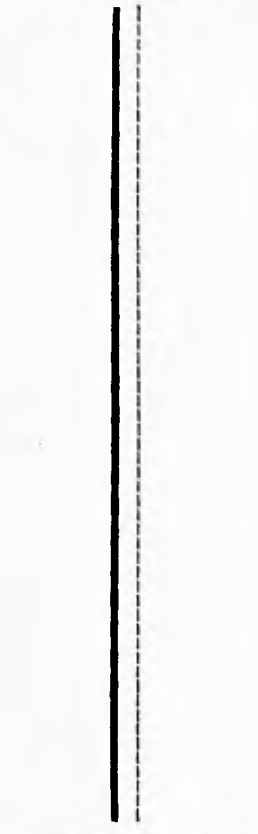
UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA



DETECCION DE FALLAS EN UN SISTEMA DINAMICO BASADO EN LA REDUNDANCIA ANALITICA

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A :

CESAR AUGUSTO MENDOZA SERRANO

DIRECTOR DE TESIS: DR. YU TANG XU

MEXICO, D. F. 1995

DEDICADO A:

Mis padres:

Porque todo lo que soy y lo que pueda llegar a ser se lo debo a ellos.
Por sus sacrificios, apoyo y amor.

Mis hermanos:

Por la comprensión y amor que me han brindado.
Porque junto a ellos haré realidad los ideales de mis padres.

Mis abuelitos:

Porque donde quiera que se encuentren sé que me apoyan en todo momento.

Mis primos:

Por todos los momentos que hemos pasado juntos y por aceptarme tal como soy.

Mis amigos:

Por su amistad y apoyo incondicional.

AGRADECIMIENTOS

A Dios por haberme dado el espíritu necesario para terminar mi carrera universitaria.

A la Universidad Nacional Autónoma de México por brindarme la oportunidad de contribuir a la grandeza de mi patria.

Al Dr. Yu Tang Xu por su paciencia, comprensión y amistad.

A todos mis profesores por su asesoría y motivación.

A la DGAPA por el apoyo al proyecto No. IN300593.

A la Fundación UNAM por el apoyo en el Programa de Iniciación Temprana a la Investigación y Docencia.

A la Fundación General Electric por haberme dado todas las herramientas necesarias para continuar con mis estudios.

A todos aquellos que ayudaron a la realización del presente trabajo.

INDICE

1. INTRODUCCION.

La detección de fallas en un sistema de control se ha convertido en un aspecto de gran importancia dentro de los procesos modernos de automatización ya que provee de los elementos necesarios de tolerancia, confiabilidad y seguridad para cualquier sistema ingenieril.

Actualmente existen complejos sistemas de control ampliamente utilizados en la industria y consisten de cientos de partes que trabajan interdependientemente y que están sujetos a un malfuncionamiento o alguna falla en forma individual, es decir, en solo una de sus partes.

Una falla del sistema en forma total puede resultar en grandes pérdidas económicas o en peligros para el personal que labora en la planta.

Normalmente, para evitar estos problemas se elabora un plan de mantenimiento con el objetivo de reemplazar las partes muy gastadas o antiguas antes de que funcionen mal o tengan alguna falla. También, paralelo a ello se diseña un esquema de monitoreo el cual detecta una falla si esta ocurre, identifica el malfuncionamiento de un componente y compensa el sistema debido a la falla del componente.

Durante las últimas dos décadas la investigación sobre detección de fallas se ha incrementado considerablemente alrededor del mundo. Este desarrollo fue y sigue siendo estimulado principalmente por la obtención de sistemas de control automáticos cada vez más complejos con algoritmos cada vez más elaborados y con una creciente correspondencia de confiabilidad y seguridad de los sistemas de control.

Es así como surge una nueva metodología para la detección de fallas y es la redundancia analítica en la cual se hace uso del modelo matemático del sistema o de partes del mismo. Varios enfoques de detección de fallas que utilizan redundancia analítica han sido reportados. El número de los diferentes enfoques, sin embargo, puede ser concentrado en sólo algunos conceptos básicos, entre ellos se encuentran[1,2]: "el filtro de detección" de Beard & Jones (1973), "la prueba de innovación usando un único filtro Kalman" de Mehra and Peshon (1971) o "bancos de filtros Kalman u observadores Luemberger" de Clark et al. (1975), Montgomery & Caglayan (1974), "el enfoque del espacio de paridad de Deckert el al.

(1975)", "la técnica de estimación de parámetros" de Kitamura (1980) y aplicación de sistemas expertos de Tzafestas (1989) .

El objetivo de este trabajo es aplicar el enfoque de redundancia analítica para la detección de fallas en un sistema de control, esto es, a través del análisis del comportamiento del residuo vectorial obtenido por medio de diferencias entre el sistema real y el modelo de estimaciones.

Se busca encontrar nuevos resultados acerca del uso de los estimadores[5] como opción dentro del enfoque de redundancia analítica aplicados a un sistema hidráulico con características semejantes al analizado en este trabajo.

Esta tesis está organizada de la siguiente manera: Este primer capítulo presenta el objetivo y el problema a analizar en este trabajo en forma general, así como los trabajos reportados en el área.

En el segundo capítulo se discuten los diferentes métodos y enfoques de detección de fallas. Se estudian diferentes criterios en cuanto a la selección del mejor enfoque de detección de fallas. Además, se describe detalladamente el enfoque que se usará en éste trabajo así como también se hacen algunas consideraciones sobre la robustez..

En el tercer capítulo se describe el sistema físico así como los procesos de calibración utilizados. Se obtiene el modelo matemático y se le aplica el algoritmo de redundancia analítica.

En el capítulo cuatro se presentan sistemáticamente los resultados obtenidos apoyados en diferentes gráficas. En el capitulo cinco se hace el correspondiente análisis de los resultados obtenidos y en el capítulo seis se hacen las conclusiones pertinentes.

En el capítulo siete se agrega además el material bibliográfico utilizado así como también se incluye en forma de apéndices el listado de los programas fuente (diseño y aplicación) para la implementación del algoritmo de redundancia analítica . Finalmente en el apéndice C se ilustra la planta por medio de fotografías.

2. ESQUEMAS DE DETECCION DE FALLAS.

2.1 Concepto de falla.

Existen muchas definiciones para el término de falla, pero en el contexto de este trabajo, el concepto de falla en un sistema de control será entendido como cualquier clase de malfuncionamiento en la dinámica actual del sistema o planta, que conlleva a una anomalía inaceptable para la operación general del sistema.

Tales malfuncionamientos pueden ocurrir en los sensores (instrumentos), en los actuadores o en los componentes del proceso.

Con respecto a los diferentes sectores donde las fallas pueden ocurrir se pueden distinguir enfoques hacia la detección de fallas en instrumentos, hacia los actuadores y hacia los componentes.

2.2 Subsistemas de una planta bajo un esquema de detección de fallas.

Para la detección de fallas en un sistema de control se pueden considerar en una planta automatizada tres tipos de subsistemas principales[1]:

a) Actuadores.
b) Estructura principal (o proceso).
c) Instrumentación o sensores.

Los actuadores son los mecanismos que se encargan de ejecutar las señales provenientes de la parte de control, en motores, calderas, hornos, etc. En la estructura principal o proceso se lleva a cabo el análisis y realización de algoritmos de control. Finalmente la instrumentación consiste de varios sensores o transductores los cuales se encargan de obtener las señales del sistema para que .estas sean procesadas.

En general, un esquema de monitoreo de fallas está especialmente diseñado para detectar y corregir cualquier malfucionamiento en alguno de los tres subsistemas.

Los esquemas principales buscan detectar fallas en los sensores, y una vez detectados suelen ser corregidos por técnicas de swictcheo electrónico, no requiriendo la reconfiguración de las partes mecánicas del sistema. La compensación de la falla en los actuadores es un tanto más difícil, pués no solo se tienen que redireccionar señales eléctricas, sino también se debe componer o sustituir algún elemento mecánico. Mientras que la compensación en la estructura principal es aún más complicada.

Los algoritmos de detección de fallas son generalmente implementados en procesadores digitales. Estos algoritmos consisten principalmente en técnicas de procesamiento de señales.

Evidentemente que la confiabilidad del sistema en su totalidad depende de la computadora (o procesador) y de los componentes físicos del sistema. Se puede asumir que es más difícil que ocurra una falla en una computadora que en los sensores, actuadores y componentes del sistema. Esto se debe principalmente a que la computadora puede permanecer en ambientes físicos bien acondicionados para su operación mientras que los componentes físicos del sistema están expuestos a ambientes más desfavorables.

De acuerdo a lo anterior, en este trabajo se asumirá que la falla existente ocurre en el sistema o planta y que la computadora es confiable en su totalidad.

2.3 Enfoques para la detección y aislamiento de fallas.

Para poder seleccionar el enfoque que se daría para la detección de fallas en este trabajo fue necesario conocer los principales enfoques que con este fin se han hecho[3]. Estos recaen en dos grupos:

a) Enfoques que no hacen uso del modelo matemático de la planta.
b) Enfoques que hacen uso del modelo matemático de la planta.

2.3.1 Enfoques que no hacen uso del modelo matemático de la planta.

Como su nombre lo indica, estos no requieren el modelo matemático del sistema físico, sino que hacen uso de instalación de más sensores, fijación de límites, análisis frecuencial etc. A continuación se describen algunos métodos:

2.3.1.1 Revisión por límite.

En este método se realizan mediciones en la planta y se comparan con límites prefijados. Un sobrepaso de dichos límites indica una situación de falla. En muchos sistemas existen dos niveles de límites. El primero sirve como aviso de alerta y el segundo indica una acción de emergencia.

2.3.1.2 Instalación de sensores especiales.

Este método es similar al anterior, en tanto que usan sensores de límites pero que básicamente realizan un chequeo de límites en hardware como pueden ser en límites de temperatura o presión o aquellos que miden variables especiales como el sonido, elongación, vibración, etc.

2.3.1.3 Análisis de frecuencia.

Este tipo de modelo también supone realizar mediciones en las plantas y es un tanto menos general. Algunas plantas tienen un espectro de frecuencia determinado bajo condiciones normales de operación y cualquier desviación de dicho espectro sugiere alguna anormalidad.

Para el aislamiento de la falla, se puede considerar que ciertos tipos de falla tienen una característica especial en el espectro de frecuencias, lo que puede ser utilizado para su aislamiento.

2.3.1.4 Sistemas expertos.

Este método se caracteriza por evaluar los "síntomas" obtenidos por hardware o software. El sistema consiste usualmente por combinaciones de ciertas reglas lógicas, como por ejemplo:

"si" *síntoma* "y" *síntoma* "entonces" *conclusión*, donde cada conclusión puede a su vez servir como un síntoma para la siguiente regla hasta llegar a la conclusión final, lo que vendría siendo la falla específica.

El método tiene la característica especial de que puede trabajar por hardware o software y además interactuar con un operador humano

2.3.1.5 Redundancia por hardware.

Este tipo de método es de los más utilizados actualmente y que tiene cierto peso en este trabajo en lo que respecta a la idea de redundancia.

La redundancia por hardware, consiste en que varios elementos de hardware como pueden ser actuadores, sensores de medición, componentes del proceso, etc. son usualmente distribuidos espacialmente en el sistema para dar protección contra daños localizados.

Tales esquemas operan típicamente a través de configuraciones de redundancia triple o cuatruple para verificar la consistencia de salidas redundantes (o medidas). Así, por ejemplo tres o más sensores se instalan para medir la misma variable cuando se podría instalar solo un sensor si este fuera totalmente confiable.

Las señales de estos sensores son monitoreadas por un circuito lógico el cual ignora las pequeñas diferencias en dichas señales tales como el ruido eletrónico, tolerancias de manufactura, errores inherentes en los instrumentos, pero que logra detectar si un sensor está fallando cuando su señal se desvía demasiado del valor promedio que proveen los otros sensores (asumiendo claro, que los otros sensores mantienen una pequeña diferencia entre sus valores).

Debido a su sencillez, este tipo de enfoque es muy utilizado y es esencial en sistemas de control que requieren altísimos niveles de seguridad y confiabilidad, como es el caso de naves espaciales, aviones, plantas químicas y nucleares, entre otros.

Desafortunadamente este enfoque presenta ciertas dificultades y desventajas. Entre las principales se encuentra el costo extra que representa instalar más de un sensor para medir una variable.

Además, requiere espacio adicional para colocar el equipo. Así por ejemplo, en un avión de guerra se sacrificaría espacio para instalar sensores de redundancia enfocados a la detección de fallas en vez de ser utilizado para armamento, combustible o equipos de orientación.

Por otra parte, se supone que los sensores "redundantes" tienen una espectativa de vida similar y por lo que cuando un sensor, perteneciente a un grupo de sensores, falla es probable que los otros tengan algún malfuncionamiento en cualquier momento.

2.3.2 Enfoques que hacen uso del modelo matemático de la planta.

Debido a las dificultades que presenta la detección de fallas desde un enfoque de redundancia por hardware, se concibió en los inicios de la década de los setentas nuevas formas de detectar algún malfuncionamiento en el sistema o planta que evitaran dichas dificultades y que mejoraran la confiabilidad del sistema en su totalidad.

Estos nuevos enfoques están basados en la idea de que diferentes sensores miden diferentes señales y que por lo tanto producen señales totalmente diferentes que pueden ser usados en un esquema de comparación más sofisticado y poder detectar la falla en forma más específica.

El razonamiento para esta idea se basa en el hecho de que aún cuando los sensores miden diferentes variables, todos ellos son consecuencia de una misma dinámica de estado del sistema y por lo tanto estan funcionalmente relacionados.

Este enfoque se le conoce como":redundancia inherente" o "redundancia funcional" para distinguirlos de la redundancia por hardware o redundancia física, pero también son conocidos como "redundancia analítica" y "redundancia artificial".[1]

Dicho enfoque, puede utilizar diferentes técnicas de procesamiento de señales, tales como estimación de estados, filtrado adaptable, variables de umbrales lógicos, teoria de las decisiones estadísticas, operaciones combinacionales, las cuales son ejecutadas en circuitos electrónicos o en computadoras digitales de alta velocidad.

2.3.2.1 Descripción del método utilizado.[2]

El esquema de detección de fallas utilizado en este trabajo se basa principalmente en la obtención de un residuo a través de redundancia analítica, esto es, la diferencia de las salidas del modelo matemático con respecto al comportamiento real de la planta se compara con un valor de umbral para la detección de la falla.

Para ello se debe obtener el modelo matemático de la planta:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}, \mathbf{u})$$

$$\mathbf{y}(t) = g(\mathbf{x}, \mathbf{u})$$

donde:

- **x:** es el vector de estados del sistema
- **u:** es el vector de entradas.
- **y:** es el vector de salidas.

el cual se linealiza en un punto de operación (x_o) de acuerdo a la Serie de Tailor obteniendo:

$$\dot{\mathbf{z}}(t) = \mathbf{A}\mathbf{z}(t) + \mathbf{B}\mathbf{u}$$
$$\mathbf{w}(t) = \mathbf{C}\mathbf{z}(t)$$

donde:

$$\mathbf{z} = \mathbf{x} - \mathbf{x}_o$$
$$\mathbf{w} = \mathbf{y} - \mathbf{y}_o$$

Así,el modelo discreto estará dado por:

$$\mathbf{z}(k+1) = \mathbf{A}_d\, \mathbf{z}(k) + \mathbf{B}_d\, \mathbf{u}(k)$$
$$\mathbf{w}(k) = \mathbf{C}\, \mathbf{z}(k)$$

Basado en este, el estimador de estado es:

$$\hat{\mathbf{z}}(k+1) = \mathbf{A}_d\, \mathbf{z}(k) + \mathbf{B}_d\, \mathbf{u}(k) + \mathbf{K}\,(\mathbf{w}(k) - \hat{\mathbf{w}}(k))$$
$$\mathbf{w}(k) = \mathbf{C}\, \hat{\mathbf{z}}\,(k)$$

en donde K es la ganancia del estimador que se obtiene utilizando *Filtros Kalman*[5]. De ésta manera podemos obtener un residuo definido como:

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

que es la diferencia entre las salidas del sistema real y las salidas del sistema estimado. Dicho residuo es necesario filtrarlo para atenuar el ruido que se pueda presentar y lo hacemos a través del siguiente algoritmo:

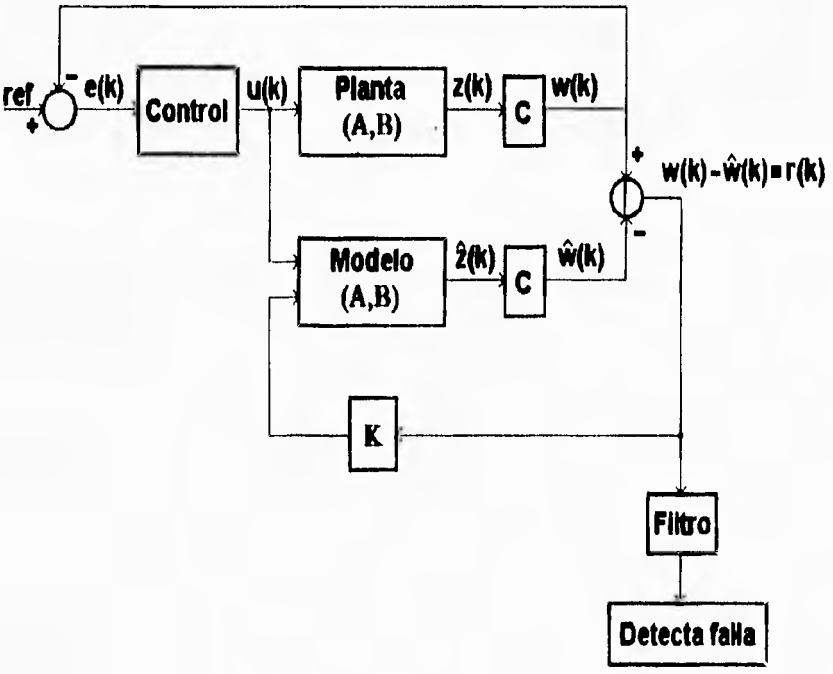
$$\mathbf{r}_m\,(t) = \frac{1}{T} \sum_{k=t-T}^{t} \mathbf{r}(k)$$

siendo T la longitud de la ventana de tiempo. Se observa que $\mathbf{r}_m(t)$ puede obtenerse de la siguiente manera recursiva:

$$\mathbf{r}_m\,(t+1) = \mathbf{r}_m\,(t) + \frac{1}{T}\mathbf{r}(t+1) - \frac{1}{T}\mathbf{r}(t+1-T)$$

Este residuo promedio se compara con un valor de umbral; cuando el residuo promedio sobrepasa dicho valor de umbral, se dice que se ha detectado una falla. La selección del valor de umbral se hace en base a la sensibilidad y porcentaje de falsas alarmas que se deseen.

El siguiente diagrama ilustra el método descrito anteriormente:



2.4 Criterios para la selección de un buen esquema de detección fallas.

Los esquemas de detección de fallas en tiempo real usando redundancia funcional o analítica aun no son muy comunes y las técnicas de diseño para hacerlos robustos no han sido desarrollados totalmente.

Algunos de los criterios para poder catalogar a un esquema de detección de fallas como bueno o malo se presentan a continuación. Dichos criterios pueden tener un mayor o menor peso de acuerdo a las necesidades requeridas por el sistema de control.[1]

a) Rapidez de detección.
b) Sensibilidad a fallas incipientes.
c) Porcentaje de falsas alarmas.
d) Fallas sin detectar.
e) Identificación de fallas incorrectas.

2.4.1 Rapidez de detección.

La rapidez en la detección de una falla puede llegar a ser de vital importancia en aquellos sistemas en donde si la falla persiste por más de una fracción de segundo puede dañar grandemente el sistema de control. Un ejemplo de ello es en las aplicaciones aeroespaciales, donde la rapidez de detección juega un papel importantísimo.

Por otra parte, en algunos sistemas de control se sacrifica la rapidez en la detección de la falla para evitar falsas alarmas o poder detectar fallas menores. Ello dependerá directamente en el criterio del diseñador.

2.4.2 Sensibilidad a una falla incipiente.

Cuando se habla de una falla incipiente, se está hablando de una falla que es muy pequeña o se desarrolla muy lentamente. En algunos sistemas, la detección de este tipo de fallas suele ser de mayor importancia, principalmente en aquellos en que se busca mejorar las operaciones de mantenimiento de las plantas a través de la temprana detección del equipo gastado, en donde la rapidez en la detección pasa a un segundo termino con respecto a la sensibilidad. A veces la rapidez en la detección y la sensibilidad son igualmente importantes, en esos casos se requieren esquemas de detección más complejos, quizá que usen redundancia física (hardware) y analítica al mismo tiempo.

2.4.3 Porcentaje de falsas alarmas.

Estos son generalmente indicativos de una operación pobre del sistema de detección de fallas. Un porcentaje pequeño de falsas alarmas durante una operación normal del sistema es inaceptable porque rápidamente resta confiabilidad al esquema de detección. Sin embargo, un sistema de detección de fallas puede llegar a tener un porcentaje de falsas alarmas aceptable cuando se está monitoreando sistemas o plantas con un comportamiento inusual, por ejemplo si se lleva a la planta o sistema de control a ambientes diferentes al acostumbrado.

En otras aplicaciones, pequeñas fallas pueden ser tan serias que es preferible reaccionar a falsas alarmas, reeplazando el componente dudoso por uno de buen funcionamiento y evitar así un deterioro en el sistema de control.

2.4.4 Fallas sin detectar.

Exiten ocasiones en que ocurre una falla y esta no es detectada. Esto puede ser aceptable en algunos esquemas de detección de fallas en donde por ejemplo la falla ocurre en un sensor sin importancia, pero por otro lado, la omisión en la detección de un malfuncionamiento puede ser demasiado grave si este ocurre en un sensor fundamental del sistema de control.

2.4.5 Identificación de fallas incorrectas.

Otro malfuncionamiento del sistema de detección es cuando ocurre una falla y se identifica al componente incorrecto. Esto está relacionado con la no detección y con las falsas alarmas.

Al haber detectado la falla pero identificando mal el componente, entonces la compensación se hará sobre una falla errónea por lo tanto en un componente que quizá se encuentre en perfectas condiciones, ocasionando terribles consecuencias en el sistema.

2.5 Robustez.

La robustez, vista desde el punto de vista de detección de fallas, es el rango de operación en que puede trabajar satisfactoriamente el esquema de detección, bajo las condiciones del sistema para el que fue

diseñado; por lo que la robustez debe ser medida con respecto a las variables asociadas con dichas condiciones.

Para los fines de este trabajo, es importante establecer ciertos rangos de robustez, por ello es necesario conocer algunos de los problemas y dificultades que se pueden tener.

2.5.1 Problemas en la robustez debido a los parámetros de la planta.

El mayor problema en el campo de la robustez de esquemas de detección de fallas es causado por incertidumbres en los parámetros físicos de la planta. En estos esquemas, las señales redundantes son combinadas lógicamente para identificar fallas que son generadas dentro del subsistema de detección, usando por ejemplo, técnicas de estimación de estados usualmente basados en teoria de sistemas lineales.

Los estimadores de estado son esencialmente modelos matemáticos de la dinámica de una planta, por lo que dependen de las características físicas de la planta, como pueden ser: propiedades de la masa, momentos de inercia, parámetros de circuitos eléctricos y electrónicos, fuerzas aerodinámicas e hidrodinámicas, propiedades de trasferencia de calor, etc.

Sin embargo, aunque el modelo matemático sea preciso (por ejemplo de un sistema lineal e invariante , que es el sistema dinámico más simple), siempre habrá algún parámetro que solo sea una aproximación. Esto es muy común en flujos de fluidos y transferencia de calor. Consecuentemente, siempre tendrán un error y por ello siempre existirá algún peligro de falsas alarmas o quizá ni siquiera se detecte la falla.

2.5.2 Problemas de Robustez debido a no-linealidades o dinámicas desconocidas.

En general, todas las plantas físicas son no lineales, aun cuando muchas de ellas tengan un comportamiento casi lineal.

Un esquema de detección de fallas, basado en modelos lineales, puede ser satisfactorio en los casos en que el comportamiento de la planta es lineal en cierto rango y bajo ciertas condiciones y cuando una planta no lineal es linealizada a través de ciertos métodos matemáticos.

Sin embargo, fuera de los rangos especificados, la planta presenta no-linealidades que no pueden ser modeladas exactamente por el esquema de detección de fallas y que podrían ser interpretadas como fallas.

2.5.3 Problemas en la robustez debido a perturbaciones y ruido.

Todos los sistemas dinámicos están sujetos a entradas no consideradas por el diseñador. Estas entradas, conocidas como perturbaciones, son generalmente aleatorias y generadas por el medio ambiente en que se encuentra el sistema.

Además los sensores, no logran liberarse en su totalidad, del ruido electrónico que se superpone a las señales que producen. Dicho ruido es también aleatorio y originado en una fuente diferente.

2.5.4 Problemas de robustez debido a las diferentes tipos de falla.

Un componente dado puede fallar de diferentes formas. Por ejemplo, un sensor puede sufrir un cambio en su factor de escala, una polarización puede ser no constante, una no-linealidad debido al uso o a la fricción, histéresis, exceso de ruido.

Algunos esquemas de detección de fallas se diseñan para detectar diferentes tipos de fallas, pero si la falla ocurre y no se encuentra dentro del repertorio del esquema de detección de fallas éste no será detectada, originando un problema de robustez.

Podemos decir que es preferible un esquema que detecte cualquier tipo y que indique el componente que falla aun cuando no especifique que tipo de falla es.

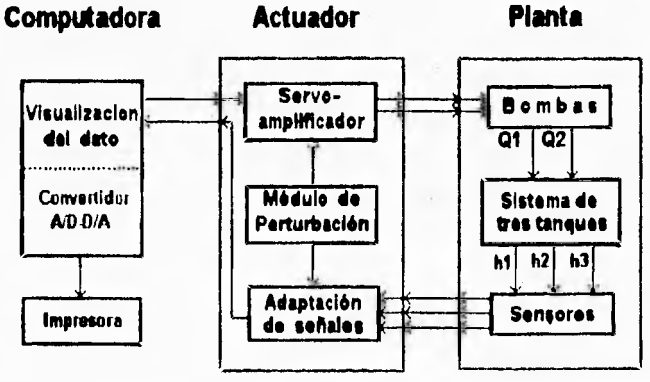
3. APLICACION DEL METODO.

3.1 Equipo de los tres tanques (Sistema DTS200).

El equipo de los tres tanques (sistema DTS-200) fue comprado en diciembre de 1993 a Amira mbH, empresa alemana, y fue instalado en el Laboratorio de Control de la División de Estudios de Posgrado de la Facultad de Ingeniería en junio de 1994. Consta principalmente de tres tanques cilíndricos apoyados en una base que sirve como contenedor de agua. Dos bombas extraen el agua de dicho contenedor y la depositan en los tanques de los extremos. Cada tanque está provisto de un sensor piezoresistivo para obtener la lectura de nivel.

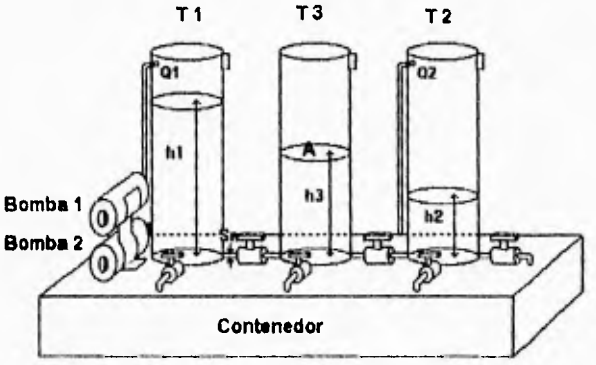
El equipo también incluye una tarjeta de adquisición de datos analógico/digital y digital/analógico para PC con software de demostración y un módulo de perturbaciones eléctricas.

El equipo está dividido en tres secciones funcionales: la planta, el actuador y la computadora[10], mostrados en la siguiente figura:



3.1.1 La planta.

La estructura de la planta puede verse en la figura siguiente:



Como se observa en la gráfica la planta consiste de tres tanques cilíndricos de acrílico (tanque 1: T1, tanque2: T2, tanque 3: T3) con una sección transversal "A". Los tanques se conectan en forma serial a través de tubos de conexión y válvulas mecánicas. También se conectan a un contenedor de agua común, igualmente con tubos de conexión y válvulas mecánicas.

El área transversal de dichos tubos de conexión (Sn) es el mismo para todos. Las bombas 1 y 2 toman agua del contenedor para alimentar a los tanques de los extremos (tanque 1 y tanque 2).

La altura máxima de los tanques es de 62 cm. Si el nivel de agua rebasara dicho límite por algún mal diseño o mal funcionamiento del sistema de control las bombas se apagaran automáticamente.

Se ha definido entonces:

Area transversal de los cilindros (A) = 0.0154 m^2
Area transversal de los tubos de conexión (Sn) = 0.00005 m^2
Altura máxima = 62 cm. (+/- 1 cm).
Flujo máximo de las bombas ($Q1máx$ y $Q2máx$) = 100 ml/seg.

Las bombas son manejados por corriente directa. Las válvulas mecánicas sirven para simular fugas en el sistema o para mantener un flujo

entre los tanques. Cada válvula tiene un coeficiente de flujo (a_{zn}) el cuál es calculado con el programa de demostración del equipo.

Los flujos de las bombas van a servir como entradas de la planta, mientras que las alturas medidas van a ser tomadas como las salidas de nuestro sistema.

Para medir las alturas se utilizan sensores piezoresistivos de presión diferencial y están colocados en la parte superior de cada tanque. El sensor 1 es el colocado en el tanque 1 y el sensor 2 en el tanque 2 y el sensor 3 es el colocado en el tanque 3.

3.1.2 Equipo de computo.

El equipo de computo utilizado fungirá como el controlador del sistema. También en él se incluye el esquema de detección de fallas utilizado en este trabajo.

Se supone de antemano que el equipo de computo es totalmente confiable, que se encuentra en un ambiente de trabajo adecuado para su correcto funcionamiento pués el esquema de detección de fallas así lo requiere.

Se requiere entonces de una computadora con procesador 486 y que trabaje a 33 Mhz, esto con el fin de que el tiempo de muestreo del sistema de control no varíe. Sin embargo es posible utilizar otra computadora que trabaje a diferente velocidad únicamente haciendo una modificación en el software elaborado específicamente para este trabajo.

Es indispensable contar con una tarjeta de adquisición de datos. La utilizada para este trabajo es una DAC 2614 de conversión analógica/digital y digital/analógica.

Para el diseño y el análisis de detección de fallas se utiliza el paquete de MATLAB. Además, para poder llevar a cabo los objetivos de esta tesis fue necesario elaborar un software en lenguaje C, apoyándose en funciones del mismo lenguaje provistas por el fabricante.

El software desarrollado sirve para implementar y sintonizar el controlador. También, dentro de los avances de este trabajo están la implementación en tiempo real de los algoritmos de detección de fallas.

3.1.3 Actuador.

El actuador consta de tres partes principales:

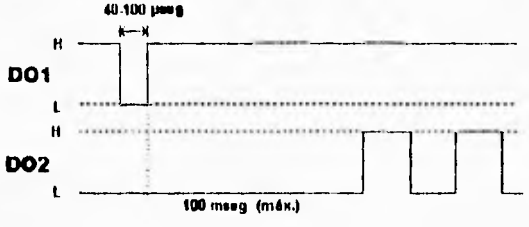
a) Servoamplificador (etapa de liberación de etapa de potencia).
b) Módulo de perturbaciones.
c) Unidad de adaptación de señales.

3.1.3.1 Servoamplificador.

El servoamplificador recibe la señal procesada por la computadora y le da la potencia necesaria para que las bombas puedan funcionar adecuadamente acorde con la señal procesada. El servoamplificador tiene uno de las aspectos de mayor importancia para el funcionamiento del equipo y es la liberación de la etapa de potencia.

La liberación de la etapa de potencia constituyó una de las partes más difíciles de este trabajo. Esta etapa brinda seguridad al equipo en caso de alguna falla en el programa, pués hace que las bombas se detengan inmediatamente.

Existen dos maneras para liberar la etapa de potencia[10] y son las siguientes: a) la primera, es moviendo el jumper JP4 del actuador en dirección del panel de enfrente (ver manual) b) en forma computacional. Esta manera requiere mandar dos señales a los pines 11 y 13 del conector de entradas y salidas digitales del actuador. El software los identifica como DO1 y DO2 respectivamente. Las señales deben seguir el patrón del diagrama de tiempos mostrado en la figura siguiente:



Esto es, estando DO1 en nivel lógico alto, debe darse un pulso de nivel lógico bajo de una duración de alrededor de 40-100 mseg, mientras que DO2 estando en nivel lógico bajo, terminando el pulso en DO1, debe recibir en no más de 100 mseg. un tren de pulsos en un rango de frecuencia de 10 Hz a 1 KHz. Este tren de pulsos debe mantenerse en ese pin todo el tiempo en que se desea que las bombas esten trabajando, independientemente de la potencia de las mismas.

3.1.3.2 Módulo de perturbación.

El actuador está equipado con interruptores y potenciómetros los cuales pueden simular fallas en los niveles de medición. Con el interruptor se simula una falla total en el sensor, indicando un nivel de altura cero; con los potenciómetros es posible simular una falla en el sensor en forma parcial a través de varios escalamientos de señal que van de cero al cien por ciento.

También existen otros 2 potenciómetros que permiten simular defectos en las bombas. Para hacer esto, la señal de control puede ser reducida hasta el cero porcentual de su valor original, lo cual es equivalente a decrementar el valor del flujo de la bomba.

3.1.3.3 Unidad de adaptación de señales.

El propósito de la unidad de adaptación de señales es adaptar los niveles de voltaje de la planta y del convertidor. De esta manera, los voltajes de salida de los sensores son adaptados a la máxima resolución de los convertidores analógico/digital y por otra parte el rango de los voltajes de salida de los convertidores digiral/analógico son adaptados al servoamplificador de la bomba correspondiente.

3.2 Calibración

Era importante dentro de los objetivos de este trabajo tener valores bastante precisos y para ello fue necesario calibrar a conciencia el equipo de los tres tanques.

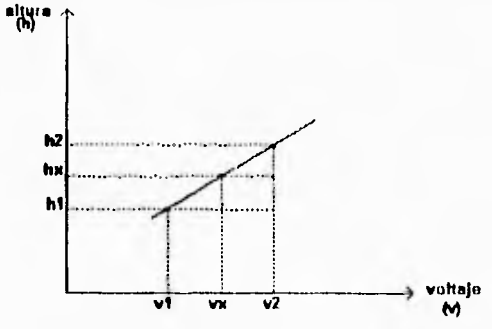
Para lograr esto, se debió tomar en cuenta que cada sensor piezoresistivo de cada tanque proporciona una señal de voltaje diferente

entre si, aunque esten midiendo el mismo nivel de agua, por lo que cada sensor se debe calibrar por separado.

La técnica utilizada fue la siguiente: A cada tanque se le suministraba agua cada cinco centimetros y en cada uno de esos incrementos se tomaba la lectura en voltaje de cada sensor. Este proceso se hizo diez veces y los promedios de lectura en voltaje de cada incremento se muestran en la tabla siguiente:

Altura centimetros	Sensor tanque 1 voltaje	Sensor tanque 2 voltaje	Sensor tanque 3 voltaje
0	9.222	8.924	9.212
5	7.874	7.570	7.848
10	6.465	6.159	6.427
15	5.070	4.753	5.001
20	3.674	3.343	3.597
25	2.285	1.953	2.196
30	0.895	0.557	0.793
35	- 0.490	- 0.830	- 0.602
40	- 1.870	- 2.210	- 1.989
45	- 3.245	- 3.589	- 3.368
50	- 4.618	- 4.953	- 4.751
55	- 6.001	- 6.331	- 6,141
60	-.7.318	- 7.650	- 7.458

Para asignar el valor de voltaje a su correspondiente nivel de altura se hace una interpolación basada en el siguiente algoritmo:



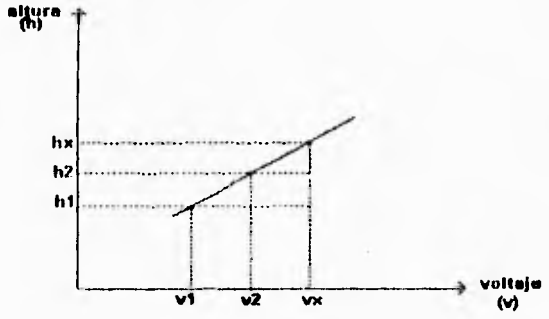
De la figura anterior, por la teoria de los triángulos semejantes se deduce que:

$$\frac{V_2 - V_1}{h_2 - h_1} = \frac{V_x - V_1}{h_x - h_1}$$

si despejamos h_x de la ecuación anterior obtenemos la altura correspondiente al voltaje mandado por el sensor:

$$h_x = \frac{V_x - V_1}{V_2 - V_1}(h_2 - h_1) + h_1$$

Ocasionalmente el valor del nivel de altura no se encuentra intermedio entre los valores de la tabla y se encuentra en alguno de los extremos. Si el valor del voltaje es mayor al voltaje que corresponde a una altura cero entonces un voltaje mayor a ese será siempre de cero. Si el voltaje es menor al correspondiente a la altura máxima 60 cm. su aplica otro algoritmo, de acuerdo a la siguiente figura:



De acuerdo a la figura anterior, y según la teoria de triángulos semejantes, el valor de la altura para un voltaje menor a la altura máxima es:

$$h_x = \frac{V_x - V_2}{V_2 - V_1}(h_2 - h_1) + h_2$$

Para la expresión anterior en particular, la máxima altura es de 60 cm. por lo que $h_2 = 60$ cm., la diferencia h_2-h_1 es igual a 5 para todos los casos. Verifíquese en la tabla. Si el incremento promedio de voltaje entre

cada altura de *5* cm. es de *1.318* se puede asumir entonces que : V_2-V_1 = *1.318*. Simplificando el algoritmo tenemos:

$$h_x = 60 + (V_2 - V_1)\left(\frac{5}{1.318}\right)$$

Con el algoritmo anterior podemos obtener valores extrapolados de niveles de altura.

Además de calibrar las señales que proporcionan los sensores, es importante también conocer en forma exacta las señales que el controlador manda a las bombas, esto es, parametrizar los flujos (señal procesada) a voltajes (señal que se manda a las bombas).

Con este fin, se ingenió un algoritmo, de tal manera que se pudiera realizar la parametrización de flujos a voltajes con solo medir los niveles de altura, aprovechando que ya se podía tener un valor confiable de altura.

Sea entonces,

$$f = \frac{v}{t}$$

donde f : flujo (cm^3 por segundo)
v : volúmen (cm^3).
t : tiempo (segundos)

Ahora, en nuestro caso el volúmen es:

$$v = r^2 \pi \Delta h$$

donde "r" es el radio del tanque cilindrico y Δh es el incremento de nivel de altura que existe cada 5 segundos. De acuerdo a esto, se tiene que:

$$f = \frac{\pi r^2 \Delta h}{5}$$

despejando Δh y considerando también que el radio es de 7 cm. obtenemos la expresión siguiente:

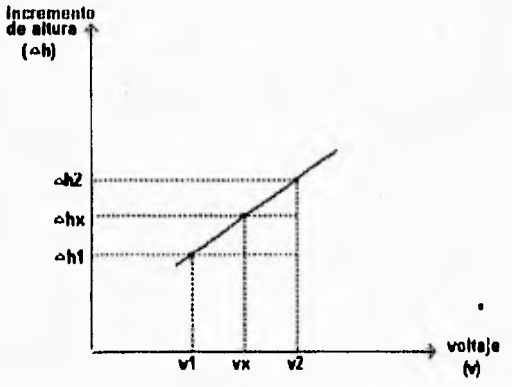
$$\Delta h = \frac{5f}{49\pi}$$

Con la expresión anterior es posible relacionar la señal de flujo que es mandada por el controlador hacia las bombas. y establecer un parámetro entre flujo y voltaje aplicado a las bombas.

En la tabla siguiente, para cada voltaje indicado en la tabla se obtuvo un incremento de altura correspondiente, así que cuando a las bombas se le aplica un voltaje de -10 volts se obtiene un flujo máximo pues el incremento de altura fue el mayor.

Voltaje (volts)	Δh tanque 1 (cm)	Δh tanque 2 (cm)
10	0	0
9	0.01	0.09
8	0.212	0.183
7	0.381	0.398
6	0.599	0.628
5	0.768	0.823
4	1.011	1.010
3	1.184	1.229
2	1.403	1.439
1	1.522	1.658
0	1.800	1.820
- 1	1.813	2.041
- 2	2.135	2.342
- 3	2.146	2.421
- 4	2.366	2.712
- 5	2.498	2.805
- 6	2.650	2.946
- 7	2.769	3.102
- 8	2.950	3.274
- 9	3.115	3.414
-10	3.252	3.233

Para obtener el algoritmo, nos apoyamos en la gráfica siguiente:



Los algoritmos de interpolación propuestos son algebraicamente y en su concepción los mismos que los desarrollados para la calibración de las alturas, por lo que solamente se escribirá la expresión para obtener el voltaje aplicado a las bombas dependiendo de la señal de flujo procesada en el controlador:

$$V_x = \frac{\Delta h_x - \Delta h_2}{\Delta h_2 - \Delta h_1}(V_2 - V_1) - V_2$$

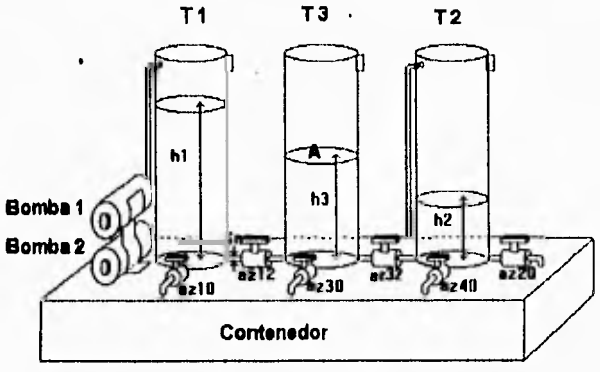
En la anterior expresión los Δh son los valores de una sola columna de la tabla anterior y se puede emplear para ambas columnas.

3.3 MODELO MATEMATICO.

El modelo matemático es indispensable dentro del enfoque de la redundancia analítica para la detección de fallas en un sistema de control. En el capítulo anterior se dió una especificación general del sistema, las partes principales y la calibración del mismo. Ahora, és esta sección se deducirá el modelo matemático de la planta.

3.3.1 Especificación de variables y parámetros.

Para poder obtener un modelo matemático preciso es importante especificar todas y cada una de las variables y parámetros que intervienen en el sistema. Para ello, se utilizará en el diagrama esquemático de la planta siguiente:

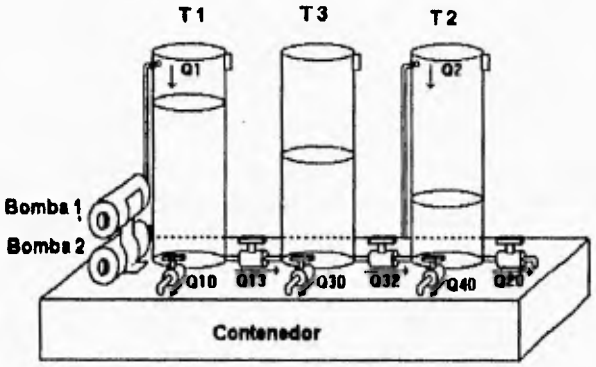


´ donde:

h1 : altura tanque 1
h2 : altura tanque 2
h3 : altura tanque 3
T1 : tanque 1
T2 : tanque 2
T3 : tanque 3
Q1 : flujo de alimentación de la bomba 1
Q2 : flujo de alimentacion de la bomba 2
Sn : sección transversal de los tubos de conexión a tanques y reservorio.
A : sección transversal del cilindro.
az13 : coeficiente de flujo de tanque 1 a 3.
az32 : coeficiente de flujo de tanque 3 a 2.
az10 : coeficiente de flujo de tanque 1 a reservorio.
az20 : coeficiente de flujo de tanque 2 a reservorio.
az30 : coeficiente de flujo de tanque 3 a reservorio.
az40 : coeficiente de flujo de tanque 2 a reservorio.:

Para los flujos existentes entre cada tanque y del tanque al contenedor también se les denotó de alguna manera y se les dió una

dirección la cual fue tomada por convención, y estas están especificadas en el diagrama siguiente:



3.3.2 Modelo matemático en función de los flujos.

Sabiendo que el flujo másico que entra por cualquier tubo de conexión es el mismo que el que sale y tomando en cuenta la dirección de los flujos especificados en el diagrama anterior podemos observar que el cambio de volúmen con respecto al tiempo es:

$$A\frac{dh_1}{dt} = Q_1 - Q_{13} - Q_{10}$$

$$A\frac{dh_2}{dt} = Q_{32} + Q_2 - Q_{20} - Q_{40}$$

$$A\frac{dh_3}{dt} = Q_{13} - Q_{32} - Q_{30}$$

y que en variables de estado es:

$$A\frac{d}{dt}\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \begin{bmatrix} -Q_{13} - Q_{10} \\ Q_{32} - Q_{20} - Q_{40} \\ Q_{13} - Q_{32} - Q_{30} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} \qquad \textbf{(3.1)}$$

es decir:

$$A\frac{d}{dt}\mathbf{h} = \mathbf{A(Q)} + \mathbf{B(Q_i)} \qquad \textbf{(3.2)}$$

donde:

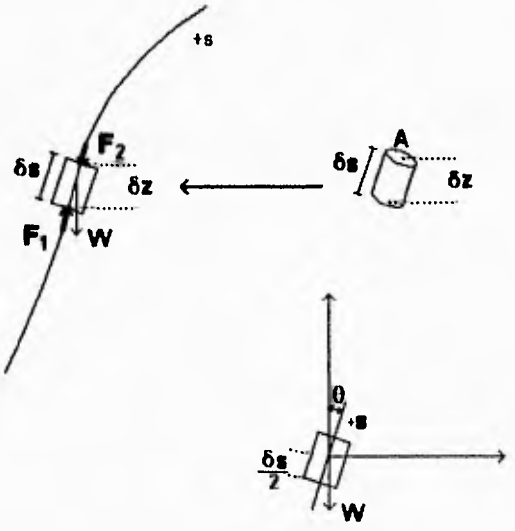
$\mathbf{h}$	: Vector de alturas
$\mathbf{A(Q)}$	: Matriz que depende del flujo entre tanques
$\mathbf{B(Q_i)}$	: Matriz que depende de los flujos de entrada

3.3.2.1 Modelado de los flujos entre tanques y contenedor.

Debido a que el algoritmo de detección de fallas se basa en los niveles de altura es conveniente poner la ecuación en variables de estado en función de las alturas.

Para lograr esto es necesario obtener el modelo matemático de los flujos entre tanques y contenedor. Esto es lo que trata esta sección.

Considerando los flujos de los tubos de conexion en una sola dirección y en una línea de corrimiento "s". Si una sección volumétrica de estos flujos es tomada y modelada en un diagrama de fuerzas que actuan en ella[8,9].



De acuerdo al diagrama anterior tenemos:

$$\delta z = \delta s \cos\theta \qquad \textbf{(3.3)}$$

Debido a la 1a. Ley de Newton, un cambio de velocidad debe estar asociado a una fuerza, se espera entonces que la presión de un fluido cambie de un punto a otro, entonces:

$$\begin{aligned} \textstyle\sum Fs &= mas \\ F_1 - F_2 - W\cos\theta &= mas \end{aligned} \qquad \textbf{(3.4)}$$

Como la fuerza aplicada al fluído es $F=PA$ donde P es la presión ejercida sobre el fluído y A es el área transeversal de la sección volumétrica, como lo indica la figura. Con lo que respecta a un fluído en movimiento tenemos que:

$$F_1 = (P - P_{dinamica1})\,\delta A \quad \text{y} \quad F_2 = (P + P_{dinamica2})\,\delta A \qquad \textbf{(3.5)}$$

Es evidente que la presión ejercida sobre el fluido (P), es la presión estática en ambos lados de la sección volumétrica tomada. Si sustituímos la ecuación 3.5 en la ecuación 3.4 tenemos:

$$\delta A(P - P_{dinamica1}) - \delta A(P + P_{dinamica2}) - W\cos\theta = mas \qquad \textbf{(3.6)}$$

Ya que $P_1 = P_2 = P$, por ser la misma presión que se aplica a la sección volumétrica y además:

$$P_{dinamica1} = P_{dinamica2} = \frac{\partial P}{\partial s}\frac{\delta s}{2} \qquad \textbf{(3.7)}$$

a lo largo de la línea de flujo "s" entonces al sustituir en la ecuación 3.6 se tendrá:

$$\delta AP - \delta AP - \frac{\partial P}{\partial s}\frac{\delta s}{2}\delta A - \frac{\partial P}{\partial s}\frac{\delta s}{2}\delta A - W\cos\theta = mas$$

y como $W = \vartheta g \delta A \delta s$ porque $W = mg$ y $m = \delta A \delta s \vartheta$ (ϑ : densidad) al sustituir esto en la ecuación anterior:

$$-\frac{\partial P}{\partial s}\delta s \delta A - \vartheta g \delta A \delta s \cos\theta = \delta A \delta s \vartheta\, as$$

recordando que: $\delta z = \delta s \cos\theta$ obtenemos que:

$$-\frac{\partial P}{\partial s}\delta s\delta A - \vartheta g\delta A\delta z - \delta A\delta s\vartheta a_s = 0$$

si dividimos ambos lados de la ecuación anterior entre $\delta A\delta s$ y consideramos que:

$$\frac{\partial z}{\partial s} = \frac{\delta z}{\delta s}$$

veremos que ésta se transforma a la siguiente expresión:

$$-\frac{\partial P}{\partial s} - \vartheta g\frac{\partial z}{\partial s} - \vartheta a_s = 0 \qquad \textbf{(3.8)}$$

ahora si tomamos la aceleración a lo largo de "s" de acuerdo a:

$$a_s = \frac{dv}{dt} = \frac{\partial v}{\partial s}\frac{\partial s}{\partial t} + \frac{\partial v}{\partial t}\frac{ds}{dt}$$

observamos que la velocidad es: $v = \frac{\partial s}{\partial t}$ que nos permite simplificar la expresión para la aceleración a:

$$a_s = \frac{\partial v}{\partial s}v + \frac{\partial v}{\partial t}$$

que al sustituir en la ecuación 3.8:

$$-\frac{\partial P}{\partial s} - \vartheta g\frac{\partial z}{\partial s} - \vartheta\left(\frac{\partial v}{\partial s}v + \frac{\partial v}{\partial t}\right) = 0$$

Y como en éste trabajo se va a considerar un flujo permanente

$$\frac{\partial v}{\partial t} = 0$$

simplifica a :

$$-\frac{\partial P}{\partial s} - \vartheta g\frac{\partial z}{\partial s} - \vartheta v\frac{\partial v}{\partial s} = 0 \qquad \textbf{(3.9)}$$

Ya que en la ecuación anterior se manejan parciales con respecto a una variable, entonces es posible hacer la siguiente simplificación:

$$\vartheta g dz + dP + \vartheta v dv = 0$$

Si integramos y dividimos entre la densidad (ϑ) y además a esta última la consideramos constante debido a que el equipo que estamos utilizando maneja agua destilada. Entonces de un punto 1 a un punto 2 de la línea de flujo se tiene:

$$\int g dz + \int \frac{dP}{\vartheta} + \int v dv = 0$$

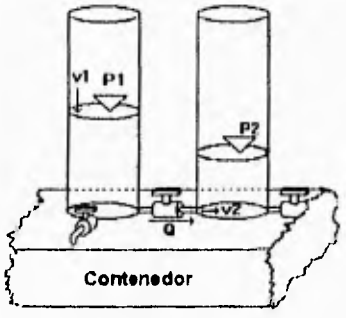
$$gz\Big|_1^2 + \frac{P}{\vartheta}\Big|_1^2 + \frac{v^2}{2}\Big|_1^2 = C$$

El primer término a la izquierda de la expresión anterior corresponde a la energía potencial, el segundo al trabajo de flujo y el tercero a la energía cinética. Evaluando en la expresión anterior:

$$g(z_2 - z_1) + \frac{P_2 - P_1}{\vartheta} + \frac{v_2^2 - v_1^2}{2} = 0 \qquad \textbf{(3.10)}$$

Que en realidad viene siendo de alguna manera una extensión simplificada de la ecuación de Bernoulli[8,9]:

Para el caso de los tubos conectados entre dos tanques:
En la figura siguiente consideramos a $P_2 = P_1 = Patm \approx 0$ y ya habíamos supuesto que: ϑ = *constante*. Además como la velocidad a la que baja el nivel de agua del tanque es mucho menor con respecto a la velocidad con la que sale el flujo de agua asumimos que: $v_2 >> v_1$ y $v_1 \approx 0$



con lo que se tiene:

$$v_2 = \sqrt{(z_1 - z_2)2g}$$

además ya que el flujo es la velocidad por el área transversal del tubo de interconexión se tiene que:

$$Q = v\,Sn$$

y al sustituir la velocidad:

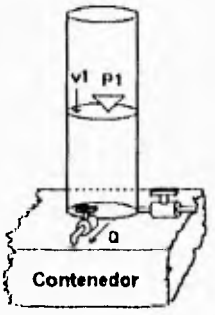
$$Q = Sn\sqrt{(z_1 - z_2)2g}$$

La expresión anterior sirve para calcular el flujo de agua en los tubos de conexión de los tanques y es conocida como la forma generalizada de Torricelli[8,9]. Ahora, debido a que la dirección del flujo va a depender de las alturas de los tanques en cuestión se le agrega la función signo que nos indicará la dirección del flujo. Además como el flujo no es ideal, es decir, existen pérdidas debido al rozamiento es necesario considerar un coeficiente de flujo, con lo que finalmente obtenemos:

$$Q = a\,Sn\;signo(z_1 - z_2)\sqrt{2g(z_1 - z_2)} \qquad \textbf{(3.11)}$$

La ecuación anterior se usará para calcular los flujos entre los tubos de conexión de los tanques.

Para el caso de un tubo de conexión entre tanque y contenedor:



De acuerdo a la figura anterior es fácil darse cuenta que una altura es cero (z_2 0) lo que reduce la ecuación 3.12 a:

$$Q = a_z S_n \sqrt{2g z_1} \qquad \textbf{(3.12)}$$

Ahora solo sustituiremos las alturas z por su correspondiente altura, según el tanque que se este analizando junto con el coeficiente de flujo adecuado. Por lo tanto, los flujos por los tubos de conexión son los siguientes:

$$Q_{13} = a_{z13}\, S_n\, signo(h_1 - h_3)\sqrt{|h_1 - h_3|2g}$$
$$Q_{32} = a_{z32}\, S_n\, signo(h_3 - h_2)\sqrt{|h_3 - h_2|2g}$$
$$Q_{10} = a_{z10}\, S_n \sqrt{2gh_1}$$
$$Q_{20} = a_{z20}\, S_n \sqrt{2gh_2}$$
$$Q_{30} = a_{z30}\, S_n \sqrt{2gh_3}$$
$$Q_{40} = a_{z40}\, S_n \sqrt{2gh_2}$$

finalmente al sustituir los flujos:

$$A\frac{d}{dt}\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \begin{bmatrix} -a_{z13}\, S_n\, signo(h_1 - h_3)\sqrt{|h_1 - h_3|2g} - a_{z10}\, S_n \sqrt{2gh_1} \\ a_{z32}\, S_n\, signo(h_3 - h_2)\sqrt{|h_3 - h_2|2g} - a_{z20}\, S_n \sqrt{2gh_2} - a_{z40}\, S_n \sqrt{2gh_2} \\ a_{z13}\, S_n\, signo(h_1 - h_3)\sqrt{|h_1 - h_3|2g} - a_{z32}\, S_n\, signo(h_3 - h_2)\sqrt{|h_3 - h_2|2g} - a_{z30}\, S_n \sqrt{2gh_3} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix}$$

(3.14)

Esta última ecuación es el modelo matemático del sistema de los tres tanques en función de las alturas.

3.3.3 Modelo matemático simplificado orientado a detección de fallas.

En el modelo matemático obtenido en la sección anterior se han considerado todos los flujos que pueden existir en la planta. Con el objetivo de poder aplicar el esquema de detección de fallas, algunos flujos se deben discriminar, considerando las válvulas cerradas, de tal manera que cuando alguna de ellas se abra simule una falla en el sistema.

Si suponemos que las válvulas que producen los flujos Q_{10} (válvula 1), Q_{30} (válvula 3) y Q_{40} (válvula 2) están cerradas implica que dichos flujos son cero, por lo que la ecuación de flujos queda reducida a:

$$A\frac{d}{dt}\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \begin{bmatrix} -Q_{13} \\ Q_{32} - Q_{20} \\ Q_{13} - Q_{32} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} \qquad \textbf{(3.15)}$$

y en función de las alturas:

$$A\frac{d}{dt}\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \begin{bmatrix} -a_{z13}\, Sn\ signo(h_1 - h_3)\sqrt{|h_1 - h_3|2g} \\ a_{z32}\, Sn\ signo(h_3 - h_2)\sqrt{|h_3 - h_2|2g} - a_{z20}\, Sn\sqrt{2gh_2} \\ a_{z13}\, Sn\ signo(h_1 - h_3)\sqrt{|h_1 - h_3|2g} - a_{z32}\, Sn\ signo(h_3 - h_2)\sqrt{|h_3 - h_2|2g} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix}$$

(3.16)

3.3.4 Linealización.

En ésta sección se hace la linealización[4] del sistema modelado en base a varias consideraciones importantes enfocadas al control y detección de fallas Se obtuvo un modelo matemático de un sistema dinámico con características no lineales. Aunque es posible hacer el análisis en forma no lineal, en este trabajo se trabajará en forma lineal alrededor de un punto.

En general, el sistema dinámico no lineal obtenido lo podemos representar de la siguiente manera:

$$\frac{d\mathbf{h}}{dt} = f(\mathbf{h}, \mathbf{Q})$$

donde:

$\mathbf{h}$	: Vector de estados (alturas).
$\mathbf{Q}$	: Vector de entradas (flujos de las bombas).
$f(\mathbf{h}, \mathbf{Q})$	: Función vector

El objetivo de esta sección es, entonces, poder representar el sistema de la siguiente manera:

$$\dot{\mathbf{h}} = \mathbf{A}\mathbf{h} + \mathbf{B}\mathbf{Q}$$

Como se puede observar la matriz $\mathbf{A}$ está en función de las alturas (estados), por lo que se hace necesario aplicar un método de

linealización para poder separar el vector **h** de la matriz **A**. Mientras que la matriz **B** y el vector de flujos **Q** se encuentran tal como los deseamos.

Si linealizamos la ecuación en variables de estado a través de una expansion en Series de Tailor alrededor de un punto de operación ($\mathbf{h_o}$,$\mathbf{Q_o}$) donde $\mathbf{h_o}$ es el vector de alturas alrededor del cual se linealiza y $\mathbf{Q_o}$ el vector de flujos de entrada alrededor del cual también linealizamos. Por tanto,

$$\dot{h}_i = f_i(\mathbf{h_o}, \mathbf{Q_o}) + \sum_{j=1}^{n} \left.\frac{\partial f_i(\mathbf{h},\mathbf{Q})}{\partial h_j}\right|_{h=h_o} (h_j - h_{oj}) + \sum_{j=1}^{p} \left.\frac{\partial f_i(\mathbf{h},\mathbf{Q})}{\partial Q_j}\right|_{Q=Q_o} (Q_j - Q_{oj})$$

Descartamos los términos de la expansión mayores a 1 y tendremos sencillamente:

$$\dot{\mathbf{h}} - f(\mathbf{h_o}, \mathbf{Q_o}) = \left.\frac{\partial f(\mathbf{h},\mathbf{Q})}{\partial \mathbf{h}}\right|_{h=h_o} (\mathbf{h} - \mathbf{h_o}) + \left.\frac{\partial f(\mathbf{h},\mathbf{Q})}{\partial \mathbf{Q}}\right|_{Q=Q_o} (\mathbf{Q} - \mathbf{Q_o}) \qquad \textbf{(3.17)}$$

Anteriormente se había mencionado que $f(\mathbf{h_o},\mathbf{Q_o})$ es un punto de operación fijo (constante), es decir, es el flujo en estado estacionario $\mathbf{Q_o}$ proporcionado por la ley de control que hace que se mantengan los estados deseados $\mathbf{h_o}$.

Proponemos el siguiente cambio de variables:

$$\mathbf{z} = \mathbf{h} - \mathbf{h_o}$$

$$\dot{\mathbf{z}} = \dot{\mathbf{h}} - \dot{\mathbf{h}}_o = \dot{\mathbf{h}} - f(\mathbf{h_o}, \mathbf{Q_o})$$

$$\mathbf{A} = \left.\frac{\partial f(\mathbf{h},\mathbf{Q})}{\partial \mathbf{h}}\right|_{h=h_o}$$

$$\mathbf{B} = \left.\frac{\partial f(\mathbf{h},\mathbf{Q})}{\partial \mathbf{Q}}\right|_{Q=Q_o}$$

$$\mathbf{v} = \mathbf{Q} - \mathbf{Q_o}$$

Sustituyendo en la ecuación 3.17:

$$\dot{\mathbf{z}} = \mathbf{Az} + \mathbf{Bv}$$

Es evidente que para aplicar el esquema de detección de fallas bajo el enfoque que se le está dando a este trabajo, se necesita hacer el análisis en un punto fijo de operación. Un enfoque diferente sería utilizar un punto variable de operación lo que implicaría utilizar esquemas de detección y control no lineales, que no son parte de esta tesis.

Hay que recordar también, que por convención de la dirección de los flujos, esta se cumple si $h_1 > h_3 > h_2$. Una consideración diferente haría replantear la dirección de los flujos, así como el modelo matemático, que algebraicamente, sería cuestión de un cambio de signos en la matriz **A**.

Aplicando $\mathbf{A} = \left.\frac{\partial f(\mathbf{h},\mathbf{Q})}{\partial \mathbf{h}}\right|_{h=h_0}$ para obtener la matriz **A** tendremos:

$$\mathbf{A} = \begin{bmatrix} -\frac{\partial}{\partial h_1}(\mathbf{Q}_{13}) & -\frac{\partial}{\partial h_2}(\mathbf{Q}_{13}) & -\frac{\partial}{\partial h_3}(\mathbf{Q}_{13}) \\ \frac{\partial}{\partial h_1}(\mathbf{Q}_{32}-\mathbf{Q}_{20}) & \frac{\partial}{\partial h_2}(\mathbf{Q}_{32}-\mathbf{Q}_{20}) & \frac{\partial}{\partial h_3}(\mathbf{Q}_{32}-\mathbf{Q}_{20}) \\ \frac{\partial}{\partial h_1}(\mathbf{Q}_{13}-\mathbf{Q}_{32}) & \frac{\partial}{\partial h_2}(\mathbf{Q}_{13}-\mathbf{Q}_{32}) & \frac{\partial}{\partial h_3}(\mathbf{Q}_{13}-\mathbf{Q}_{32}) \end{bmatrix}$$

Obteniendo cada uno de los elementos de la matriz:

$$-\frac{\partial}{\partial h_1}(\mathbf{Q}_{13}) = -\frac{\partial}{\partial h_1}(az_{13}\,Sn\sqrt{2g(h_1-h_3)}) = -az_{13}\,Sn\frac{g}{\sqrt{2g(h_1-h_3)}}$$

$$-\frac{\partial}{\partial h_2}(\mathbf{Q}_{13}) = 0$$

$$-\frac{\partial}{\partial h_3}(\mathbf{Q}_{13}) = -\frac{\partial}{\partial h_3}(az_{13}\,Sn\sqrt{2g(h_1-h_3)}) = az_{13}\,Sn\frac{g}{\sqrt{2g(h_1-h_3)}}$$

$$\frac{\partial}{\partial h_1}(\mathbf{Q}_{32}-\mathbf{Q}_{20}) = 0$$

$$\frac{\partial}{\partial h_2}(\mathbf{Q}_{32}-\mathbf{Q}_{20}) = \frac{\partial}{\partial h_2}(az_{32}\,Sn\sqrt{2g(h_3-h_2)} - az_{20}\,Sn\sqrt{2gh_2})$$

$$= -az_{32}\,Sn\frac{g}{\sqrt{2g(h_3-h_2)}} - az_{20}\,Sn\frac{g}{\sqrt{2gh_2}}$$

$$\frac{\partial}{\partial h_3}(\mathbf{Q}_{32}-\mathbf{Q}_{20}) = \frac{\partial}{\partial h_3}(az_{32}\,Sn\sqrt{2g(h_3-h_2)} - az_{20}\,Sn\sqrt{2gh_2})$$

$$= az_{32}\,Sn\frac{g}{\sqrt{2g(h_3-h_2)}}$$

$$\frac{\partial}{\partial h_1}(\mathbf{Q}_{13}-\mathbf{Q}_{32}) = \frac{\partial}{\partial h_1}(az_{13}\,Sn\sqrt{2g(h_1-h_3)} - az_{32}\,Sn\sqrt{2g(h_3-h_2)})$$

$$= a_{z13}\, Sn \frac{g}{\sqrt{2g(h_1 - h_3)}}$$

$$\frac{\partial}{\partial h_2}(Q_{13} - Q_{32}) = \frac{\partial}{\partial h_2}\left(a_{z13}\, Sn \sqrt{2g(h_1 - h_3)} - a_{z32}\, Sn \sqrt{2g(h_3 - h_2)}\right)$$

$$= a_{z32}\, Sn \frac{g}{\sqrt{2g(h_3 - h_2)}}$$

$$\frac{\partial}{\partial h_3}(Q_{13} - Q_{32}) = \frac{\partial}{\partial h_3}\left(a_{z13}\, Sn \sqrt{2g(h_1 - h_3)} - a_{z32}\, Sn \sqrt{2g(h_3 - h_2)}\right)$$

$$= -a_{z13}\, Sn \frac{g}{\sqrt{2g(h_1 - h_3)}} - a_{z32}\, Sn \frac{g}{\sqrt{2g(h_3 - h_2)}}$$

De ésta manera la matriz A tiene la siguiente forma:

$$\mathbf{A} = \begin{bmatrix} -\dfrac{a_{z13}\, Sng}{\sqrt{2g(h_1 - h_3)}} & 0 & \dfrac{a_{z13}\, Sng}{\sqrt{2g(h_1 - h_3)}} \\ 0 & -\dfrac{a_{z32}\, Sng}{\sqrt{2g(h_3 - h_2)}} - \dfrac{a_{z20}\, Sng}{\sqrt{2gh_2}} & \dfrac{a_{z32}\, Sng}{\sqrt{2g(h_3 - h_2)}} \\ \dfrac{a_{z13}\, Sng}{\sqrt{2g(h_1 - h_3)}} & \dfrac{a_{z32}\, Sng}{\sqrt{2g(h_3 - h_2)}} & -\dfrac{a_{z13}\, Sng}{\sqrt{2g(h_1 - h_3)}} - \dfrac{a_{z32}\, Sng}{\sqrt{2g(h_3 - h_2)}} \end{bmatrix}_{\mathbf{h}=\mathbf{h_o}}$$

Finalmente, la ecuación en variables de estado en forma linealizada es:

$$A\frac{d}{dt}\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \begin{bmatrix} -\dfrac{a_{z13}\, Sng}{\sqrt{2g(h_1 - h_3)}} & 0 & \dfrac{a_{z13}\, Sng}{\sqrt{2g(h_1 - h_3)}} \\ 0 & -\dfrac{a_{z32}\, Sng}{\sqrt{2g(h_3 - h_2)}} - \dfrac{a_{z20}\, Sng}{\sqrt{2gh_2}} & \dfrac{a_{z32}\, Sng}{\sqrt{2g(h_3 - h_2)}} \\ \dfrac{a_{z13}\, Sng}{\sqrt{2g(h_1 - h_3)}} & \dfrac{a_{z32}\, Sng}{\sqrt{2g(h_3 - h_2)}} & -\dfrac{a_{z13}\, Sng}{\sqrt{2g(h_1 - h_3)}} - \dfrac{a_{z32}\, Sng}{\sqrt{2g(h_3 - h_2)}} \end{bmatrix}_{\mathbf{h}=\mathbf{h_o}} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix}$$

(3.18)

Ahora, se sustituyen los valores reales de la planta en la ecuación anterior:

$$A = 0.0154\ m^2$$
$$Sn = 0.00005\ m^2$$

$$g = 9.81\ {}^{m}/_{s^2}$$
$$az13 = 0.4475$$
$$az23 = 0.4342$$
$$az20 = 0.5971$$

Se fijan valores de las alturas alrededor de los cuáles se quiere que opere el esquema de detección de fallas. La selección de dichos valores, tiene la única restricción de $h_1 > h_3 > h_2$ que fue con la que se linealizó. Una selección diferente implicaría la utilización del software de diseño elaborado en éste trabajo. Así

h1 = 40 cm.
h2 = 20 cm.
h3 = 30 cm.

De ésta manera la ecuación de coeficientes constantes es:

$$\frac{d}{dt}\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \begin{bmatrix} -1.01756\,E-2 & 0 & 1.01756\,E-2 \\ 0 & -1.94739\,E-2 & 9.87322\,E-2 \\ 1.01756\,E-2 & 9.87322\,E-2 & -2.00489\,E-2 \end{bmatrix} \cdot \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} + \begin{bmatrix} 64.93506 & 0 \\ 0 & 64.93506 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix}$$

(3.19)

Mientras que el vector de salidas y es:

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}$$

3.3.5 Discretización[5].

Es evidente que al estar trabajando con un sistema de computo se van a manejar señales digitales por lo que resulta indispensable discretizar el sistema obtenido.

De acuerdo a esto, la expresión general para la solución de la ecuación en variables de estado es[4]:

$$\mathbf{z}(t) = e^{\mathbf{A}(t-t_0)}\mathbf{z}(t_0) + \int_{t_0}^{t} e^{\mathbf{A}(t-\tau)}\mathbf{B}\,\mathbf{u}(\tau)\,d\tau$$

Como se quiere emplear dicha solución en un periodo de muestreo, ésta se modifica un poco (haciendo: $t = kT + T$ y $t_0 = kT$):

$$\mathbf{z}(k\,t+T) = e^{\mathbf{A}T}\mathbf{z}(kT) + \int_{kT}^{kT+T} e^{\mathbf{A}(kT+T-\tau)}\mathbf{B}\,\mathbf{u}(\tau)\,d\tau$$

Este resultado no depende del tipo de retención ya que **u** se especifica en función de su historia de tiempo contínuo $\mathbf{u}(\tau)$ sobre el intervalo de muestreo. Una suposición válida es la de retención de orden cero sin retraso, es decir:

$$\mathbf{u}(\tau) = \mathbf{u}(kT)\ , \qquad kT \le \tau < kT + T$$

Para facilitar la solución se define:

$$\eta = kT + T - \tau$$

por lo tanto:

$$\mathbf{z}(kt+T) = e^{\mathbf{A}T}\mathbf{z}(kT) + \int_{0}^{T} e^{\mathbf{A}\eta}\,d\eta\ \mathbf{B}\,\mathbf{u}(kT)$$

de la ecuación anterior hacemos:

$$\Phi = e^{\mathbf{A}T} = \mathbf{I} + \mathbf{A}T + \frac{\mathbf{A}^2T^2}{2!} + \frac{\mathbf{A}^3T^3}{3!} + \cdots$$

$$\Phi = \mathbf{I} + \mathbf{A}T\Psi \qquad \textbf{(3.20)}$$

donde

$$\Psi = \mathbf{I} + \frac{\mathbf{A}T}{2!} + \frac{\mathbf{A}^2T^2}{3!} + \cdots \qquad \textbf{(3.21)}$$

y además:

$$\Gamma = \int_{0}^{T} e^{\mathbf{A}\eta}\,d\eta\,\mathbf{B}$$

entonces tendremos:

$$\mathbf{z}(kt+T) = \Phi\,\mathbf{z}(kT) + \Gamma\,\mathbf{u}(kT)$$

y como Γ se puede evaluar término por término para dar:

$$\Gamma = \sum_{k=0}^{\infty} \frac{\mathbf{A}^k T^{k+1}}{(k+1)!}\mathbf{B} = \sum_{k=0}^{\infty} \frac{\mathbf{A}^k T^k}{(k+1)!}T\mathbf{B} = \Psi T\,\mathbf{B} \qquad \textbf{(3.22)}$$

Para Ψ sólo tomamos los dos primeros términos, pues los de mayor orden resultan insignificantes, así:

$$\Psi = \mathbf{I} + \frac{\mathbf{A}T}{2!} \qquad \textbf{(3.23)}$$

Fijando el tiempo de muestreo en T= 250 mseg., tiempo adecuado porque el sistema es bastante lento, y sustituyendo las matrices A y B en la ecuación 3.20, ecuación 3.21 y ecuación 3.23 se obtiene:

$$\Psi = \begin{bmatrix} 0.99873 & 1.0465E-6 & 1.26875E-3 \\ 1.0465E-6 & 0.99757 & 1.23009E-3 \\ 1.26875E-3 & 1.23009E-3 & 0.99750 \end{bmatrix}$$

$$\Phi = \begin{bmatrix} 0.99746 & 3.1265E-6 & 2.5343E-3 \\ 3.1265E-6 & 0.99515 & 2.4561E-3 \\ 2.5343E-3 & 2.4561E-3 & 0.995007 \end{bmatrix}$$

$$\Gamma = \begin{bmatrix} 16.2131 & 0 \\ 0 & 16.2131 \\ 2.059E-2 & 1.996E-2 \end{bmatrix}$$

Finalmente nuestra ecuación en variables de estado queda como sigue:

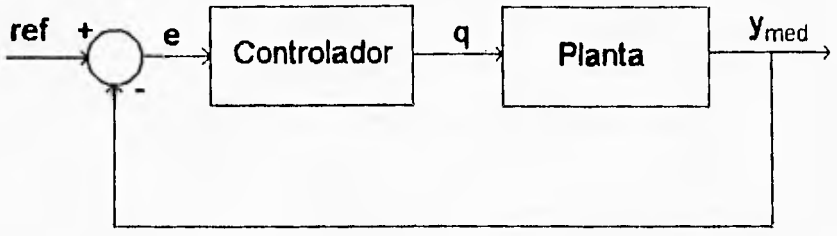
$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}_{k+1} = \begin{bmatrix} 0.99746 & 3.1265E-6 & 2.5343E-3 \\ 3.1265E-6 & 0.99515 & 2.4561E-3 \\ 2.5343E-3 & 2.4561E-3 & 0.995007 \end{bmatrix} \cdot \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}_k + \begin{bmatrix} 16.2131 & 0 \\ 0 & 16.2131 \\ 2.059E-2 & 1.996E-2 \end{bmatrix} \cdot \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix}$$

3.4 Controlador.

Una de las herramientas importantes para poder llevar a cabo el esquema de detección de fallas en éste trabajo fue la implementación del controlador para el sistema. En esta sección se hace el análisis de controlabilidad y observabilidad se describe la forma de implementación del controlador.

3.4.1 Implementación del controlador.

El controlador se basa en la figura esquemática siguiente[6]:



Se usaron las acciones de control proporcional (P), integral (I) y derivativo (D) para conformar las acciones proporcional-integrativo (PI) y proporcional-integral-derivativo (PID) las cuales se aplicaron en los esquemas de detección de fallas. La acción P se usó básicamente para la sintonización[4].

3.4.1.1 Acciones de control.

Se define primeramente la señal de error como:

$$e = y_{ref} - y_{med}$$

donde y_{med} es la altura medida y y_{ref} es la altura de referencia fijada para el controlador.

3.4.1.1.1 Acción proporcional.

Para un control de acción proporcional, la salida de control "q" es:

$$q(t) = K_p\, e(t)$$

donde K_p es la sensibilidad proporcional o ganancia ajustable. En magnitud de transformadas de Laplace tenemos:

$$\frac{Q(s)}{E(s)} = K_p$$

3.4.1.1.2 Acción integral.

En la acción integral, la salida de control q(t) varía proporcionalmente a la señal de error actuante:

$$\frac{dq(t)}{dt} = K_i\, e(t)$$

ó

$$q(t) = K_i \int_0^t e(t)\, dt$$

donde Ki es una constante regulable en que:

$$K_i = \frac{1}{T_i}$$

A Ti se le conoce como constante de integración. En magnitud de transformada de Laplace:

$$\frac{Q(s)}{E(s)} = \frac{K_i}{s} = \frac{1}{T_i\, s}$$

Con respecto a la acción de control derivativo, esta no puede actuar solo, porque sólo es efectiva únicamente durante periodos transitorios.

3.4.1.1.3 Acción de control proporcional-integral (PI).

Esta queda definida por la acción conjunta de las anteriores acciones de control, asi:

$$q(t) = K_p\, e(t) + \frac{K_p}{T_i} \int_0^t e(t)\, dt$$

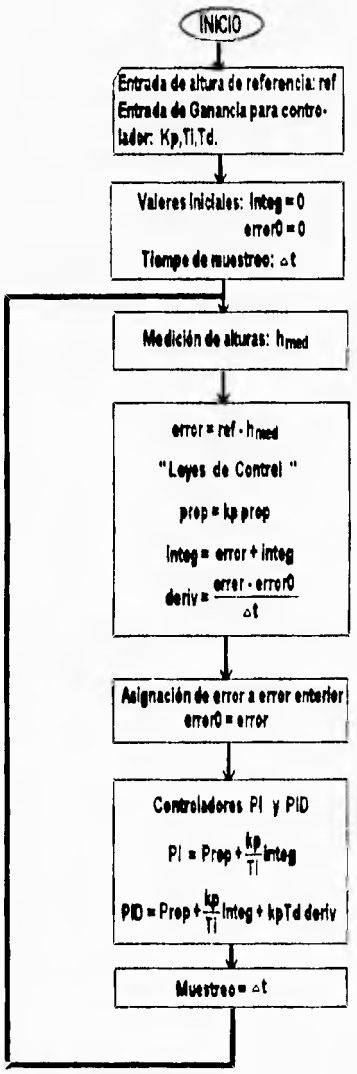
3.4.1.1.4 Acción de control proporcional-integral-derivativa (PID).

Esta incluye todas las acciones de control tratadas anteriormente y se define en la siguiente ecuación:

$$q(t) = K_p\, e(t) + \frac{K_p}{T_i} \int_0^t e(t)\, dt + K_p T_d \frac{de(t)}{dt}$$

Nótese que aquí ya se incluye la acción derivativa afectada por una constante Td se le conoce como tiempo derivativo.

La implementación del controlador proporcional-integral (PI) y proporcional-integral-derivativo (PID) se basó en aplicar los pasos del diagrama siguiente en la computadora:



3.4.2 Sintonización.

Para obtener las ganancias adecuadas para el controlador proporcional-integral (PI) y proporcional-integral-derivativo (PID) de tal

manera que el error del sistema fuese muy cercano a cero se utilizó el método de oscilaciones sostenidas de Ziegler-Nichols (1942).

En este método los parámetros del controlador se calculan en base a las características dinámicas del proceso. Dichas características son representadas por la ganancia límite (de un controlador proporcional) y el periodo de oscilación de las salidas del sistema.

De acuerdo a lo anterior se realimentó el sistema con un controlador proporcional con ganancia Ku hasta que las salidas tenían oscilaciones sostenidas. De ellas se evalúa el periodo de oscilación (Tu) y se aplican las expresiones propuestas por Ziegler-Nichols para éste método.

Para un controlador PI[7]:

$$Kp = \frac{Ku}{2.2} \qquad Ti = \frac{Tu}{1.2}$$

Y para un controlador PID[7]:

$$Kp = \frac{Ku}{1.7} \qquad Ti = \frac{Tu}{2} \qquad Td = \frac{Tu}{8}$$

Posteriormente se hizo un ajuste sobre los valores obtenidos para finalmente obtener:

Para la bomba 1:

	Kp	Ti	Td
PI	**13**	**685**	**-**
PID	**13**	**681**	**0.1**

Y para la bomba 2:

	Kp	Ti	Td
PI	**13**	**293**	**-**
PID	**13**	**291**	**0.09**

3.4.3 Valores de flujos en los puntos de operación.

Es importante conocer el valor de los flujos de entrada en estado estacionario cuando estos se encuentran en los puntos de operación. Esto porque son utilizados al aplicar el algoritmo de detección de fallas.

Para ello, únicamente se observó el flujo que existía cuando se habían alcanzado los puntos de operación durante el funcionamiento normal del controlador. Se determinó así, el valor de los flujos como sigue:

$Q_1 = 31$ ml/seg.

$Q_2 = 28$ ml/seg.

3.5 Ganancia del estimador

La ganancia del estimador es importante para lograr que las características de error entre el modelo real y el modelo estimado tiendad a cero. Para ello se obtiene la matriz de ganancia del estimador (**K**) a través del programa de diseño elaborado en este trabajo y que puede obtener dicha ganancia utilizando *Filtros Kalman* .

Asi la matriz de ganancia del estimador, para los puntos de linealización seleccionados es:

$$\mathbf{K} = \begin{bmatrix} 0.9962 & 0 & 0.0013 \\ 0 & 0.9962 & 0.0012 \\ 0.0013 & 0.0012 & 0.0011 \end{bmatrix}$$

Nota: Es posible, también con el programa de diseño elaborado obtener otra ganancia basada en otros puntos de operación o linealización.

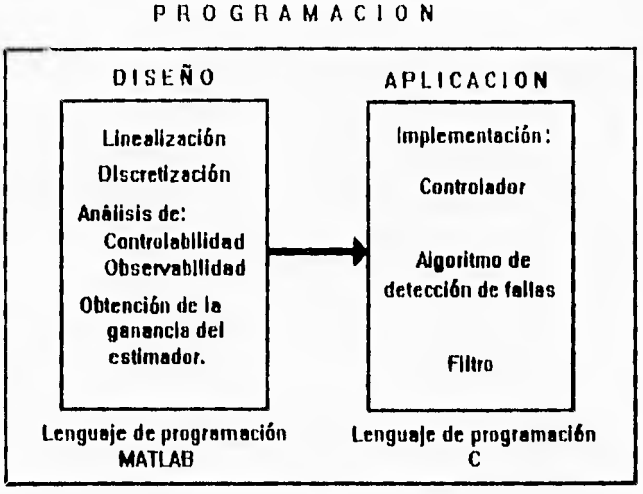
3.6 Filtro

El filtro, como ya se dijo anteriormente es utilizado para disminuir las señales de ruido aleatorio, en base a la teoría explicada en la sección 2.3.2.1. Esto es, se obtiene un promedio del residuo de 10 muestras a lo largo del comportamiento de la planta y si este es mayor a el voltaje de umbral se detecta la falla. Entre más grande sea la ventana de tiempo o número de muestras que se promedien, disminuirá el porcentaje de falsas alarmas, pero es posible que fallas de pequeña duración no sean detectadas.

3.7 Programación.

El software utilizado se basa principalmente en dos partes:

a) Diseño.
b) Aplicación.

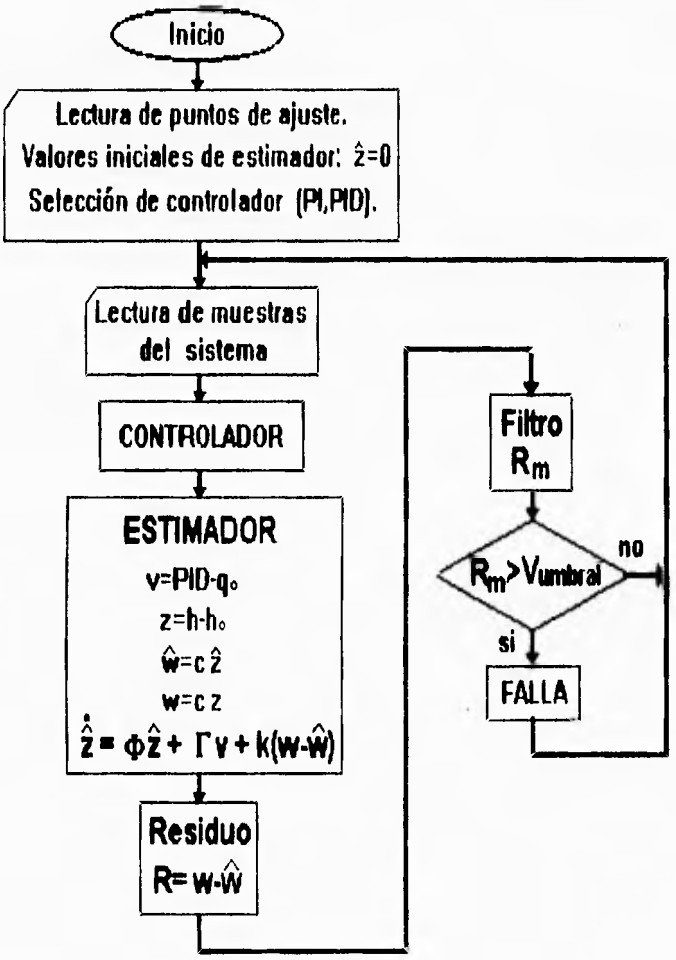


En la parte del diseño, se programó en lenguaje de Matlab, que es un paquete con que dispone la División de Estudios de Posgrado. En este programa únicamente es necesario proporcionar las alturas alrededor del cual se desea linealizar y automáticamente obtiene las matrices linealizadas del sistema, así como también realiza la discretización habiendo dado previamente el tiempo de muestreo.

Además nos muestra las matrices de controlabilidad y observabilidad verificando si el sistema es controlable y observable, así mismo se obtiene la ganancia del estimador utilizando filtros Kalman.

En la parte de la aplicación del esquema de detección de fallas se hizo en lenguaje C, apoyándose en algunos objetos definidos por el fabricante dentro del mismo lenguaje de programación. En ésta parte el usuario tiene la posibilidad de seleccionar entre las diferentes acciones de control P, PI, PID para llevar a los puntos de ajuste deseados, además a los controles PI y PID se le agregó la opción de detectar fallas.

Este programa se basa en el siguiente diagrama de bloques de la página siguiente:


Inicio
Lectura de puntos de ajuste.
Valores iniciales de estimador: $\hat{z}=0$
Selección de controlador (PI,PID).
Lectura de muestras del sistema
CONTROLADOR
ESTIMADOR
v=PID-q_o
z=h-h_o
$\hat{w}=c\,\hat{z}$
w=c z
$\dot{\hat{z}} = \Phi\hat{z} + \Gamma v + k(w-\hat{w})$
Residuo
$R= w-\hat{w}$
Filtro
R_m
$R_m > V_{umbral}$
no
si
FALLA

4. RESULTADOS.

A continuación se presentan los resultados de los diferentes experimentos a los que condujo la aplicación del esquema de redundancia analítica.

Condiciones generales de los experimentos:

Puntos de operación: h1 = 40 cm.
h2 = 20 cm.
h3 = 30 cm
Controlador: Proporcional-Integral-Derivativo
Sintonización: Oscilaciones sostenidas de Ziegler-Nichols
Ganancias: Bomba 1:
Kp = 13
Ti = 685
Td = 0.1
Bomba 2:
Kp = 13
Ti = 293
Td = 0.09

Flujos de entrada en puntos de operación en estado estacionario y sin falla:

Flujo 1 (Q1) = 31 ml/seg.
Flujo 2 (Q2) = 28 ml/seg.

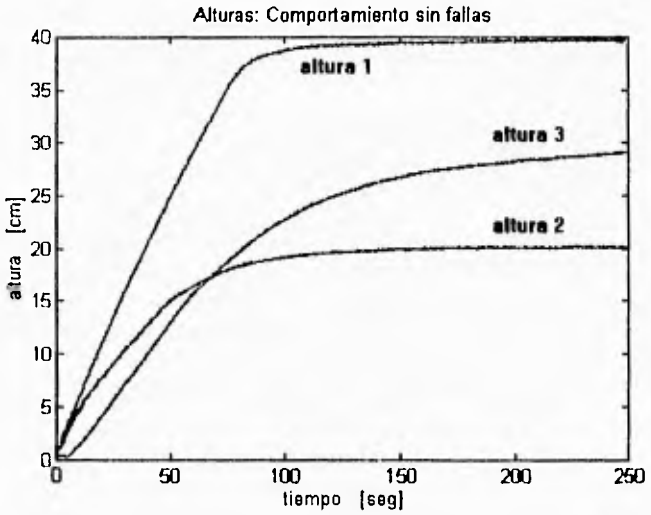
Experimento 1

Condiciones:

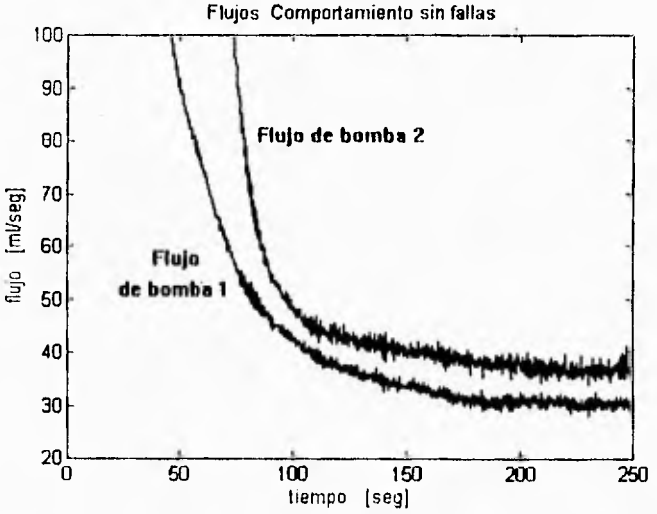
Se presenta el sistema trabajando normalmente.
Condiciones iniciales: h1 = h2 = h3 = 0 cm.

Resultados:

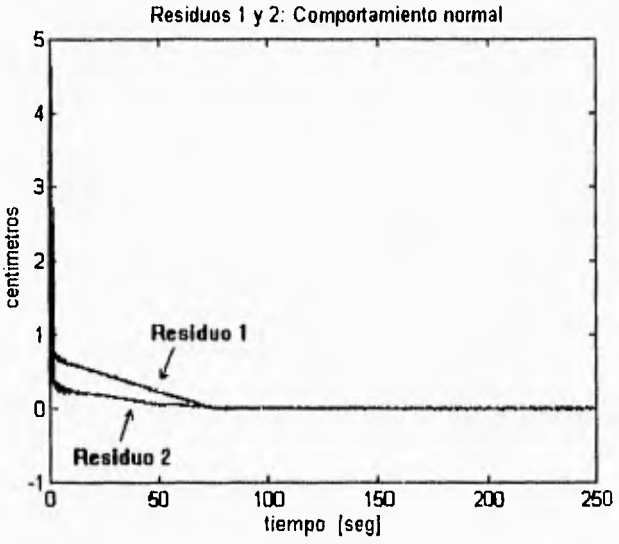
Gráfica 1



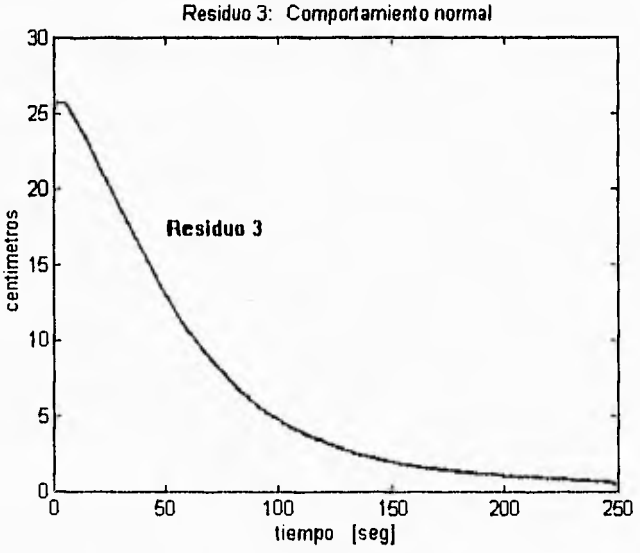
Gráfica 2



Gráfica 3



Gráfica 4



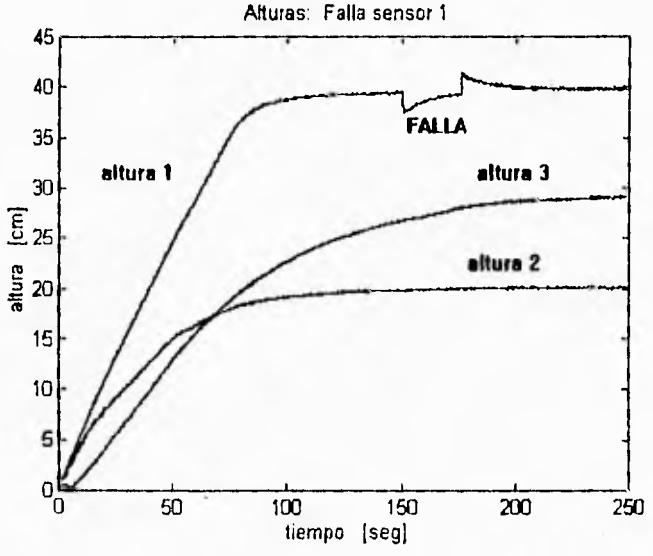
Experimento 2:

Condiciones:

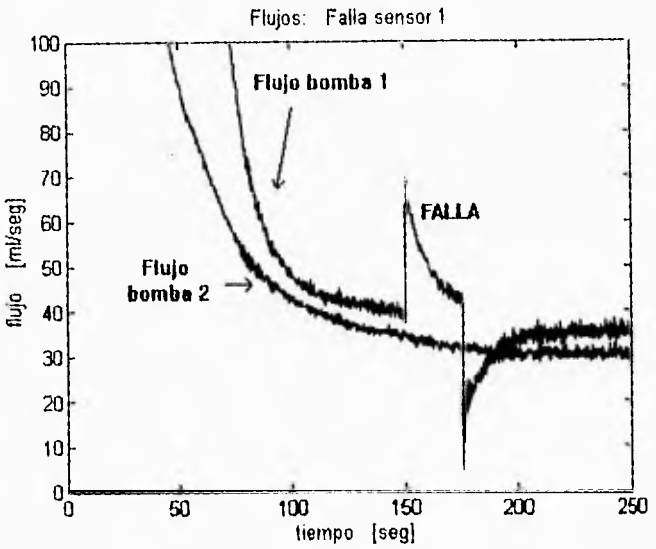
Se introdujo una falla en el sensor 1 después de 150 segundos.
El sensor solo detecta el 90% de la lectura real.
Duración de la falla: 25 seg.
Condiciones iniciales: h1 = h2 = h3 = 0 cm

Resultados:

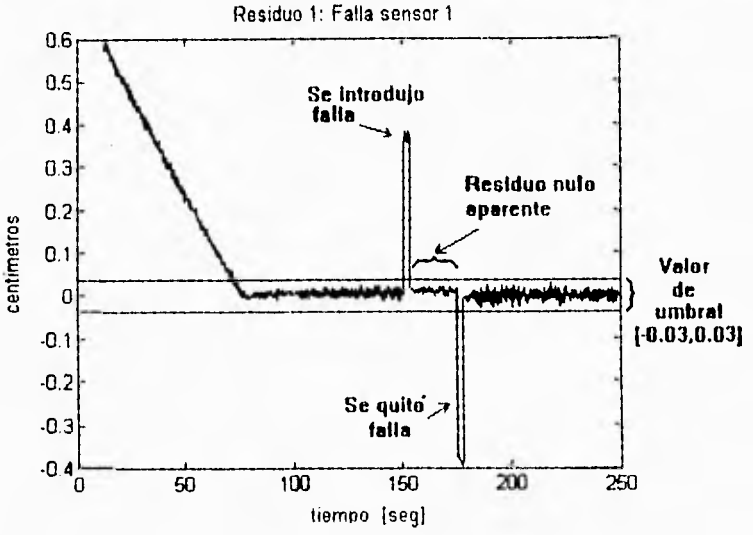
Gráfica 5



Gráfica 6



Gráfica 7



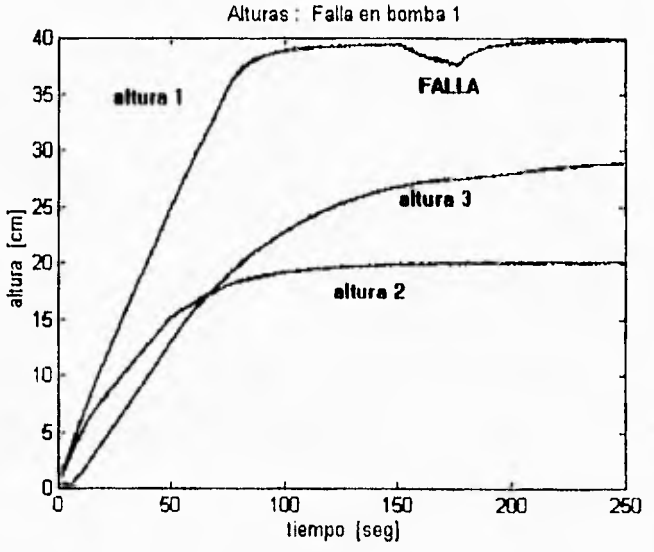
Experimento 3:

Condiciones:

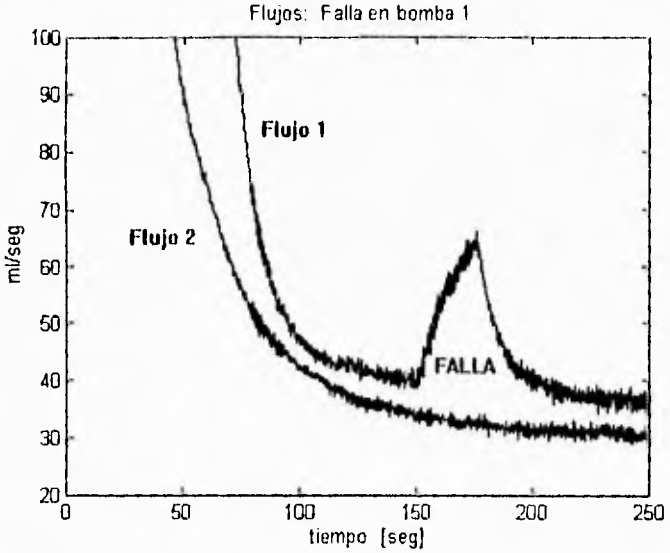
Se introdujo una falla en la bomba 1 después de 150 segundos.
La bomba funciona al 70% de su capacidad total.
Duración de la falla: 25 segundos.
Condiciones iniciales: h1 = h2 = h3 = 0 cm

Resultados:

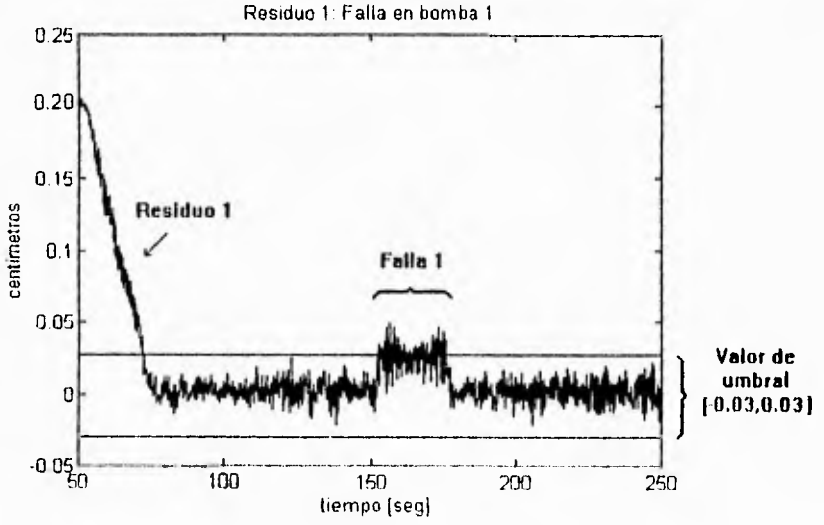
Gráfica 8



Gráfica 9



Gráfica 10



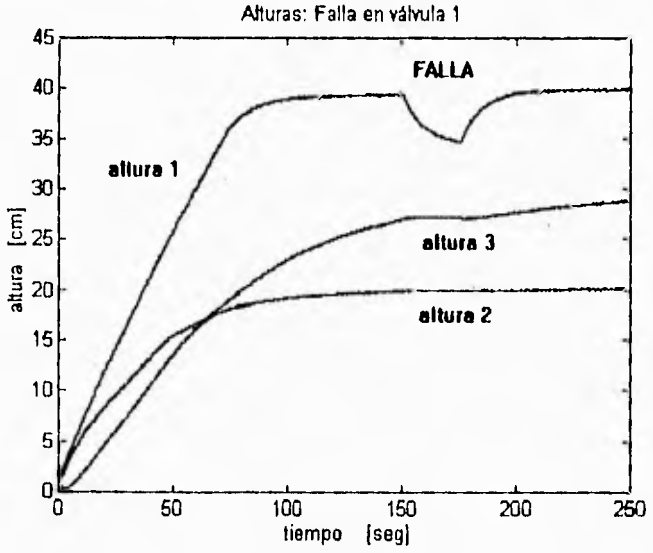
Experimento 4:

Condiciones:

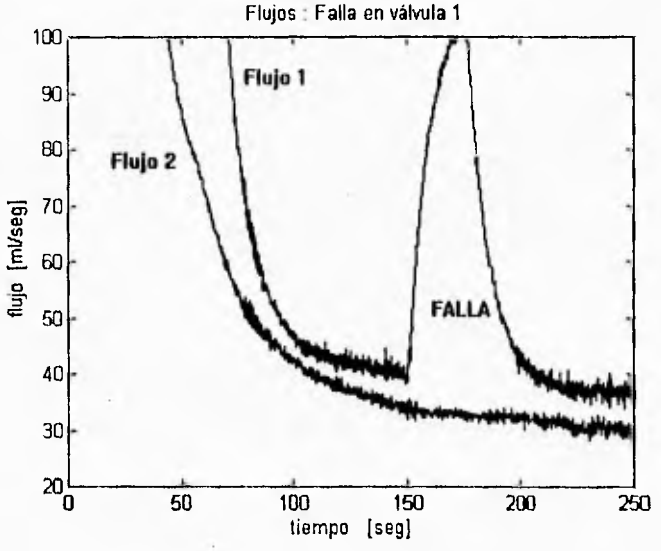
Se introdujo una falla en la válvula 1 después de 150 segundos.
Válvula totalmente abierta.
Duración de la falla: 25 seg.
Condiciones iniciales: h1 = h2 = h3 = 0 cm

Resultados:

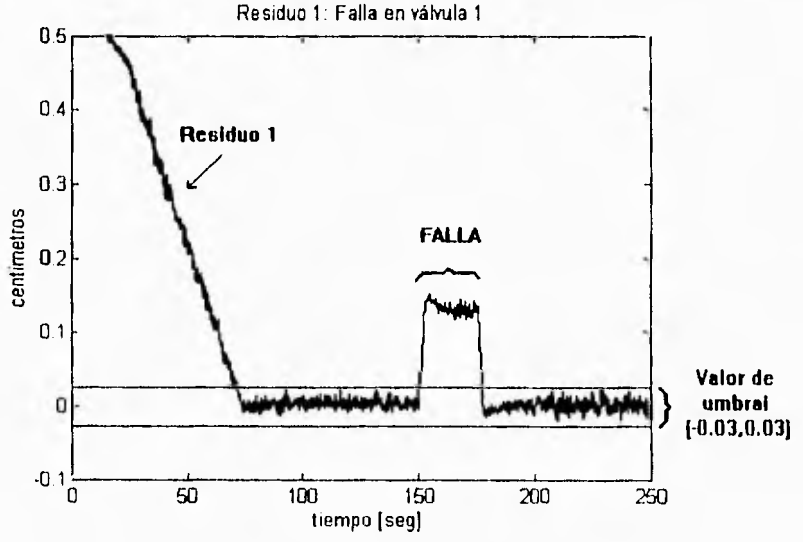
Gráfica 11



Gráfica 12



Gráfica 13



Experimento 5

Condiciones:

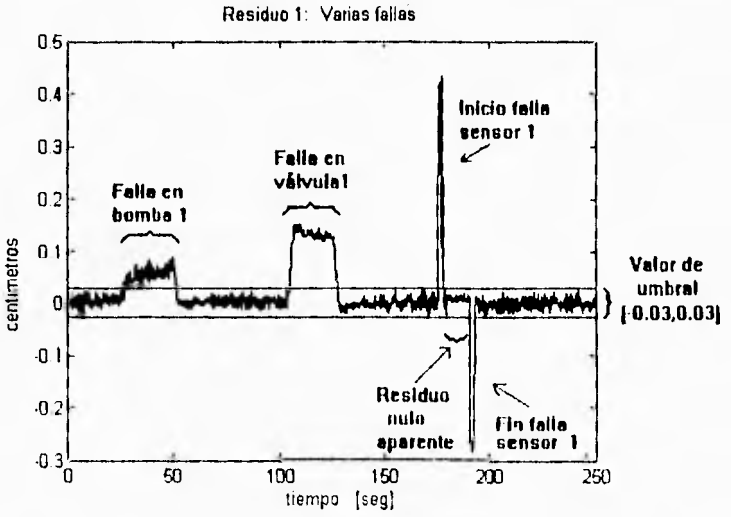
Se introdujeron diferentes fallas por un periodo de tiempo determinado.

Falla en:	Inicio (seg)	Fin (seg)	Sensibilidad
Válvula 3	5	12	Abierta totalmente
Bomba 1	25	50	50 %
Bomba 2	65	90	50 %
Válvula 1	100	125	Abierta totalmente
Válvula 2	140	165	Abierta totalmente
Sensor 1	175	190	90 %
Sensor 2	200	215	90 %

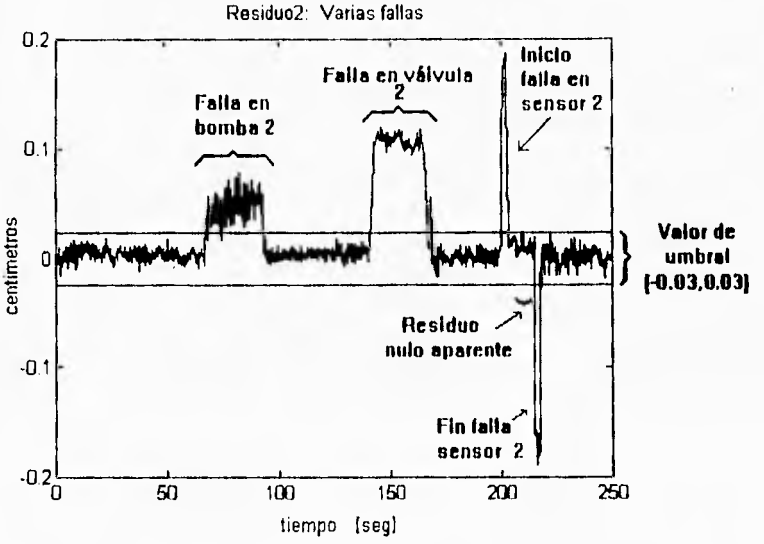
Condiciones iniciales: h1 = 40 cm.
h2 = 20 cm.
h3 = 30 cm

Resultados:

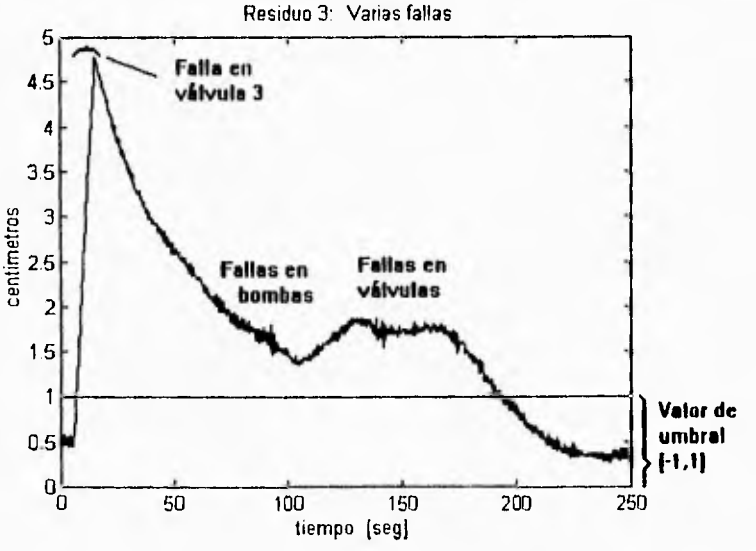
Gráfica 14



Gráfica 15



Gráfica 16



Experimento 6.

Condiciones:

Se intrudujeron combinaciones de fallas al mismo tiempo por un periodo determinado.

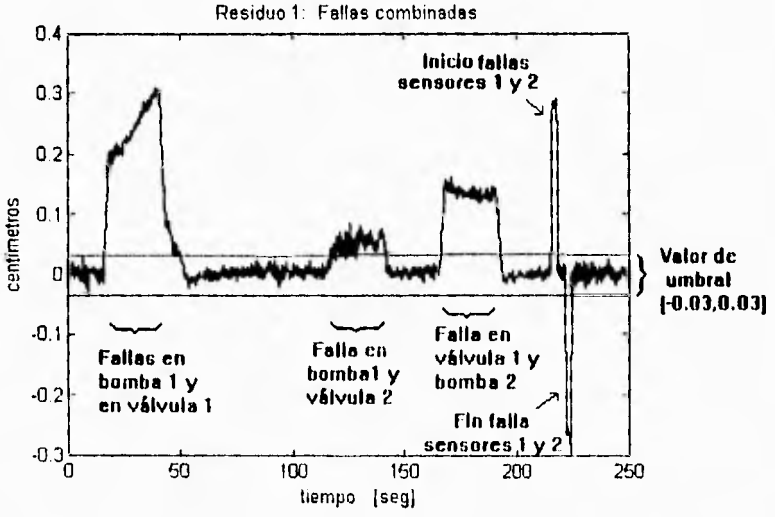
Fallas combinadas:	Inicio (seg)	Fin (seg)
Bomba 1 y válvula 1	**15**	**40**
Bomba 2 y válvula 2	**65**	**90**
Bomba 1 y válvula 2	**115**	**140**
Bomba 2 y válvula 1	**165**	**190**
Sensor 1 y 2	**210**	**220**

Falla en bombas de 50% de su capacidad real.
Falla en válvulas: totalmente abiertas.
Sensibilidad sensor: 90 %

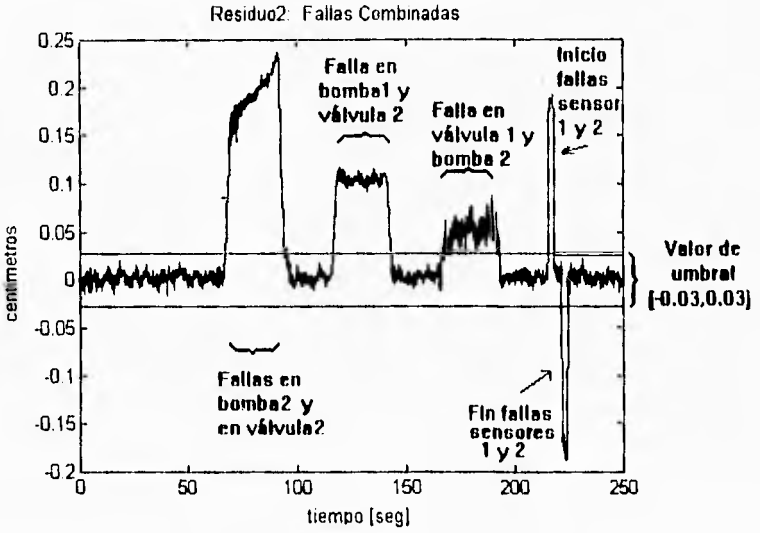
Condiciones iniciales: h1 = 40 cm.
h2 = 20 cm.
h3 = 30 cm

Resultados:

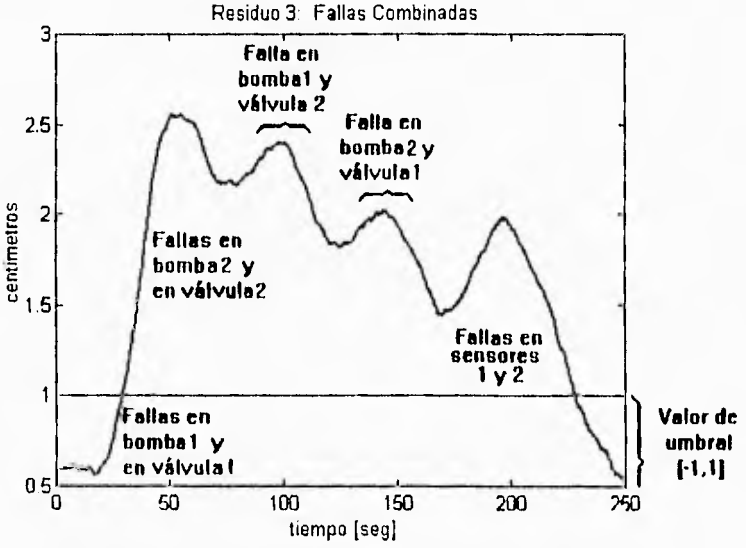
Gráfica 17



Gráfica 18



Gráfica 19



5. ANALISIS DE RESULTADOS.

Experimento 1.

En este experimento se logra apreciar el buen funcionamiento del controlador PID al llevar los estados del sistema a los puntos de operación deseados. Vea gráfica 1.

En la gráfica 2 se observa el comportamiento de las señales de control (flujos de entrada), las cuales tienden a un flujo constante en estado estacionario.

Según las gráficas 3 y 4 los residuos convergen a cero, esto quiere decir que no existe falla, al ser las salidas del estimador casi iguales a las salidas reales del sistema.

Los residuos 1 y 2 tienen una magnitud que oscila entre -0.2 y 0.2 cm., lo que da una idea de la magnitud que se le debe dar al valor de umbral. Se observa que entre más chico dicho valor de umbral aumenta la sensibilidad pero también aumenta el porcentaje de falsas alarmas.

Nótese que el residuo 3 (gráfica 4) converge a cero de una manera más lenta, esto se debe a que no tiene un flujo de entrada que le permita llegar al estado deseado en forma más rápida. También , se observa que el valor de umbral debe ser mayor por la misma razón de que no llega al estado deseado rápidamente.

Experimento 2.

En este experimento se introdujo una falla en el sensor 1 (solo detecta el 90% de la altura real) después de 150 segundos y con 25 segundos de duración, lo que se aprecia perfectamente en la gráfica 5. Nótese que el cambio en la lectura al introducirse el error es casi instantáneo midiendo 36 cm. cuando debían ser 40.

El estar sensando una altura menor a la de operación (40 cm.) el flujo también se incrementa casi instantaneamente (gráfica 6) y va disminuyendo conforme alcanza la altura deseada. Es importante hacer notar que la altura a la que se está ajustando es equivocada debido a la falla en el sensor (es mayor a la deseada). Esto se observa claramente, cuando al quitar la falla (aproximadamente en el segundo 175) la altura real es mayor a la altura de operación (vea gráfica 5) lo que hace que el flujo

disminuya rápidamente, para posteriormente converger al flujo en estado estacionario que se presenta cuando el sistema no tiene falla.

En la gráfica 7 se observa el comportamiento del residuo. Al introducir la falla, se rebasa casi instantáneamente el valor de umbral, para converger a cero posteriormente. Es evidente que el residuo existente después de introducir la falla es aparente, pues este es casi cero sólo porque se volvió a alcanzar el punto de operación deseado, pero dicho punto de operación es alcanzado con un sensor defectuoso.

Al quitar la falla, en la gráfica 7, se vuelve a rebasar el valor de umbral, pero esto es en referencia al sistema con falla. El residuo posterior a la falla es real pues convergió a cero en base a lecturas sin falla.

Experimento 3.

En este experimento se introdujo una falla en la bomba 1 después de 150 segundos y por un periodo de 25 segundos, lo que se ilustra en la figura 8. Después de 175 segundos, que es cuando se quita la falla, se observa en la misma gráfica como se vuelve a obtener la altura deseada.

En la gráfica 9, se observa que el flujo se incrementa con objeto de tratar de alcanzar los puntos de ajuste deseados.

En la gráfica 10, se observa claramente que el valor de umbral es rebasado por la magnitud de la señal de residuo al introducirse la falla.

Experimento 4.

En este experimento se introdujo una falla, al provocar una fuga en el tanque 1, lo que se simula abriendo una de las válvulas que se conecta al contenedor de agua.

Al provocar dicha fuga, el nivel de agua comienza a disminuir (gráfica 11), lo que hace que el controlador incremente el flujo de la bomba (gráfica 12), así al quitar la falla en el segundo 175, el nivel de agua regresa a su punto de operación, mientras que el flujo disminuye conforme se alcanzan dichos niveles de agua deseados.

En la gráfica 13, se aprecia en forma clara como al introducirse la falla, la magnitud del residuo rebasa el valor de umbral.

Experimento 5.

En el experimento 5 se introdujeron varias y direntes tipos de fallas.

En la gráfica 14 se observa que cada que se introducía una falla que afectara la salida 1, es decir la altura del tanque 1, se rebasan los niveles del valor de umbral. Se repite también lo analizado con respecto a la falla de un sensor, esto es, la existencia de un residuo nulo aparente, ya que aun habiendo falla, la magnitud del residuo es menor al valor de umbral debido a que se alcanzaron las alturas deseadas, pero según el sensor defectuoso. Esto mismo se presenta en lo que se refiere al residuo 2 (gráfica 15).

En la gráfica 16, se aprecia el incremento del residuo al provocar una fuga en el tanque 3 (simulada al abrir una válvula que se conecta al contenedor). Se observa también la influencia que tienen las fallas en otras partes del sistema, más específicamente las ocurridas en las válvulas (las que simulan fugas de los tanques) pues existe también un incremento en la magnitud del residuo. También se aprecia que es menos notoria la influencia de las fallas en las bombas.

Experimento 6.

En este experimento se introdujeron diferentes fallas al mismo tiempo., las cuales hacen que las magnitudes de los residuos aumenten y sobrepasen el voltaje de umbral, detectandose así la falla..

Así para la gráfica 17 y 18 se observa como afectan las diferentes fallas lo que se ve reflejado en los residuos 1 y 2.

Se nota en dichas gráficas, como al existir fallas combinadas, el residuo de las mismas se suma, sobrepasando fácilmente el valor de umbral.

En la grafica 19, correspondiente al residuo 3, es evidente que para su salida correspondiente es más difícil alcanzar un residuo próximo a cero por su interdependencia con los otros dos tanques. Esto se nota, al incrementarse la magnitud del residuo, no importando donde ocurra la falla.

6. Conclusiones.

Este trabajo logró mostrar que es posible detectar fallas en tiempo real en un sistema hidraúlico utilizando el enfoque de redundancia analítica.

Se comprobó también que el incrementar la sensibilidad (es decir, bajar los valores de umbral) puede aumentar los porcentajes de falsas alarmas.

Además es posible detectar fallas de componente (válvulas), actuadores (bombas) e instrumentos (sensores) pués una falla en ellos ocasiona una discrepancia entre el modelo real y el modelo estimado originando así que la magnitud del residuo sea mayor a la del valor de umbral. Al lograrse esto, se puede detectar la falla.

El hecho de manejar un residuo vectorial, nos permite identificar en que tanque ocurrió la falla y un posterior análisis gráfico puede guiarnos a conocer si la falla es de componente, actuador o instrumento.

Para este sistema en particular, se encontró un resultado bastante interesante, el cuál se origina en la falla en instrumentos (sensores). A este se le denominó "Residuo nulo aparente".

Con ello se quiere decir, que el residuo puede estar muy por abajo del valor de umbral y existir una falla, contradiciendo la teoría de redundancia analítica.

Sin embargo, esto se debe principalmente al uso de estimadores que se van actualizando de acuerdo a la dinámica del sistema, así al introducir una falla en el sensor, el controlador intenta llegar al punto de operación deseado (lo cual finalmente logra), haciendo que el residuo sea cero. Pero nótese que se alcanzó dicho punto de operación en base a lecturas erróneas del sensor.

Se comprobó también que las fallas pueden ser aditivas, facilitando su detección.

Para trabajos a futuro, es conveniente aplicar métodos de identificación para obtener las matrices A, B y C del sistema. También se sugiere manejar un modelo no lineal para aumentar la robustez del esquema de detección de fallas.

Se propone además, lograr identificar el tipo de falla detectada y no sólo detectarla y saber dónde se origina, que es la propuesta de este trabajo.

Bibliografía.

[1]PATTON,RON; FRANK, PAUL & CLARK,ROBERT, "Fault Diagnosis in Dynamic Systems", Prentice-Hall, Inglaterra (1989), Cap. I y II .
[2]FRANK,PAUL, "Fault Diagnosis in Dynamic Systems Using Analitical and Knowledge-based Redundancy", Automática, Vol.26, No. 3, Inglaterra (1990), pp. 459-474.
[3]JANOS,J.GERTLER,"Survey of Model-Based Failure Detection and Isolation in Complex Plants", IFAC Proceedings Series, Vol 7. (1987).
[4]OGATA,KATSUHIKO, "Modern Control Engineering", Prentice-Hall, EEUU (1970), Cap. IV,V.
[5]FRANKLIN,GENE & POWELL,DAVID, "Digital Control of Dynamic Systems", Addson-Wesley, EEUU (1980), pp. 145-148.
[6]FRANKLIN,GENE & POWEL,DAVID, "Control de Sistemas Dinámicos con Realimentación", Addison-Wesley, EEUU (1991), pp. 342-344.
[7]RODRIGUEZ,FRANCISCO,"Manual de Prácticas de Laboratorio de Control Digital", DIECC, Facultad de Ingeniería, UNAM.
[8]MASSEY, B.STANFORD,"Mechanics of fluids", Van Nostrand Reinhold Co., Inglaterra (1979), Cap. III.
[9]JAMES,JOHN & HABERMAN, WILLIAM, "Introducción a la mecánica de los fluidos", Prentice-Hall, EEUU (1974), Cap. III.
[10]VARIOS AUTORES,"Documentation DTS200 Three-Tank-System", Amira GmbH, Alemania (1993).

PAGINACION VARIA

COMPLETA LA INFORMACION

APENDICE A

En el siguiente apéndice se muestra la estructura del lenguaje fuente de MATLAB..(Versión 4.0) que corresponde a la parte de diseño del software desarrollado en este trabajo.

```
%       Universidad  Nacional Autonoma de México
%       Facultad de Ingenieria
%       Elaborado por:  César Augusto Mendoza Serrano
%       25 de enero de 1995.

clc
disp('      ********************************************************************************')
disp('      * El siguiente programa discretiza el sistema:                                 *')
disp('      *    dh = Ah + BQ                                                              *')
disp('      *     y = Ch                                                                   *')
disp('      *                                                                              *')
disp('      *                                                                              *')
disp('      * Ademas verifica que este sea controlable  y                                 *')
disp('      * observable, obteniendo su respectiva matriz.                                *')
disp('      * Finalmente se obtiene la matriz de ganancia del                             *')
disp('      * estimador.                                                                  *')
disp('      ********************************************************************************')
disp(' ')
disp(' ')
disp('      Ahora se procede a la introduccion de los elementos ')
disp('      de la matriz A,B y C.    Se  recuerda que las di-      ')
disp('      mensiones son de 3x3, 3x2 y 3x3 respectivamente.   ')
disp(' ')
disp('                    Cualquier tecla para continuar ')
pause

%*******************************************************************************************
%*****************  Creación de las matrices del sistema ***********************************
%*******************************************************************************************

% MATRICES DE RUIDO EN EL PROCESO Y COVARIANZA

 g= [.5 0; 0 .5; 0 0];

 q= [.5 0; 0 .5];

 r= [.5 0 0; 0 .5 0; 0 0 .5];
```

Apéndice A

```
% ASIGNACION DE VALORES A LOS PARAMETROS DEL SISTEMA

% Coeficientes de flujo

   az13= .4475;
   az32= .4342;
   az20= .5971;
   az10=  0;
   az30=  0;
   az40=  0;

 % Parametros del sistema

    A= 0.0154;        % Area transversal de los tanques. [m2]
   Sn= 5e-5;          % Area transversal de los tubos.   [m2]
    g= 9.81;          % Acceleración de la gravedad [m2/seg]

% INTRODUCCION DE ALTURAS ALREDEDOR DE LAS CUALES SE LINEALIZA

clc
disp(' ')
disp ('    Alturas alrededor de las cuales se linealizará ')
h1=input('Altura de tanque 1 [metros] : ');
h2=input('Altura de tanque 2 [metros]: ');
h3=input('Altura de tanque 3 [metros]: ');

% OBTENCION DE LA MATRIZ A

a11=(-az13*Sn*sign(h1-h3)*g/(sqrt(2*g*abs(h1-h3)))-az10*Sn*g/(sqrt(2*g*h1)))/A;
a12= 0;
a13= (az13*Sn*sign(h1-h3)*g/(sqrt(2*g*abs(h1-h3))))/A;
a21=0;
a22=(-az32*Sn*sign(h3-h2)*g/(sqrt(2*g*abs(h3-h2)))-az20*Sn*g/(sqrt(2*g*h2))-
az40*Sn*g/(sqrt(2*g*h2)))/A;
a23=(az32*Sn*sign(h3-h2)*g/(sqrt(2*g*abs(h3-h2))))/A;
a31=(az13*Sn*sign(h1-h3)*g/(sqrt(2*g*abs(h1-h3))))/A;
a32=(az32*Sn*sign(h3-h2)*g/(sqrt(2*g*abs(h3-h2))))/A;
a33=(-az13*Sn*sign(h1-h3)*g/(sqrt(2*g*abs(h1-h3)))-az32*Sn*sign(h3-h2)*g/(sqrt(2*g*abs(h3-
h2)))-az30*Sn*g/(sqrt(2*g*h3)))/A;

disp (' ')
disp ('                    cualquier para continuar')
pause

% OBTENCION DE LA MATRIZ B

b11=1/A;
b12=0;
b21=0;
b22=1/A;
b31=0;
b32=0;
```

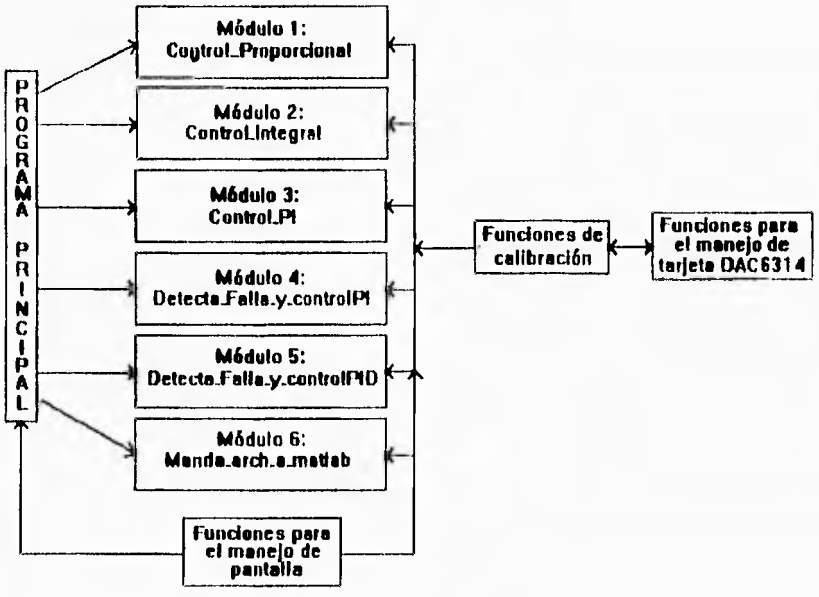
```
% OBTENCION DE LA MATRIZ C

c11=1;
c12=0;
c13=0;
c21=0;
c22=1;
c23=0;
c31=0;
c32=0;
c33=1;


% CREACION DE LAS MATRICES DEL SISTEMA A PARTIR DE LOS DATOS
INTRODUCIDOS

clc
a=[a11 a12 a13; a21 a22 a23; a31 a32 a33];
b=[b11 b12; b21 b22; b31 b32];
c=[c11 c12 c13; c21 c22 c23; c31 c32 c33];
disp(' ')
disp('      Para el sistema linealizado las         ')
disp('      matrices introducidas :')
disp('      dh= Ah + BQ        y = Ch ')
disp(' ')
a
b
c
disp ('                    Cualquier tecla para continuar')
pause


%      SE PROCEDE A REALIZAR LA DISCRETIZACION DEL SISTEMA
clc
disp(' ')
disp('          D I S C R E T I Z A C I O N')
disp(' ')
t=input('   Tiempo de muestreo: ');
[phi,gamma]=c2d(a,b,t);
disp(' ')
disp('    El sistema discretizado: h(k+1)= phi h(k) +  gamma u')
disp(' ')
disp('  donde:')
phi
gamma
disp(' ')
disp('                Cualquier tecla para continuar')
pause


%  SE HACE EL ANALISIS DE CONTROLABILIDAD Y OBSERVABILIDAD
clc
matcont=ctrb(phi,gamma);
```

```
matobsv=obsv(phi,c);
disp(' La matriz de controlabilidad es: ')
matcont
disp(' ')
disp('La matriz de observabilidad es: ')
matobsv disp(' ')
disp('                Cualquier tecla para continuar')
pause
clc
disp(' ')
disp(' ')
rangocont= rank (matcont);
rangobsv= rank (matobsv);
if rangocont == 3
    disp('      El sistema es controlable' )
    disp(' ')
else  disp('      El sistema no es controlable')
end

if rangobsv == 3
    disp('      El sistema es observable')
    disp(' ')
else disp('      El sistema no es observable')
end

%  OBTENCION DE LA GANANCIA  VECTORIAL  DEL ESTIMADOR

l=  dlqe(phi,gamma,c,q,r);
disp (' ')
disp (' ')
disp (' El estimador lineal cuadrático discreto es:')
l
% FIN DEL PROGRAMA DE DISEÑO
```

APÉNDICE B

En este apéndice se lista el programa fuente en lenguaje Borland C++, (Versión 3.1) de el algoritmo de detección de fallas, también se incluyen las rutinas de manejo de pantalla, controladores, calibración. Además se agregan algunas rutinas proporcionadas por la empresa alemana Amira GmbH para el manejo de la tarjeta DAC6314.

El programa está organizado de acuerdo al diagrama de bloques siguiente:



En el programa principal se fijan los puntos de operación así como también se selecciona el módulo con el que se quiere trabajar. Cada módulo lleva el nombre de la función que realiza.

Las funciones para el manejo de la pantalla interactúan entre el programa principal y los módulos.

Para la implementación del controlador, cada módulo sigue el diagrama de flujo de la página 41. Los módulos cuatro y cinco implementan el algoritmo de detección de fallas de acuerdo con el diagrama de la página 45.

Las funciones de calibración como son AltenCentimetros, De_flujo_a_volt, se utilizan para transformar las lecturas, dadas en volts, en centimetros, y los flujos, dados por el controlador, en voltaje para el actuador respectivamente.

Las funciones para el manejo de la tarjeta DAC6314, que se encuentran al principio del programa fueron proporcionadas por el fabricante y se da la información de las mismas en el manual correspondiente[10].

La definición de los objetos utilizados en el programa se encuentra en el archivo io2.h.

El listado del programa fuente es el siguiente:

FUNCIONES PARA EL MANEJO DE LA TARJETA DAC6314

```
/*
  Universidad Nacional Autónoma de México.
  Facultad de Ingeniería.
  Elaborado por: César Augusto Mendoza Serrano.
  4 de febrero de 1995.
  Definicion de objetos amira-PC-IO-Karte
  M. Dabrowski
*/
#include <time.h>
#include <dos.h>
#include <conio.h>
#include <string.h>
#define DIM 1000

#include <conio.h>
#include <stdio.h>
#define BYTE    unsigned char
#define BOOL    unsigned int

#include "c:\tt\prj\p338\c\io2.h";

/* Definiciones en "IO.H"  */

void PCIO( int adress )
{
        base            = adress;
        data_port       = base+3;
        cmd_port        = base+2;
        output_status   = 0 ;
        out_reg         = 0 ;
        gate_int  = 0 ;
        DAC1    = 0x20;
```

```
        DAC2    = 0x40;
        ADC     = 0x00;
        DIGITAL_IN      = 0x60;
        DIGITAL_OUT  = 0xa0;
        HCTL_RESET   = 0x80;
        HCTL_HIGH     = 0x88;
        HCTL_LOW      = 0x89;
        TIMER           = 0xE0;
        DIGITAL_OUT2    = 0xC0;
};

void SelectChannel( int channel )
{
        SelectAdress ( DIGITAL_OUT ) ;
        output_status = (output_status & 0x8F) | ( ((BYTE)channel<<4) & 0x70 ) ;
        outp ( data_port, output_status ) ;
};

BYTE ReadDigitalInputs( void )
{
        SelectAdress ( DIGITAL_IN ) ;
        return (inp ( data_port )  & 0x0f ) ;
};

void DigOut ( BYTE data )
{
        SelectAdress ( DIGITAL_OUT ) ;
        output_status = (output_status & 0xf0) | (data & 0x0f) ;
        outp ( data_port, output_status ) ;
};

BOOL ReadDigital( int channel )
{
        BYTE mask;
        mask = 0x01<<channel;
        return( ( ReadDigitalInputs() & mask ) ? 1 : 0 );
};

void WriteDigital( int channel, int value )
{
        out_reg = value ? (out_reg | (0x01<<channel)) : (out_reg &~(0x01<<channel));
        DigOut( out_reg );
};

int ReadAnalogInt( int channel )
{
        int adc_word ;
        SelectChannel ( channel ) ;
        SelectAdress ( ADC ) ;
        inp ( data_port ) ;                 /* Empieza conversion */
        SelectAdress ( DIGITAL_IN ) ;

        while(!( inp( data_port ) & 0x80) );   /* Ocupado */
```

```
        SelectAdress ( ADC+1 ) ;
        adc_word  = ( inp ( data_port ) << 8 ) & 0x0F00 ;
        SelectAdress ( ADC ) ;
        adc_word |= inp ( data_port ) ;
        return (2048-adc_word) ;              /* rango: +-2048 */
};

float ReadAnalogVolt( int channel )
{
        return( (float)ReadAnalogInt(channel) / 204.8 );
};

float ReadAnalogVoltMean( int channel, int repeat )
{
        float   mean;
        int     count;

        mean = 0.0;
        for( count=0; count<repeat; count++ )
                mean += ReadAnalogVolt( channel );
        mean /= (float)repeat;
        return( mean );
};

void WriteAnalogInt( int channel, int value )
{
        BYTE addr;
        value+=2048;
        addr = (channel==0) ? DAC1 : DAC2 ;
        SelectAdress ( addr++ ) ;
        outp ( data_port, value & 0x0f );
        SelectAdress ( addr++ ) ;
        outp ( data_port, (value>>4) & 0x0f ) ;
        SelectAdress ( addr++ ) ;
        outp ( data_port, (value>>8) & 0x0f ) ;
        SelectAdress ( addr ) ;
        outp ( data_port, 0 ) ;
};

void WriteAnalogVolt( int channel, float value )
{
        float val;
        /* Control de sobrecapacidad */
        if( value > 9.9 ) val = 9.9;    /* Maximo: 9.9951 */
        else if( value < -9.9 ) val = -9.9;
        else val = value;
        WriteAnalogInt( channel, (int)(val*204.8));
};

void ResetHCTL( void )
{
        SelectAdress( HCTL_RESET );
        inp( data_port );
};
```

```
int    ReadHCTL( void )
{
        int     wert = 0;
        SelectAdress( HCTL_HIGH );
        wert = inp( data_port ) << 8;
        SelectAdress( HCTL_LOW );
        wert |= inp( data_port );
        return( wert );
};

void SetINT( int val )
{
        gate_int = val ?
            ( gate_int | 0x02  ) :
            ( gate_int &~ 0x02 );
        SelectAdress( DIGITAL_OUT2 );
        outp( data_port, gate_int );
};

void SetTimer( unsigned long time )
{
        unsigned int w;

        /* Parte baja de palabra */
        w = ( unsigned int )time;
        SelectAdress( TIMER+3 );
        outp( data_port, 0x76 ); /* Contador1: primer lsb, modo 3 */

        SelectAdress( TIMER+1 );
        outp( data_port, ( unsigned char)w );
        outp( data_port, ( unsigned char)(w>>8) );

        /* Parte alta de palabra */
        w = ( unsigned int )(time>>16);
        SelectAdress( TIMER+3 );
        outp( data_port, 0xb6 ); /* Contador2: primer lsb, modo 3 */

        SelectAdress( TIMER+2 );
        outp( data_port, ( unsigned char)w );
        outp( data_port, ( unsigned char)(w>>8) );
};

void SetCounter( unsigned int count )
{
        /* setzen */
        SelectAdress( TIMER+3 );
        outp( data_port, 0x36 ); /* Contador0: primer lsb , modo 3 */

        SelectAdress( TIMER+0 );
        outp( data_port, ( unsigned char)count );
        outp( data_port, ( unsigned char)(count>>8) );
};
```

```
unsigned long GetTimer( void )
{
        unsigned long wert=0;

        /*Parte baja palabra */
        SelectAdress( TIMER+3 );
        outp( data_port, 0x46 ); /* Contador1: en volatil, modo 3 */
        SelectAdress( TIMER+1 );
        wert  = (unsigned long)inp( data_port );
        wert |= (unsigned long)inp( data_port ) << 8;

        /*Parte alta palabra */
        SelectAdress( TIMER+3 );
        outp( data_port, 0x86 ); /* Contador2: en volatil, modo 3 */

        SelectAdress( TIMER+2 );
        wert |= (unsigned long)inp( data_port ) << 16;
        wert |= (unsigned long)inp( data_port ) << 24;

        return wert;
};

unsigned int GetCounter( void )
{
        unsigned int wert=0;

        /* Contador */
        SelectAdress( TIMER+3 );
        outp( data_port, 0x06 ); /* Contador0: en volatil, modo 3 */
        SelectAdress( TIMER+0 );
        wert  = ( unsigned int )inp( data_port );
        wert |= ( unsigned int )inp( data_port ) << 8;
        return wert;
};

void GateTimer( int val )
{
        gate_int = val ?
            ( gate_int | 0x01  ) :
            ( gate_int &~ 0x01 );
        SelectAdress( DIGITAL_OUT2 );
        outp( data_port, gate_int );
};

void GateCounter( int val )
{
        gate_int = val ?
            ( gate_int | 0x04  ) :
            ( gate_int &~ 0x04 );
        SelectAdress( DIGITAL_OUT2 );
                outp( data_port, gate_int );
};
```

```
/*************************************************************************
* Variables globales del programa                                        *
*************************************************************************/

int   num_dat=0;                    /* Apuntador de los vectores para matlab  */
float altura1[DIM],                 /* Vectores de alturas para los archivos  */
    altura2[DIM],                   /* de matlab.                             */
    altura3[DIM],
    t_muestreo[DIM],                /* Vector de muestras -- arch. en matlab  */
    Residuo1[DIM],                  /* Vectores de residuos para arch. en     */
    Residuo2[DIM],                  /* matlab.                                */
    Residuo3[DIM],
    alt_estimad1[DIM],              /* Vectores de alturas estimadas para     */
    alt_estimad2[DIM],              /* arch. en matlab                        */
    alt_estimad3[DIM],
    err_alt1_real_estim[DIM],       /* Vectores de errores entre la altura    */
    err_alt2_real_estim[DIM],       /* real y estimada para arch. en matlab   */
    err_alt3_real_estim[DIM],
    fluj_bomba1[DIM],               /* Vectores de flujo para arch. en matlab */
    fluj_bomba2[DIM],
    potv1,potv2;                    /* Potencias para las bombas              */

float Ti1,Ti2,                      /* Ganancias de ley integral              */
    Kp1,Kp2,                        /* Ganancias de ley proporcional          */
    Ki1,Ki2,                        /* Ganancias integrales (Kp/Ti)           */
    Td1,Td2,                        /* Ganancias de ley derivativa            */
    h1,h2,h3,                       /* Altura de los tanques                  */
    err1,err2,                      /* Errores. h deseada - h medida          */
    err1_anterior,                  /* Errores un tiempo de muestreo          */
    err2_anterior,                  /* anterior a err1 y err2                 */
    integral1,                      /* Leyes de control integral              */
    integral2,
    proporcional1,                  /* Leyes de control proporcional          */
    proporcional2,
    PI1,PI2,                        /* Leyes de control PI2                   */
    PID1,PID2,                      /* Leyes de control PID                   */
    derivativo1,                    /* Leyes de control derivativo            */
    derivativo2;
int   cont=0,j,aux=0,i=0,elemento;
    char  ch,ch2;

/*************************************************************************
* Variables para el proceso de deteccion de fallas                       *

Tenemos que:  dh=Ah+Bu
                y=Ch

          h1     a11 a12 a13   h1       b11 b12    q1
        d h2  =  a21 a22 a23 * h2   +    b21 b22 *  q2
          h3     a31 a32 a33   h3       b31 b32

                 c11 c12 c13     h1
              y  = c21 c22 c23  *  h2
```

```
                                    h3
*********************************************************************/

float a11=.99746,                   /* Las a's corresponden a los elementos   */
      a12=0,                        /* de la matriz A listada anteriormente   */
      a13=.00253,
      a21=0,
      a22=.99515,
      a23=.00246,
      a31=.00253,
      a32=.00246,
      a33=.9950,
      b11=16.21315,                 /* Las b's corresponden a los elementos   */
      b12=0,                        /* de la matriz B listada anteriormente   */
      b21=0,
      b22=16.194329,
      b31=.02059,
      b32=.01996,
      c11=1,                        /* Las c's corresponden a los elementos   */
      c12=0,                        /* de la matriz C listada anteriormente   */
      c13=0,
      c21=0,
      c22=1,
      c23=0,
      c31=0,
      c32=0,
      c33=1,
      Zestim1=0,                    /* Las Zestim son las alturas estimadas   */
      Zestim2=0,                    /* para el tanque 1,2,3.                  */
      Zestim3=0,                    /* unidad: metros                         */
      uo1=33,                       /* Flujos de Q1 y Q2 en estado            */
      uo2=28,                       /* estacionario.  [ml/seg]                */
      ho1=40,                       /* Alturas alrededor de las               */
      ho2=20,                       /* cuales se linealiza.                   */
      ho3=30,                       /*  unidad: centimetros                   */
      k11=.9962,                    /* Las k's representan los                */
      k12=0,                        /* valores de los elementos de            */
      k13=.0013,
      k21=0,                        /* de la matriz de ganancia del           */
      k22=.9962,                    /* estimador.                             */
      k23=.0012,
      k31=.0013,
      k32=.0012,
      k33=.0011,
      v1,v2,                        /* Entrada (cambio de variable)           */
      w1,w2,w3,                     /* Salida  (cambio de variable)           */
      Westim1,Westim2,Westim3,       /* Salidas (alturas) estimadas            */
      res1,res2,res3,               /* Residuos                               */
      z1,z2,z3,                     /* Alturas (cambio de variable)           */
      Residuo_prom1=0,              /* Residuos promedio                      */
      Residuo_prom2=0,
      Residuo_prom3=0,
      Residuo_suma1=0,              /* Suma de residuos en una venta          */
      Residuo_suma2=0,              /* na                                     */
```

```
    Residuo_suma3=0,
    Resid1[11]={0,0,0,0,0,0,0,0,0,0,0},  /* Ventana de resi                */
    Resid2[11]={0,0,0,0,0,0,0,0,0,0,0},  /* duos ,                         */
    Resid3[11]={0,0,0,0,0,0,0,0,0,0,0};
```

FUNCIONES DE CALIBRACIÓN.

```
/*****************************************************************************
* La siguiente funcion devuelve la altura en centimetros del tanque 1,2 o 3   *
* Se utiliza de la siguiente manera:                                          *
*     y = AltenCentimetros (1);                                               *
*                    donde "y" (tipo float)  recibe el valor de               *
*                    la altura en centimetros del tanque 1                    *
*****************************************************************************/

float AltenCentimetros (int altura) {
 float tanque[3][14] ={ {0 ,  9.222,7.874,6.465,5.07,3.674,2.285,0.895,-0.490,
                          -1.870,-3.245,-4.618,-6.001,-7.318},
                        {1,  8.924,7.570,6.159,4.753,3.343,1.953,0.557,-0.830,
                          -2.210,-3.589,-4.953,-6.331,-7.650},
                        {2,  9.212,7.848,6.427,5.001,3.597,2.196,0.793,-0.602,
                          -1.989,-3.368,-4.751,-6.141,-7.458}};

/* En la definicion de la matriz anterior "0" corresponde tanque uno, "1" a    */
/* a tanque dos y "2" a tanque 3.                                              */
/* Ademas a cada 5 cm. le corresponde un voltaje. Ej. 9.222 es para 0 cm       */
/* 7.874 es para 5 cm. y asi sucesivamente, lo cual es para calibracion        */

int  j,                          /* variable auxiliar para apuntador de matriz  */
    canal=altura-1;              /* canal de la tarjeta.                        */
float alt_volt,                  /* altura mostrada en volts                    */
     alt_cm;                     /* altura en cm.                               */
    alt_volt=ReadAnalogVolt(canal);

    for (j=0;j<=13;j++)   {
          if (alt_volt > tanque[canal][j+1]) {
                          alt_cm=(((alt_volt-tanque[canal][j])/(tanque[canal][j+1]
                                -tanque[canal][j]))*5)+(5*(j-1));
                          j=13;  }
          else {
                          alt_cm=60+(tanque[canal][13]-alt_volt)*(5/1.318);  };
                          };

return (alt_cm+1);      /*  Se le suma 1 solo por efectos de factor de  */
                        /*  calibracion dinamica                        */
};


/*****************************************************************************
*  En la siguiente funcion se hace arrancar a las bombas con los parametros:  *
*   tiempo:  lo que dura prendida la bomba.                                   *
*   pot1: potencia (en flujo) de la bomba 1                                   *
*   pot2: potencia (en flujo) de la bomba 2                                   *
*****************************************************************************/
```

```
void Arranca_bombas ( int tiempo, float pot1, float pot2)
{
int  i;
int  DecSeg=tiempo*10;          /* Se multiplica por 10 para hacer          */
                                /* que el ciclo dure 1 segundo              */
float potv1,potv2;
     potv1=De_flujo_a_volt(1,pot1);
     potv2=De_flujo_a_volt(2,pot2);

    for(i=1;i<=DecSeg;i++) {
          WriteDigital(1,1);    /*Aseguramos los estados de las            */
          WriteDigital(2,0);    /*de las salidas digitales                 */
          WriteDigital(1,0);    /* Se hace que se mande el tren*/
          WriteDigital(1,1);    /* de pulsos por el canal 2    */
    /*    int delay (100);*/
          WriteAnalogVolt(0,potv1);
          WriteAnalogVolt(1,potv2);
          if (AltenCentimetros(1)>61) {
                potv1=-10;}
          else {if (AltenCentimetros(2)>61) {
                      potv2=-10;}
                else {} }
};
};

/*****************************************************************************
* La siguiente funcion convierte el flujo de la bomba y lo asigna a una varia-*
* de tipo float.                                                              *
* Parametros: bomba: la bomba uno o la bomba dos                              *
*          flujo: el flujo en ml/seg                                          *
*****************************************************************************/

float De_flujo_a_volt(int bomba, float flujo)
{
float tanque [2] [22] ={{0, 0.00001,0.01,0.212,0.381,0.599,0.768,1.011,1.184,
                         1.403,1.522,1.800,1.813,2.135,2.146,2.366,
                         2.498,2.650,2.769,2.950,3.115,3.252},
                        {1, 0.00001,0.09,0.183,0.398,0.628,0.823,1.010,1.229,
                         1.439,1.658,1.820,2.041,2.342,2.421,2.712,
                         2.805,2.946,3.102,3.274,3.414,3.233}};

/* La matriz anterior el "0" es para bomba 1 y "1" para bomba 2 .      */
/* corresponde al incremento de altura cada 5 seg                      */

int  aux1=-11,j,canal=bomba-1;
float voltaje_de_flujo,inch;

    inch=(flujo*5)/(3.1416*49);          /* El *5 es porque se toma el     */
    if (flujo>99) { inch=3.23;};         /* incremento de altura cada      */
    if (flujo<=0) { inch=0.00001;};      /* 5 segundos                     */
    for (j=0;j<=20;j++)   {
          if (inch < tanque[canal[ [j+1[) {   /*FORMULA DE INTERPOLACION LINEAL*/
                voltaje_de_flujo=(((inch-tanque[canal] [j[)/(tanque[canal]{j+1[
```

```
                    -tanque[canal][j]))*1)+(aux1);
                    j=23;  }
            else {
                    voltaje_de_flujo=10+(tanque[canal][13]-inch);  };
                    aux1++;
                        };
return (voltaje_de_flujo);
   };

/*****************************************************************************
* La siguiente funcion asigna a una variable de tipo entero el valor de      *
* la tecla proveniente del teclado sin importar cuando se halla tecleado     *
*****************************************************************************/

int   tecla (void){
    if (!kbhit())
            return (0);
            return (getch());
   };
```

FUNCIONES DE MANEJO DE PANTALLA.

```
/*****************************************************************************
* Las siguientes funciones se encargan de  hacer marcos en la pantalla       *
*****************************************************************************/
void pantalla (int renglon,int columna,int longx, int longy,char simbolo,
            int color){

   int i=0,j=0;
   textcolor(color);
   for (j=renglon;j<=(renglon+longy);j++){
          for (i=columna;i<=(columna+longx);i++) {
             gotoxy(i,j);
             cprintf("%c",simbolo);
             };
   };
   textcolor(0);  }
          }

void pantalla_80x24 (void) {
int columna=1,
   renglon=1,
   longx=78,
   longy=24,
   color=10;
char simbolo=176;
   pantalla (renglon,columna,longx,longy,simbolo,color);}

void Caratula (void)  {
    pantalla (8,18,44,10,219,3);
    textbackground(3);
    textcolor(1);
    gotoxy(21,9);
```

```
    cprintf("Universidad Nacional Autonoma de México");
    gotoxy(24,11);
    cprintf("Proyecto: ");
    gotoxy(34,12);
    cprintf("Detección de fallas en");
    gotoxy(34,13);
    cprintf("un sistema de control");
    gotoxy(26,16);
    cprintf("Cesar Augusto Mendoza Serrano");
    gotoxy(1,25);
    delay (5000);
  };

void Menu (void)  {
        textbackground(0);
        clrscr();
        pantalla(1,1,78,24,176,13);
        pantalla(7,14,53,10,219,7);
        textcolor(9);
        textbackground(7);
        gotoxy(33,8);
        cprintf(" M  E  N  U ");
        gotoxy(16,10);
        cprintf("1. Control P");
        gotoxy(16,11);
        cprintf("2. Control I");
        gotoxy(16,12);
        cprintf("3. Control PI");
        gotoxy(16,13);
        cprintf("4. Control PI con deteccion de fallas");
        gotoxy(16,14);
        cprintf("5. Control PID con deteccion de fallas");
        gotoxy(16,15);
        cprintf("6. Guarda proceso anterior para graficar en MATLAB ");
        gotoxy(16,16);
        cprintf("7. Fin");
    }

int seleccion (void)
{
int pos,letra;
   gotoxy(15,10);
   cprintf("□");
   pos=10;
   letra=getch();
   while (letra!=13){
         do { letra=getch();
           if (letra=='\0') {
             letra=getch();
            };
           if ((letra-55)==25) {
                          gotoxy(15,pos);
                          cprintf(" ");
                          pos++;
```

```
                        if (pos>16) {pos=10;};
                        gotoxy(15,pos);
                        cprintf("□");
            };
            if ((letra-56)==16) {
                        gotoxy(15,pos);
                        cprintf(" ");
                        pos--;
                        if (pos<10) {pos=16;};
                        gotoxy(15,pos);
                        cprintf("□"); };
          }while (letra-56!=16 && letra-55 !=25);
          letra=getch();
  } ;
  return (pos-9);
  }
```

Módulo 1

```
/*************************************************************************
* La siguiente rutina es de un control de accion proporcional, recibe las *
* alturas de referencia hdes1,hdes2 y el tiempo de muestre t que no es     *
* utilizado.                                                              *
*************************************************************************/

void Control_Proporcional (float hdes1,float hdes2,float t)
{
 char  ch;
int   cont=0,              /* Numero de muestras.              */
      i=0,j;

    textbackground(3);
    textcolor(10);
    gotoxy(27,1);
    cprintf(" Control Proporcional ");
    pantalla(9,20,25,1,219,3);
    textcolor (WHITE);
    gotoxy(20,9);
    cprintf("Ganancia para bomba1:");
    gotoxy(20,10);
    cprintf("Ganancia para bomba2:");
    gotoxy(42,9);
    scanf("%f",&Kp1);
    gotoxy(42,10);
    scanf("%f",&Kp2);
    gotoxy(55,25);
    cprintf("Teclea q para interrumpir");
    WriteDigital(1,1);
    WriteDigital(2,0);
    WriteDigital(1,0);
    WriteDigital(1,1);
    while (ch!= 'q' && ch!= 'Q')  {
```

```
        h1=AltenCentimetros(1);         /* Muestras de alturas     */
        h2=AltenCentimetros(2);
        h3=AltenCentimetros(3)-.7;      /* El -.7 es para calibrar */
        err1=hdes1-h1;                  /* Senal de error          */
        err2=hdes2-h2;
        proporcional1=Kp1*err1;         /* Ley de control          */
        proporcional2=Kp2*err2;

/* Lo siguiente es para acondicionar la senal a 100 cuando se ha      */
/* saturado a numeros demasiado grandes                               */

        if (proporcional1>100) {proporcional1=100;};
        if (proporcional2>100) {proporcional2=100;};

/* Se llenan los vectores que se usaran para generar los archivos en  */
/* matlab.                                                            */

        if (num_dat<=DIM) {
                altura1[num_dat]=h1;
                altura2[num_dat]=h2;
                altura3[num_dat]=h3;
                fluj_bomba1[num_dat]=proporcional1;
                fluj_bomba2[num_dat]=proporcional2;
                t_muestreo[num_dat]=num_dat*(t/1000);
                num_dat++;
         };

/* A continuacion el ciclo se da para: a) dar el retardo deseado      */
/* b) proporcionar el ciclo de reloj al canal 2 y poder arrancar la   */
/* bomba                                                              */
        for (j=0;j<t-7;j++) {
                delay(1);
                WriteDigital(2,i);
                i++;
                if (i==2) {i=0;};
         };

        potv1=De_flujo_a_volt(1,proporcional1);
        potv2=De_flujo_a_volt(2,proporcional2);
        WriteAnalogVolt(0,potv1);
        WriteAnalogVolt(1,potv2);
        if (AltenCentimetros(1)>61) {
                potv1=-10;}
        else {  if (AltenCentimetros(2)>61) {
                        potv2=-10;}
                 else {} };

        cont++;
        gotoxy (20,13);
        cprintf ("Altura tanque1:%f cm   ",AltenCentimetros(1));
        gotoxy (20,14);
        cprintf ("Altura tanque2:%f cm   ",AltenCentimetros(2));
        gotoxy (20,15);
        cprintf ("Altura tanque3:%f cm   ",AltenCentimetros(3)-.7);
```

```
        gotoxy (20,16);
        cprintf ("Flujo bomba1:%f ml/s  ",proporcional1);
        gotoxy (20,17);
        cprintf ("Flujo bomba2:%f ml/s  ",proporcional2);
        gotoxy (20,19);
        cprintf ("Numero de muestras: %d  ",cont);
        ch=tecla();     };
  gotoxy (50,25);
  cprintf ("Con p regresa a menu principal");
  do {
        gotoxy (79,25);
        ch=tecla();
  } while (ch!='p');

  ch='n';
  }
```

Módulo 2.

```
/*****************************************************************************
* La siguiente funcion es el de un controlador de accion integral con las    *
* referencias de alturas hdes1,hdes2 y tiempo de muestreo t                   *
*****************************************************************************/

void Control_Integral (float hdes1, float hdes2, float t)  {

int   i=0,cont=0,j;
    char  ch;

    textbackground(3);
    textcolor(10);
    gotoxy(27,1);
    cprintf(" Control Accion Integral ");
    pantalla(10,19,33,1,219,3);
    textcolor(WHITE);
    gotoxy(20,10);
    cprintf("Ganancia para bomba1 (Ki1):");
    gotoxy(20,11);
    cprintf("Ganancia para bomba2 (Ki2):");
    gotoxy(49,10);
    scanf("%f",&Ki1);
    gotoxy(49,11);
    scanf("%f",&Ki2);
    gotoxy(55,25);
    cprintf("Teclea q para interrumpir");
    pantalla(13,21,30,6,219,3);
    integral1=0;
    integral2=0;
    WriteDigital(1,1);
    WriteDigital(2,0);
    WriteDigital(1,0);
    WriteDigital(1,1);
```

```
    while (ch!= 'q' && ch!= 'Q')  {
            h1=AltenCentimetros(1);     /* Muestras para las alturas*/
            h2=AltenCentimetros(2);     /* (estados)               */
            h3=AltenCentimetros(3)-.7;
            err1=hdes1-h1;              /* Senal de error          */
            err2=hdes2-h2;

/****************** Ley de control ************************************/
            integral1=(t/1000)*err1+Ki1*integral1;
            integral2=(t/1000)*err2+Ki2*integral2;
            if (integral1>100) {integral1=100;};
            if (integral2>100) {integral2=100;};

/******** Se llenan los vectores para generar los archivos en matlab **********/

            if (num_dat<=DIM) {
                    altura1[num_dat]=h1;
                    altura2[num_dat]=h2;
                    altura3[num_dat]=h3;
                    fluj_bomba1[num_dat]=integral1;
                    fluj_bomba2[num_dat]=integral2;
                    t_muestreo[num_dat]=num_dat*(t/1000);
                    num_dat++;
             };

/********* A continuacion el ciclo nos da el retardo requerido y proporciona ***/
/******** el ciclo de reloj necesario para que la etapa de potencia sea liberada***/
            for (j=0;j<=t-6;j++) {      /* -4 es para calibrar lo  */
                    delay(1);           /* que dura la funcion     */
                    WriteDigital(2,i);  /* Nos da el ciclo de reloj*/
                    i++;
             if (i==2) {i=0;};
             };
            potv1=De_flujo_a_volt(1,integral1);
            potv2=De_flujo_a_volt(2,integral2);
            WriteAnalogVolt(0,potv1);
            WriteAnalogVolt(1,potv2);
            if (AltenCentimetros(1)>61) {
                    potv1=-10;}
            else { if (AltenCentimetros(2)>61) {
                            potv2=-10;}
                     else {} };
            cont++;
            gotoxy (22,14);
            cprintf ("Altura tanque1: %f cm.",AltenCentimetros(1));
            gotoxy (22,15);
            cprintf ("Altura tanque2: %f cm.",AltenCentimetros(2));
            gotoxy (22,16);
            cprintf ("Altura tanque3: %f cm.",AltenCentimetros(3)-.7);
            gotoxy (22,17);
            cprintf ("Flujo bomba1: %f ml/s",integral1);
            gotoxy (22,18);
            cprintf ("Flujo bomba2: %f ml/s",integral2);
            gotoxy (23,21);
```

```
            cprintf (" Numero de muestras: %d ",cont);
            ch=tecla();
  };
    gotoxy (50,25);
    cprintf ("Con p regresa a menu principal");
    do {
            gotoxy (79,25);
            ch=tecla();
     } while (ch!='p');
    ch='n';
   }
```

Módulo 3.

```
/*****************************************************************************
* La siguiente funcion es un controlados de accion proporcional-integral
* que recibe las alturas de referencia hdes1,hdes2 y el tiempo de muestreo      *
* t como parametros .                                                           *
*****************************************************************************/

void Control_PI (float hdes1,float hdes2,float t)  {
int   cont=0,          /* Muestras                */
      j,i=0;
    char ch;

    textbackground(3);
    textcolor(10);
    gotoxy(31,1);
    cprintf("Control Accion PI");
    pantalla(9,17,48,5,219,3);
    gotoxy(20,10);
    cprintf("Ganancia proporcional para bomba1 (Kp1):");
    gotoxy(20,11);
    cprintf("Ganancia proporcional para bomba2 (Kp2):");
    gotoxy(20,12);
    cprintf("Ganancia integral para bomba1 (Ti1):");
    gotoxy(20,13);
    cprintf("Ganancia integral para bomba2 (Ti2):");
    gotoxy(62,10);
    scanf("%f",&Kp1);
    gotoxy(62,11);
    scanf("%f",&Kp2);
    gotoxy(59,12);
    scanf("%f",&Ti1);
    gotoxy(59,13);
    scanf("%f",&Ti2);
    gotoxy(55,25);
    cprintf("Teclea q para interrumpir");
    pantalla(16,19,30,4,219,3);
    integral1=0;
    integral2=0;
    WriteDigital(1,1);      /* Se aseguran los estados de las entradas      */
```

```
    WriteDigital(2,0);      /* digitales                                    */
    WriteDigital(1,0);
    WriteDigital(1,1);
    while (ch!= 'q' && ch!= 'Q') {
            h1=AltenCentimetros(1);   /* Muestras o estados                 */
            h2=AltenCentimetros(2);
            h3=AltenCentimetros(3)-.7;    /* El -.7 es factor calibracion   */

/*************** Ley accion proporcional ********************************/
            proporcional1=Kp1*(hdes1-h1);
            proporcional2=Kp2*(hdes2-h2);
/*************** Senal de error *****************************************/
            err1=hdes1-h1;
            err2=hdes2-h2;
/*************** Ley accion integral ************************************/
            integral1=(t/1000)*err1+integral1;
            integral2=(t/1000)*err2+integral2;
/************** Accion PI ***********************************************/
            PI1=proporcional1+(Kp1/Ti1)*integral1;
            PI2=proporcional2+(Kp2/Ti2)*integral2;
/************* Si la accion PI se satura se coloca a 100 ml/s ***********/
           if (PI1>100) {PI1=100;};
           if (PI2>100) {PI2=100;};
/*************** Se llenan los vectores para generar los archivos para matlab******/
            if (num_dat<DIM) {
                    altura1[num_dat]=h1;
                    altura2[num_dat]=h2;
                    altura3[num_dat]=h3;
                    fluj_bomba1[num_dat]=PI1;
                    fluj_bomba2[num_dat]=PI2;
                    t_muestreo[num_dat]=num_dat*(t/1000);
                    num_dat++;
                    };

/* El siguiente ciclo proporciona el retardo requerido y da la senal de  */
/* reloj requerida para liberar la etapa de potencia.                     */
            for (j=0;j<=t-8;j++) {
                    delay(1);
                    WriteDigital(2,i);
                    i++;
                    if (i==2) {i=0;};
                    };

            potv1=De_flujo_a_volt(1,PI1);
            potv2=De_flujo_a_volt(2,PI2);
            WriteAnalogVolt(0,potv1);
            WriteAnalogVolt(1,potv2);
            cont++;
            gotoxy (20,16);
            cprintf ("Altura tanque1: %f cm",AltenCentimetros(1));
            gotoxy (20,17);
            cprintf ("Altura tanque2: %f cm",AltenCentimetros(2));
            gotoxy (20,18);
            cprintf ("Altura tanque3: %f cm",AltenCentimetros(3)-.7);
```

```
            gotoxy (20,19);
            cprintf ("Flujo bomba1: %f ml/s",Pl1);
            gotoxy (20,20);
            cprintf ("Flujo bomba2: %f ml/s",Pl2);
            gotoxy (20,23);
            cprintf (" Numero de muestras: %d ",cont);
            ch=tecla();
    };
    gotoxy (50,25);
    cprintf ("Con p regresa a menu principal");
    do {
            gotoxy (79,25);
            ch=tecla();
    } while (ch!='p');
    ch='n';
   }
```

Módulo 4.

```
/*****************************************************************
* La siguiente rutina es la encargada de la deteccion de fallas  *
* usando un control de accion PI y recibiendo como parametros    *
* las alturas de referencia hdes1,hdes2 asi como tambien el       *
* tiempo de muestreo.                                            *
*****************************************************************/

void DetectaFalla_y_controlPI (float hdes1,float hdes2,float t)  {
    char  ch,ch2;
     int   cont=0,j,aux=0,i=0,elemento;

    textbackground(3);
    textcolor(10);
    gotoxy(16,1);
    cprintf("Deteccion de Fallas con controlador de acción PI");
    pantalla(9,17,48,3,219,3);
    gotoxy(18,9);
    cprintf("Ganancia proporcional para bomba1 (Kp1):");
    gotoxy(18,10);
    cprintf("Ganancia proporcional para bomba2 (Kp2):");
    gotoxy(18,11);
    cprintf("Ganancia integral para bomba1 (Ti1):");
    gotoxy(18,12);
    cprintf("Ganancia integral para bomba2 (Ti2):");
    gotoxy(59,9);
    scanf("%f",&Kp1);
    gotoxy(59,10);
    scanf("%f",&Kp2);
    gotoxy(57,11);
    scanf("%f",&Ti1);
    gotoxy(57,12);
    scanf("%f",&Ti2);
    gotoxy(55,25);
```

```
    cprintf("Teclea q para interrumpir");
    pantalla(15,8,33,6,219,3);
    integral1=0;
    integral2=0;
    num_dat=0;
    WriteDigital(1,1);          /* Se aseguran los estados de */
    WriteDigital(2,0);          /* los canales.               */
    WriteDigital(1,0);
    WriteDigital(1,1);

/*************************************************************************
*    Empieza los muestreos                                               *
*************************************************************************/
    while (ch!= 'q' && ch!= 'Q') {
            h1=AltenCentimetros(1);
            h2=AltenCentimetros(2);
            h3=(AltenCentimetros(3)-.7);   /*El -.7 es sumando calibracion*/
/******************* Parte proporcional*********************************/
            proporcional1=Kp1*(hdes1-h1);
            proporcional2=Kp2*(hdes2-h2);
/******************* Error *********************************************/
            err1=hdes1-h1;
            err2=hdes2-h2;
/******************* Parte integral ************************************/
            integral1=(t/1000)*err1+integral1;
            integral2=(t/1000)*err2+integral2;
/******************* Accion controlador PI *****************************/
            PI1=(proporcional1+(Kp1/Ti1)*integral1);
            PI2=(proporcional2+(Kp2/Ti2)*integral2);
/*************************************************************************/
            v1=(PI1-uo1)/1e6;          /* Divido entre 1e6 para pasar  */
            v2=(PI2-uo2)/1e6;          /* cm^3 y ml. a m^3             */
            z1=(h1-ho1)/100;           /* Divido entre 100 para pasar  */
            z2=(h2-ho2)/100;           /* de cm a m.                   */
            z3=(h3-ho3)/100;
/****** Obtencion de los estados estimados y de los reales ***********/
/****** asi como de las salidas estimadas previo cambio de ***********/
/******              variables.                          ***********/
            Westim1=c11*Zestim1+c12*Zestim2+c13*Zestim3;
            Westim2=c21*Zestim1+c22*Zestim2+c23*Zestim3;
            Westim3=c31*Zestim1+c32*Zestim2+c33*Zestim3;
            w1=c11*z1+c12*z2+c13*z3;
            w2=c21*z1+c22*z2+c23*z3;
            w3=c31*z1+c32*z2+c33*z3;

            Zestim1=a11*Zestim1+a12*Zestim2+a13*Zestim3+b11*v1+b12*v2
                    +k11*(w1-Westim1)+k12*(w2-Westim2)+k13*(w3-Westim3);
            Zestim2=a21*Zestim1+a22*Zestim2+a23*Zestim3+b21*v1+b22*v2
                    +k21*(w1-Westim1)+k22*(w2-Westim2)+k23*(w3-Westim3);
            Zestim3=a31*Zestim1+a32*Zestim2+a33*Zestim3+b31*v1+b32*v2
                    +k31*(w1-Westim1)+k32*(w2-Westim2)+k33*(w3-Westim3);

/***************** Obtencion del residuo instantaneo*****************/
            res1=Westim1-w1;
```

```
        res2=Westim2-w2;
        res3=Westim3-w3;
/*************************************************************************/
/**** Se obtiene el residuo promedio a traves de una ventana (vector)*/
/** que contiene los residuos instantaneos.          **************/
        Resid1[10]=res1;
        Resid2[10]=res2;
        Resid3[10]=res3;
        Residuo_suma1=0;
        Residuo_suma2=0;
        Residuo_suma3=0;
/*** Se obtiene la suma de 10 residuos instantaneos guardados en un */
/*********************  vector.     ******************************/
        for (elemento=0;elemento<10;elemento++) {
                Resid1[elemento]=Resid1[elemento+1];
                Resid2[elemento]=Resid2[elemento+1];
                Resid3[elemento]=Resid3[elemento+1];
                Residuo_suma1=Residuo_suma1+Resid1[elemento];
                Residuo_suma2=Residuo_suma2+Resid2[elemento];
                Residuo_suma3=Residuo_suma3+Resid3[elemento];
        };
        Residuo_prom1=Residuo_suma1/10;    /* Se divide entre 10 */
        Residuo_prom2=Residuo_suma2/10;    /* para obtener promedio*/
        Residuo_prom3=Residuo_suma3/10;
        gotoxy (48,15);
        cprintf(" Residuo promedio 1: %f ",Residuo_prom1*100);
        gotoxy (48,16);
        cprintf(" Residuo promedio 2: %f ",Residuo_prom2*100);
        gotoxy (48,17);
        cprintf(" Residuo promedio 3: %f ",Residuo_prom3*100);
        if (cont>300) {
                if (Residuo_prom1> 0.00025 || Residuo_prom1< -0.00025) {
                        gotoxy(48,18);
                        cprintf(" Falla 1 ");}
                else {gotoxy(48,18);
                        cprintf("         ");};
                if (Residuo_prom2> 0.00025 || Residuo_prom2< -0.00025) {
                        gotoxy(48,19);
                        cprintf(" Falla 2 ");}
                else {gotoxy(48,19);
                        cprintf("         ");};
                if (cont>550) {
                        if (Residuo_prom3> .04500 || Residuo_prom3< -0.045) {
                            gotoxy(48,20);
                            cprintf(" Falla 3 ");}
                        else {gotoxy(48,20);
                            cprintf("         ");};
                 };      };
        cont++;

/*************************************************************************/
        if (Pl1>100) {Pl1=100;};
        if (Pl2>100) {Pl2=100;};
        potv1=De_flujo_a_volt(1,Pl1);
```

```
            potv2=De_flujo_a_volt(2,Pl2);
            WriteAnalogVolt(0,potv1);
            WriteAnalogVolt(1,potv2);

/******** Aqui empieza el llenado de los vectores para MATLAB**************/
            gotoxy(4,23);
            cprintf("\n inicia muestreo");
            if (ch=='m') {aux=1;};
            if (aux==1) {
                  gotoxy (4,23);
                  cprintf (" Numero de muestras: %d ",num_dat+1);
                  if (num_dat<DIM) {
                        altura1[num_dat]=h1;
                        altura2[num_dat]=h2;
                        altura3[num_dat]=h3;
                        fluj_bomba1[num_dat]=Pl1;
                        fluj_bomba2[num_dat]=Pl2;
                        t_muestreo[num_dat]=num_dat*(t/1000);
                        Residuo1[num_dat]=res1*100;
                        Residuo2[num_dat]=res2*100;
                        Residuo3[num_dat]=res3*100;
                        alt_estimad1[num_dat]=Zestim1*100;
                        alt_estimad2[num_dat]=Zestim2*100;
                        alt_estimad3[num_dat]=Zestim3*100;
                        err_alt1_real_estim[num_dat]=(z1-Zestim1)*100;
                        err_alt2_real_estim[num_dat]=(z2-Zestim2)*100;
                        err_alt3_real_estim[num_dat]=(z3-Zestim3)*100;
                        num_dat++;
                        /*cont++;*/
                  };    };

/**Ahora se da el tiempo de muestreo requerido y la senal de reloj**/
/*** para liberar las bombas.                          **************/
            for (j=0;j<=t-9;j++)
                  { delay(1);
                  WriteDigital(2,i);
                  i++;
                  if (i==2) {i=0;};
                  };
/*************************************************************************/
            gotoxy (10,16);
            cprintf ("Altura tanque1: %f cm ",AltenCentimetros(1));
            gotoxy (10,17);
            cprintf ("Altura tanque2: %f cm ",AltenCentimetros(2));
            gotoxy (10,18);
            cprintf ("Altura tanque3: %f cm ",AltenCentimetros(3)-.7);
            gotoxy (10,19);
            cprintf ("Flujo bomba1: %f ml/s ",Pl1);
            gotoxy (10,20);
            cprintf ("Flujo bomba2: %f ml/s ",Pl2);
   /*    gotoxy (10,22);
            cprintf ("Numero de muestras: %d",num_dat+1);
   */     ch=tecla();          };
    gotoxy (50,25);
```

```
    cprintf ("Con p regresa a menu principal");
    do {  gotoxy (79,25);
            ch=tecla();
    } while (ch!='p');
    ch='n';
    /*   while (ch!='s') {ch=tecla();};  */
     }
```

Módulo 5.

```
/*****************************************************************
* La siguiente rutina es la encargada de la deteccion de fallas  *
* usando un control de accion PI y recibiendo como parametros    *
* las alturas de referencia hdes1,hdes2 asi como tambien el       *
* tiempo de muestreo.                                            *
*****************************************************************/

void DetectaFalla_y_controlPID (float hdes1,float hdes2,float t)  {

int  cont=0,j,aux=0,i=0,elemento;
    char ch,ch2;

    textbackground(3);
    textcolor(10);
    gotoxy(17,1);
    cprintf("Detección de Fallas con controlador acción PID");
    pantalla(9,17,48,5,219,3);
    gotoxy(18,9);
    cprintf("Ganancia proporcional para bomba1 (Kp1):");
    gotoxy(18,10);
    cprintf("Ganancia proporcional para bomba2 (Kp2):");
    gotoxy(18,11);
    cprintf("Ganancia integral para bomba1 (Ti1):");
    gotoxy(18,12);
    cprintf("Ganancia integral para bomba2 (Ti2):");
    gotoxy(18,13);
    cprintf("Ganancia derivativa para bomba1 (Td1):");
    gotoxy(18,14);
    cprintf("Ganancia derivativa para bomba2 (Td2):");
    gotoxy(59,9);
    scanf("%f",&Kp1);
    gotoxy(59,10);
    scanf("%f",&Kp2);
    gotoxy(55,11);
    scanf("%f",&Ti1);
    gotoxy(55,12);
    scanf("%f",&Ti2);
    gotoxy(57,13);
    scanf("%f",&Td1);
    gotoxy(57,14);
    scanf("%f",&Td2);
    gotoxy(55,25);
```

```
    cprintf("Teclea q para interrumpir");
    pantalla(17,9,32,4,219,3);
    integral1=0;
    integral2=0;
    err1_anterior=0;
    err2_anterior=0;
    num_dat=0;
    WriteDigital(1,1);          /* Se aseguran los estados de          */
    WriteDigital(2,0);          /* los canales.                        */
    WriteDigital(1,0);
    WriteDigital(1,1);

/*************************************************************************
*    Empieza los muestreos                                               *
*************************************************************************/
    while (ch!= 'q' && ch!= 'Q') {
            h1=AltenCentimetros(1);
            h2=AltenCentimetros(2);
            h3=(AltenCentimetros(3)-.7);   /*El -.7 es sumando calibracion*/
/****************** Parte proporcional*********************************/
            proporcional1=Kp1*(hdes1-h1);
            proporcional2=Kp2*(hdes2-h2);
/****************** Error *********************************************/
            err1=hdes1-h1;
            err2=hdes2-h2;
/****************** Parte integral ************************************/
            integral1=(t/1000)*err1+integral1;
            integral2=(t/1000)*err2+integral2;
/****************** Parte Derivativa **********************************/
            derivativo1= (err1-err1_anterior)/(t/1000);
            derivativo2= (err2-err2_anterior)/(t/1000);
            err1_anterior=err1;
            err2_anterior=err2;
/****************** Accion controlador PID ****************************/
          PID1=(proporcional1+(Kp1/Ti1)*integral1+(Kp1*Td1)*derivativo1);
          PID2=(proporcional2+(Kp2/Ti2)*integral2+(Kp2*Td2)*derivativo2);
/*************************************************************************/
            v1=(PID1-uo1)/1e6;          /* Divido entre 1e6 para pasar*/
            v2=(PID2-uo2)/1e6;          /* cm^3 a m^3                 */
            z1=(h1-ho1)/100;           /* Divido entre 100 para pasar*/
            z2=(h2-ho2)/100;           /* de cm a m.                 */
            z3=(h3-ho3)/100;
/****** Obtencion de los estados estimados y de los reales ****************/
/****** asi como de las salidas estimadas previo cambio de ****************/
/******              variables.                                 **********/
            Westim1=c11*Zestim1+c12*Zestim2+c13*Zestim3;
            Westim2=c21*Zestim1+c22*Zestim2+c23*Zestim3;
            Westim3=c31*Zestim1+c32*Zestim2+c33*Zestim3;
            w1=c11*z1+c12*z2+c13*z3;
            w2=c21*z1+c22*z2+c23*z3;
            w3=c31*z1+c32*z2+c33*z3;

            Zestim1=a11*Zestim1+a12*Zestim2+a13*Zestim3+b11*v1+b12*v2
                    +k11*(w1-Westim1)+k12*(w2-Westim2)+k13*(w3-Westim3);
```

```
        Zestim2=a21*Zestim1+a22*Zestim2+a23*Zestim3+b21*v1+b22*v2
                +k21*(w1-Westim1)+k22*(w2-Westim2)+k23*(w3-Westim3);
        Zestim3=a31*Zestim1*a32*Zestim2+a33*Zestim3+b31*v1+b32*v2
                +k31*(w1-Westim1)+k32*(w2-Westim2)+k33*(w3-Westim3);

/***************** Obtencion del residuo instantaneo************************/
        res1=Westim1-w1;
        res2=Westim2-w2;
        res3=Westim3-w3;
/*********************************************************************/
/**** Se obtiene el residuo promedio a traves de una ventana (vector)          */
/** que contiene los residuos instantaneos.          ***********************/
        Resid1[10]=res1;
        Resid2[10]=res2;
        Resid3[10]=res3;
        Residuo_suma1=0;
        Residuo_suma2=0;
        Residuo_suma3=0;
/*** Se obtiene la suma de 10 residuos instantaneos guardados en un  ***********/
/********************** vector.      ***********************************/
        for (elemento=0;elemento<10;elemento++) {
                Resid1[elemento]=Resid1[elemento+1];
                Resid2[elemento]=Resid2[elemento+1];
                Resid3[elemento]=Resid3[elemento+1];
                Residuo_suma1=Residuo_suma1+Resid1[elemento];
                Residuo_suma2=Residuo_suma2+Resid2[elemento];
                Residuo_suma3=Residuo_suma3+Resid3[elemento];          };
        Residuo_prom1=Residuo_suma1/10;    /* Se divide entre 10        */
        Residuo_prom2=Residuo_suma2/10;    /* para obtener promedio  /
        Residuo_prom3=Residuo_suma3/10;
        gotoxy (47,18);
        cprintf(" Residuo promedio 1: %f ",Residuo_prom1*100);
        gotoxy (47,19);
        cprintf(" Residuo promedio 2: %f ",Residuo_prom2*100);
        gotoxy (47,20);
        cprintf(" Residuo promedio 3: %f ",Residuo_prom3*100);
        if (cont>300) {
                if (Residuo_prom1>0.00025 || Residuo_prom1<-0.00025) {
                        gotoxy(57,21);
                        cprintf(" Falla 1 ");}
                else {gotoxy(57,21);
                        cprintf("         ");};
                if (Residuo_prom2>0.00025  || Residuo_prom2<-0.00025) {
                        gotoxy(57,22);
                        cprintf(" Falla 2 ");}
                else {gotoxy(57,22);
                        cprintf("         ");};
                if (cont>550) {
                        if (Residuo_prom3>0.0450 || Residuo_prom3<-0.0450) {
                            gotoxy(57,23);
                            cprintf(" Falla 3 ");}
                        else {gotoxy(57,23);
                            cprintf("         ");}                  }
        };
```

```
        cont++;

/*************************************************************************/
        if (PID1>100) {PID1=100;};
        if (PID2>100) {PID2=100;};
        potv1=De_flujo_a_volt(1,PID1);
        potv2=De_flujo_a_volt(2,PID2);
        WriteAnalogVolt(0,potv1);
        WriteAnalogVolt(1,potv2);
/******** Aqui empieza el llenado de los vectores para MATLAB***************/
        if (ch=='m') {aux=1;}
        else {  gotoxy(5,23);
                cprintf("m inicia muestreo");
                }
        if (aux==1) {
                if (num_dat<DIM) {
                 gotoxy (5,23);
                 cprintf (" Numero de muestras: %d ",num_dat+1);
                        altura1[num_dat]=h1;
                        altura2[num_dat]=h2;
                        altura3[num_dat]=h3;
                        fluj_bomba1[num_dat]=PID1;
                        fluj_bomba2[num_dat]=PID2;
                        t_muestreo[num_dat]=num_dat*(t/1000);
                        Residuo1[num_dat]=res1*100;
                        Residuo2[num_dat]=res2*100;
                        Residuo3[num_dat]=res3*100;
                        alt_estimad1[num_dat]=Zestim1*100;
                        alt_estimad2[num_dat]=Zestim2*100;
                        alt_estimad3[num_dat]=Zestim3*100;
                        err_alt1_real_estim[num_dat]=(z1-Zestim1)*100;
                        err_alt2_real_estim[num_dat]=(z2-Zestim2)*100;
                        err_alt3_real_estim[num_dat]=(z3-Zestim3)*100;
                        num_dat++;
                        /*cont++;*/
                };
    };

/**Ahora se da el tiempo de muestreo requerido y la senal de reloj**/
/*** para liberar las bombas.                          **************/
        for (j=0;j<=t-9;j++)
                { delay(1);
                WriteDigital(2,i);
                i++;
                if (i==2) {i=0;};
                };
/*************************************************************************/
        gotoxy (10,17);
        cprintf ("Altura tanque1: %f cm ",AltenCentimetros(1));
        gotoxy (10,18);
        cprintf ("Altura tanque2: %f cm ",AltenCentimetros(2));
        gotoxy (10,19);
        cprintf ("Altura tanque3: %f cm ",AltenCentimetros(3)-.7);
        gotoxy (10,20);
```

```
            cprintf ("Flujo bomba1: %f ml/s ",PID1);
            gotoxy (10,21);
            cprintf ("Flujo bomba2: %f ml/s ",PID2);
            ch=tecla();          };
    gotoxy (50,25);
    cprintf ("Con p regresa a menu principal");
    do {  gotoxy (50,79);
            ch=tecla();
    } while (ch!='p');
    ch='n';
    /*   while (ch!='s') {ch=tecla();};  */
     }
```

Módulo 6.

```
/*************************************************************************
* La siguiente funcion genera un archivo para matlab a partir de        *
* vectores .                                                             *
*************************************************************************/

void manda_arch_a_matlab (void)  {

int  i=0;
    FILE *arch_salida;
    char nom_arch[20];
    char archivo[20];/* = "c:\\matlab\\cesar\\";*/

    pantalla(2,2,77,22,176,14);
    pantalla(8,12,55,8,219,5);
    textbackground(13);
    gotoxy(15,10);
    cprintf("A continuacion ");
    gotoxy (15,10);
    cprintf("A continuacion se grabara en un archivo de salida");
    gotoxy (15,11);
    cprintf("las lecturas de las alturas y el flujo de las bom-");
    gotoxy (15,12);
    cprintf("bas, durante las primeras 1000 muestras y poder ser");
    gotoxy (15,13);
    cprintf("utilizadas posteriormente en MATLAB.  ");
    gotoxy (18,15);
    cprintf ("Nombre de archivo de salida: ");
    gets(archivo);
    gotoxy(18,48);
    gets(archivo);
    puts(archivo);
    if ((arch_salida = fopen(archivo,"a")) == NULL)
            {  printf("Error en la apertura:");
               delay(5000);
               return;
            }
    fprintf(arch_salida,  "dat = [ \n");                for(i=1;i<=num_dat-2;i++)
```

```
            {fprintf(arch_salida,"%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f\n",
            altura1[i],altura2[i],altura3[i],fluj_bomba1[i],
            fluj_bomba2[i],t_muestreo[i],Residuo1[i],Residuo2[i],Residuo3[i],
            alt_estimad1[i],alt_estimad2[i],alt_estimad3[i],
            err_alt1_real_estim[i],err_alt2_real_estim[i[,
            err_alt3_real_estim[i]);
            };
    fprintf(arch_salida, " ]; \n");
    fprintf(arch_salida, "altura1 = dat(:,1); ");
    fprintf(arch_salida, "altura2 = dat(:,2); ");
    fprintf(arch_salida, "altura3 = dat(:,3); ");
    fprintf(arch_salida, "fluj_bomba1 = dat(:,4); ");
    fprintf(arch_salida, "fluj_bomba2 = dat(:,5); ");
    fprintf(arch_salida, "t_muestreo = dat(:,6); ");
    fprintf(arch_salida, "Residuo1 = dat(:,7); ");
    fprintf(arch_salida, "Residuo2 = dat(:,8); ");
    fprintf(arch_salida, "Residuo3 = dat(:,9); ");
    fprintf(arch_salida, "alt_estimad1 = dat(:,10); ");
    fprintf(arch_salida, "alt_estimad2 = dat(:,11); ");
    fprintf(arch_salida, "alt_estimad3 = dat(:,12); ");
    fprintf(arch_salida, "err_alt1_real_estim = dat(:,13); ");
    fprintf(arch_salida, "err_alt2_real_estim = dat(:,14); ");
    fprintf(arch_salida, "err_alt3_real_estim = dat(:,15); ");

/*************** Cerramos el archivo ****************************/
  /* strcpy(archivo,"") ; */
   rewind(arch_salida);
   fclose(arch_salida);
    return;
  }
```

PROGRAMA PRINCIPAL.

```
/*****************************************************************/
/*****************************************************************/
main()
{
 float   t,                    /* Tiempo de muestreo en segundos      */
         hdes1,hdes2;          /* Alturas deseadas para los tanques   */
 int     opcion;               /* Opcion para el menu                 */
 char    ch='n';

 PCIO(0X300);
 textbackground(0);
 clrscr();
 pantalla_80x24();
 Caratula ();
 clrscr();

  do { Menu ();
       opcion=seleccion();
```

```
        if (opcion!=6 && opcion!=7) {
                textbackground(7);
                clrscr();
                pantalla(2,2,77,22,176,11);
                pantalla(5,17,48,2,219,1);
                textbackground(1);
                textcolor(WHITE);
                gotoxy (18,5);
                cprintf ("Altura deseada tanque 1:      cm (máx. 60 cm)");
                gotoxy (18,6);
                cprintf ("Altura deseada tanque 2:      cm (máx. 60 cm)");
                gotoxy (18,7);
                cprintf ("Tiempo de muestreo:       ms  (mínimo 5 ms)");
                gotoxy (44,5);
                scanf("%f",&hdes1);
                gotoxy (44,6);
                scanf("%f",&hdes2);
                gotoxy (40,7);
                scanf("%f",&t);
                };

switch (opcion) {
        case 1: Control_Proporcional (hdes1,hdes2,t);
                ch='n';
                break;
        case 2: Control_Integral (hdes1,hdes2,t);
                ch='n';
                break;
        case 3: Control_PI   (hdes1,hdes2,t);
                ch='n';
                break;
        case 4: DetectaFalla_y_controlPI (hdes1,hdes2,t);
                ch='n';
                break;
        case 5: DetectaFalla_y_controlPID (hdes1,hdes2,t);
                ch='n';
                break;
        case 6: ch='s';
                break;
        case 7: ch='f';
                break;
        };
  clrscr();
  if (ch=='s' || ch=='S') {
         manda_arch_a_matlab ();
         num_dat=0;
         };
  clrscr();
  opcion=0;
 } while (ch!='f');
       return(0);
};
```

APÉNDICE C

En las páginas siguientes se muestra el equipo de los tres tanques (DTS200).

La primera ilustración muestra como se encuentra conformado el equipo. A la izquierda se encuentra la planta, con sus tres tanques, sus bombas y sus válvulas.

En la parte central se aprecia el actuador, el cuál contiene el servoamplificador, el módulo de perturbación y la unidad de adaptación de señales. En la izquierda, se observa la computadora que es utilizada para procesar los algoritmos de control y detección de fallas.

La segunda ilustración se muestra la planta en forma individual. Las cajas negras que se encuentran en la parte superior de cada tanque son los sensores. En la parte inferior izquierda se observa a las bombas y su conexión con el contenedor y con los tanques 1 y 2.

La tercera ilustración, muestra en forma particular una de las válvulas mecánicas a través de la cual se introduce una falla al sistema.

APÉNDICE C

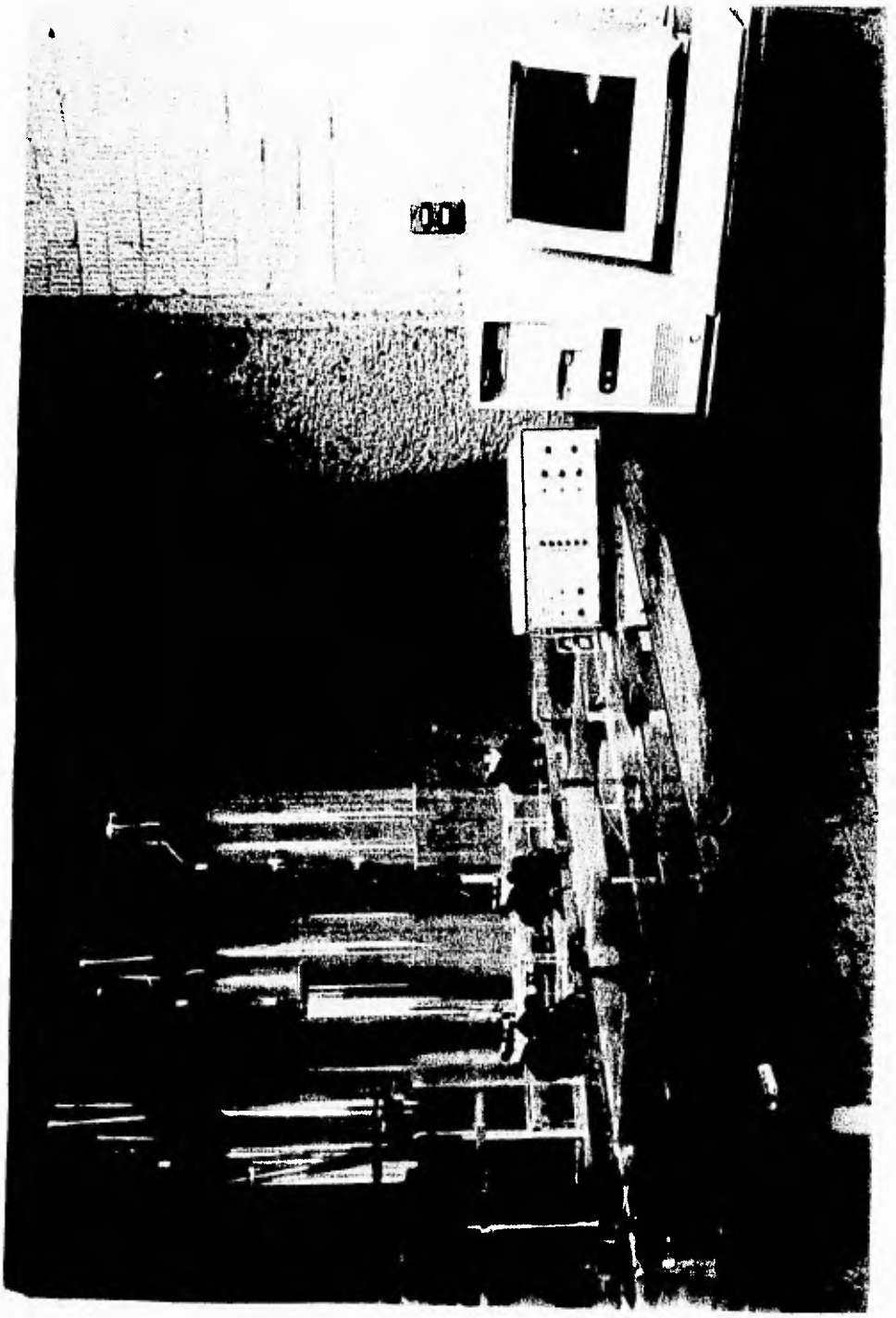
En las páginas siguientes se muestra el equipo de los tres tanques (DTS200).

La primera ilustración muestra como se encuentra conformado el equipo. A la izquierda se encuentra la planta, con sus tres tanques, sus bombas y sus válvulas.

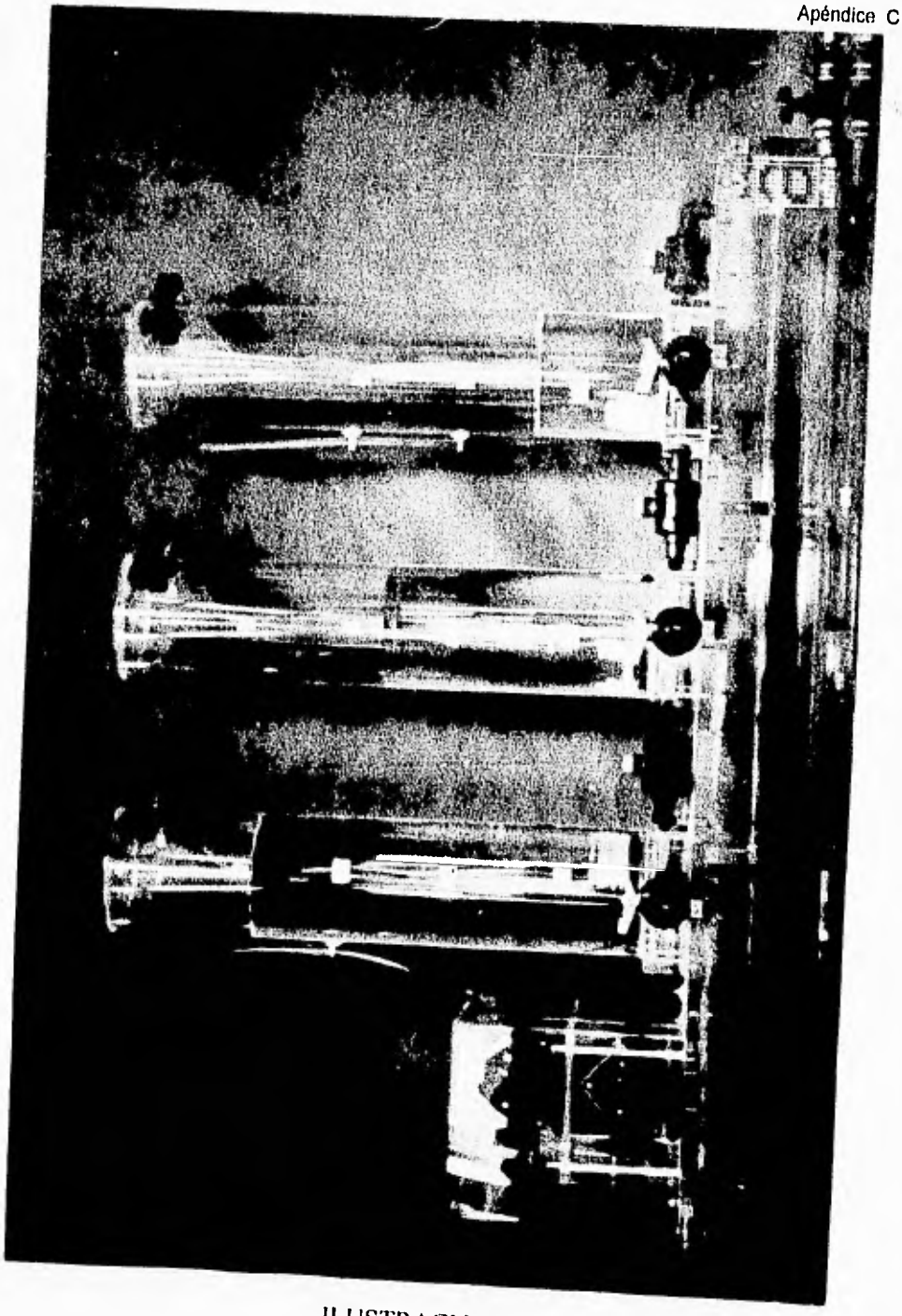
En la parte central se aprecia el actuador, el cuál contiene el servoamplificador, el módulo de perturbación y la unidad de adaptación de señales. En la izquierda, se observa la computadora que es utilizada para procesar los algoritmos de control y detección de fallas.

La segunda ilustración se muestra la planta en forma individual. Las cajas negras que se encuentran en la parte superior de cada tanque son los sensores. En la parte inferior izquierda se observa a las bombas y su conexión con el contenedor y con los tanques 1 y 2.

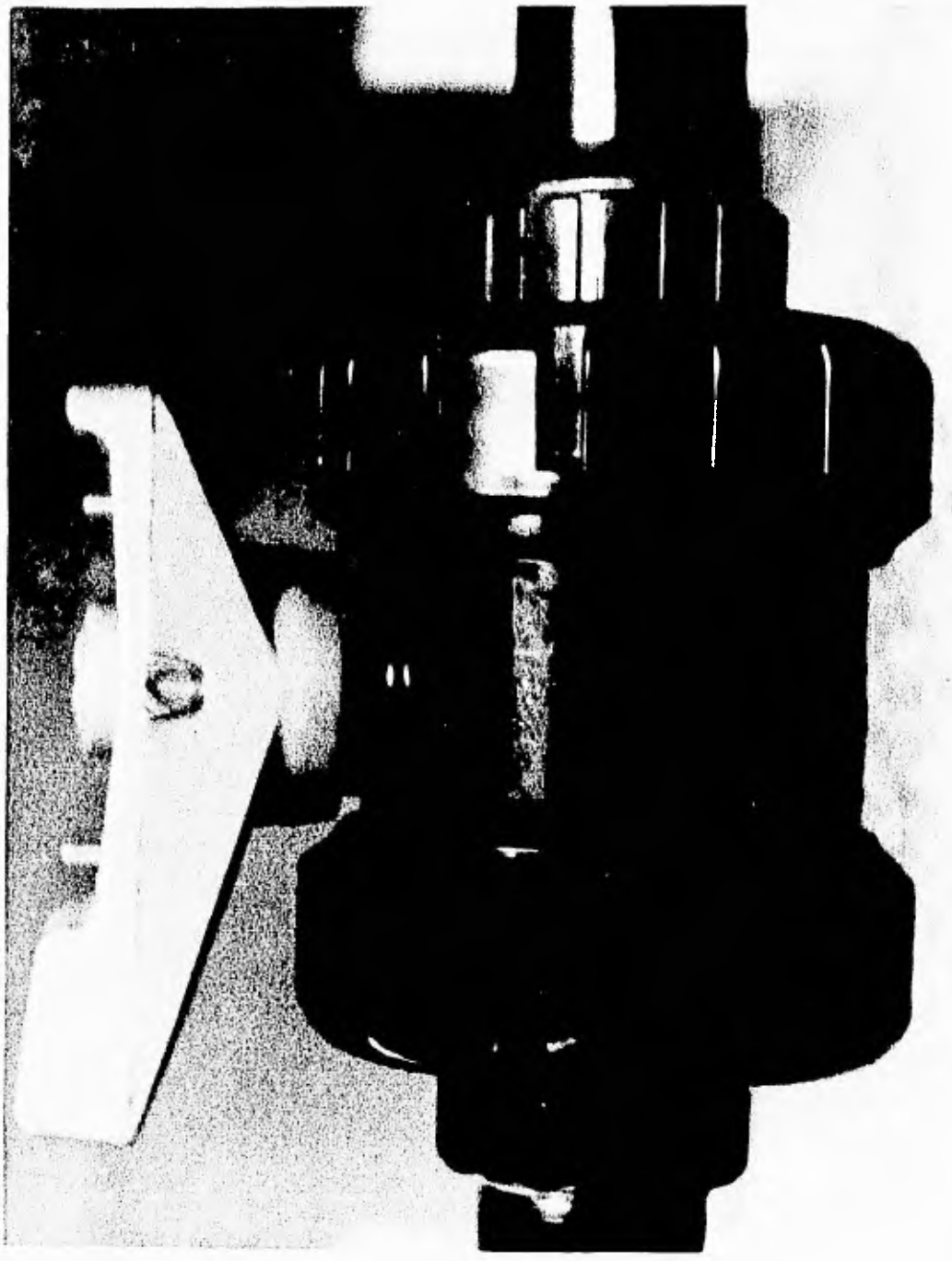
La tercera ilustración, muestra en forma particular una de las válvulas mecánicas a través de la cual se introduce una falla al sistema.



ILUSTRACION 1



ILUSTRACION 2



ILUSTRACION 3