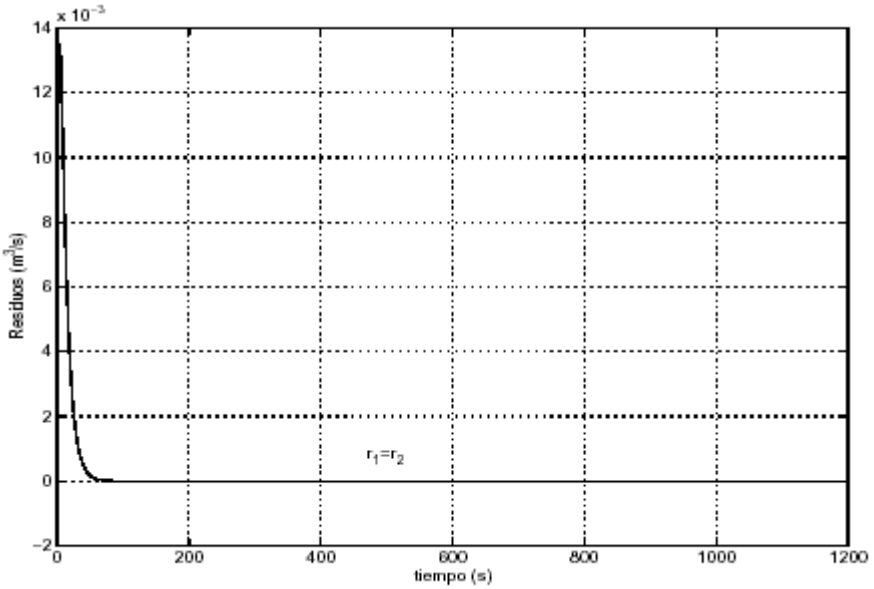

Fig. 5.4 Evolución de los residuos con un punto de operación del 9.7% del gasto nominal a los 600 s

V.2 Resultados experimentales

Se presentan aquí los resultados experimentales de las pruebas hechas en la planta piloto de la tubería, donde se adquirieron datos de presión y gasto a la entrada y salida de la tubería.

Las pruebas que se realizaron son de diferentes tipos:

a) Observación del estado estacionario del sistema de la tubería y el fluido, donde se puede ajustar el tiempo de observación desde 1 minuto hasta 2 horas.

b) Comportamiento de los residuos al provocar una fuga en un tiempo establecido.

c) Análisis del estado estacionario del sistema al cambiar el punto de operación por medio de una entrada en el modelo de *Simulink*.

d) Comparación de desempeño entre los dos tipos de sensores de flujo.

V.2.1 Resultados experimentales obtenidos con medidores de gasto ultrasónicos

En esta sección se presentan resultados de la evolución de los residuos con datos reales fuera de línea. Las mediciones de gasto se tomaron con los sensores de ultrasonido. Estos sensores tienen un tiempo de refrescamiento de aproximadamente 4.4 s, dato que no fue suministrado por el fabricante y que fue necesario identificar empíricamente.

El intervalo de operación del inversor considerado es de 45 a 60 *Hz* obteniéndose gastos de 0.020 a 0.015 m^3/s y presiones de 9.1455 a 5.6280 *m*, respectivamente.

A continuación se presentan gráficas de los resultados del algoritmo de detección con datos reales con una fuga provocada a los 600 *s* mediante la apertura de la servo válvula localizada a 49 *m*. También se muestran resultados cuando no se presenta ninguna fuga pero sí un cambio de punto de operación. Para eliminar los efectos de ruido de las mediciones, se filtran los residuos con un filtro pasa-bajas con un ancho de banda de 0.01 *rad/s*.

En la figura 5.6 se muestran los datos de gasto y presión en los extremos de la tubería con una apertura del 75% de la servo válvula. El gasto de fuga corresponde a 1.1 × 0−3 m^3/s; es decir, el 6.1% del gasto nominal.

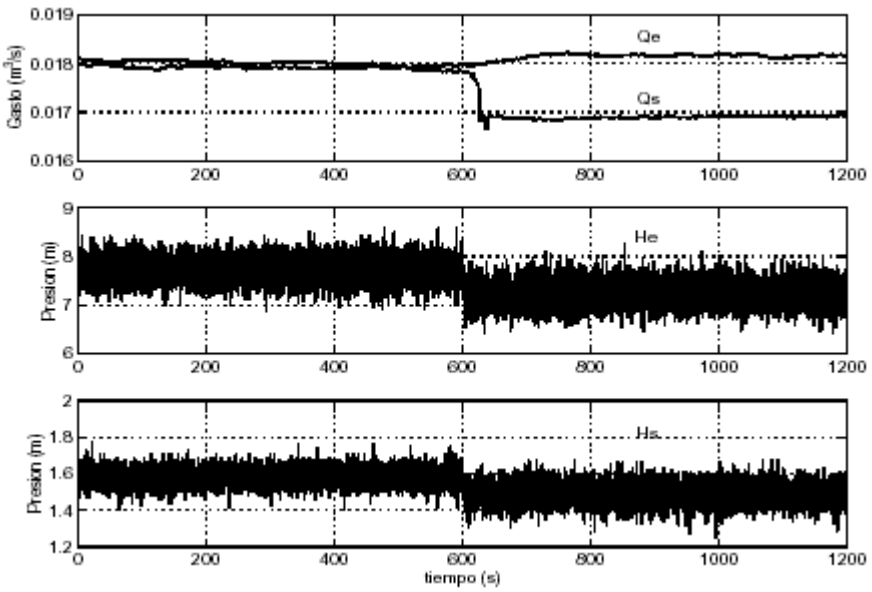

Fig. 5.6 Gastos y presiones en los extremos del ducto con una fuga del 6.1% del gasto nominal localizada a 49 *m*

En la fig. 5.7 se muestran los residuos y se observa que éstos se mantienen cercanos a cero hasta que se presenta una fuga a los 600 *s*. Al presentarse la fuga, el residuo r1 se hace diferente de cero y el residuo r2 se mantiene casi cero. Esto indica que la fuga es cercana a la fuga 1 que para diseño está en 43.96 *m* considerada en el diseño del generador de residuos.

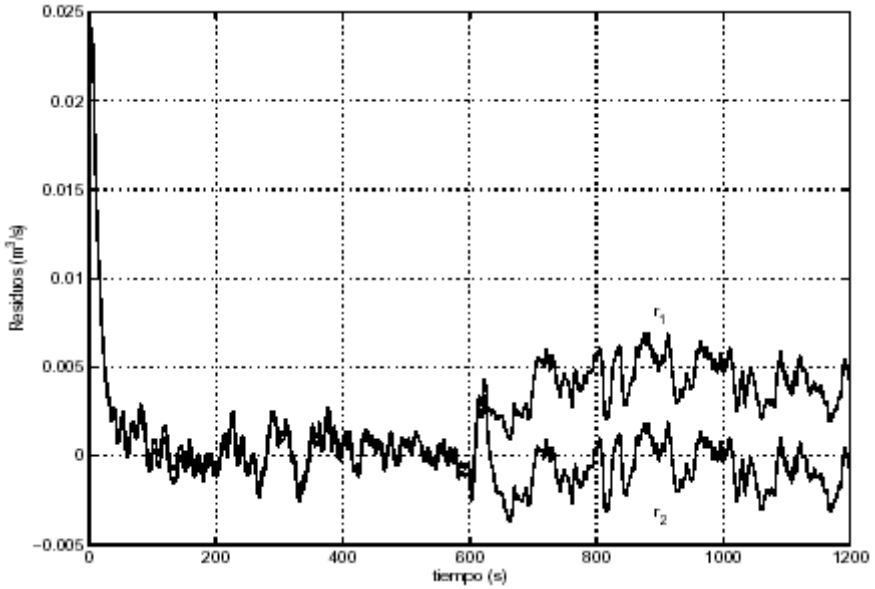

Fig. 5.7 Evolución de los residuos con una fuga del 6.1% del gasto nominal localizada a 49 *m*

En la figura 5.8 se muestran los datos de gasto y presión con una fuga provocada del 60% de apertura de la servo válvula. Con un gasto de fuga de $0.5 \times 10^{-3}\, m^3/s$; esto es, el 3.1% del gasto nominal.

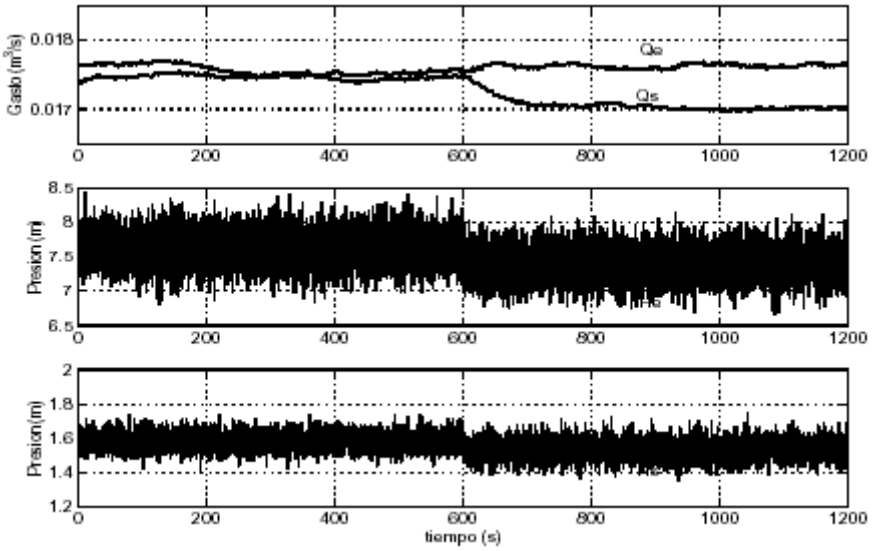

Fig. 5.8 Gastos y presiones en los extremos del ducto con una fuga del 3.1% del gasto nominal localizada a 49 m

En la fig. 5.9 se ve que los residuos se mantienen cercanos a cero hasta los 600 s cuando aparece una fuga. El residuo r1 se desvía de cero mientras que r2 permanece cerca de ese valor. Esto indica que la fuga se encuentra próxima a los 43.96 *m* considerados en el diseño.

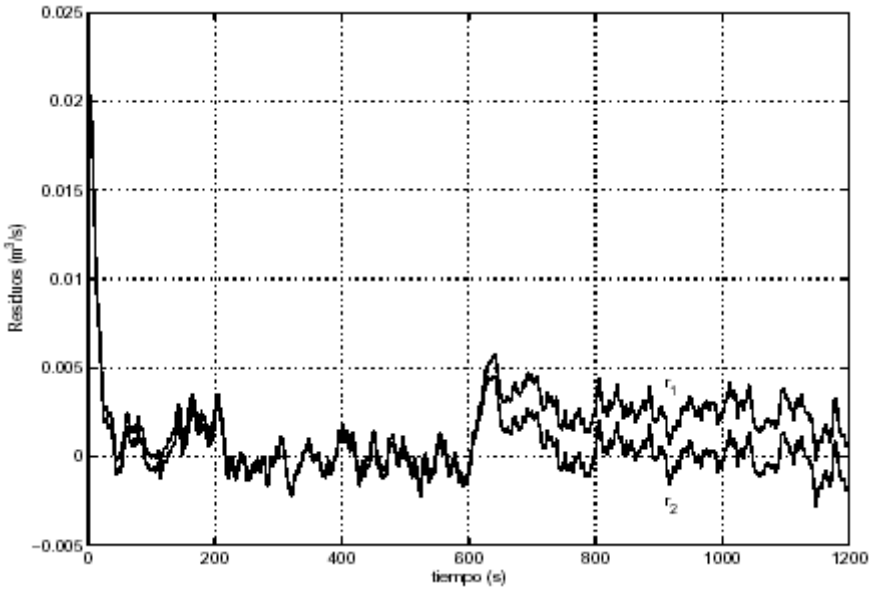

Fig. 5.9 Evolución de los residuos con una fuga del 3.1% del gasto nominal localizada a 49 m

En las siguientes figuras se muestra el desempeño del generador de residuos ante un cambio de punto de operación. En la figura 5.10 se muestran datos de gastos y presión en los extremos de la tubería con un cambio de punto de operación de 0.0170 a 0.0186 m^3/s, es decir, del 9.1% del gasto nominal a los 600 s.

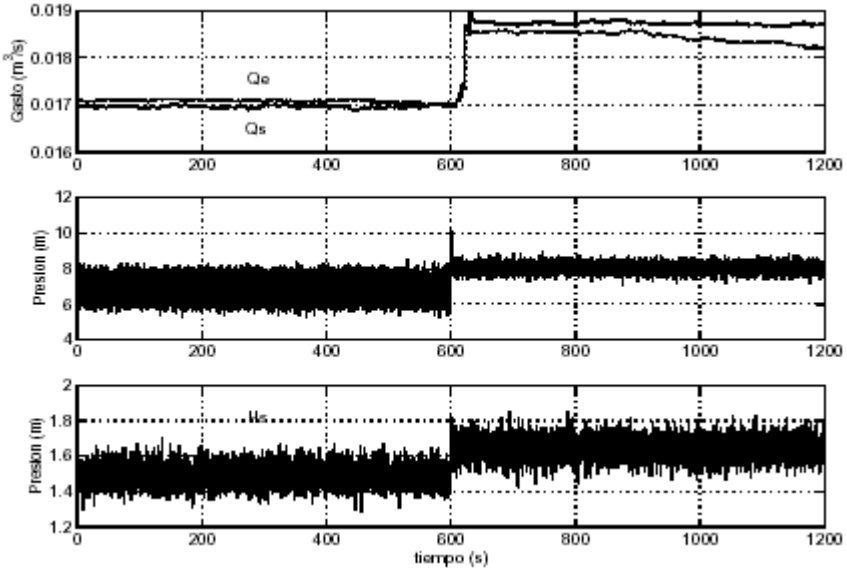

Fig. 5.10 Gastos y presiones en los extremos del ducto con un cambio de punto de operación del 9.1% del gasto nominal a los 600 seg.

En la figura 5.11 se muestran los residuos y se observa que son robustos a estos cambios. Sin embargo, se puede observar que los generadores de residuos indican una falsa alarma a los 600 s y se mantiene durante el transitorio, esto se debe a que el modelo del fluido aproximado para el diseño de los observadores no describe satisfactoriamente

comportamiento transitorio de los flujos reales de la tubería. Esto se debe al bajo orden considerado en el modelo de parámetros concentrados.

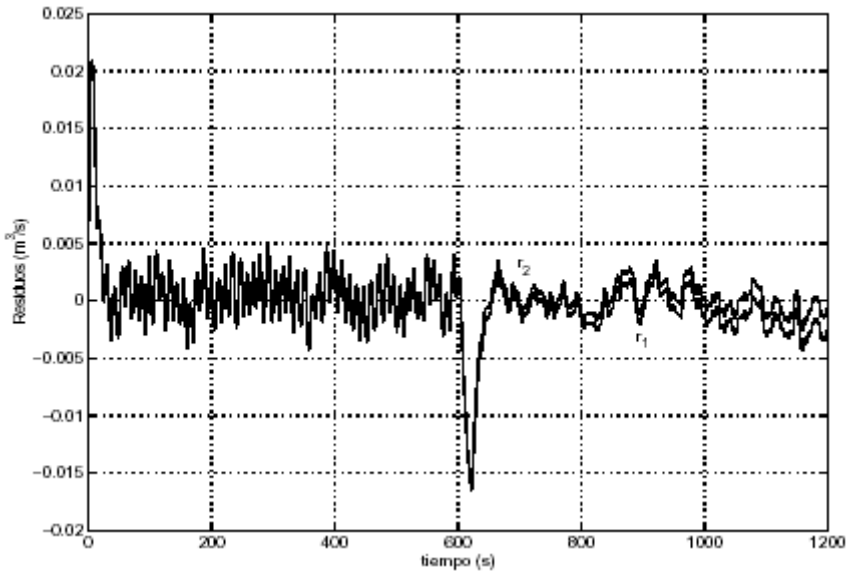

Fig. 5.11 Evolución de los residuos usando datos reales cambiando punto de operación del 9.1% del gasto nominal a los 600 s

V.2.2 Resultados experimentales obtenidos con sensores de gasto de propela

En esta sección se muestran los resultados con mediciones de gasto con los sensores de propela. Estos sensores tienen un tiempo de respuesta de aproximadamente 0.01 *s* lo que permite ver la respuesta transitoria de los gastos reales de los extremos de la tubería.

En la figura 5.12 se muestran las señales medidas de gasto y presión en los extremos del ducto, se provoca un cambio de punto de operación del 11.5% del gasto nominal a los 380 *s*.

Se observa en la fig. 5.13 que el residuo presenta una falsa alarma durante el transitorio del cambio de punto de operación. Sin embargo, el pico que se presenta al realizar el cambio de punto de operación tiene menor magnitud (0.005 m^3/s) que el presentado con las mediciones con los sensores de ultrasonido (0.015 m^3/s). Esto se justifica porque la respuesta de los sensores de propela es más rápida que la de los de ultrasonido y se obtiene un mejor seguimiento de los gastos durante el transitorio. Las falsas alarmas en el transitorio se deben principalmente a que la respuesta real y la del modelo son diferentes.

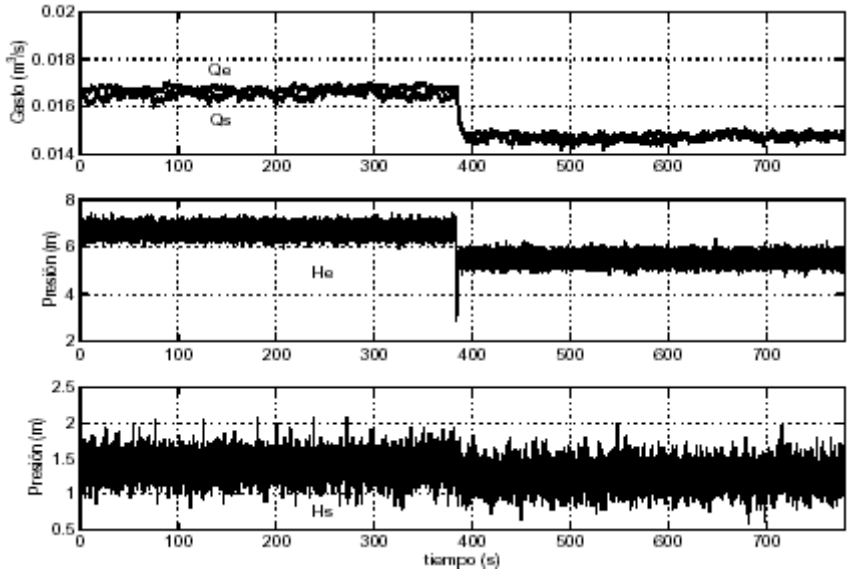

Fig. 5.12 Gastos y presiones en los extremos del ducto con un cambio de punto de operación del 11.5% del gasto nominal a los 380 s

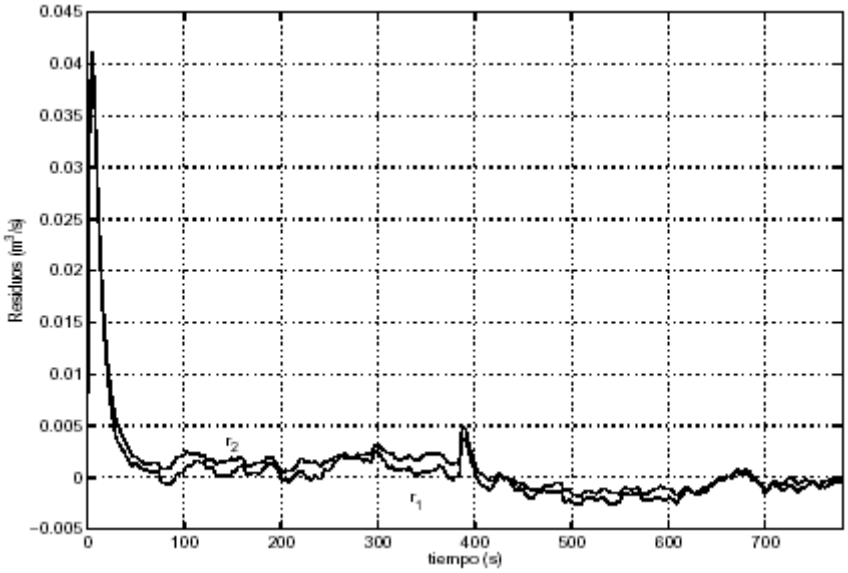

Fig. 5.13 Evolución de los residuos con un cambio de operación del 11.5% del gasto nominal a los 380 s

En el Capítulo IV se concluyó que el problema de detección de fallas se puede resolver sólo bajo dos suposiciones para dos fugas. Debido a esto, se utiliza el modelo con tres secciones y dos fugas distribuidas uniformemente a lo largo del ducto para el diseño de los generadores de residuo. Sin embargo, la respuesta transitoria de este modelo no representa las dinámicas reales, esto se puede ver en las dos figuras siguientes.

En la figura 5.14 se muestra la respuesta transitoria de los gastos reales y los gastos generados por el modelo en los extremos de la tubería. Se puede ver que existe una diferencia entre la señal real y la del modelo durante la parte transitoria.

En la figura 5.15 se compara la respuesta transitoria de las presiones reales y las presiones del modelo en los extremos de la tubería. Se observa que existe una diferencia entre las señales de datos reales y las señales obtenidas del modelo.

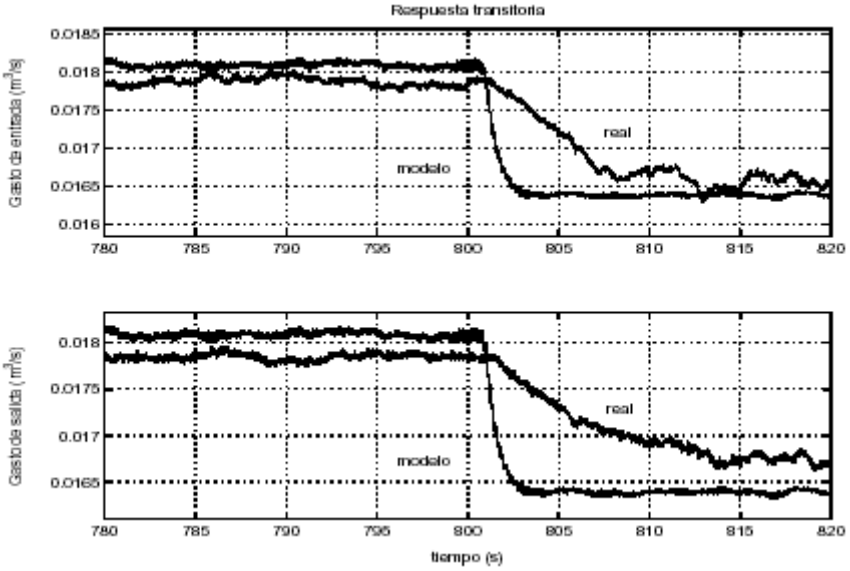

Fig. 5.14 Respuesta transitoria de los gastos reales con tres secciones

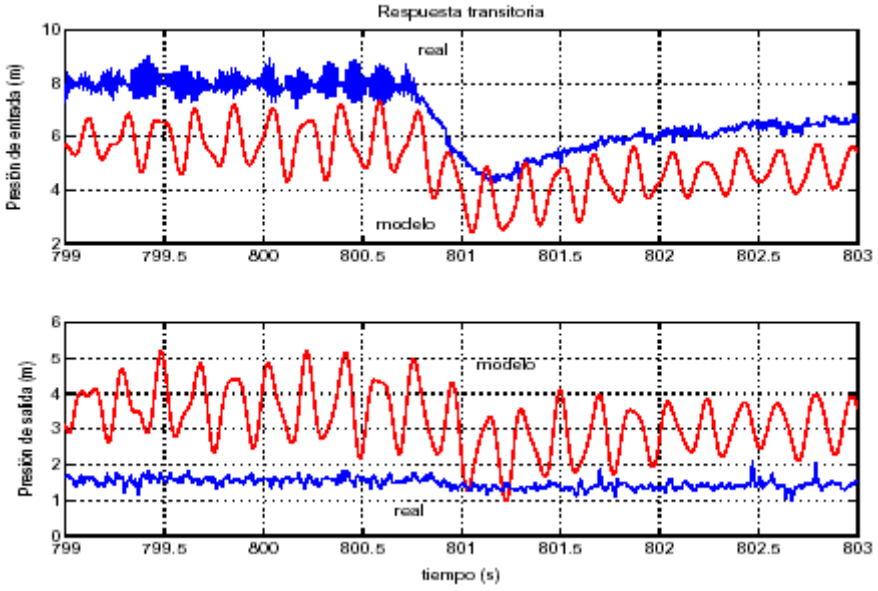

Fig. 5.15 respuesta transitoria de las presiones reales con tres secciones

Sin embargo, el problema de detección con observadores no lineales para un mayor número de secciones con las dos fugas localizadas arbitrariamente no tiene solución. Por lo tanto, en general, resulta imposible generar subsistemas desacoplados para un mayor número de secciones.

V. 3 Resultados en tiempo real

El objetivo principal de esta Tesis es validar en tiempo real la detección de fugas en un ducto utilizando mediciones a la entrada y salida de la tubería, en esta sección se presentan las lecturas obtenidas y sus resultados.

a) Cambio de frecuencia de operación

Como primera parte, se realizó una prueba variando la frecuencia de operación de la bomba que hace circular el agua en la tubería de 45 a 50 y 55 Hertz, siendo el objetivo observar el comportamiento de las variables durante estos cambios. Como se observa en la fig. 5.16, los gastos y presiones reflejan estos incrementos, no así los *residuos* que se mantienen alrededor de su valor inicial.

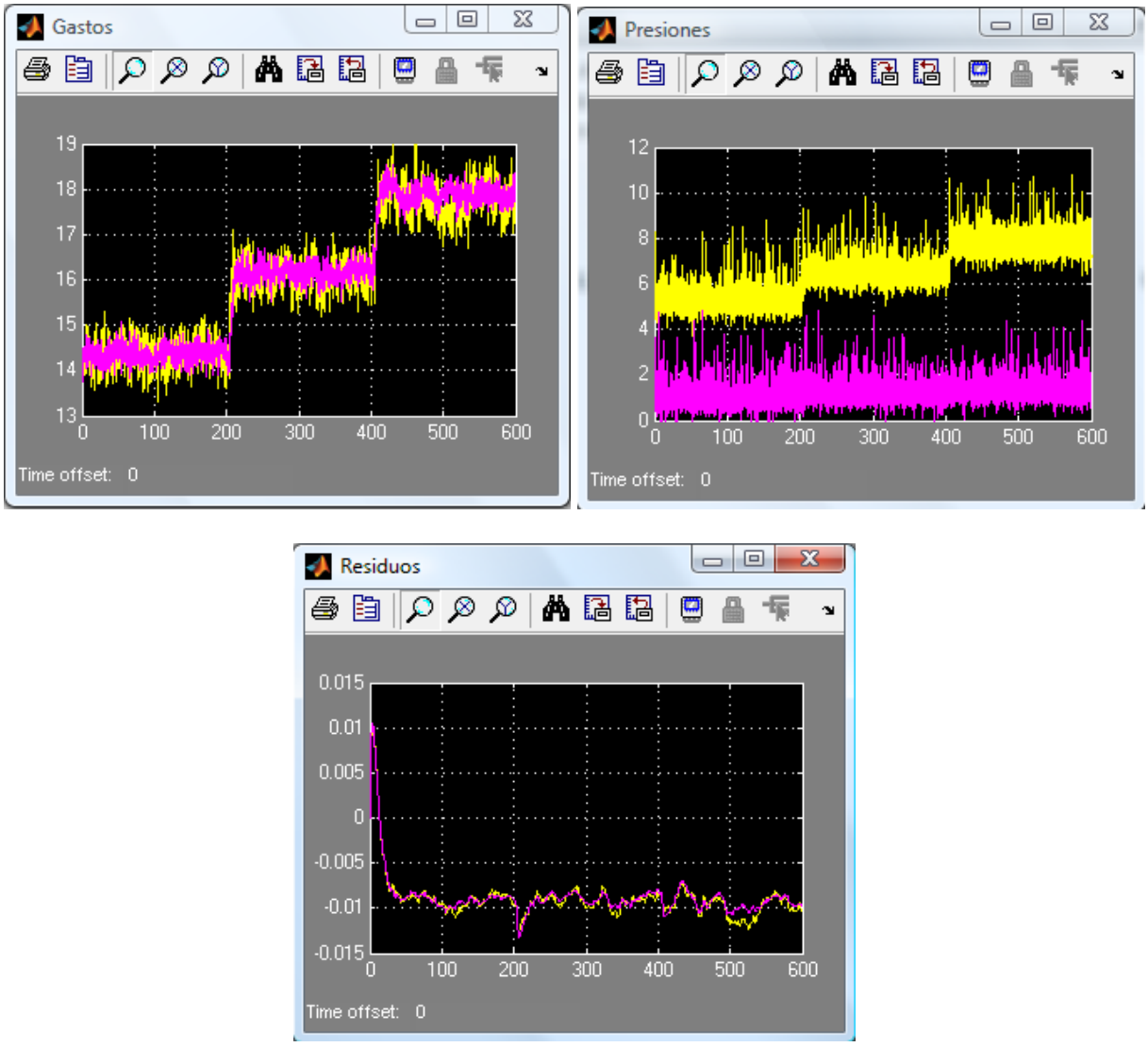

Fig. 5.16 Lecturas en cambio de operación

b) Fugas a diferentes frecuencias de operación

La segunda parte de pruebas consistión en la generación de una fuga a los 150 *s* con 100% de apertura de la servo-válvula a diferentes frecuencias de operación.

En la fig. 5.17 se presentan las lecturas de la prueba a 45 Hertz, con fuga del 8% del gasto nominal, donde se aprecia su efecto en gastos y presiones, así como en la variación del residuo asociado a dicha fuga.

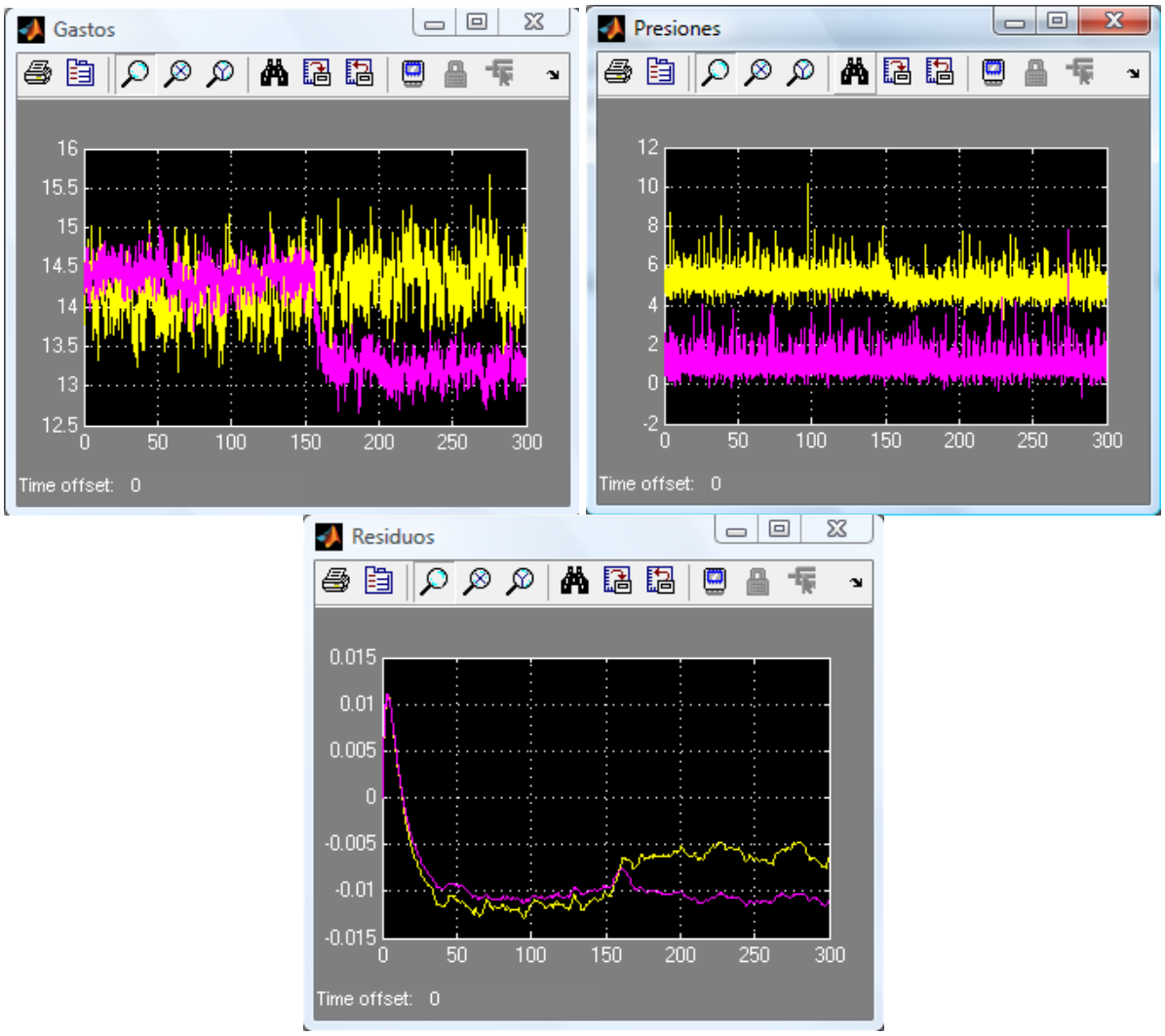

Fig. 5.17 Efecto de fuga a 45 Hertz de operación

En la fig. 5.18 se presentan las lecturas de la prueba a 50 Hertz, con fuga del 8% del gasto nominal, donde se aprecia su efecto en gastos y presiones, así como en la variación del residuo asociado a dicha fuga.

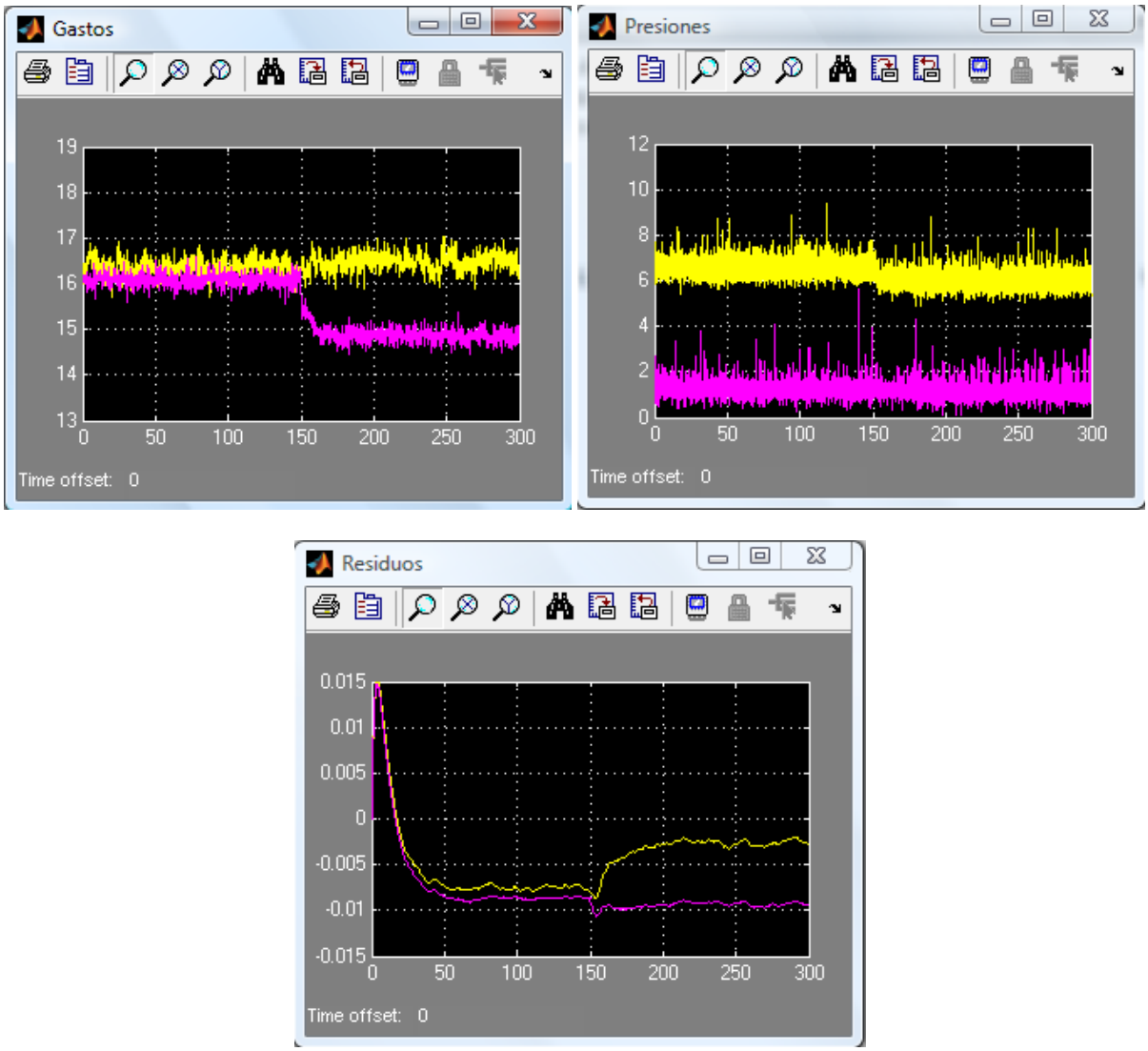

Fig. 5.18 Efecto de fuga a 50 Hertz de operación

En la fig. 5.19 se presentan las lecturas de la prueba a 50 Hertz, con fuga del 8% del gasto nominal, donde se aprecia su efecto en gastos y presiones, así como en la variación del residuo asociado a dicha fuga.

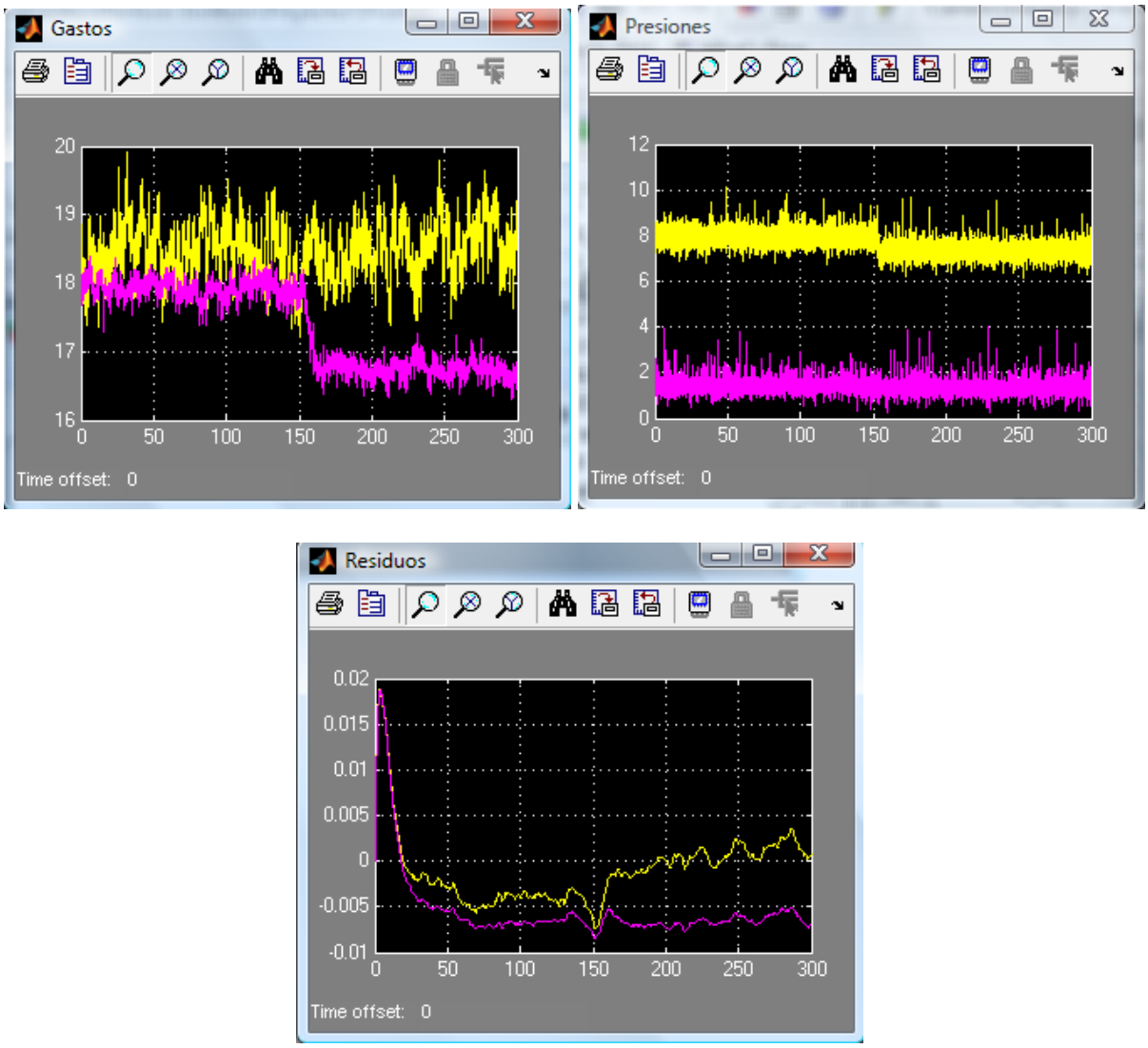

Fig. 5.19 Efecto de fuga a 55 Hertz de operación

De estos resultados descritos anteriormente se identifican los siguientes problemas con el esquema de detección utilizado:

• Existe un problema de robustez en la localización de las fugas, ya que sólo se pueden diseñar dos generadores de residuos.

• El transitorio del modelo no refleja el comportamiento real de los gastos y presiones de los extremos del ducto, lo que provoca falsas alarmas en los residuos.

• El esquema de detección imposibilita la detección de más de dos fugas en posiciones fijas de la tubería.

Capítulo VI

Conclusiones

Al principio de este trabajo se plantearon tres objetivos principales, los cuales se analizan en esta sección.

El primer objetivo fue diseñar un sistema de adquisición de datos y control para la tubería piloto que se encuentra en el Instituto de Ingeniería de la UNAM y que se describe en el capítulo II, utilizando las herramientas ya mencionadas y que constituyó la mayor parte de este trabajo.

Este objetivo se cumplió arrojando valiosos resultados como la creación de un controlador para la tarjeta *PCAO2DC* desde *Matlab*, y con el que no se contaba anteriormente en la coordinación de automatización, así como una guía para este tipo de diseños que no se encontraban documentados, permitiendo con esto que futuras generaciones de investigadores y alumnos tengan una referencia al momento de realizar esta tarea.

Otra aportación es la calibración de sensores de presión y flujo utilizados en la parte de pruebas y que harán más fácil su uso posterior, además de descubrir las limitaciones en la exactitud de los sensores de flujo digitales, pues nunca pudo lograrse un error cero aún sin fugas; también se mejoró y extendió el manejo del inversor digital encargado de la operación de la bomba, descubriendo algunas limitantes que tiene, tales como no poder utilizar la tierra como común para más de una salida. Además se obtuvo el módulo de fricción para la tubería por medio de los datos experimentales, en donde se detectó que los voltajes a las salidas de la tarjeta presentan ligera inestabilidad, pues variaban en el tiempo, aunque no de manera significativa, éstas si pueden tener una repercusión en la operación de la bomba; en cuanto a la servo válvula utilizada, ahora se cuenta con un instrumento que permite generar fugas con gran precisión en tiempo y tamaño, queda aún pendiente la obtención de su relación corriente-apertura; también se concluyó que la bomba auxiliar encargada de reenviar el agua de las fugas al depósito principal es insuficiente para esta tarea, debiendo ser reemplazada por una de mayor capacidad.

El segundo objetivo consistió en integrar en este sistema el algoritmo de detección de fugas diseñado por la Dra. Visairo y que contiene el modelo físico de la tubería y la dinámica de las fugas (Visairo, 2004), aunque esta fuera del alcance de este trabajo su análisis y deducción. Los resultados usando estas herramientas permiten concluir específicamente bajo qué condiciones es posible generar subsistemas desacoplados de perturbaciones o fallas de no interés y sensibles a las fallas de interés. De los resultados obtenidos se puede concluir que sólo bajo dos condiciones se pueden detectar dos fugas simultáneas, la primera es que las presiones en las fugas estén directamente relacionadas con las salidas y la segunda que se supongan conocidas las posiciones de las fugas en el análisis. Para un caso de mas fugas resulta imposible el desacoplo de una fuga sin perder la sensibilidad de las otras en las salidas. Por lo tanto, el problema general de detección de múltiples fugas simultáneas no tiene solución.

La utilización del modelo en Simulink del algoritmo no presentó mayor dificultad pues se integró a los modelos diseñados para la adquisición de datos, primero en simulación utilizando datos obtenidos experimentalmente y posteriormente en pruebas en tiempo real en la tubería.

Por último se tomó la validación del modelo localizador de fugas. De las pruebas en la tubería se puede concluir que para lograr tener una mayor repetitibilidad en las mediciones debe buscarse contar con condiciones semejantes, esto sobre todo en el nivel de agua del tanque, pues es crucial que no se obstruya la salida del agua en la descarga, lo que podría trabajar como un freno al fluido, alterando las mediciones.

Los resultados en simulación del algoritmo de detección permiten afirmar que la detección es satisfactoria cuando las fugas se encuentran exactamente en las posiciones consideradas en el algoritmo. Sin embargo, este algoritmo no es robusto ante incertidumbres en la posición. También se puede decir que los residuos son robustos ante variaciones en el punto de operación.

Los resultados experimentales muestran que el modelo matemático del fluido utilizado para el diseño de los observadores no describe la dinámica real del sistema y de aquí que existan falsas alarmas en los residuos al presentarse cambios de punto de operación. Se hizo una comparación de las señales reales del sistema con las señales generadas por el modelo con diferente número de secciones. De aquí se concluyó que el problema del transitorio se puede minimizar aumentando el número de secciones en el modelo.

Los sensores de flujo representan otro punto de cuidado, pues su desempeño se ve afectado en gran medida por la selección de una localización óptima, obtenida ésta por prueba y error, es recomendable mantenerla. La habilidad en la colocación y el ajuste de los soportes, demanda especial cuidado.

Hoy en día, el sistema SCADA desarrollado en esta tesis es usado en proyectos de investigación en el área de control de procesos, enfocado a buscar una solución al problema de detección de múltiples fugas simultáneas en un ducto. El problema se divide en detección, localización e identificación de las fugas, todas ellas en función del buen desempeño del sistema de adquisición de datos y control usado, y en espera de futuras aplicaciones.

Apéndice A

Diseño del controlador para la Tarjeta *PCAO2DC*

Targets de *Real-Time Workshop*

Real-Time Workshop traduce el modelo *Simulink* de su sistema a código *C ANSI*. El código generado proporciona un modelo residente en memoria y totalmente instrumentado, que es muy útil para verificar y depurar. Una plantilla de *target* especifica el entorno de *target* en el que se correrá su código generado. *Real-Time Workshop* soporta numerosos entornos de target para todas las fases del proceso de desarrollo del producto. Estos incluyen configuraciones *ready-to-run* y *targets* de otros fabricantes. También puede desarrollar sus propios targets personalizados.

Cuando se elige una plantilla de *target*, se elige implícitamente un formato de código. El formato de código determina el estilo y el esquema general del código generado.

S-Functions

El target de *S-Functions* posibilita traducir un subsistema o un diagrama de bloques de *Simulink* a una *S-Function* (función de sistema) de *Simulink*. Una *S-function* es una descripción en lenguaje de ordenador de un bloque *Simulink*. Usando el *target* de función S usted puede generar un código para un modelo que se usará más tarde como componente en un modelo mayor. Esto le permite acelerar simulaciones, reutilizar códigos y compartir componentes de modelos sin exponer los detalles del modelo fuente registrado. Puede incluir en el mismo modelo múltiples casos de la misma función *S*, manteniendo cada uno estructuras de datos independientes.

Dentro de la realización del sistema SCADA para el proyecto, fue necesario el desarrollo de un Controlador (Driver) para la Tarjeta *PCAO2DC* del fabricante *National Instruments* que no se encuentra soportada por *Matlab*, proceso que se describe a continuación y que comienza con la definición de controlador para nuestra área de interés:

Controlador es un programa que permite la comunicación entre la computadora y un dispositivo periférico que se desee manejar desde ésta.

Desarrollo de aplicaciones utilizando la *Matlab C* Math Library

La construcción y desarrollo de aplicaciones utilizando esta librería es un proceso de amplias perspectivas una vez se tiene un dominio adecuado de su operativa. El producto está dividido en dos categorías (como librerías objeto): la librería (built-in library) contiene versiones de las funciones de *Matlab* en lenguaje *C* del tipo numérico, lógico y utilidades. Por otra parte la librería de toolboxes (toolbox library) contiene versiones compiladas de la mayoría de archivos M de *Matlab* para cálculo numérico, análisis de datos y funciones de acceso a archivos y matrices.

Utilización de *Matlab* y de su compilador

Para construir una aplicación del tipo 'stand alone' que incorpore código originalmente desarrollado como archivos M de *Matlab* , deberán de seguirse los pasos siguientes:

- Utilizar el compilador de *Matlab* para convertir archivos M en *C* mediante la utilización de la instrucción mcc -e (la cual es externa a *Matlab*).
- Compilar el código *C* fuente en código objeto utilizando un compilador *ANSI C*.
- Enlazar el código resultante con la *Matlab C* Math Library y con cualquier tipo de archivos y programas específicos que hayan sido previamente definidos por el usuario.

Generación Automática de archivos MEX.

El compilador de *Matlab* automatiza la creación de archivos MEX de *C* (*Matlab* Ejecutables). Los archivos MEX contienen código objeto que es dinámicamente enlazado como 'runtime' en el entorno *Matlab* por el intérprete del programa.

El proceso en cuestión se realiza en tres pasos:

- El compilador de *Matlab* traduce las funciones *Matlab* en sus funciones equivalente en lenguaje C.
- La instrucción *Matlab* cmex llama al compilador y al enlazador del sistema para construir un fichero MEX objeto.
- El intérprete de *Matlab* enlaza automáticamente la función de *Matlab* como 'runtime'.

Mientras se efectúa una conversión de los archivos M en archivos MEX, el compilador realiza llamadas a las rutinas de la librería *C* para muchas de las instrucciones contenidas en el propio núcleo de *Matlab*. Existen algunas funciones, incluyendo las rutinas 'Handle Graphics', para las cuales se generan de nuevo llamadas 'callbacks' a *Matlab*.

Pueden convertirse convenientemente archivos M en código fuente *C* para incorporarlos posteriormente en los archivos externos desarrollados en lenguaje C, si ese es el caso. Esta opción es ideal para usuarios que quieren sacar la máxima ventaja de Matlab desde cualquier otra aplicación o producir código *C* eficiente a partir de los algoritmos desarrollados con *Matlab*. Los desarrollos del tipo 'stand-alone' requieren para ello de la *Matlab C* Math Library. Obsérvese que las funciones gráficas de *Matlab* no están incluidas.

Para construir aplicaciones 'stand-alone' se debería seguir los siguientes pasos:

- Utilizar el compilador de *Matlab* para convertir archivos M en C con la instrucción externa mcc -e.
- Compilar el código *C* fuente en código objeto utilizando un compilador *C*.
- Enlazar el código resultante con las librerías matemáticas *C* de *Matlab* y los archivos específicos de que dispongamos.

Dicho controlador se diseñó utilizando lo que en *Matlab* y *Simulink* se conoce como *S-Function*, que es la descripción en lenguaje computacional de un bloque de *Simulink*, que puede ser escrita en diferentes lenguajes de programación, teniendo para ello plantillas prediseñadas dónde se inserta el código que se requiera para su función, en este caso se seleccionó el lenguaje C por su previo conocimiento, siendo compilada como un archivo MEX usando la utilería *mex* del *Matlab*.

El controlador realiza la tarea de controlar la apertura y cierre de la servo válvula instalada que provocará las fugas, así como la operación y arranque de la bomba hidráulica que provoca el flujo en la tubería, siendo controlados desde el mismo modelo de control en Simulink.

La tarjeta cuenta con canales de salida analógica para corriente y voltaje, con rangos de 4-20 mA y 0-10 V, los cuales fueron usados como salidas a dichos dispositivos en el sistema de tubería. La programación del driver se hizo de manera que se ingresara el porcentaje de la salida de corriente máxima de los canales de la tarjeta desde el modelo de control en *Simulink*.

El código del programa que realizará la comunicación entre *Matlab* y la tarjeta *PC-AO-2DC* se programó en lenguaje *C*. Este tipo de archivos definen el block de *S-Function* que proveerá información acerca del modelo a *Simulink* durante la simulación. Como la

ejecución procede, *Simulink*, el resolvedor de ecuaciones y el archivo MEX interactúan para ejecutar una tarea específica. Estas tareas incluyen definir condiciones iniciales y características del bloque y sus salidas.

Simulink interactúa con el archivo *C* MEX que define la *S-Funtion* llamando métodos que esta implementa. Cada método ejecuta una tarea predeterminada. La *S-Function* es libre de ejecutar la tarea de acuerdo a la funcionabilidad implementada.

Configuración de la Tarjeta

El usuario puede configurar la Tarjeta *PC-AO-2DC* de *National Instruments*, usando una dirección base en el rango de 100 a 3E0 hexadecimal definida desde el sistema operativo. Esta tarjeta es completamente compatible con especificaciones estándares Intel-Microsoft en la administración de sus recursos (National Instruments, 1996 PC-AO-2DC User Manual).

Los registros de la tarjeta se dividen en tres grupos: Configuración y Calibración que controla la operación de la tarjeta; Salida Analógica que controla los 12 bits DACs y el grupo de bits para I/O Digital.

Particularmente es de interés el grupo de bits encargado de las salidas analógicas de la tarjeta, DAC0 controla la salida analógica del Canal 0. DAC1 controla la salida analógica del Canal 1. Estos DAC deben escribirse individualmente.

El DAC en cada salida analógica genera un voltaje proporcional a la entrada V_{ref} multiplicado por el código digital cargado en el DAC. Los DACs tienen una interfaz de 8 bits. Cada DAC puede ser cargado con un código digital de 12 bits escribiendo primero el bajo byte y después el alto byte. El voltaje a la salida es actualizado por la PC-AO-2DC en sus pines de conexión DAC0OUT y DAC1OUT tan pronto es escrito el alto byte.

Direcciones	Base + 04 (hex)	Carga DAC0 bajo byte
	Base + 05 (hex)	Carga DAC0 alto byte
	Base + 06 (hex)	Carga DAC1 bajo byte
	Base + 07 (hex)	Carga DAC1 alto byte

DAC*x*H

7	6	5	4	3	2	1	0
D15	D14	D13	D12	D11	D10	D9	D8
{——————Sign Extension Bits——————}							

DAC*x*L

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0

Código del Controlador

Simulink cuenta con plantillas predeterminadas que facilitan la programación de las *S-Functions* de acuerdo al lenguaje de programación seleccionado. A continuación se muestra el cuerpo de nuestra función que utiliza la plantilla para *C*, describiendo cada parte de ella:

```
#define S_FUNCTION_NAME lectura                          /* Nombre de la s-function */
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"                                    /* Llamada a librerías de c */
#include "stdio.h"
#include "conio.h"
#include "math.h"

/*Los tamaños de arreglo son usados por Simulink para determinar las características del bloque S-function (numero de entradas, salidas, estados, etc.)*/
static void mdlInitializeSizes(SimStruct *S)
{
   ssSetNumSFcnParams(S, 0);                             /*Número de parámetros esperados */
   if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
   return;
   }
   ssSetNumContStates(S, 0);                             /* Número de estados continuos */
   ssSetNumDiscStates(S, 0);                             /* Número de estados discretos */
   if (!ssSetNumInputPorts(S, 1)) return;
   ssSetInputPortWidth(S, 0, 1);
   ssSetInputPortDirectFeedThrough(S, 0, 0);

   if (!ssSetNumOutputPorts(S,1)) return;
   ssSetOutputPortWidth(S, 0, 1);
   ssSetOutputPortWidth(S, 1, 1);
   /*ssSetOutputPortWidth(S, 2, 1);*/

   ssSetNumSampleTimes(S, 1);                            /*Número de tiempos de muestreo */

   ssSetNumRWork(       S, 0);            // number of real work vector elements
   ssSetNumIWork(       S, 0);            // number of integer work vector elements
   ssSetNumPWork(       S, 0);            // number of pointer work vector elements
   ssSetNumModes(       S, 0);            // number of mode work vector elements
   ssSetNumNonsampledZCs( S, 0);          // number of nonsampled zero crossings

   ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);   // general options (SS_OPTION_xx)

} //* end mdlInitializeSizes

/*Esta function es usada para especificar el tiempo de muestreo de la S-function. Si la S-function es continua debe especificarse  un tiempo de muestreo de 0.0*/
static void mdlInitializeSampleTimes(SimStruct *S)
```

```
{
   // Register one pair for each sample time
   ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
   ssSetOffsetTime(S, 0, 0.0);
} // end mdlInitializeSampleTimes
/*                                          Function:                                          mdlOutputs
=====================================================
 /* En esta función, se calculan las salidas de la S-function. Generalmente las salidas estan situadas en
el vector de salidas   ssGetOutputPortSignal.*/
static void mdlOutputs(SimStruct *S, int_T tid)
{
   int       LB = 0x124;                                    /*Dirección hex bajo byte*/
   int       HB = 0x125;                                    /*Dirección hex alto byte*/
   float      A1 = 2.1;                                     /*Factor bajo byte*/
   float      A2 = .15;                                     /*Factor alto byte*/
   int       LD = 0x00;                                     /*Valor inicial bajo byte*/
   int       HD = 0x00;                                     /*Valor inicial alto byte*/
   int_T      i;                                            /*Contador*/
   int_T      a;                                            /*Porcentaje de operación*/

   InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
   real_T  *salida = ssGetOutputPortRealSignal(S,0);
   int_T     width = ssGetOutputPortWidth(S,0);

   for (i=0; i<width; i++){
      salida[i] = *uPtrs[i];
      a = salida[i];
      LD = a * A1;
      HD = a * A2;
      outp(LB,LD);
      outp(HB,HD);
}
} // end mdlOutputs

static void mdlTerminate(SimStruct *S)
{}

/*=============================*
 * Required S-function trailer *
 *=============================*/

#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

Dentro de la investigación realizada en el Instituto de Ingeniería de la UNAM, es frecuente que se necesite desarrollar controladores para establecer comunicación entre los Software utilizados (particularmente *Matlab*), y algún dispositivo que no esté soportado por éste u otro paquete de cómputo.

Hasta el día de hoy, cada nuevo investigador que necesitase diseñar uno de estos programas, se encontraba con que no existe información documental sobre el proceso de desarrollo de controladores, por lo que perdía mucho tiempo no sólo en la elaboración del mismo, sino además en la investigación necesaria para iniciar dicha tarea.

Es por eso que aquí se presenta esta pequeña guía con la idea de ahorrar mucho de ese tiempo y proveer a quien necesite de valiosa ayuda pues no es cosa sencilla.

MÈTODO:

1. Tener a la mano toda la documentación necesaria en el proceso, tal como el manual de usuario y de programador si existiera de la tarjeta de Adquisición de Datos, los cuales se pueden obtener en los sitios Web de los fabricantes. De la misma manera es importante tener los manuales de usuario de Matlab y Simulink, así como la Guía de Programación de *S-Functions* de *Matlab*.
2. Definir la función y alcances a realizar por el controlador, es decir, la tarea en específico que realizará y para lo cual se desarrollará. Lograr una comprensión total de la forma de operar de la tarjeta en lo referente a su envío y recepción de datos, es decir, la secuencia que sigue así como el tratamiento que da a la información a nivel de registros.
3. Analizar las diferentes opciones que ofrece *Matlab* para desarrollar *S-Functions*, desde lenguajes de programación hasta el uso de archivos auxiliares llamados "Head". Para este punto es recomendable contar ya con un esbozo de el controlador, es decir, tener definidas ya cuáles serán sus entradas, salidas y el tratamiento u operaciones que se realizarán con los datos, esto para tomar las decisiones que permitan desarrollar un controlador que sea lo más sencillo posible y que cubra nuestras necesidades.
4. Es importante conocer lo básico del sistema operativo con el que se trabaja, y de qué manera permite la programación de registros, pues es a este nivel que trabajará el controlador, aunque en su mayoría son compatibles con *Matlab*. Es recomendable hacer uso de técnicas de programación como los diagramas de flujo y el pseudo código para el desarrollo del controlador.
5. Ya seleccionado el tipo de *S-Function* que se usará, e identificadas las diferentes secciones que la integran, se procede a la programación de la misma, donde se programarán las tareas que realizará.
6. Ya que se tenga el programa, éste se integrará en un modelo de *Simulink* utilizando el bloque llamado *S-function*, al que se asignará el nombre del programa del controlador. Dentro de este modelo será donde el controlador realizará sus funciones, así que será aquí donde se implementen sus entradas y salidas entre *Matlab* y la computadora.

Apéndice B

Operación del sistema de tubería

Aquí se da una serie de pasos para la correcta operación del sistema de tubería y equipo que permita obtener pruebas con mejores resultados:

1) Energización del sistema con protección adecuada para el cuidado de los sistemas electrónicos.
2) Revisión del nivel del tanque buscando que sea el mismo para todas las pruebas y que la descarga a la salida quede por descubierta, así como la limpieza del agua.
3) Inicio de operación del inversor ya sea de forma manual o a través de la interfaz con la computadora sin toma de lecturas esperando que la circulación de líquido se estabilice por varios minutos.
4) Verificación del equipo en estado permanente además de la instalación de cualquier equipo eventual y su cableado.
5) Inicio de la prueba en forma, es decir, funcionamiento de la tubería y adquisición de datos.
6) Vigilancia del experimento durante todo su desarrollo por medio del osciloscopio de *Matlab* donde se registran las lecturas en tiempo real, lo que puede incluir provocar una fuga en un tiempo específico.
7) Fin de la prueba, terminando operación del equipo del inversor y bomba eléctrica, cierre de servo válvula, guardado de datos en archivo .mat y almacenamiento del mismo para su análisis.
8) Corte de alimentación de energía.

REFERENCIAS

Chaudry, M.C (1979). *Applied Hydraulic Transients*. Van Nortrand Reinhold Co, New York, USA.

Dietel, (2001*) Cómo programar en C/C++*. Mc Graw Hill, México

Gertler, J. (1991). Analytical redundancy methods in fault detection and isolation, survey and synthesis. Fault Detection Supervision and Safety for Technical Processes,

Mitsubishi , Mitsubishi Electric, Manual de Usuario para el Inversor digital. USA.

National Instruments (1996*), Lab-PC 1200 User Manual*. Ed National instruments, USA.

National Instruments (1996), *PC-AO-2DC User Manual*. Ed. National Instruments, USA.

Nicholas, Edward (1999), *Leak detection on pipelines in unsteady Flow*. Revista Scientific Software Núm. 6 pp 13-20, USA

Ogata, Katshuilko (2001), *Teoría de Control Automático*, Ed. Mc Graw Hill, México.

Patton, R., Frank, P., and Clark, R. (2000). Issues of Fault Diagnosis for Dynamic Systems. Springer-Verlag, London, Great Britain.

Real Time Windows Target, *User´s Guide*. Editado por Math Works, USA

Rosas Jaimes, Oscar (2001), Localización borrosa de fugas en tuberías. Tesis FI UNAM, México.

Panametrics, *Two Chanel Transport Flowmeter*. Manual editado por Panametric Inc.

Schreier, G., Frank, P., and Kratz, F. (1998). Stability discussion of an observer for a class of nonlinear systems. Annual Meeting of the IAR, Mulhouse, France

Simulink, *Using Simulink*,. Editado por Math Works. USA

Simulink, *Writing* S-*Functions*. Editado por Math Works. USA

Thomas, P. (1999). Simulation of industrial processes for control engineers. Butterworth Heinemann, Great Britain.

Verde, Cristina y Carrera, Rolando (2001*), Localizador automático de fugas en un ducto*. Revista Ingeniería Hidráulica en México, México.

Verde, cristina y Carrrera, rolando (1999), *Configuración y calibración de la planta piloto para detección fugas en tuberías*. Reporte II UNAM, México.

Visairo Cruz, Nancy, (2004*), Detección y Localización de Fugas en un ducto*. Reporte de Investigación.II UNAM, México

En Internet
http://www.automatas.org/redes/scadas.htm
http://www.aurigacorp.com.pe/aurigatech/paginas/ai_automatizacion.html