Universidad de Huelva

Escuela Politécnica Superior

Departamento de Ciencias Agroforestales

SISTEMA DE CONTROL BORROSO PARA LA REGULACIÓN DE BALSAS DE PLANTAS DE ACUICULTURA SEMI-INTENSIVA

Trabajo profesional fin de carrera presentado por Elena Barrios Díaz en satisfacción de los requisitos necesarios para optar al grado de Ingeniero Técnico Industrial (Especialidad Mecánica), dirigido por la Dra. Inmaculada Pulido Calvo, profesora del área de Mecánica de Fluidos del Departamento de Ciencias Agroforestales.

La alumna, La directora,

Elena Barrios Díaz Dra. Inmaculada Pulido Calvo

Huelva, marzo de 2009

AGRADECIMIENTOS

En primer lugar, quisiera agradecer a la Dra. Inmaculada Pulido Calvo y al Dr. Juan Carlos Gutiérrez Estrada la enorme oportunidad que me han ofrecido al trabajar conmigo durante la elaboración de este proyecto fin de carrera. Espero sinceramente que hayan disfrutado tanto como yo.

Mi reconocimiento al Dr. Ignacio de la Rosa Lucas por su cordial disposición a facilitarme la información y datos relacionados con el capítulo *zona de estudio* de este trabajo.

También quisiera agradecer su comprensión a todos mis amigos, a los que tantas veces he dejado “abandonados” porque tenía que dedicar mi tiempo a la realización de este proyecto, en especial a Montserrat Valls Vitrià, autora de la fotografía que he elegido como portada del mismo.

Por último, me gustaría agradecer a mi familia todo el apoyo que me han ofrecido, no sólo durante la elaboración de este trabajo, sino a lo largo de toda mi vida. Espero que mis padres, a quienes admiro profundamente, puedan sentirse orgullosos.

ÍNDICE

1. INTRODUCCIÓN 1

1.1. Introducción a la lógica borrosa 4

1.2. Introducción a Microsoft Visual Basic 6 7

1.2.1. Entornos de desarrollo visual 8

1.2.2. Visual Basic 8

1.3. Introducción a la acuicultura 9

1.3.1. Los orígenes de la acuicultura 10

1.3.2. Sistemas de producción acuícola 10

1.3.3. Acuicultura marina 12

1.4. Antecedentes 12

1.5. Objetivos específicos 14

2. ZONA DE ESTUDIO 15

2.1. Introducción 16

2.2. La dorada 21

2.3. La lubina 22

2.4. Proceso de producción 23

3. MATERIAL Y MÉTODOS 25

3.1. Lógica borrosa 26

3.1.1. Sistemas de control borroso 26

3.1.2. Funciones de inclusión o pertenencia de conjuntos borrosos 29

3.1.3. Particiones borrosas 29

3.1.4. Reglas borrosas 30

3.1.5. Métodos de inferencia borrosa y desborrosificación 31

3.2. Microsoft Visual Basic 6 32

3.2.1. Proyectos 32

3.2.2. Archivos de un proyecto 33

3.2.3. Diseñadores y editores 34

3.2.4. Formularios 35

3.2.5. Controles ActiveX 35

3.2.6. Propiedades 36

3.2.7. Eventos y métodos 37

3.2.8. Programación estructurada y orientación a objetos 37

3.2.9. Funciones y procedimientos 38

3.2.10. Cuestiones de ámbito 39

3.2.11. Clases y objetos 39

3.3. Caudal en compuertas de corriente subálvea 40

3.3.1. Cálculo de las curvas del coeficiente de descarga 42

3.4. Evaluación de los modelos 44

3.5. Aplicación informática "*Sistema de control borroso para la regulación de balsas de plantas de acuicultura semi-intensiva*" 45

4. RESULTADOS 50

4.1. Resultados obtenidos al utilizar la memoria asociativa borrosa inicial A 51

4.2. Resultados obtenidos al utilizar una memoria asociativa borrosa B menos restrictiva que la inicial 54

4.3. Resultados obtenidos al utilizar una memoria asociativa borrosa C más restrictiva que la inicial 56

5. DISCUSIÓN 61

6. CONCLUSIONES 64

REFERENCIAS 66

ANEXO 74

a) Código fuente del formulario "Programa.frm" 75

b) Código fuente del formulario "FAM.frm" 97

c) Código fuente del formulario "Variables.frm" 98

d) Código fuente del formulario "CT.frm" 99

e) Código fuente del formulario "CNA.frm" 100

f) Código fuente del formulario "CA.frm" 101

1. INTRODUCCIÓN

1.1. Introducción a la lógica borrosa

1.2. Introducción a Microsoft Visual Basic 6

1.2.1. Entornos de desarrollo visual

1.2.2. Visual Basic

1.3. Introducción a la acuicultura

1.3.1. Los orígenes de la acuicultura

1.3.2. Sistemas de producción acuícola

1.3.3. Acuicultura marina

1.4. Antecedentes

1.5. Objetivos específicos

1. INTRODUCCIÓN

El campo de la acuicultura ha sido considerado por muchos de sus profesionales más como un arte que como una ciencia, ya que el éxito de las operaciones acuícolas ha sido atribuido a la intuición del director de la planta de acuicultura más que a la interpretación de la fisiología, la ecología y el comportamiento de las especies cultivadas. Por este motivo, los directores de las explotaciones acuícolas han sido, generalmente, reacios a confiar sus cultivos a sistemas de control automatizado. Sin embargo, recientes investigaciones y operaciones comerciales han comenzado a adoptar nuevas tecnologías, por lo que la acuicultura como ciencia está evolucionando (Lee, 2000).

La acuicultura ha sufrido un proceso de desarrollo continuado en los últimos años, iniciándose como una actividad económica complementaria de la pesca extractiva y siendo en la actualidad una actividad independiente capaz de conseguir productos de elevada calidad. Según la FAO (2000), la acuicultura proporcionó a principios de este siglo alrededor del 35% del pescado para consumo humano a nivel mundial, estimándose que para el año 2010 este valor alcance el 40%. En consecuencia, a lo largo de los próximos años, la acuicultura debe posicionarse como una industria sólida y consolidada capaz de conseguir una producción competitiva y sostenible desde un punto de vista económico, social y ambiental.

En Europa, la producción acuícola de especies de alto valor comercial supone en la actualidad el 6.4% de la producción mundial, del cual el 74% se corresponde con la producción de peces. En lo que respecta a la posición de los distintos países en la producción acuícola europea desde 1975, Francia y España han ocupado las dos primeras posiciones de forma alternativa, si bien, a partir de 1997, nuestro país se alzó al primer puesto, registrando a finales del 2003 un nivel de producción 3.4 veces superior a la media europea. De estas cifras se deduce que España es la primera potencia en el contexto europeo.

En España, atendiendo a la distribución de la producción acuícola por Comunidades Autónomas, un 85% se concentra en Galicia. En segundo lugar se sitúa Andalucía (2.7%), seguida por la Región de Murcia (2.4%) y la Comunidad Valenciana (1.8%). La concentración de la producción acuícola en Galicia sobre el conjunto de España es consecuencia de la elevada producción de mejillón. Si sólo se considera la producción piscícola los porcentajes de participación de las anteriores Comunidades Autónomas son: 19.9% en Galicia, 14.2% en Andalucía, 14.1% en Murcia y 10.4% en la Comunidad Valenciana. Por tanto, es de destacar que Andalucía es la segunda región española en producción piscícola.

El 60% de la producción del sector acuícola andaluz procede de cultivos semi-intensivos que se caracterizan, de forma general, por instalaciones en tierra cerca de la costa que toman el agua de una fuente de suministro (río o mar) mediante una estación de bombeo. El agua se distribuye mediante canales, normalmente sin revestir, a los distintos estanques o balsas (normalmente sin revestir también) donde se procede a la cría de una especie acuícola determinada.

En este contexto, uno de los objetivos horizontales prioritarios en este tipo de plantas de acuicultura es el control de la calidad del agua en las balsas de producción. Así, la densidad de los peces en las balsas está directamente relacionada con el consumo de oxígeno y la acumulación de catabolitos tóxicos (principalmente amonio), factores que determinarán la regulación del caudal de agua de entrada y salida en las balsas y, por tanto, las necesidades de agua a bombear desde la fuente de suministro. La consecución de este objetivo global en el sector acuícola sustenta, asimismo, las actuales políticas de los países industrializados que se basan principalmente en crecimientos económicos sostenibles considerando la calidad de la producción.

Para esta regulación de caudales en las balsas de cultivo, se ha utilizado en este trabajo un sistema de control borroso cuyas metodologías se incluyen dentro del campo de conocimiento denominado “Inteligencia Computacional” o “Soft-computing” (Zadeh, 1965). La adopción de esta modelación heurística obedece esencialmente a los resultados satisfactorios que se están obteniendo en la solución de problemas de temáticas diferenciadas (ASCE, 2000a,b; Marsili-Libelli, 2004; Gutiérrez-Estrada *et al*., 2005; Pulido-Calvo y Portela, 2007a,b; Pulido-Calvo y Gutiérrez-Estrada, 2009).

Los modelos de lógica borrosa están diseñados para tratar de imitar cómo el cerebro humano tiende a clasificar información procedente de datos imprecisos tales como: la temperatura del agua está “fría”, “templada” o “caliente”. En los modelos de lógica borrosa, la información se procesa en términos de conjuntos borrosos, los cuales se precisan a través de la definición de una función de pertenencia asociada. La inferencia específica es procesada entonces mediante el conjunto borroso combinado con algunas reglas borrosas (es decir, *si* la temperatura del agua de la balsa es “muy alta” *y* el nivel de amonio es “muy alto” *entonces* la apertura de la compuerta que regula el paso de agua a la balsa es “muy grande”).

Con este proyecto se pretenden evaluar e indicar posibles líneas de actuación en la gestión de los sistemas de almacenamiento de agua en sistemas acuícolas semi-intensivos de producción que satisfagan los criterios de funcionamiento en cuanto a cantidad y calidad de la producción. La consecución del objetivo planteado culmina con el desarrollo de una aplicación de ordenador en un entorno amigable (Visual Basic) que se constituye como un “sistema de ayuda a la toma de decisiones” dinámico, multidisciplinar,

interactivo y de fácil manejo para los técnicos responsables del diseño y la gestión de los sistemas de producción.

La aplicación y evaluación de los modelos desarrollados se ha realizado en la empresa "*Langostinos de Huelva, S.A.*", planta acuícola de cultivo semi-intensivo de principalmente dorada (*Sparus aurata*), ubicada en el término municipal de Cartaya (provincia de Huelva), con una producción media anual de 600 toneladas. Según las estadísticas de producción de 2005, en Andalucía destaca el cultivo de dorada con porcentajes del 57%, 55% y 49% de la producción total en las fases de criadero, preengorde y engorde, respectivamente.

En las secciones siguientes de este capítulo se muestra una breve introducción a las metodologías y herramientas utilizadas en este proyecto (lógica borrosa y Visual Basic), así como una descripción global de conceptos relacionados con el conocimiento de las actividades de producción acuícola. El capítulo culmina con una relación de trabajos que han servido de antecedentes, así como con la enumeración de los objetivos específicos del proyecto.

1.1. Introducción a la lógica borrosa

Los sistemas basados en lógica borrosa emulan la manera en que el cerebro razona o piensa. La denominada lógica borrosa (*fuzzy logic*) permite tratar información imprecisa en términos de conjuntos borrosos o difusos (imprecisos, en definitiva). Estos conjuntos borrosos se combinan en reglas para definir acciones como, por ejemplo, "*si* la temperatura es alta, *entonces* enfría mucho". De esta manera, los sistemas de control basados en lógica borrosa combinan unas variables de entrada (definidas en términos de conjuntos borrosos) por medio de grupos de reglas que producen uno o varios valores de salida. Los sistemas borrosos permiten modelar cualquier proceso no lineal y aprender de los datos haciendo uso de determinados algoritmos de aprendizaje. Además, gracias a la simplicidad de los cálculos (sumas y comparaciones, fundamentalmente), normalmente pueden realizarse en sistemas baratos y rápidos.

Hay que señalar que dentro de los sistemas borrosos se incluyen diversas teorías, como la teoría de conjuntos borrosos, extensión de la teoría de conjuntos clásica, o la lógica borrosa, que puede ser considerada una ampliación de las lógicas *n*-valuadas propuestas por Lukasiewiez en 1930, que son a su vez extensión de la lógica trivaluada (verdadero, falso e indeterminado) propuesta por Kosko en 1993.

En la teoría conjuntos clásica, algo está incluido completamente en un conjunto o no lo está en absoluto. Esta situación puede describirse asignando un 1 a todos los elementos incluidos en el conjunto y un 0 a los no incluidos. La función que asigna estos valores es denominada función de inclusión o pertenencia (*membership function*). Los conjuntos borrosos permiten describir el grado de pertenencia o inclusión de un objeto al concepto dado por la etiqueta que le da nombre asignando un número real entre 0 y 1.

Las bases teóricas de la lógica borrosa (*fuzzy logic*) fueron enunciadas en 1965 por Lofti A. Zadeh, profesor de Ingeniería Eléctrica en la Universidad de California en Berkeley; sin embargo, Zadeh no presenta la teoría básica de los controladores borrosos hasta 1973. Aunque al principio su trabajo fue recibido fríamente (especialmente en Estados Unidos), a partir de él otros investigadores comenzaron a aplicar la lógica borrosa a diversos procesos. Así, desde Mamdani, quien aplica la lógica borrosa a un sistema de control de vapor, se han venido sucediendo numerosas aplicaciones. Otra de las más clásicas quizás sea la de Smidth y sus colaboradores, quienes en 1980 aplican esta técnica al control de hornos rotativos de una cementera.

Pero, sin duda alguna, donde más éxito han tenido los sistemas borrosos ha sido en Japón. En este sentido, una breve historia sería:

- **1972:** Grupo de trabajo en sistemas *fuzzy* en el TIT (*Tokio Institute of Technology*).
- **1972-1981:** Informe anual *General Problems on Fuzzy Systems*.
- **1972:** *Workshop* sobre sistemas borrosos por la sociedad para la instrumentación e ingenieros de control.
- **1983:** Aplicación del control borroso al proceso de purificación de agua, Fuji Electric y TIT.
- **1984:** Capítulo IFSA (*International Fuzzy Systems Association*) de Japón.
- **1983:** Primer *symposium* anual sobre sistemas borrosos.
- **1987:** Aplicación del control borroso al tren metropolitano de la ciudad de Sendai (Japón), desarrollado por Hitachi.
- **1987:** Primera “moda” *fuzzy*.
- **1987:** Segundo congreso IFSA, Tokio.
- **1988:** *Workshop* internacional sobre aplicación de sistemas borrosos en Iizuka.

- **1989-1994:** Proyecto LIFE (*Laboratory for International Fuzzy Engineering research*), Ministerio de Comercio Internacional e Industria de Japón (MITI).
- **1989-1993:** Proyecto de investigación en sistemas borrosos (STA).
- **1989:** Sociedad para la teoría y sistemas *fuzzy* (SOFT).
- **1989:** Aplicación de control borroso a unidad de suministro de agua caliente para uso doméstico, desarrollada por Matsushita.
- **1989:** Premio Honda al profesor Zadeh.
- **1990:** Conferencia sobre lógica borrosa y redes neuronales en Iizuka.
- **1990:** Reunión chino-japonesa sobre conjuntos y sistemas borrosos.
- **1993:** *Second IEEE International Conference on Fuzzy Systems*, San Francisco.
- **1994:** *Third IEEE International Conference on Fuzzy Systems*, Orlando.
- **1995:** *Fourth IEEE International Conference on Fuzzy Systems*, Yokohama.
- **1996:** *Fifth IEEE International Conference on Fuzzy Systems*, New Orleans.
- **1997:** *Sixth IEEE International Fuzzy Systems Conference FUZZ-IEEE'97*, Barcelona.
- **1998:** *IEEE International Fuzzy Systems Conference FUZZ-IEEE'98*, Anchorage.
- **1999:** *IEEE International Fuzzy Systems Conference FUZZ-IEEE'99*, Seúl.
- **2000:** *IEEE International Fuzzy Systems Conference FUZZ-IEEE'2000*, San Antonio (Texas).

Especial mención merece la creación de LIFE (*Laboratory for International Fuzzy Engineering research*) en marzo de 1989, auspiciado por el Ministerio de Comercio Internacional e Industria de Japón (MITI). Su capital es de compañías privadas japonesas y del propio Ministerio al 50% y su Presidente está en el Instituto de Tecnología de Tokio (TIT). En su sede trabajan en la actualidad alrededor de 30 investigadores a tiempo completo.

En los Estados Unidos y Europa solamente se empezó a dar importancia a la lógica borrosa cuando empezó a llegar información desde Japón sobre sus muchas aplicaciones prácticas. A partir de entonces, numerosas empresas

norteamericanas (como NASA, Boeing, Ford, Rochwell o Bell) comenzaron a aplicar la lógica borrosa (Martín-del-Brío y Sanz-Molina, 2001).

1.2. Introducción a Microsoft Visual Basic 6

La aparición de Windows 3.0 marcó, sin duda, un antes y un después en el mundo de los compatibles PC. Esta versión de Windows consiguió lo que las anteriores no habían conseguido: que muchos usuarios abandonaran poco a poco el entorno basado en texto para avanzar hacia un entorno gráfico. Hoy es indiscutible la aceptación de este tipo de entorno, calculándose que millones de personas en todo el mundo utilizan Windows. A este éxito contribuyó de forma muy importante Windows 95, un Windows que dejó de ser un entorno gráfico para convertirse en un sistema operativo completo, y más recientemente Windows NT, que poco a poco fue haciéndose un hueco en la empresa y estaciones de trabajo de profesionales con requerimientos mayores de estabilidad y seguridad.

Este hecho conllevó que los desarrolladores tuvieran que adaptarse a este nuevo mundo pasando de un medio, en el que la aplicación que se estaba ejecutando era dueña del sistema, a otro donde las aplicaciones tenían que cooperar entre sí. De una filosofía de codificación secuencial a otra dirigida por eventos. En muchos casos fue necesario abandonar el lenguaje que hasta entonces se utilizaba, para utilizar el hasta entonces lenguaje por excelencia sobre Windows, C.

Paradójicamente, aunque para el usuario un entorno gráfico es sinónimo de mayor facilidad de uso, para el programador suele ser todo lo contrario significando una cantidad de trabajo cada vez mayor. Es lógico, generalmente cuanto más fácil de usar sea un programa más complejo será éste desde el punto de vista de su codificación, ya que el programador debe tener en cuenta elementos que antes se dejaban al usuario.

Por fortuna este panorama está cambiando con rapidez y en el mercado han ido apareciendo nuevos lenguajes y herramientas de desarrollo que facilitan en mucho la tarea del programador.

1.2.1. Entornos de desarrollo visual

Desde la aparición de los primeros lenguajes de programación actuales, FORTRAN, COBOL y BASIC, hasta hace varios años, los entornos de programación han ido evolucionando lentamente. Los compiladores por línea de comandos dejaron paso a los llamados “entornos de desarrollo integrados”, desde los que es posible escribir el programa, compilarlo y depurarlo, sustituyendo en gran parte a los editores, compiladores tradicionales y depuradores separados. Sin embargo, ha sido sobre la plataforma Windows donde se ha evolucionado con mayor rapidez, pasando de estos EDI a los llamados entornos de desarrollo visual.

Estos entornos se caracterizan, principalmente, porque el programador desarrolla su aplicación básicamente a partir del diseño de una interfaz. Crea una ventana, introduce en ella diversos componentes que simbolizan datos o acciones a llevar a cabo, establece las propiedades de esos componentes y, donde es preciso, añade algo de código. El tiempo de desarrollo de cualquier aplicación se ve reducido de meses o semanas a días. Además, es posible olvidarse por completo de cómo se gestiona el ratón, de restablecer la porción de pantalla que había debajo al ocultar una ventana, de permitir al usuario mediante ciertas teclas moverse de un lugar a otro. Todo esto y más queda en manos del sistema y, en parte, del lenguaje. Por lo tanto, el programador puede centrarse en lo que realmente le interesa, el funcionamiento lógico de la aplicación, y no en la interacción del usuario con ésta.

1.2.2. Visual Basic

Visual Basic fue uno de los primeros entornos de desarrollo visual para Windows. Pretendía facilitar el desarrollo de programas Windows a todos los programadores sin necesidad de tener que aprender otro lenguaje. Visual Basic no se parece nada, en apariencia, a aquel GwBasic o QuickBasic que los programadores acostumbraban a utilizar sobre DOS, aunque, por supuesto, sigue siendo BASIC. Eso sí, un BASIC muy evolucionado, que incorpora múltiples tipos de datos, la posibilidad de crear funciones y procedimientos, estructuras de control típicas de Pascal o C, orientación a objetos, capaz de crear nuevos componentes, etc.

La mayor parte de los programadores en un momento u otro han utilizado BASIC, es más, la mayoría de ellos aprendieron a programar con este lenguaje. Por ello, Visual Basic abre la puerta a la programación sobre Windows a prácticamente todo el mundo y no sólo a aquellos que conocen C y

el funcionamiento de Windows a fondo. Si el programador desea desarrollar sus propias aplicaciones para este entorno de forma simple, desde luego Visual Basic es el lenguaje apropiado.

Desde Visual Basic se tiene al alcance la mayoría de las posibilidades Windows, se puede crear aplicaciones MDI, con soporte OLE, multimedia, acceso a bases de datos, etc. Bastante más de lo que actualmente ofrecen muchos compiladores de C o C++. Si los objetos facilitados con Visual Basic y el propio lenguaje no cubren una cierta área, siempre se tendrá acceso a la API de Windows, por lo que prácticamente se tienen todas las puertas abiertas. A pesar de ello, Visual Basic, como se ha dicho antes, sigue siendo BASIC, por lo que siempre existirán operaciones, generalmente de bajo nivel, que son más accesibles a lenguajes como C.

Visual Basic 6.0 nos permite, además de lo dicho anteriormente, desarrollar aplicaciones de 32 bits generando verdadero código nativo, lo que redunda en una mayor velocidad de ejecución. La creación de componentes ActiveX es una tarea fácil con Visual Basic 6.0, ya que abre las puertas al desarrollo de objetos que podrán ser usados no sólo en las aplicaciones creadas por el usuario, sino en cualquier otro entorno que soporte este tipo de componentes o incluso en páginas web. Visual Basic 6.0 puede ser ampliado, es posible añadir otras opciones al menú a partir de programas externos e incluso escribiendo esas nuevas opciones en el propio Visual Basic. Además de aplicaciones de tipo “estándar”, también se pueden crear aplicaciones IIS, para trabajar conjuntamente con un servidor web, e incluso aplicaciones DHTML (Charte, 1998).

1.3. Introducción a la acuicultura

El término acuicultura engloba todas las actividades que tienen por objeto la producción, crecimiento (desarrollo) y comercialización de organismos acuáticos, animales o vegetales, de aguas dulces, salobres o saladas. Esto implica el control de las diferentes etapas, desde el huevo hasta la cosecha, proporcionando a los organismos los medios adecuados para su crecimiento y engorde. Algas, moluscos (malacocultura), crustáceos (carcinocultura) y peces (piscicultura) son los grandes grupos objetivo de la acuicultura.

1.3.1. Los orígenes de la acuicultura

La acuicultura se remonta a tiempos muy remotos. Existen referencias de prácticas de cultivo de mújol y carpa en la antigua China, Egipto, Babilonia, Grecia, Roma y en otras culturas euroasiáticas y americanas.

Las referencias más antiguas datan de en torno al año 3500 a. C. en la antigua China. Además, en el año 1400 a. C. ya existían leyes de protección frente a los ladrones de pescado. El primer tratado sobre el cultivo de carpa data del año 475 a. C., atribuido al chino Fan-Li, también conocido como Fau Lai.

Entre griegos y romanos, existen numerosas referencias. Aristóteles y Plinio el Viejo escribieron sobre el cultivo de ostras. Plinio, en concreto, atribuye al general romano Lucinius Murena el invento del estanque de cultivo y cita las grandes ganancias de su explotación comercial en el siglo I. Séneca también tuvo su opinión sobre la piscicultura, la cual fue bastante crítica: "*La invención de nuestros estanques de peces, esos recintos diseñados para proteger la glotonería de las gentes del riesgo de enfrentarse a las tormentas*".

En la cultura occidental actual, la acuicultura no recobró fuerza hasta la Edad Media en monasterios y abadías, aprovechando estanques alimentados por cauces fluviales en los que el cultivo consistía en el engorde de carpas y truchas.

En el año 1758 se produjo un importante descubrimiento, la fecundación artificial de huevos de salmones y truchas por Stephen Ludvig Jacobi, un investigador austríaco, aunque su investigación no salió del laboratorio y quedó en el olvido.

En 1842, dos pescadores franceses, Remy y Gehin, obtuvieron puestas viables totalmente al margen del hallazgo de Jacobi. Lograron alevines de trucha, que desarrollaron en estanque con éxito. El descubrimiento llevó a la Academia de Ciencias de París a profundizar en el hallazgo y con ello a la creación del Instituto de Huninge, el primer centro de investigación en acuicultura.

1.3.2. Sistemas de producción acuícola

Como en cualquier sistema de producción agropecuaria, existen diferentes tipos de cultivos según la intensidad y tecnificación del cultivo. En los

sistemas de producción acuícola es posible diferenciar entre sistemas de producción extensiva y sistemas de producción semi-intensiva o intensiva.

Los sistemas de producción extensiva son sistemas cultivo de baja intensidad y tecnología, en los que se aprovechan condiciones naturales favorables. Los cultivos extensivos más conocidos son los de organismos filtradores marinos, como ostras, almejas y mejillones, y los de macroalgas marinas, que se realizan directamente sobre fondos arenosos de áreas intermareales o sobre estructuras apoyadas en el fondo, como estacas y mesas de cultivo, o flotantes, como bateas y líneas. En ellos se procede a la siembra, siendo el proceso de alimentación y engorde es natural.

A pesar de ser sistemas extensivos, pueden alcanzar unos niveles de productividad muy elevados. Es el caso del cultivo de mejillón en las rías gallegas, donde la gran riqueza de las aguas y las beneficiosas condiciones ambientales disparan las tasas de crecimiento y calidad del producto.

Los sistemas extensivos son bastante utilizados en la producción de fitoplancton y zooplancton en climas cálidos, con grandes dosis de radiación solar. Balsas de agua enriquecidas con nutrientes minerales se utilizan para la producción de microalgas, destinadas a alimentación humana, cosmética o herbodietética, o como alimento de un segundo cultivo extensivo de zooplancton, utilizado posteriormente en alimentación larvaria de peces y crustáceos.

La piscicultura extensiva es algo anecdótico. Existen experiencias con lagunas oligotróficas sembradas con nutrientes minerales para activar la producción de fitoplancton y activar toda la cadena trófica, con el objetivo de cosechar posteriormente especies de peces para consumo, pero a esto no se le puede llamar propiamente acuicultura.

Por otra parte, los sistemas de producción semi-intensiva e intensiva son sistemas de cultivo más controlados y de mayor rendimiento, en los que el grado de tecnología e intervención es mucho mayor a los extensivos.

Los cultivos de peces en jaulas flotantes, directamente en el mar o en lagos, son sistemas semi-intensivos. El agua es la del medio, sin ningún sistema de bombeo, pero se aportan alimentos y se realiza un mínimo control del cultivo. También son sistemas semi-intensivos los cultivos en estanques y canales en circuito abierto o semi-abierto aprovechando aguas corrientes, algo muy frecuente en truchicultura.

Los cultivos intensivos se realizan normalmente en instalaciones separadas del medio natural, en tanques o piscinas aisladas con sistemas técnicos de captación y recirculación de agua, con un control total del medio y de los individuos. Son mucho más caros que los procesos menos tecnificados,

pero el aumento de rendimiento o la necesidad de un mayor control de la producción es determinante.

A menudo, se emplean cultivos superintensivos en los que se utilizan técnicas de acuariología, como recirculación de agua, control de temperatura y fotoperíodo o monitorización de parámetros, en las fases más delicadas de la cría.

1.3.3. Acuicultura marina

La acuicultura marina comprende los cultivos de especies propiamente marinas, tanto de peces como de algunos invertebrados, como el pulpo. Tiene una gran importancia económica y, en el caso de muchas especies, la producción de cultivo casi ha sustituido por completo a las capturas pesqueras.

Algunas de las especies más importantes son el rodaballo, la dorada, la lubina, el bacalao, la corvina y la anguila. Los cultivos de otras especies aún están en desarrollo, como el pulpo, el besugo o el lenguado, entre otras.

Una variante de acuicultura marina es el llamado engrasado de Atún Rojo, que se cultiva en jaula a partir de ejemplares salvajes. Tras un proceso de engorde son vendidos posteriormente en el mercado japonés, donde es un preciado producto (*http://es.wikipedia.org/wiki/Acuicultura*).

1.4. Antecedentes

Existen pocos estudios rigurosos sobre el diseño y planificación de los sistemas de distribución de agua de las plantas de acuicultura con la finalidad de conseguir crecimientos económicos sostenibles considerando la calidad de la producción. Así, los trabajos encontrados acerca de este tipo de instalaciones son mínimos, abarcando cuestiones demasiado generalistas que no permiten una optimización real de la instalación fluidomecánica ni evalúan la influencia de estos factores en la definición de la estrategia de producción (Kerr, 1981; Colt *et al*., 2006; Pulido-Calvo *et al*., 2008).

La optimización de sistemas urbanos de abastecimiento de agua ha sido ampliamente estudiado por autores como Tarquin y Dowdy (1989), Zessler y Shamir (1989), Brion y Mays (1991), Jowitt y Germanopoulos (1992), Ulanicki *et al*. (1993), Nitivattananon et al. (1996), Cembrano *et al*. (2000), León et al.

(2000) y Biscos *et al*. (2003). El principal objetivo de estos modelos es definir el esquema de operación óptimo de las estaciones de bombeo y depósitos de regulación en un período de 24 horas. En sistemas de suministro de agua para riego, Buchleiter y Heermann (1986, 1990), Moradi-Jalal *et al*. (2003), Pulido-Calvo *et al.* (2003, 2006a) y Planells *et al.* (2005) han desarrollado métodos para optimizar el tipo y número de bombas así como el esquema de operación del sistema de distribución y almacenamiento de agua considerando tanto el coste de inversión inicial como el coste de energía consumida. Aplicaciones de estos modelos muestran ahorros de entre el 20% y el 25% en el coste total anual del sistema de distribución de agua.

En el diseño y la gestión de sistemas de distribución de agua de plantas de acuicultura, algunos autores como Kerr (1981) y Colt *et al*. (2006) propusieron criterios de diseño para las estaciones de bombeo de agua. Pulido-Calvo *et al*. (2006b, 2008) presentaron la aplicación de ordenador ACUIGES que cuenta con dos módulos independientes, para las selecciones óptimas de los grupos de bombeo y los diámetros de las tuberías que impulsan y conducen el agua hasta los depósitos de cultivo en plantas de acuicultura intensiva. Otros sistemas expertos desarrollados para la planificación y control de instalaciones de acuicultura han sido desarrollados por Ernst *et al*. (2000), Lee (2000), Lee *et al*. (2000) y Gutiérrez-Estrada *et al*. (2005).

Los modelos de lógica borrosa han sido aplicados con éxito en el manejo y la planificación de recursos hídricos (Bardossy, 1996). Así, cabe destacar su uso en la caracterización y predicción de inundaciones (Cheng, 1999; Akter y Simonovic, 2005; Yu y Chen, 2005; Aqil *et al*., 2006), en la regulación de compuertas de canales, embalses y depósitos de regulación (Lian *et al.*, 1998; Dubrovin *et al.*, 2002; Hasebe y Nagayama, 2002; Bagis, 2003; Bagis y Karaboga, 2004; Begovich *et al.*, 2005; Karaboga *et al.*, 2008), en la evaluación de la calidad y del nivel de parámetros físico-químicos del agua (Chang *et al.*, 2001; Murnleitner *et al*., 2002; Chen *et al.*, 2003; Liou *et al.*, 2003; Ning y Chang, 2004; Lee y Chang, 2005; Chanona *et al*., 2006; Lai *et al.*, 2007; Li *et al*., 2007; Nasiri *et al.*, 2007), en la estimación de demandas de sistemas de distribución de agua (Moreno, 2006; Pulido-Calvo y Gutiérrez-Estrada, 2009), etc.

En el caso de los sistemas de producción acuícola, destacan las aplicaciones de los modelos borrosos en el diseño de un sistema experto para el diagnóstico de enfermedades (Gutiérrez-Estrada *et al*., 2005) y de un controlador para la regulación de los sistemas de alimentación (Papandroulakis *et al.*, 2000).

1.5. Objetivos específicos

Con este proyecto se pretende desarrollar un modelo para la regulación de los caudales de entrada en balsas de sistemas de distribución de agua de plantas de producción acuícola semi-intensiva, considerando como restricciones los niveles de calidad permitidos de los parámetros físico-químicos del agua (temperatura y nivel de amonio). Para la consecución de este objetivo global, se han planteado los siguientes objetivos específicos:

a) Desarrollo de un sistema de control borroso que, utilizando como datos de entrada (conjuntos borrosos de entrada) la temperatura y nivel de amonio del agua almacenada en la balsa, tenga como resultado (conjunto borroso de salida) los grados de apertura o cierre de las compuertas encargadas de la entrada de los caudales de agua.

b) Comparación y discusión de los resultados obtenidos con el sistema de control borroso desarrollado con los datos reales de funcionamiento de las balsas de cultivo de la planta de acuicultura "*Langostinos de Huelva*".

c) Integración del modelo desarrollado en una aplicación de ordenador en un entorno amigable (Visual Basic) que se constituya como un "sistema de ayuda a la toma de decisiones" dinámico, multidisciplinar, interactivo y de fácil manejo para los técnicos responsables del diseño y la gestión de sistemas de producción piscícola. Este modelo integral ha sido concebido como una herramienta metodológica general de posible utilización en cualquier piscifactoría.

2. ZONA DE ESTUDIO

2.1. Introducción

2.2. La dorada

2.3. La lubina

2.4. Proceso de producción

2. ZONA DE ESTUDIO

2.1. Introducción

Con objeto de contrastar el funcionamiento de la aplicación informática que se desarrolla en este trabajo, se aplicará y evaluará en la empresa "*Langostinos de Huelva, S.A.*", la cual posee una explotación dedicada a la acuicultura marina en régimen semi-intensivo (*figura 2.1*) ubicada en el término municipal de Cartaya (provincia de Huelva).

Figura 2.1. *Localización de la planta de acuicultura de la empresa "Langostinos de Huelva, S.A."*

La instalación consiste fundamentalmente en un conjunto de balsas (*figura 2.2*) delimitadas por muros de tierra, ocupando una superficie total aproximada de 90 hectáreas. La balsa de mayor tamaño (40 hectáreas, aproximadamente) es la encargada de recibir el aporte de agua necesario por medio de una serie de bombas instaladas en el río Piedras y en el caño de la margen izquierda del mismo. Además, actúa como depósito regulador, ya que desde ella se realiza la distribución de agua hacia las demás balsas.

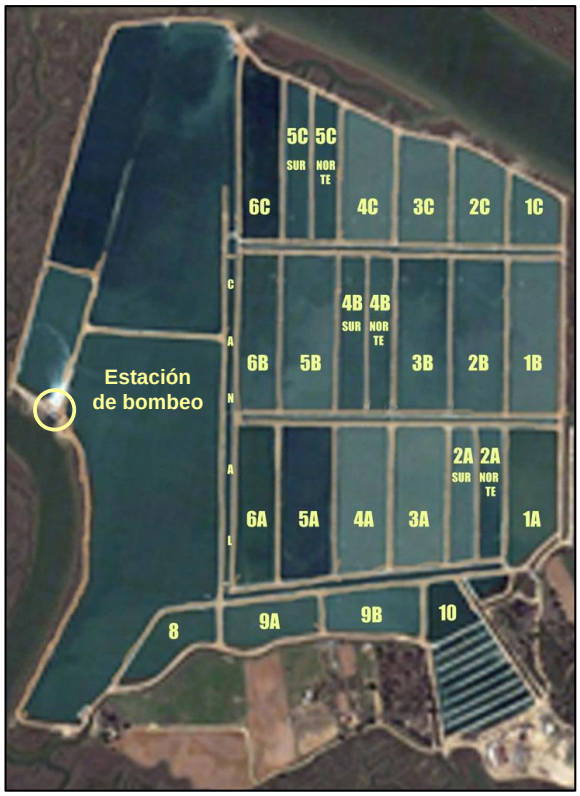


Figura 2.2. *Vista de las balsas de cultivo de la empresa "Langostinos de Huelva, S.A."*

En lo referente a este trabajo, los datos diarios de entrada al sistema borroso de temperatura y nivel de amonio serán los medidos en la balsa 5C Norte del 21 de abril al 29 de mayo de 2008 (*tabla 2.1*). También se midieron en este periodo las alturas de la lámina de agua, antes y después de las compuertas (*figura 2.3*). Estos datos nos permitirán comparar los caudales reales de entrada a las balsas con los obtenidos mediante el modelo de lógica borrosa desarrollado.

La empresa "*Langostinos de Huelva, S.A.*" lleva más de 15 años dedicada a la acuicultura marina a través del cultivo de numerosas especies, como la almeja, las ostras plana y rizada, el langostino o diferentes especies de peces, como dorada, lubina, lenguado, anguila, etc.

En la actualidad, la principal actividad de la empresa se centra en el cultivo en régimen semi-intensivo de dorada y lubina.

Fecha	Temperatura (° C)	Nivel de amonio (mg/l)	Altura y_1 (m)	Altura y_3 (m)
21/04/08	17	1.6	1.21	0.75
22/04/08	16	0.5	1.16	0.75
23/04/08	18	1.8	1.27	0.81
24/04/08	22	2	1.24	0.89
25/04/08	21	1	1.28	0.79
26/04/08	23	1.4	1.19	0.59
27/04/08	22	1.2	1.29	0.76
28/04/08	22	0.8	1.25	0.74
29/04/08	20	1	0.99	0.98
30/04/08	21	1.2	1.08	0.68
01/05/08	19	0.8	1.10	0.70
02/05/08	19	0.5	1.15	0.66
03/05/08	20	1.2	1.16	0.84
04/05/08	20	1.2	1.21	0.71
05/05/08	21	0.5	1.28	0.98
06/05/08	21	0.5	1.38	0.94
07/05/08	22	0.8	1.21	0.92
08/05/08	23	0.2	1.36	0.95
09/05/08	21	1.2	0.75	0.73
10/05/08	20	0.65	1.05	0.90
11/05/08	18	0.65	1.10	0.72
12/05/08	18	0.8	1.11	0.65
13/05/08	19	0.8	1.17	1.05
14/05/08	19	0.2	1.35	1.14
15/05/08	19	0.2	1.24	0.87
16/05/08	20	0.8	1.28	0.85
17/05/08	21	0.8	1.30	0.90
18/05/08	22	1.2	1.33	0.80
19/05/08	23	0.2	1.20	0.62
20/05/08	23	0.8	1.10	0.75
21/05/08	21	0.2	1.26	0.80
22/05/08	19	0.2	1.20	0.90
23/05/08	19	0.8	1.21	0.84
24/05/08	19	0.8	1.19	0.85
25/05/08	19	0.8	1.19	0.85
26/05/08	19	0.5	1.27	0.86
27/05/08	23	0.5	1.25	0.90
28/05/08	22.7	0.5	1.25	0.95
29/05/08	23.5	0.5	1.25	0.95

Tabla 2.1. *Datos diarios de temperatura, nivel de amonio y altura de la lámina de agua antes y después de las compuertas*

Figura 2.3. *Fotografía de la efectuación de mediciones en la compuerta de la balsa 5C Norte, realizada por Juan Carlos Gutiérrez Estrada*

Figura 2.4. *Fotografía de una de las balsas de cultivo de la empresa "Langostinos de Huelva, S.A.", realizada por Juan Carlos Gutiérrez Estrada*

Figura 2.5. *Fotografía de la compuerta de una de las balsas de cultivo de la empresa "Langostinos de Huelva, S.A.", realizada por Juan Carlos Gutiérrez Estrada*

2.2. La dorada

La dorada (*Sparus aurata*) es un pez marino de la familia de los Espáridos, muy estimado por la calidad de su carne. Puede alcanzar unos 70 cm de longitud y el cuerpo, como en otros miembros de su familia, es ovalado, alto y comprimido. La cabeza es grande, con mandíbulas fuertes. El dorso es gris y el vientre blanco y ostenta una banda longitudinal negra en la aleta dorsal. Los flancos presentan reflejos amarillos. Entre los ojos, que se encuentran en posición elevada, tiene una mancha dorada flanqueada por sendas líneas oscuras, y en cada opérculo presenta una mancha negra (*figura 2.6)*. Se alimenta de peces y sus dientes, molariformes, le permiten romper las cubiertas de los moluscos y crustáceos de los que también se alimenta.

Figura 2.6. *Dorada (Sparus aurata)*

De carácter sedentario, forma grupos pequeños que nadan sobre fondos arenosos, entre las rocas o en las praderas de posidonias. Viven en aguas poco profundas, especialmente los jóvenes, sin sobrepasar normalmente los 30 m de profundidad a dichas edades, y alcanzando los 150 m en el caso de los adultos. Gusta de las aguas salobres. En verano se queda cerca de la orilla, descendiendo a mayores profundidades en invierno, cuando se reproduce. Es una especie hermafrodita protándrica (son machos entre la edad de 1 y 2 años y hembras entre los 2 y los 3 años). La reproducción tiene lugar entre octubre y diciembre. Se distribuye por el Mediterráneo y el Atlántico oriental, entre las islas Británicas y cabo Verde, incluyendo las islas Azores y las islas Canarias.

Se captura con artes diversas, como palangre de fondo, redes de enmalle, trasmallos y redes de arrastre. También se pesca deportivamente a caña desde la costa. En España se ha desarrollado enormemente el cultivo intensivo en piscifactorías, lo que ha contribuido a popularizar este pescado.

2.3. La lubina

La lubina (*Dicentrarchus labrax*) es un pez de aspecto elegante de la familia de los Serránidos cuya área de distribución se extiende por el mar Mediterráneo y el océano Atlántico, desde Mauritania hasta Noruega, incluyendo el Báltico y el sur del mar del Norte. El cuerpo es alargado y robusto, mide entre 10 y 90 cm de longitud, y está cubierto de escamas de gran tamaño. La boca es protráctil con los labios carnosos y en el ángulo superior del opérculo presenta dos espinas cortas. Su color varía desde el gris oscuro en el dorso hasta el blanco en la parte ventral (*figura 2.7*).

Figura 2.7. *Lubina (Dicentrarchus labrax)*

De hábitos litorales, tiene preferencia por las aguas poco profundas y oxigenadas, aunque puede descender por debajo de los 100 m de profundidad o bien penetrar en las aguas dulces. Los peces jóvenes forman cardúmenes, mientras que cuando se hacen adultos se convierten en solitarios y territoriales. Los machos adquieren la madurez sexual a los dos años, mientras que las hembras lo hacen a los tres. La época de puesta ocurre desde enero a marzo, siendo probable que la hembra realice varios desoves durante ese periodo. En la zona del canal de la Mancha el desove tiene lugar de marzo a junio. Es muy voraz y come crustáceos, gusanos, peces, erizos de mar y otros animales marinos.

Es una especie muy apreciada por su valor culinario y en la pesca deportiva. Comercialmente se pesca con diversos artes, como el trasmallo, el palangre, el curricán y la red de arrastre, si bien la mayor parte de la producción actualmente comercializada se genera en piscifactorías.

2.4. Proceso de producción

La dorada y la lubina son dos especies con las que se lleva trabajando más de 30 años y se puede considerar que su cultivo está bastante consolidado con un suministro constante al mercado.

El ciclo de producción de ambas especies es similar. Los machos y hembras se colocan en el criadero en tanques con desagüe superficial y, cuando están maduros, las hembras emiten los huevos y los machos el esperma, fecundándose en el agua.

Las hembras son capaces de poner varios millones de huevos a lo largo de varias semanas de los cuales sólo sobrevive un pequeño porcentaje. A las 48 horas de incubación en agua, los huevos darán lugar a las larvas, las cuales son muy pequeñas, ya que pesan tan sólo 1 mg, y extremadamente sensibles.

Durante los primeros 30 días se les aporta un alimento microscópico vivo llamado rotífero, que a su vez es alimentado con algas microscópicas muy seleccionadas. Toda esta cadena de alimentación se denomina “alimento vivo” y se completa con el uso de otro zooplancton llamado artemia hasta los 2 meses de vida. Posteriormente, se les empieza a aportar alimentos inertes llamados piensos, compuestos fundamentalmente por harina de pescado. La calidad del agua debe ser óptima en cuanto a temperatura y otros parámetros físico-químicos.

A pesar de todos los cuidados, la mortalidad en la fase larvaria suele ser mayor del 90%. A los 2 meses, las larvas supervivientes pesan 0.1 gramos y siguen creciendo en tanques de alevinaje hasta tamaños de 30 a 50 gramos (8 meses de vida), momento en el que son trasladadas en camiones especiales a los centros de crianza donde se completa el ciclo de producción.

La crianza puede desarrollarse en sistemas extensivos o semi-intensivos en esteros o en sistemas intensivos en tanques o en viveros flotantes. En el caso de la empresa “*Langostinos de Huelva, S.A.*”, la crianza tiene lugar de manera semi-intensiva en esteros, lo que otorga a las especies un alto valor gastronómico debido a las condiciones de cultivo, únicas por salinidad, temperatura y por la vegetación y algas marinas que se desarrollan en este entorno natural.

La crianza hasta tamaños de 400 gramos puede completarse en 18 meses, alcanzándose tallas de 1 kg a los 3 años. La alimentación se basa en dietas equilibradas elaboradas con harinas de pescado y productos vegetales.

Las granjas marinas implican una abúndate avifauna, como garzas, águilas pescadoras o cormoranes, haciendo necesario el uso de redes para evitar la depredación.

Durante todo el proceso de crianza no existe ninguna manipulación de los peces, a excepción de su control sanitario.

Una vez alcanzada la talla y bajo pedido, las doradas o lubinas se extraen de las balsas con redes y se sumergen en agua con hielo para conseguir un sacrificio mediante el descenso de temperatura, lo cual resulta menos traumático que el tradicional de muerte mediante asfixia. Inmediatamente, las doradas o lubinas se clasifican y colocan en cajas con hielo en salas de manipulación bajo los pertinentes controles higiénicos y sanitarios.

Como puede apreciarse, se trata de un proceso de producción muy largo y cuidadoso. El resultado final es la dorada o lubina de crianza, productos disponibles todo el año de gran frescura y alta calidad nutritiva con garantía sanitaria.

3. MATERIAL Y MÉTODOS

3.1. Lógica borrosa

3.1.1. Sistemas de control borroso

3.1.2. Funciones de inclusión o pertenencia de conjuntos borrosos

3.1.3. Particiones borrosas

3.1.4. Reglas borrosas

3.1.5. Métodos de inferencia borrosa y desborrosificación

3.2. Microsoft Visual Basic 6

3.2.1. Proyectos

3.2.2. Archivos de un proyecto

3.2.3. Diseñadores y editores

3.2.4. Formularios

3.2.5. Controles ActiveX

3.2.6. Propiedades

3.2.7. Eventos y métodos

3.2.8. Programación estructurada y orientación a objetos

3.2.9. Funciones y procedimientos

3.2.10. Cuestiones de ámbito

3.2.11. Clases y objetos

3.3. Caudal en compuertas de corriente subálvea

3.3.1. Cálculo de las curvas del coeficiente de descarga

3.4. Evaluación de los modelos

3.5. Aplicación informática "*Sistema de control borroso para la regulación de balsas de plantas de acuicultura semi-intensiva*"

3. MATERIAL Y MÉTODOS

3.1. Lógica borrosa

La teoría de conjuntos borrosos parte de la teoría clásica de conjuntos, añadiendo una función de pertenencia al conjunto, la cual es definida como un número real entre 0 y 1. Así, se introduce el concepto de conjunto o subconjunto borroso asociado a un determinado valor lingüístico, definido por una palabra, adjetivo o etiqueta lingüística A. Para cada conjunto o subconjunto borroso se define una función de pertenencia o inclusión $\mu_A(t)$, que indica el grado en el que la variable t está incluida en el concepto representado por la etiqueta A.

Los conjuntos borrosos permiten agrupar objetos o sucesos por el valor de una cierta magnitud; por ejemplo, la regulación de la apertura o cierre de la compuerta por la que pasa el caudal de agua hasta las balsas de producción respecto a su temperatura y nivel de amonio. Así, en el modelo de lógica borrosa desarrollado en este trabajo se tienen como conjuntos borrosos de entrada la temperatura y el nivel de amonio y de salida la altura de la compuerta. Estos conjuntos borrosos se procesan mediante la combinación de reglas borrosas como *si* la temperatura es “muy alta” *y* el nivel de amonio es “muy alto”, *entonces* la altura de la compuerta es “muy alta”. Los modelos de lógica borrosa combinan una o más señales de entrada, las cuales se definen por los conjuntos borrosos, con un grupo de reglas borrosas para producir una salida que puede ser comparada con los valores reales observados (Marsili-Libelli, 2004; Gutiérrez-Estrada *et al.*, 2005; Pulido-Calvo y Gutiérrez-Estrada, 2009).

3.1.1. Sistemas de control borroso

Los sistemas expertos de control borroso basados en reglas, conocidos como controladores borrosos o FLC, o también como sistemas de inferencia borrosa o FIS, son sin duda la aplicación más extendida de la lógica borrosa.

Como se muestra en la *figura 3.1*, para controlar un proceso o sistema se hace uso de un módulo controlador, que recibe como entradas una o varias variables de control, llamadas generalmente referencias $\bar{R}$, y una o varias variables de salida del propio proceso $\bar{S}$, produciendo como salida una o varias variables, que se conocen como actuadores $\bar{A}$. Un primer bloque realiza un

preprocesado de las variables de entrada, que proporciona el vector de entradas al controlador borroso o FLC. El controlador borroso aplica la entrada que recibe a la base de reglas para obtener la salida.

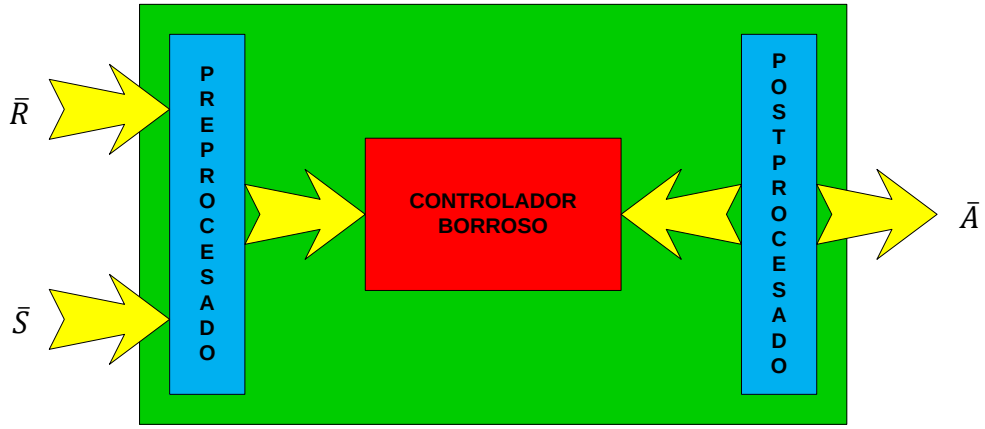


Figura 3.1. *Estructura de un controlador*

La estructura interna de un controlador borroso o FLC se muestra en la *figura 3.2*. Un primer elemento, llamado borrosificador, realiza la conversión de valores discretos a términos borrosos. Su salida es utilizada por el dispositivo de inferencia borrosa para aplicarla a cada una de las reglas de las bases de reglas, siguiendo el método de inferencia seleccionado. La salida de este bloque puede ser M conjuntos borrosos, un único conjunto borroso o M escalares.

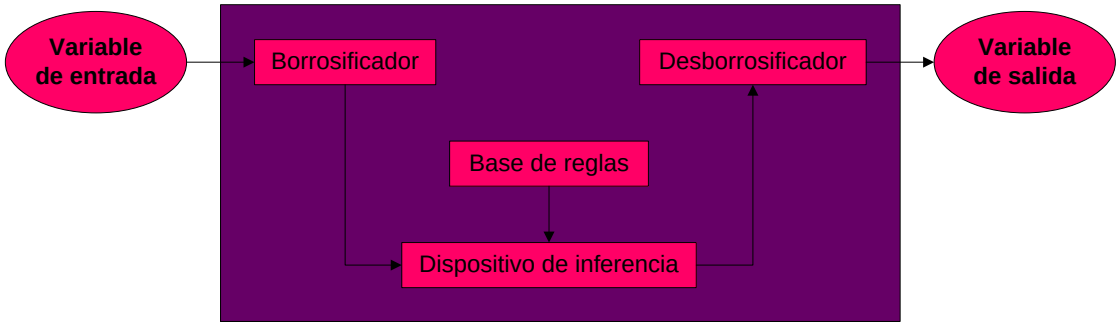


Figura 3.2. *Controlador borroso o FLC*

Las etapas de la estructura interna de un controlador borroso se detallan a continuación.

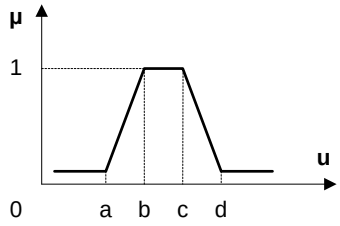
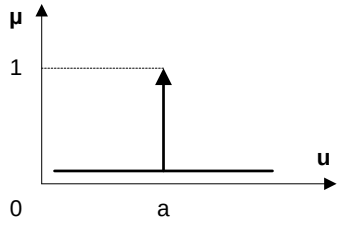
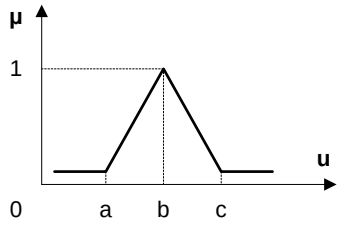
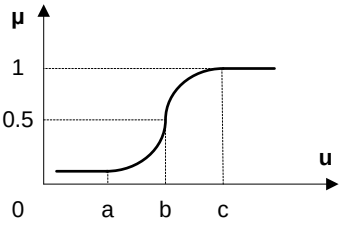
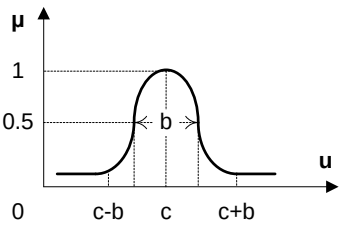
Función de tipo trapezoidal	
$$S(u;a,b,c,d)=\begin{cases}0 & u<a\\ (u-a)/(b-a) & a\le u\le b\\ 1 & b\le u\le c\\ (d-u)/(d-c) & c\le u\le d\\ 0 & u<d\end{cases}$$	
Función de tipo singleton	
$$S(u;a)=\begin{cases}1 & u=a\\ 0 & u\neq a\end{cases}$$	
Función de tipo T (triangular)	
$$T(u;a,b,c)=\begin{cases}0 & u<a\\ (u-a)/(b-a) & a\le u\le b\\ (c-u)/(c-b) & b\le u\le c\\ 0 & u<c\end{cases}$$	
Función de tipo S	
$$S(u;a,b,c)=\begin{cases}0 & u<a\\ 2\cdot[(u-a)/(c-a)]^2 & a\le u\le b\\ 1-2\cdot[(u-a)/(c-a)]^2 & b\le u\le c\\ 0 & u<c\end{cases}$$	
Función de tipo π	
$$\pi(u;b,c)=\begin{cases}S(u;c-b,c-(b/2),c) & u\le c\\ 1-S(u;c-b,c-(b/2),c) & u\ge c\end{cases}$$	

Figura 3.3. *Funciones de pertenencia más frecuentes (Martín-del-Brío y Sanz-Molina, 2001)*

3.1.2. Funciones de inclusión o pertenencia de conjuntos borrosos

La función de inclusión o pertenencia de un conjunto borroso consiste en un conjunto de pares ordenados $F = \{(u, \mu_F(u))/u \in U\}$, si la variable es discreta, o una función continua, si no lo es. El valor de $\mu_F(u)$ indica en grado en que el valor u de la variable U está incluido en el concepto representado por la etiqueta F. Para la definición de estas funciones de pertenencia se utilizan convencionalmente ciertas familias de formas estándar. Las más frecuentes son la función de tipo trapezoidal, singleton, T (triangular), S y π, las cuales se describen en la *figura 3.3*.

3.1.3. Particiones borrosas

Para que el controlador borroso funcione, es necesario definir las particiones de las variables del controlador, entendiéndose por partición un conjunto de los conjuntos borrosos que se han definido para la variable A (en nuestro caso "temperatura").

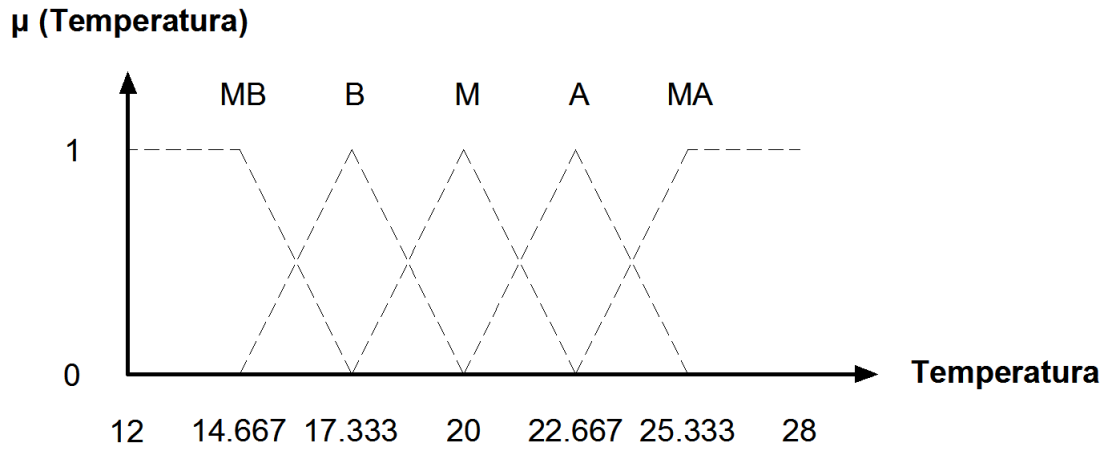


Figura 3.4. *Conjuntos borrosos para la variable "temperatura"*

Una partición de A es uno de os subconjuntos que pueden formarse con los elementos (términos) de $T(A)$. Así, para la variable "temperatura", una posible partición sería compuesta por cinco subconjuntos borrosos, cada uno identificado por una etiqueta, $\{Muy\ baja\ (MB), Baja\ (B), Media\ (M), Baja\ (B), Media\ (M), Alta\ (A)\ y\ Muy\ alta\ (MA)\}$, y una función de inclusión o pertenencia, $\{\mu_{Muy\ baja}(t), \mu_{Baja}(t), \mu_{Media}(t), \mu_{ltaA}(t), \mu_{Muy\ alta}(t)\}$.

Normalmente, las particiones son completas, impares y con un solapamiento del 20% al 50%. Se suelen utilizar particiones de tres, cinco o siete conjuntos, ya que la complejidad no es excesiva y permiten una precisión suficiente para la descripción de los valores de la variable.

En este trabajo, la variable "temperatura" se ha asociado a conjuntos de tipo T (triangular) con cinco particiones: Muy baja (MB), Baja (B), Media (M), Alta (A) y Muy alta (MA). A continuación, se muestra el gráfico con un nivel de solapamiento del 50% (*figura 3.4*).

3.1.4. Reglas borrosas

Las reglas borrosas combinan uno o más conjuntos borrosos de entrada, llamados antecedentes o premisas, a los cuales se les asocia un conjunto borroso de salida, llamado consecuente o consecuencia. Los conjuntos borrosos de entrada se relacionan a través de expresiones lógicas como *y*, *o*, etc. Estas reglas permiten expresar el conocimiento del que se dispone acerca de la relación existente entre entradas y salidas. Para expresar este conocimiento de forma completa se necesitan, generalmente, varias reglas que en su conjunto forman lo que se conoce como base de reglas.

Las bases de reglas se pueden representar bien como una tabla o como una memoria asociativa borrosa (MAB). Las MABs son matrices que representan la consecuencia de cada regla definida para cada combinación de dos entradas y permiten realizar una clara representación gráfica de las relaciones existentes entre dos variables lingüísticas de entrada y la variable lingüística de salida. Para ello, requieren que se indiquen explícitamente todas las reglas que se pueden formar con estas dos variables de entrada.

Formalmente, una base de reglas borrosas es un conjunto de reglas $R^{(l)}$ de la forma:

$$R^{(l)}\text{: } \textbf{\textit{si}}\ x_1 \text{ es } F_1^l\ \textbf{\textit{y}}\ \dots\ \textbf{\textit{y}}\ x_n \text{ es } F_n^l, \textbf{\textit{entonces}}\ y \text{ es } G^l \qquad (3.1)$$

donde F_1^l, F_n^l y G^l son conjuntos borrosos y $x = (x_1, \dots, x_n)$ e y son variables lingüísticas. En la *figura 3.5* se muestra la memoria asociativa borrosa inicial utilizada en este trabajo. A modo de ejemplo, las dos siguientes reglas pueden extraerse de la MAB:

- R1: **Si** "nivel de amonio" es MB **y** "temperatura" es MB, **entonces** "altura de la compuerta" es MB.

- R2: **Si** “nivel de amonio” es A **y** “temperatura” es M, **entonces** “altura de la compuerta” es A.

La regla R1 expresa que si el nivel de amonio es muy bajo y la temperatura es muy baja, entonces la altura hasta la que habría que abrir la compuerta es muy baja. La regla R2, por su parte, indica que si el nivel de amonio es alto y la temperatura es media, entonces la altura hasta la que habría que abrir la compuerta es alta.

		Nivel de amonio				
		MB	**B**	**M**	**A**	**MA**
Temperatura	**MB**	*M*	*M*	*A*	*MA*	*MA*
	B	*B*	*B*	*M*	*A*	*MA*
	M	*MB*	*B*	*M*	*A*	*A*
	A	*B*	*B*	*M*	*A*	*MA*
	MA	*M*	*M*	*M*	*MA*	*MA*

Figura 3.5. *Memoria asociativa borrosa inicial para la “altura de la compuerta”. A cada combinación de las variables de entrada “nivel de amonio” y “temperatura” se le asocia una consecuencia*

3.1.5. Métodos de inferencia borrosa y desborrosificación

Una vez establecida la base de reglas borrosas, es necesario seleccionar los métodos de inferencia borrosa y de desborrosificación.

Se denominan dispositivos de inferencia borrosa a aquellos sistemas que interpretan las reglas de tipo “*si – entonces”* con el fin de obtener los valores de salida a partir de los actuales valores de las variables lingüísticas de entrada al sistema. En este trabajo, se ha considerado una implicación borrosa por la regla del mínimo, mediante la cual la ecuación (3.1) puede expresarse como:

$$\mu_{F\to G}(x,y) = \min[\mu_F(x), \mu_G(y)] \qquad (3.2)$$

donde el término $\mu_F(x)$ quedará definido como el mínimo de las funciones de inclusión o pertenencia para cada término:

$$\mu_F(x) = \mu_{F_1^l \times \ldots \times F_n^l}(x) = \min\left\{\mu_{F_1^l}(x), \ldots, \mu_{F_n^l}(x)\right\} \quad (3.3)$$

El desborrosificador es la función que transforma un conjunto borroso, normalmente salida de un dispositivo de inferencia borrosa, en un valor no borroso. En este trabajo, se usa un método desborrosificador por media de centros, definido por:

$$y = \frac{\sum_{l=1}^{N} \overline{y^l}\left(\mu_F\left(\overline{y^l}\right)\right)}{\sum_{l=1}^{N}\left(\mu_F\left(\overline{y^l}\right)\right)} \quad (3.4)$$

donde y es el valor no borroso de salida, N es el número de conjuntos borrosos satisfechos, $\overline{y^l}$ representa el centro del conjunto borroso de salida y $\mu_F\left(\overline{y^l}\right)$ viene definido según la ecuación (3.3).

De este modo de obtiene el valor de la salida, el cual se corresponde con la variable “altura de la compuerta”.

3.2. Microsoft Visual Basic 6

Visual Basic no es sólo un lenguaje de programación. Es un entorno de desarrollo integrado en el que se puede desarrollar, ejecutar, probar y depurar los errores de las aplicaciones (Petroutsos, 1999).

3.2.1. Proyectos

Toda aplicación Visual Basic parte de la elaboración de un proyecto, nombre con el que se conoce al conjunto de formularios, módulos de clase, módulos de código y cualquier otro archivo necesario para una aplicación. Los formularios contienen controles y métodos y los módulos de código definición de clases, procedimientos y funciones. Archivos adicionales pueden contener recursos, diseños de informes, etc.

El proyecto establece, por lo tanto, todos los elementos que forman una aplicación. Aparece como una ventana con una lista jerárquica, en la que se pueden ver los archivos que lo forman, cada uno con su nombre de formulario

o módulo correspondiente. Aunque lo habitual es que a medida que se van creando formularios y módulos de código éstos se vayan incluyendo automáticamente en el proyecto, también es posible eliminar cualquier elemento de un proyecto.

Desde la ventana de proyecto se puede seleccionar cualquiera de los componentes de la aplicación haciendo doble clic. Si el elemento es un formulario, éste se abrirá para edición, al igual que ocurre en casa de ser un control ActiveX. Si se trata de un módulo de código, se abrirá una ventana donde es posible editarlo en forma de texto.

3.2.2. Archivos de un proyecto

La definición de un proyecto se almacena en un archivo de texto, que tendrá el nombre que el usuario le dé y la extensión VBP. En este archivo se almacena información general acerca de la aplicación: referencias a los controles que usa, el nombre del proyecto o su versión, etc. Visual Basic 6.0 permite tener abiertos varios proyectos de forma simultánea, pudiendo almacenarse la colección en un archivo que tendrá la extensión VBG.

Cada uno de los formularios contenidos en el proyecto se almacena separadamente en un archivo que tendrá el nombre del formulario y la extensión FRM. También se trata de un archivo de texto, similar a los archivos de recursos de otros lenguajes, en el que se define la posición y dimensiones del formulario y de cada uno de los controles que éste contiene. Además, contiene el código correspondiente a los métodos asociados a los eventos que se hayan definido. En caso de que el formulario cuente con elementos que no sea posible describir mediante texto, como gráficos, existirá un archivo con el mismo nombre que el archivo FRM, pero con extensión FRX. En este caso se trata de un archivo binario manipulado indirectamente al editar el formulario.

Los módulos de código independientes de los formularios se almacenan en archivos con extensión BAS, como los tradicionales programas BASIC. Estos archivos contienen tan sólo declaraciones y código, en forma de procedimientos o funciones.

En caso de que un proyecto defina alguna clase de objeto nueva, esta definición se almacenará en archivo con extensión CLS. También en este caso se trata de un archivo normal de texto, con la enumeración de las propiedades y los métodos necesarios para poder crear objetos de la clase que se define. Un proyecto puede contener también controles de usuario, cuya descripción se almacena en archivos con extensión CTX, con una estructura similar a los

archivos FRX. En caso de que el control cuente con páginas de propiedades, éstas se almacenarán en archivos con extensión PAG.

Además de formularios y código, es posible añadir a los proyectos archivos conteniendo recursos, tales como mapas de bits o cadenas de texto. Estos archivos de recursos, que tendrán extensión RES, se editan mediante una herramienta específica integrada en Visual Basic.

Los elementos mencionados son sólo algunos de los que se podrían encontrar en un proyecto, quizá los más habituales. Habría que añadir otros como los proyectos de datos, archivos con diseños de informes, diseños de HTML dinámico, etc.

3.2.3. Diseñadores y editores

Cada uno de los tipos de archivo anteriores es manipulado en Visual Basic mediante un determinado diseñador o editor, que facilita su edición.

Los archivos de proyecto o grupos de proyecto, VBP y VBG respectivamente, se editan mediante el Gestor de proyectos. En esta ventana se puede añadir y eliminar elementos, abrirlos, establecer opciones, etc.

El diseñador más conocido de Visual Basic es el de formularios. Se trata de una ventana en la que los formularios del programa aparecen como se verán finalmente, al ejecutar la aplicación, permitiéndose la inserción y manipulado de componentes. Las interfaces de los programas tradicionales se elaboran mediante el diseñador de formularios.

Una interfaz siempre cuenta, por lo general, con un código que la gestiona. Todo código Visual Basic, indistintamente de que esté asociado a un formulario, un módulo de clase o un módulo estándar, se manipula mediante el Editor de código. Se trata de un editor de texto sencillo que permite introducir, corregir y eliminar código, pero con capacidades extendidas como la diferenciación sintáctica por colores o la tecnología IntelliSense, que ofrece ayuda inmediata a muchas acciones.

Los controles de usuario se manipulan mediante un diseñador muy similar al de formularios, aunque con diferencias lógicas debidas a la distinta naturaleza de formularios y controles.

Existen diseñadores más específicos, como los que permiten la edición de módulos de datos o el diseño de informes e interfaces con HTML dinámico. Algunos tipos de módulos, cono los que contienen recursos, son editados con una herramienta independiente, aunque accesible desde Visual Basic.

3.2.4. Formularios

El formulario es en Visual Basic el punto central de cualquier aplicación estándar, pudiendo ésta contar con uno o más formularios, según las necesidades. Un formulario es una ventana Windows en la que se depositan los controles necesarios para crea la interfaz con el usuario de la aplicación. El formulario cuenta por defecto con los elementos habituales de cualquier ventana Windows, como el control de menú de sistema, los botones de minimizar y maximizar o el borde por medio del cual es posible redimensionar la ventana. La barra de título del formulario se puede utilizar para desplazarlo de un punto a otro y el título que aparece en ella es, por supuesto, establecido por el creador de la aplicación.

Las propiedades que por defecto tiene un formulario pueden modificarse durante el diseño de la aplicación, accediendo a la ventana de propiedades. En ella se seleccionará el título que se desea darle, si debe tener o no botones de minimizar y maximizar, si debe ser o no redimensionable, etc. También es posible asociar código al formulario, abriendo la ventana de código y seleccionando el evento o mensaje que provocará su ejecución.

Un formulario por sí solo tiene poca utilidad a no ser que en su interior se inserten controles. Esta es la finalidad principal de un formulario, agrupar una serie de controles por medio de los cuales poder presentar y solicitar información al usuario. Estos controles pueden ser los estándar de Windows, como botones, listas, etiquetas de texto o botones de radio, otros controles incorporados por Visual Basic, como los controles que permiten presentar datos en rejillas o en forma de esquema, o bien ser objetos externos, como una imagen de Paintbrush o un documento de Word. En general, se puede usar cualquier componente ActiveX u objeto OLE.

3.2.5. Controles ActiveX

La creación de aplicaciones con Visual Basic se basa principalmente en el uso de distintos controles. Algunos de ellos sirven para solicitar o presentar información al usuario, otros para generar un evento cada cierto tiempo, para buscar un archivo en el disco, o para contener un imagen. Estos controles, denominados tradicionalmente controles OLE, se encuentran almacenados en archivos separados ya compilados, conteniendo el código necesario para ser utilizados accediendo a sus propiedades y sus métodos.

Actualmente existen infinidad de controles ActiveX, denominación con la que se conoce a este tipo de componentes. Visual Basic facilita un número importante de ellos y en la mayoría de las ocasiones serán suficientes para el desarrollo de las diferentes aplicaciones. No obstante, existen muchos otros de terceros proveedores e incluso es posible crear controles ActiveX con Visual Basic.

Un control ActiveX no es útil sólo a la hora de crear una aplicación con Visual Basic, sino que puede ser usado con otras herramientas de programación. También es posible usarlos en la composición de páginas web, haciéndolas más dinámicas y útiles. La facilidad con la cual Visual Basic permite la creación de controles ActiveX pone esta posibilidad al alcance de la mano de cualquier programador.

3.2.6. Propiedades

Todos los objetos con los que se trabaja en Visual Basic, desde el propio formulario hasta cada uno de los controles que se puedan utilizar, tienen una serie de propiedades. Estas propiedades permiten personalizar el objeto, indicando su posición, dimensiones, color, aspecto, título, valor, etc. El trabajo de diseño en Visual Basic consiste básicamente en insertar objetos en un formulario y establecer cada una de sus propiedades.

Hay propiedades que sólo pueden ser establecidas o modificadas durante el diseño de la aplicación, otras a las que sólo se puede acceder mientras la aplicación se está ejecutando y, por último, hay un tercer grupo que pueden ser utilizadas tanto en tiempo de diseño como de ejecución. El establecimiento y obtención de propiedades es muy simple tanto en el momento en que se está diseñando la aplicación, con la ventana de propiedades, como desde el código, durante la ejecución.

Las propiedades pueden ser de distintos tipos, al igual que las variables, pudiendo contener un número entero, una cadena o un valor *booleano*, por poner un ejemplo. A la hora de utilizar o fijar los valores de una propiedad, se aplicarán, por lo tanto, las mismas reglas que para las variables.

Además, el programador puede definir sus propios objetos con sus propiedades, así como crear procedimientos que se ejecuten cada vez que se establezca u obtenga una determinada propiedad.

3.2.7. Eventos y métodos

Windows es un entorno gestionado por eventos que son generados por una acción exterior por parte del usuario, como el movimiento del ratón o la pulsación de una tecla o de uno de los botones del ratón, o bien por el propio Windows. Las aplicaciones desarrolladas para funcionar sobre este entorno tienen que trabajar dirigidas por estos eventos y no siguiendo la metodología clásica de programación utilizada por sistemas como DOS.

En Visual Basic no existe el tipo de codificación secuencial que se da en el BASIC sobre DOS, en su lugar el código se encuentra delimitado en procedimientos que son llamados, cuando es necesario, por el propio Windows.

Cada uno de los controles que se puede insertar en un formulario, incluido el propio formulario, puede recibir una serie de eventos, en unos casos comunes y en otros específicos de cada control. Por defecto Visual Basic no asocia función alguna a los controles para que respondan a estos eventos. Existe una serie de procedimientos de respuesta, llamados métodos, que en principio están vacíos y a los cuales se les puede asocia el código que se necesite.

Aunque cada uno de los controles que incorpora Visual Basic es capaz de responder a multitud de eventos, por regla general sólo se definen los métodos de algunos de ellos.

3.2.8. Programación estructurada y orientación a objetos

Los programadores desde siempre han diferenciado los lenguajes entre estructurados, tipo Pascal o C, y aquellos que no lo eran, como BASIC. Este último adolecía de las estructuras necesarias para distribuir el código de una forma clara, siendo necesario realizar saltos continuamente, lo peor, sin duda alguna, de este lenguaje.

Visual Basic, por el contrario, cuenta con múltiples estructuras de control que permiten llevar a cabo la ejecución de una aplicación completa sin necesidad de realizar un solo salto. Anteriormente, se ha expresado la posibilidad de crear procedimientos y funciones eliminando así la necesidad de utilizar uno de los saltos de BASIC, el de la instrucción GOSUB. Asimismo se cuenta con varios tipos de bucles y estructuras de decisión múltiple, que desterrarán de una vez por todas el repetitivo GOTO. De esta forma se consigue que el código sea mucho más legible y fácil de mantener.

El siguiente paso lógico en la evolución de los lenguajes viene determinado por la orientación a objetos. Con esta metodología ya no existen procedimientos o funciones aislados, sino objetos que pueden ser usados para realizar determinadas tareas. Visual Basic cuenta con algunas características de orientación a objetos, dejando definitivamente atrás los tiempos de la compleja codificación lineal y monolítica.

3.2.9. Funciones y procedimientos

Además del código contenido en los métodos, que es ejecutado en respuesta a un evento, en ocasiones una aplicación necesita realizar otro tipo de operaciones que, por claridad y facilidad de mantenimiento, no es adecuado incluir en los propios métodos. En estas ocasiones lo que se hace es crear un módulo independiente incluyendo en él el código que se necesite, debidamente estructurado en procedimientos y funciones.

Un procedimiento es un conjunto de líneas de código al que se da un nombre, usado posteriormente para ejecutar ese código. La misma definición es válida para una función, con la única diferencia de que ésta devuelve un valor al finalizar su ejecución, mientras que el procedimiento no. El concepto de procedimiento y función no es nuevo para los programadores de TurboBasic o Quick Basic, pero sí para aquellos que siempre utilizaron GwBasic y compiladores de BASIC estándar. En éstos la forma de crear trozos de código separados, que eran llamados cuando se necesitaban, eran las subrutinas, a las cuales se llamaba con GOSUB y de las que se volvía con RETURN. Pues bien, un procedimiento no es ni más ni menos que eso, pero con la diferencia de que para ejecutarlo basta con utilizar su nombre.

Tanto los procedimientos como las funciones pueden recibir parámetros, lo que facilita el paso de datos sin necesidad de tener que utilizar variables de ámbito global. Estos parámetros pueden ser pasados por valor o por referencia. En el primer caso el procedimiento o función recibe una copia de la variable que se pasa como parámetro, por lo que no puede modificar la variable original. En el segundo se recibe realmente la dirección de la variable, de tal forma que cualquier modificación que se efectúe en ella hará que, al volver del procedimiento o función, el código que realizó la llamada encuentre la variable modificada.

Dentro de una misma aplicación se pueden tener múltiples módulos de código, conteniendo cada uno de ellos declaraciones, funciones y procedimientos. Cuando se da este caso, un procedimiento o función que se encuentre en un módulo no puede llamar a otro de un módulo distinto a no ser

que éste último sea público. De forma similar a lo que ocurre con las variables, tanto los procedimientos como las funciones pueden ser públicos, utilizables desde cualquier otro punto de la aplicación, o privados, pudiéndoseles llamar tan sólo desde el módulo de código en el que están definidos.

De lo anterior se puede deducir que, si una aplicación va a contar con múltiples procedimientos y funciones, lo más adecuado es crear varios módulos de código, almacenando en cada uno de ellos aquellas funciones o procedimientos que están relacionados de alguna forma o sean dependientes unos de otros.

3.2.10. Cuestiones de ámbito

A diferencia de lo que ocurre en los intérpretes y compiladores de BASIC tradicionales, en los cuales una variable declarada en un punto del programa es accesible desde cualquier otro punto, en Visual Basic las variables pueden ser globales, accesibles desde cualquier punto, públicas en un módulo, accesibles desde los métodos y funciones contenidos en un módulo de código, o privadas, accesibles tan sólo en el ámbito en que se han declarado.

Es una práctica común que todas las variables que se utilizan en métodos, funciones o clases de objetos sean privadas, evitando así que puedan ser manipuladas desde cualquier código externo. Este tipo de variable es creado automáticamente cada vez que se ejecuta el procedimiento o función y destruido cuando éste termina. Este hecho, por otra parte, reduce el consumo global de memoria, ya que la mayor parte de las variables existen tan sólo mientras se está ejecutando la porción de código en la que se declaran y la memoria que ocupan se libera al salir.

Los especificadores de ámbito, que son los que determinan si un identificador es público o privado, no son aplicables sólo a las variables, sino que pueden aplicarse a definiciones de tipos, declaraciones y otros elementos de una aplicación.

3.2.11. Clases y objetos

Aunque Visual Basic no es completamente un lenguaje orientado a objetos, sí que tiene algunas características de estos lenguajes. Entre estas características está la de poder definir clases de objetos, estableciendo qué propiedades y métodos tendrán.

Para definir una clase se inserta en el proyecto un módulo de clase, en él se especificará cada una de sus propiedades, escribiendo el código necesario para poder obtenerlas y modificarlas, así como las declaraciones de todos sus métodos. En cada archivo de módulo de clase tan sólo se puede definir una clase, pero es posible incluir varios módulos de este tipo en un mismo proyecto.

Una clase es, en cierta forma, como una plantilla del objeto que se desea crear. En realidad, al definir la clase no se está creando el objeto, pero se están poniendo las bases para ello. Cuando en una aplicación se necesite un objeto de esta clase se creará y en ese momento se tomará la definición efectuada previamente para establecer las propiedades y métodos del nuevo objeto, a los que se podrá acceder como si se tratase de cualquier otro control Visual Basic.

¿Qué diferencia hay entre crear una clase y definir un procedimiento o una función? Básicamente una: un procedimiento o una función es un conjunto de líneas de código que se pueden ejecutar, no cuentan con propiedades ni métodos, mientras que un objeto encapsula en su interior todo lo necesario para su funcionamiento, no sólo código, también todas aquellas variables e incluso otros objetos que le sean necesarios. El objeto no se ejecuta sin más, como un procedimiento, sino que se crea, se establecen y se obtienen sus propiedades y se utilizan sus métodos. Es una entidad muy superior a un procedimiento o función (Charte, 1998).

3.3. Caudal en compuertas de corriente subálvea

Para controlar el caudal en el umbral vertedor de un derramadero, o a la entrada de un canal de irrigación o de un río que sale de un lago, se dispone de una amplia variedad de estructuras de tipo compuerta.

Se dice que el flujo bajo una compuerta es efluente libre cuando el fluido sale como un chorro de flujo supercrítico con una superficie libre abierta a la atmósfera. En estos casos se acostumbra escribir este caudal como el producto de la distancia a entre el fondo del canal y el fondo de la compuerta, por la anchura b de dicha compuerta y por la velocidad de referencia conveniente $(2 \cdot g \cdot y_1)^{1/2}$. Es decir,

$$Q = C_d \cdot b \cdot a \cdot \sqrt{2 \cdot g \cdot y_1} \qquad (3.5)$$

donde Q es el caudal que pasa por la apertura de la compuerta. El coeficiente de descarga C_d es función del coeficiente de contracción $C_c = y_3/a$ y la razón de profundidad y_1/a. Los valores representativos del coeficiente de descarga para el efluente libre (o descarga libre) de una compuerta de desagüe vertical son del orden de 0.55 hasta 0.6, como se indica en la línea roja de la *figura 3.6*.

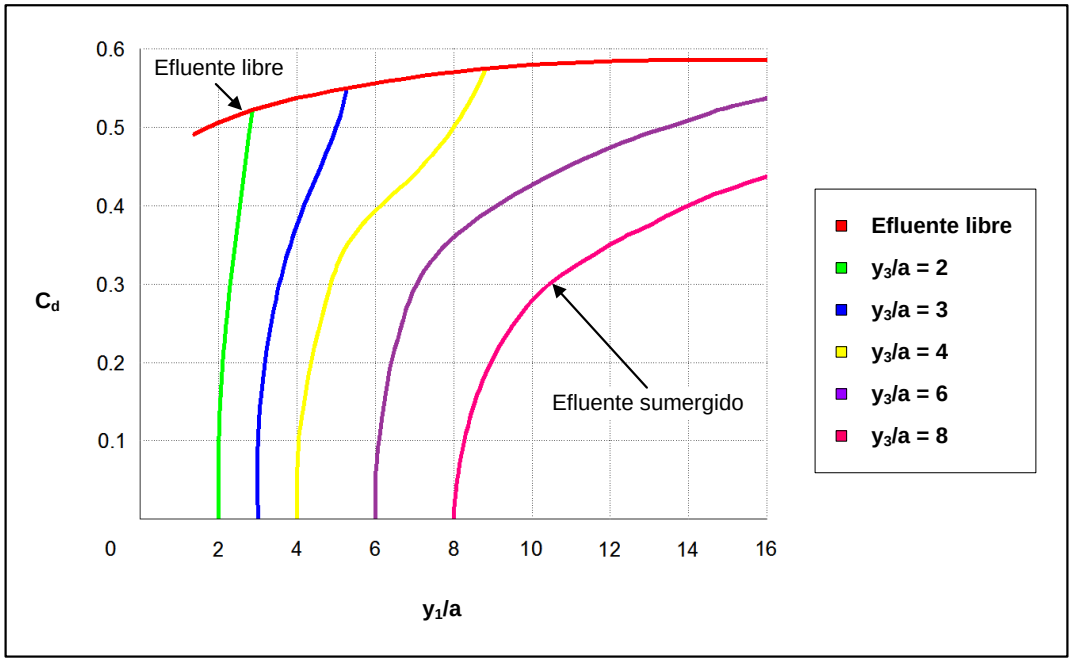


Figura 3.6. *Coeficientes de descarga característicos de compuertas de corriente subálvea*

Como se indica en la *figura 3.7*, en ciertas situaciones la profundidad corriente abajo de la compuerta es controlada por algún obstáculo corriente abajo y el chorro de agua que sale desde debajo de la compuerta está cubierto por una masa de agua bastante turbulenta.

El caudal para una sumergida (o inundada) se puede obtener a partir de la ecuación (3.5), siempre que el coeficiente de descarga se modifique según sea necesario. Valores de C_d representativos para casos de efluente sumergido se indican en la *figura 3.6* como la serie de curvas inferiores. Considérese el flujo para una compuerta y condiciones corriente arriba dadas (es decir, dado y_1/a), correspondientes a una recta vertical en la *figura 3.6*. Con $y_3/a = y_1/a$ (es decir, $y_3 = y_1$) no hay carga que active el flujo, de modo que $C_d = 0$ y el fluido es estacionario. Para una profundidad corriente arriba dada (y_1/a fija), el valor de C_d aumenta con y_3/a decreciente hasta que se alcanza el valor máximo de C_d. Este máximo corresponde a las condiciones de descarga libre y está representado por la línea de efluente libre, así identificada en la *figura 3.6*. Para valores de y_3/a con los que se obtienen valores de C_d entre cero y su

máximo, el chorro de la compuerta está cubierto (sumergido) por el agua corriente abajo y, en consecuencia, el caudal se reduce en comparación con una situación de descarga libre (Munson *et al.*, 1999).

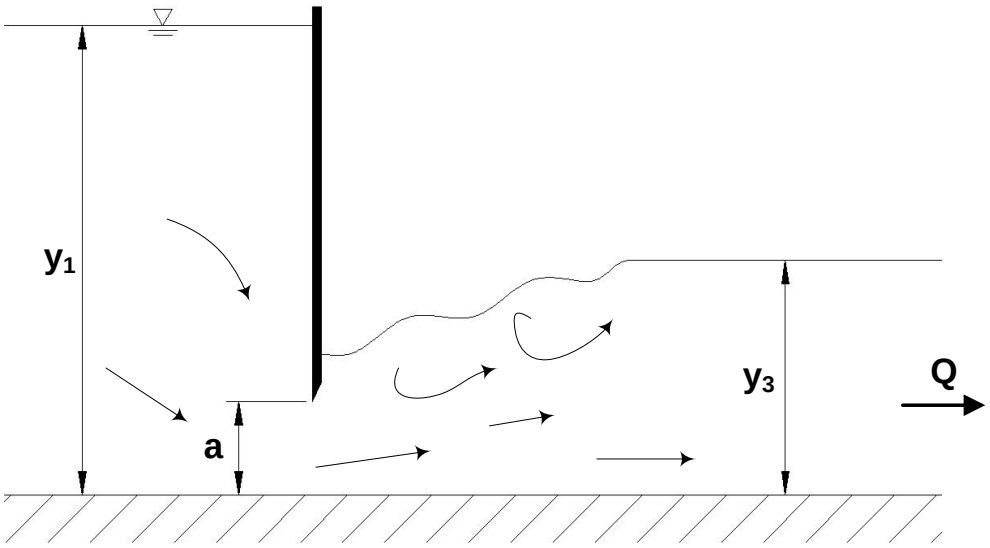


Figura 3.7. *Efluente sumergido que sale de una compuerta de desagüe*

Para el cálculo del caudal que atraviesa la compuerta de la balsa, la aplicación informática "*Sistema de control borroso para la regulación de balsas de plantas de acuicultura semi-intensiva*" utilizará la ecuación (3.5). El cálculo del caudal real aproximado que entra en la balsa se calcula considerando la ecuación (3.6) para desagües por orificios sumergidos, ya que en la planta de acuicultura en estudio la regulación del caudal se realiza mediante tablones con orificios en su parte central.

$$Q = C_d \cdot b \cdot \sqrt{2 \cdot g \cdot (y_1 - y_3)} \qquad (3.6)$$

3.3.1. Cálculo de las curvas del coeficiente de descarga

Debido al hecho de que las curvas de la *figura 3.6* son de origen experimental, se procederá al cálculo de la ecuación de cada una de ellas mediante el método de aproximación por mínimos cuadrados con polinomios de grado seis.

En la ciencia, a menudo un experimento produce un conjunto de datos $\{x_k, y_k\}_1^n$, siendo las abscisas distintas. Uno de los objetivos es obtener la expresión que relacione las variables. En un principio, se considera el caso en el que exista una relación lineal entre las variables $f(x) = y = a \cdot x + b$.

Curva y_3 / a = 2	
$f(x) = \frac{978}{851}x^6 - \frac{1593}{155}x^5 + \frac{3821}{205}x^4 + \frac{2195}{23}x^3 - \frac{36633}{76}x^2 + \frac{41876}{53}x - \frac{917}{2}$	$E_2(f) = \frac{85}{8411} = 0.0101$
Curva y_3 / a = 3	
$f(x) = -\frac{163}{1677}x^6 + \frac{1377}{550}x^5 - \frac{1015}{38}x^4 + \frac{6947}{43}x^3 - \frac{14341}{30}x^2 + \frac{61783}{77}x - \frac{18981}{34}$	$E_2(f) = \frac{69}{7525} = 0.0092$
Curva y_3 / a = 4	
$f(x) = -\frac{40}{38411}x^6 + \frac{179}{4281}x^5 - \frac{659}{951}x^4 + \frac{1591}{262}x^3 - \frac{1634}{55}x^2 + \frac{3315}{43}x - \frac{8692}{105}$	$E_2(f) = \frac{81}{25733} = 0.0031$
Curva y_3 / a = 6	
$f(x) = -\frac{8}{368233}x^6 + \frac{37}{24584}x^5 - \frac{371}{8664}x^4 + \frac{333}{520}x^3 - \frac{2672}{503}x^2 + \frac{2695}{116}x - \frac{5833}{140}$	$E_2(f) = \frac{37}{35345} = 0.0010$
Curva y_3 / a = 8	
$f(x) = -\frac{7}{114036}x^6 + \frac{59}{12957}x^5 - \frac{658}{4713}x^4 + \frac{949}{419}x^3 - \frac{8143}{397}x^2 + \frac{10530}{107}x - \frac{20522}{105}$	$E_2(f) = \frac{58}{47627} = 0.0012$

Figura 3.8. *Curvas del coeficiente de descarga calculadas mediante aproximación por mínimos cuadrados con sus respectivos errores cuadráticos medios*

A menudo, los datos que se han obtenido en la experimentación conllevan a un cierto error experimental $f(x_k) = y_k + e_k$, donde e_k es un error en la medición. El conjunto de errores de la medición $\{e_k\}$ permite definir distintos tipos de errores (error máximo, lineal, cuadrático medio…), que se pueden considerar para encontrar la mejor aproximación. Estos errores están asociados a ciertas normas, para evaluar dichos errores en este trabajo usaremos la expresión del error cuadrático medio:

$$E_2(f) = \sqrt{\frac{1}{n} \cdot \sum_1^n |f(x_k) - y_k|^2} \qquad (3.7)$$

La recta de regresión, definida como la recta $y = a \cdot x + b$ que minimiza el error cuadrático medio $E_2(f)$, se obtiene mediante las soluciones de las ecuaciones normales:

$$a = \frac{(\sum_1^n x_k \cdot y_k) - n \cdot (\sum_1^n x_k) \cdot (\sum_1^n y_k)}{(n \cdot \sum_1^n x_k^2) - (\sum_1^n x_k)^2} \quad (3.8)$$

$$b = \frac{(\sum_1^n y_k) - a \cdot (\sum_1^n x_k)}{n} \quad (3.9)$$

En muchas ocasiones, los datos deben venir ajustados no por formas lineales, sino por funciones del tipo $y = a \cdot x^m$, donde m es una constante conocida. Usando la técnica de los mínimos cuadrados, la solución se reduce a:

$$a = \frac{\sum_1^n (x_k^m \cdot y_k)}{\sum_1^n (x_k^{2m})} \quad (3.10)$$

De esta forma, las ecuaciones correspondientes a las curvas de la *figura 3.6* y sus respectivos errores cuadráticos medios, con la razón de profundidad y_1/a expresada como x y el coeficiente de descarga C_d expresado como $f(x)$, son los indicados en la *figura 3.8*.

3.4. Evaluación de los modelos

Para evaluar el comportamiento de cada una de las simulaciones realizadas se calculó el Error Cuadrático Medio (RMSE), el cual puede ser estimado como:

$$RMSE = \sqrt{\frac{\sum_{i=1}^N (Q_i - \hat{Q}_i)^2}{N}} \quad (3.11)$$

donde Q_i es el valor de caudal real en m^3/s en el momento i, y $\hat{Q}_i$ es el valor del caudal en m^3/s estimado por el modelo en el mismo instante.

Asimismo, se analizó el comportamiento medio de cada simulación mediante el análisis de la prueba estadística t de Student. Bajo las hipótesis de normalidad e igual varianza, la hipótesis de partida será:

H_0: *La media del caudal real para el período considerado es igual a la media del caudal simulado*

El t test para dos muestras independientes se basa en el estadístico:

$$t = \frac{\bar{Q}_r - \bar{Q}_s}{\sqrt{\frac{(n-1)\cdot \hat{S}_r^2 + \left(m - 1 \cdot \hat{S}_s^2\right)}{n+m-2}} \cdot \sqrt{\frac{1}{n} + \frac{1}{m}}} \tag{3.12}$$

$$\hat{S}_r^2 = \frac{1}{n-1} \cdot \sum_{i=1}^{n} \left(Q_{i,r} - \bar{Q}_r\right)^2 \tag{3.13}$$

$$\hat{S}_s^2 = \frac{1}{m-1} \cdot \sum_{i=1}^{m} \left(Q_{i,s} - \bar{Q}_s\right)^2 \tag{3.14}$$

donde $\bar{Q}_r$ y $\bar{Q}_s$ son respectivamente los caudales medios reales y estimados, $\hat{S}_r^2$ y $\hat{S}_s^2$ son las cuasivarianzas muestrales correspondientes y m y n son el número de datos disponibles en cada muestra (en nuestro caso $m = n$).

Si la hipótesis de partida es cierta, el estadístico seguirá una distribución t de Student con $n + m - 2$ grados de libertad. De ser así, el valor obtenido debería estar dentro del rango de mayor probabilidad según esta distribución.

Normalmente se toma como referencia el rango de datos en el que se concentra el 95% de la probabilidad (nivel de significación: p = 0.05). Si el valor de p es muy pequeño (normalmente menor a 0.05), es poco probable que se cumpla la hipótesis de partida y se debería rechazar.

Para comprobarlo, basta con comparar el valor del estadístico t con el correspondiente en la tabla para los oportunos grados de libertad y nivel de significación. El número que determina su intersección es el valor crítico correspondiente. De este modo, si el estadístico que se obtiene toma un valor mayor, se dirá que la diferencia es significativa.

3.5. Aplicación informática "*Sistema de control borroso para la regulación de balsas de plantas de acuicultura semi-intensiva*"

La aplicación informática "*Sistema de control borroso para la regulación de balsas de plantas de acuicultura semi-intensiva*" de varios formularios con

su correspondiente código, el cual se presenta en el documento Anexo de este trabajo.

El formulario principal (*figura 3.9*), "Programa.frm", es el encargado de realizar los cálculos derivados de la aplicación de la lógica borrosa. En la parte superior del mismo, se encuentran los conjuntos borrosos de las variables de entrada, "temperatura" y "nivel de amonio", y de la variable de salida, "altura de la compuerta".

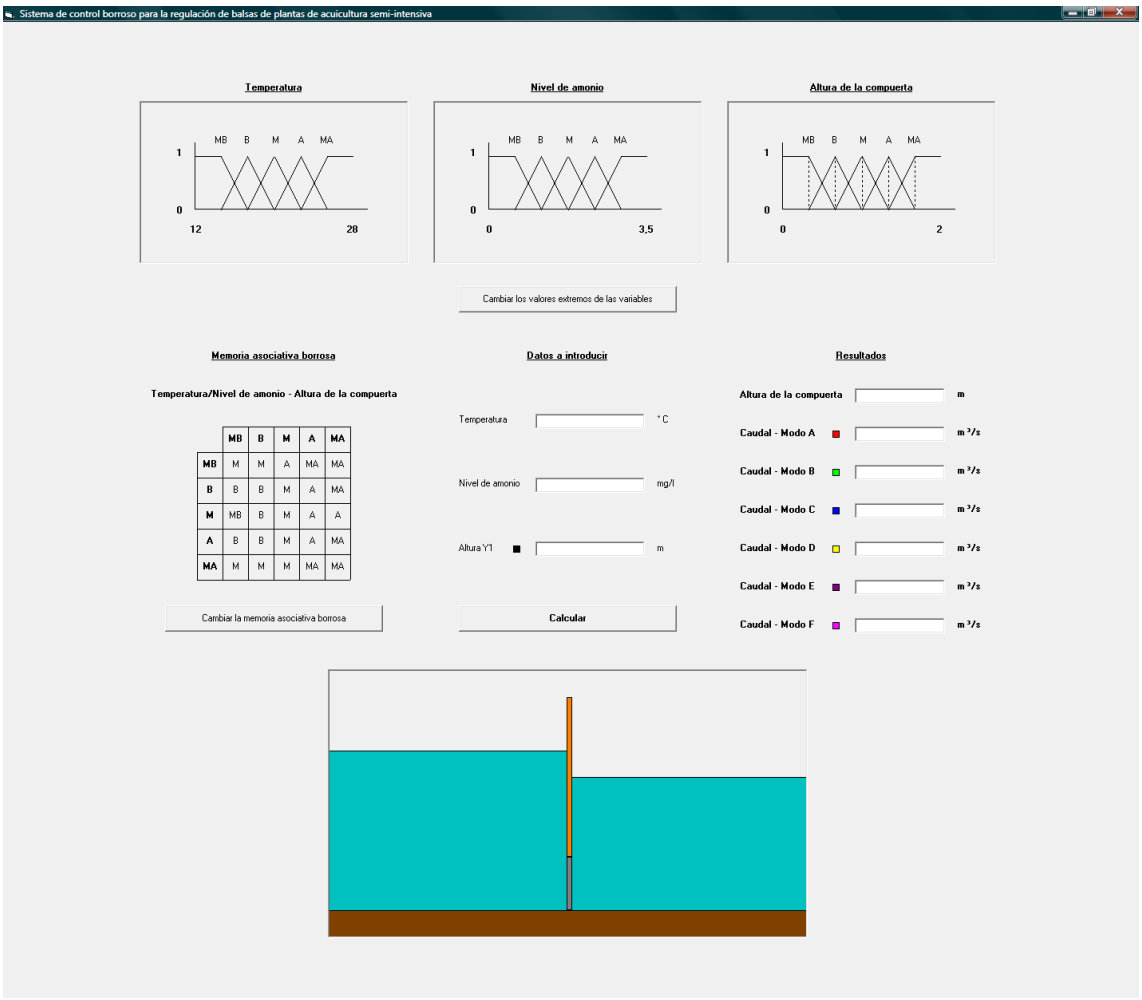


Figura 3.9. *Formulario principal, "Programa.frm"*

Los valores mínimos y máximos de estas variables pueden ser modificados mediante el formulario "Variables.frm" (*figura 3.10*), al cual se accede al pulsar sobre el botón *Cambiar los valores extremos de las variables*. Una vez que aparezca la ventana de este formulario, se podrá elegir entre cambiar los valores extremos de la variable "temperatura" (*figura 3.11*), "nivel de amonio" (*figura 3.12*) o "altura de la compuerta" (*figura 3.13*). Estos formularios hacen posible la utilización de esta aplicación como herramienta en

cualquier piscifactoría, ya que los límites de las variables de entrada y salida pueden ser establecidos en cualquier momento por el usuario.

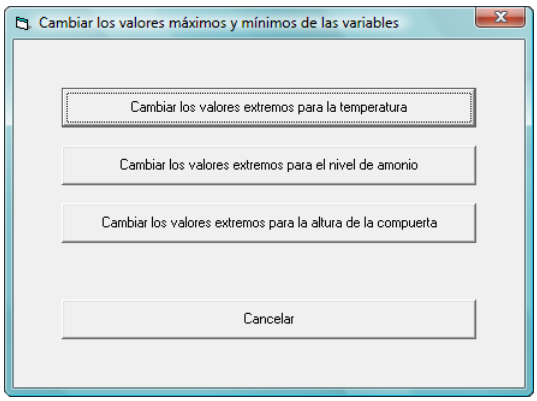


Figura 3.10. *Formulario "Variables.frm", el cual permite cambiar los valores extremos de las diferentes variables*

Figura 3.11. *Formulario "CT.frm", el cual permite cambiar los valores extremos de la variable de entrada "temperatura"*

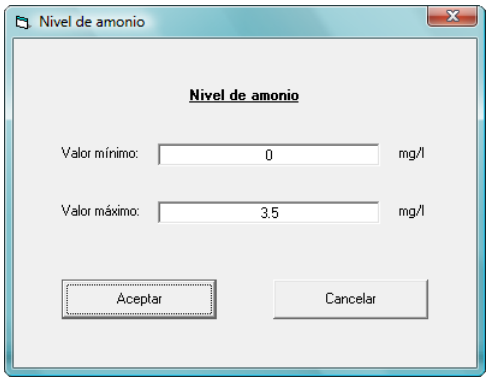


Figura 3.12. *Formulario "CNA.frm", el cual permite cambiar los valores extremos de la variable de entrada "nivel de amonio"*

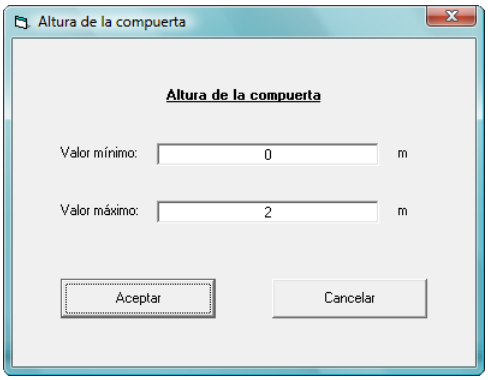


Figura 3.13. *Formulario "CA.frm", el cual permite cambiar los valores extremos de la variable de salida "altura de la compuerta"*

En la parte central izquierda del formulario "Programa.frm" se halla presentada la memoria asociativa borrosa que establece las reglas borrosas entre las variables de entrada y la variable de salida. Las reglas presentadas en esta memoria asociativa borrosa pueden ser modificadas a través del formulario "FAM.frm" (*figura 3.14*), al cual se tiene acceso al hacer click sobre botón *Cambiar memoria asociativa borrosa*.

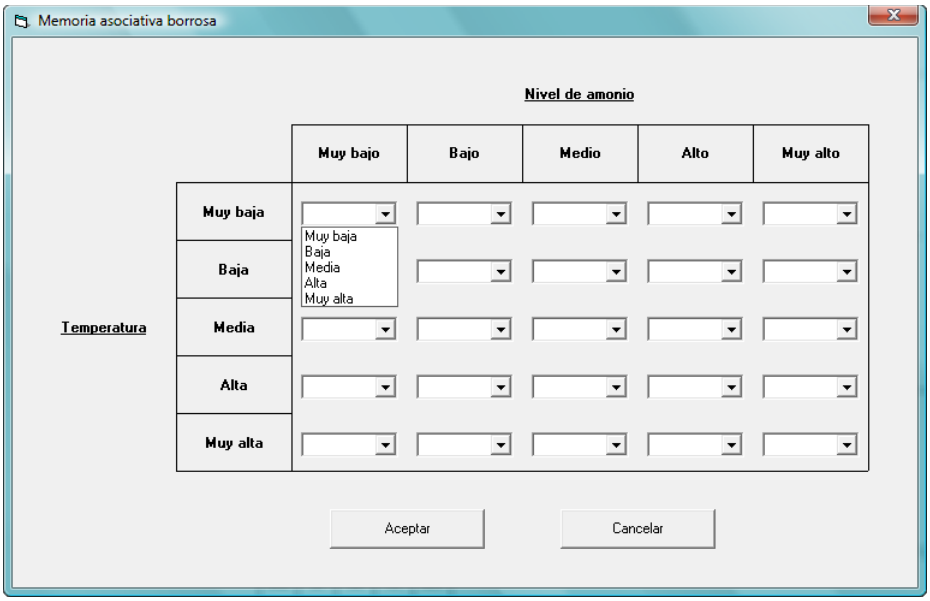


Figura 3.14. *Formulario "FAM.frm", el cual permite cambiar las reglas borrosas presentadas en la memoria asociativa borrosa*

Justo al lado de la memoria asociativa borrosa, en el centro del formulario "Programa.frm", se encuentra una de las zonas más importantes del mismo, ya que los controles que en ella se hallan son los encargados de recoger la información que el usuario transmite al programa, estos son, la

"temperatura", el "nivel de amonio" y la "altura y_1". Una vez que el usuario haya rellenado correctamente los espacios destinados a la recogida de los datos correspondientes a dichas variables, se encuentra en la disposición de hacer click sobre el botón *Calcular*. Mediante esta acción, se ejecutará la subrutina principal del formulario y se realizarán los cálculos pertinentes, ya expuestos en los anteriores apartados.

Finalmente, se mostrarán los resultados obtenidos en la zona central derecha del formulario "Programa.frm", al mismo tiempo que en la parte superior del mismo se dibujan las disposiciones de los valores de los conjuntos borrosos de entrada y de salida y en la parte inferior se representa gráficamente la posición tanto de la compuerta de la balsa como de las alturas y_1 e y_3, como muestra la *figura 3.15*.

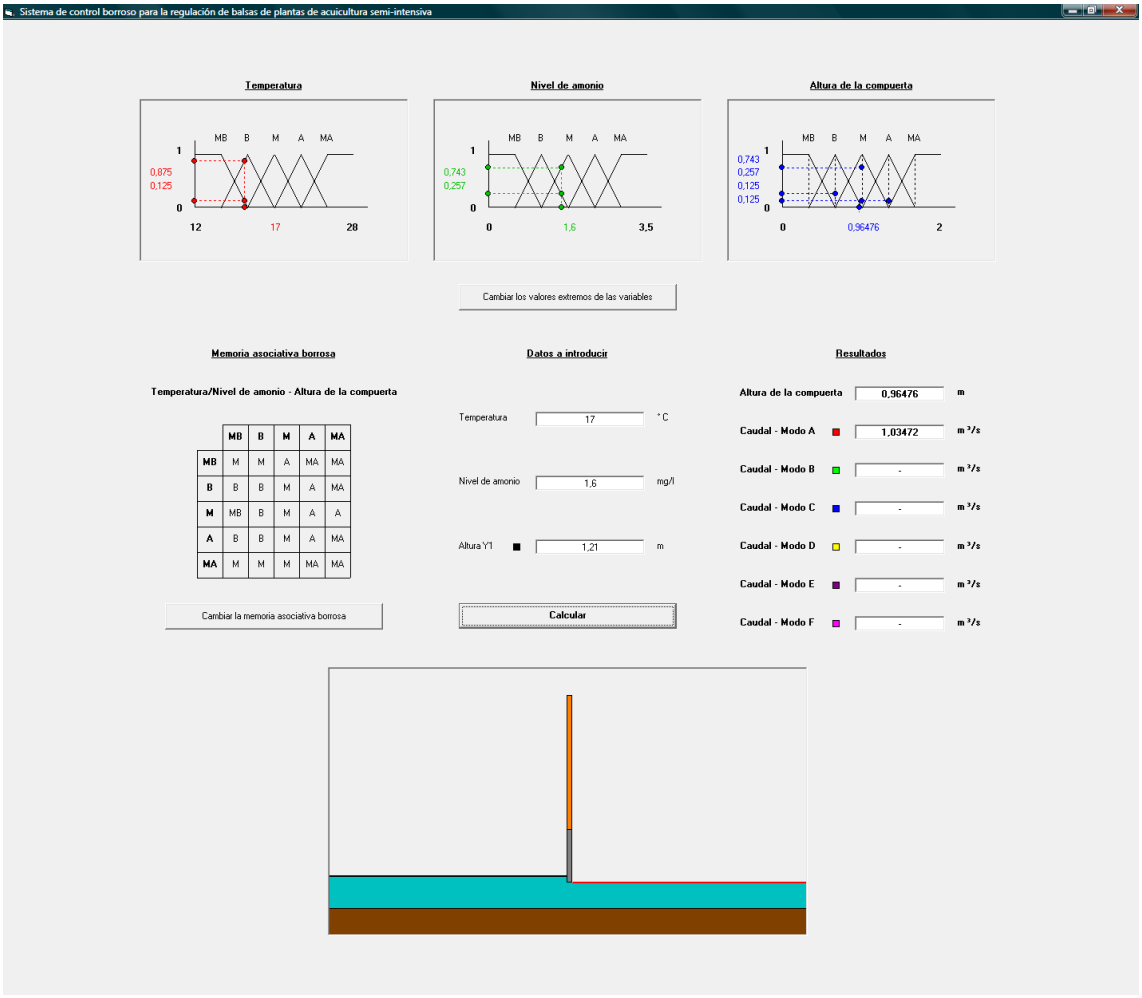


Figura 3.15. *Demostración de los cálculos realizados por la aplicación desarrollada*

4. RESULTADOS

4.1. Resultados obtenidos al utilizar la memoria asociativa borrosa inicial A

4.2. Resultados obtenidos al utilizar una memoria asociativa borrosa B menos restrictiva que la inicial

4.3. Resultados obtenidos al utilizar una memoria asociativa borrosa C más restrictiva que la inicial

4. RESULTADOS

Los resultados presentados en este capítulo son los derivados de establecer los valores de las variables de entrada y de salida al sistema de control borroso entre los siguientes rangos:

- Los valores de la variable de entrada “temperatura”, oscilarán entre 12 y 28 ºC.
- Por su parte, los valores de la variable de entrada “nivel de amonio”, oscilarán entre 0 y 3.5 mg/l.
- Finalmente, los valores de la variable de salida “altura de la compuerta”, oscilarán entre 0 y 2 m.

Además, el valor fijado para la anchura de la compuerta que regula la entrada y salida del caudal de agua hacia y desde la balsa es de 0.5 m.

A continuación, se exponen las relaciones de resultados obtenidos utilizando distintas memorias asociativas borrosas, una inicial, otra menos restrictiva que ésta y una última más restrictiva que la primera.

4.1. Resultados obtenidos al utilizar la memoria asociativa borrosa inicial A

		Nivel de amonio				
		MB	***B***	***M***	***A***	***MA***
Temperatura	***MB***	*M*	*M*	*A*	*MA*	*MA*
	B	*B*	*B*	*M*	*A*	*MA*
	M	*MB*	*B*	*M*	*A*	*A*
	A	*B*	*B*	*M*	*A*	*MA*
	MA	*M*	*M*	*M*	*MA*	*MA*

Figura 4.1. *Memoria asociativa borrosa A*

Al utilizar las bases de reglas borrosas expuestas en la memoria asociativa borrosa A (*figura 4.1*), la aplicación informática desarrollada

proporciona la relación de resultados de la altura de la compuerta y el caudal que pasa a través de la misma presentada en la *tabla 4.1*.

Fecha	Altura de la compuerta (m)	Caudal estimado (m^3/s)	Caudal real aproximado (m^3/s)
21/04/08	0.965	1.035	0.751
22/04/08	0.833	0.870	0.709
23/04/08	1.049	1.155	0.751
24/04/08	1.151	1.259	0.655
25/04/08	0.606	0.321	0.775
26/04/08	0.840	0.887	0.858
27/04/08	0.701	0.756	0.806
28/04/08	0.611	0.202	0.791
29/04/08	0.571	0.543	0.111
30/04/08	0.701	0.701	0.700
01/05/08	0.547	0.092	0.700
02/05/08	0.458	0.415	0.775
03/05/08	0.686	0.706	0.626
04/05/08	0.686	0.719	0.783
05/05/08	0.458	0.573	0.607
06/05/08	0.458	0.076	0.735
07/05/08	0.611	0.635	0.596
08/05/08	0.708	0.782	0.709
09/05/08	0.701	0.597	0.157
10/05/08	0.371	0.429	0.429
11/05/08	0.599	0.597	0.683
12/05/08	0.611	0.612	0.751
13/05/08	0.547	0.304	0.384
14/05/08	0.458	0.658	0.507
15/05/08	0.458	0.530	0.674
16/05/08	0.457	0.574	0.726
17/05/08	0.547	0.450	0.700
18/05/08	0.701	0.766	0.806
19/05/08	0.708	0.742	0.843
20/05/08	0.733	0.742	0.655
21/05/08	0.458	0.552	0.751
22/05/08	0.458	0.481	0.607
23/05/08	0.547	0.360	0.674
24/05/08	0.547	0.335	0.646
25/05/08	0.547	0.335	0.646
26/05/08	0.458	0.563	0.709
27/05/08	0.708	0.755	0.655
28/05/08	0.671	0.712	0.607
29/05/08	0.771	0.827	0.607

Tabla 4.1. *Resultados obtenidos utilizando la memoria asociativa borrosa A*

En la *figura 4.2* se muestra la representación gráfica de la variación de la altura de la compuerta a lo largo del tiempo, mientras que en la *figura 4.3* se muestra la representación gráfica de la variación del caudal, tanto estimado como real aproximado, a lo largo del tiempo.

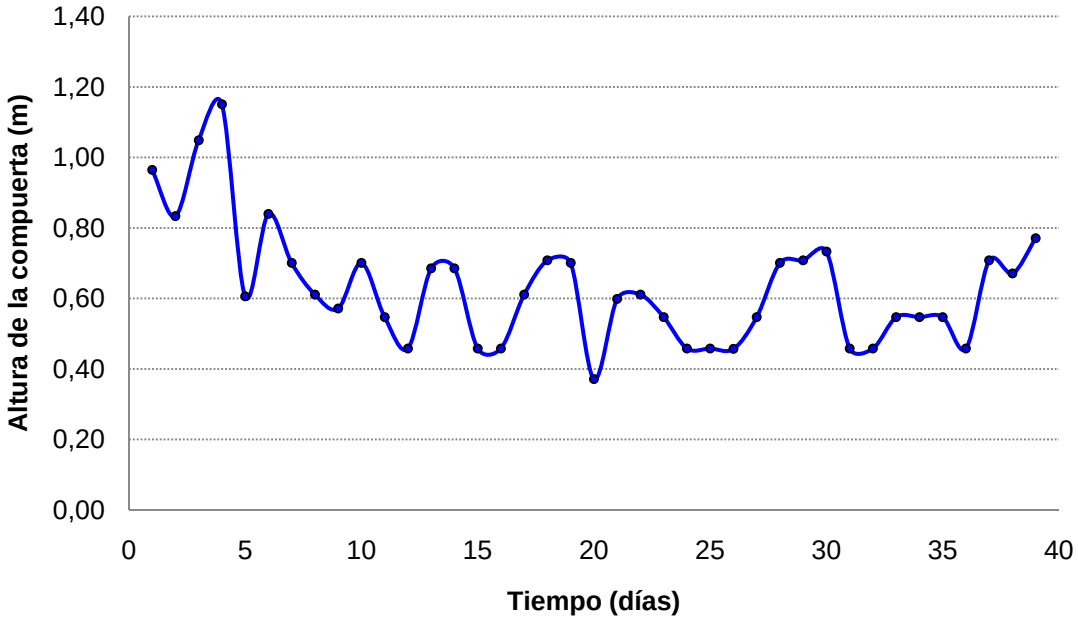


Figura 4.2. *Representación gráfica de la variación de la altura de la compuerta frente al tiempo (MAB A)*

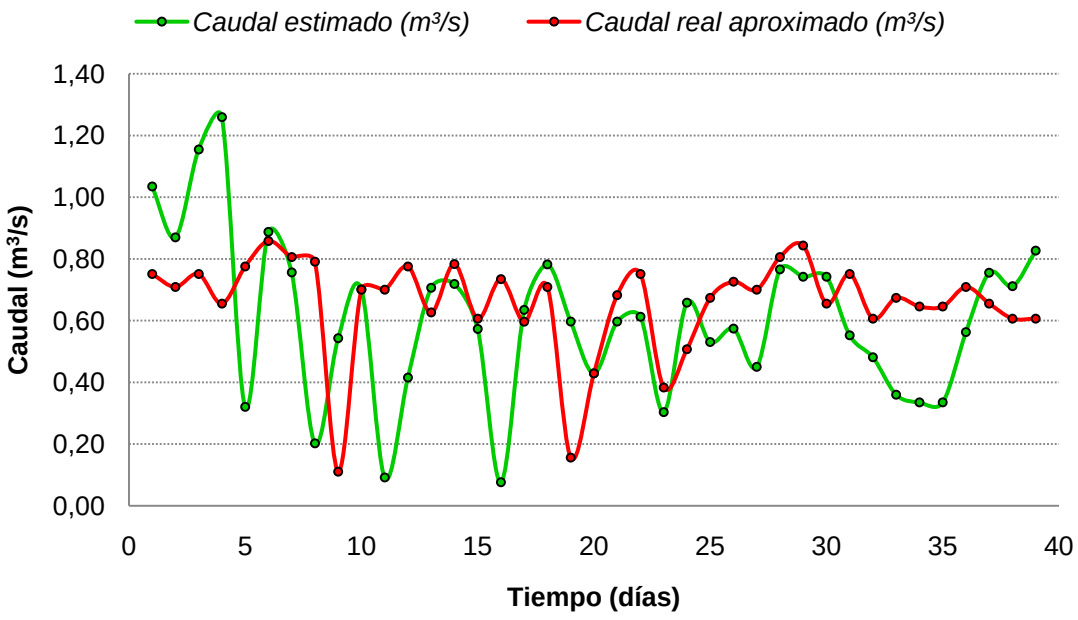


Figura 4.3. *Representación gráfica de la variación del caudal frente al tiempo (MAB A)*

4.2. Resultados obtenidos al utilizar una memoria asociativa borrosa B menos restrictiva que la inicial

Temperatura \ Nivel de amonio	MB	B	M	A	MA
MB	B	B	M	A	MA
B	MB	MB	B	M	A
M	MB	MB	B	M	M
A	MB	MB	B	M	A
MA	B	B	M	A	MA

Figura 4.4. *Memoria asociativa borrosa B*

Al utilizar las bases de reglas borrosas expuestas en la memoria asociativa borrosa B (*figura 4.4*), la aplicación informática desarrollada proporciona la relación de resultados de la altura de la compuerta y el caudal que pasa a través de la misma presentada en la *tabla 4.2*.

Fecha	Altura de la compuerta (m)	Caudal estimado (m^3/s)	Caudal real aproximado (m^3/s)
21/04/08	0.631	0.658	0.751
22/04/08	0.500	0.366	0.709
23/04/08	0.715	0.768	0.751
24/04/08	0.817	0.877	0.655
25/04/08	0.333	0.288	0.775
26/04/08	0.540	0.348	0.858
27/04/08	0.368	0.272	0.806
28/04/08	0.333	0.273	0.791
29/04/08	0.333	0.422	0.111
30/04/08	0.368	0.469	0.700
01/05/08	0.333	0.192	0.700
02/05/08	0.333	0.225	0.775
03/05/08	0.352	0.207	0.626
04/05/08	0.352	0.241	0.783
05/05/08	0.333	0.288	0.607
06/05/08	0.333	0.339	0.735
07/05/08	0.333	0.254	0.596
08/05/08	0.375	0.302	0.709
09/05/08	0.368	0.089	0.157
10/05/08	0.333	0.138	0.429

11/05/08	0.333	0.192	0.683
12/05/08	0.333	0.200	0.751
13/05/08	0.333	0.235	0.384
14/05/08	0.333	0.324	0.507
15/05/08	0.333	0.268	0.674
16/05/08	0.333	0.288	0.726
17/05/08	0.333	0.299	0.700
18/05/08	0.368	0.292	0.806
19/05/08	0.375	0.191	0.843
20/05/08	0.400	0.451	0.655
21/05/08	0.333	0.278	0.751
22/05/08	0.333	0.249	0.607
23/05/08	0.333	0.254	0.674
24/05/08	0.333	0.244	0.646
25/05/08	0.333	0.244	0.646
26/05/08	0.333	0.283	0.709
27/05/08	0.375	0.239	0.655
28/05/08	0.338	0.271	0.607
29/05/08	0.438	0.562	0.607

Tabla 4.2. *Resultados obtenidos utilizando la memoria asociativa borrosa B*

En la *figura 4.5* se muestra la representación gráfica de la variación de la altura de la compuerta a lo largo del tiempo, mientras que en la *figura 4.6* se muestra la representación gráfica de la variación del caudal, tanto estimado como real aproximado, a lo largo del tiempo.

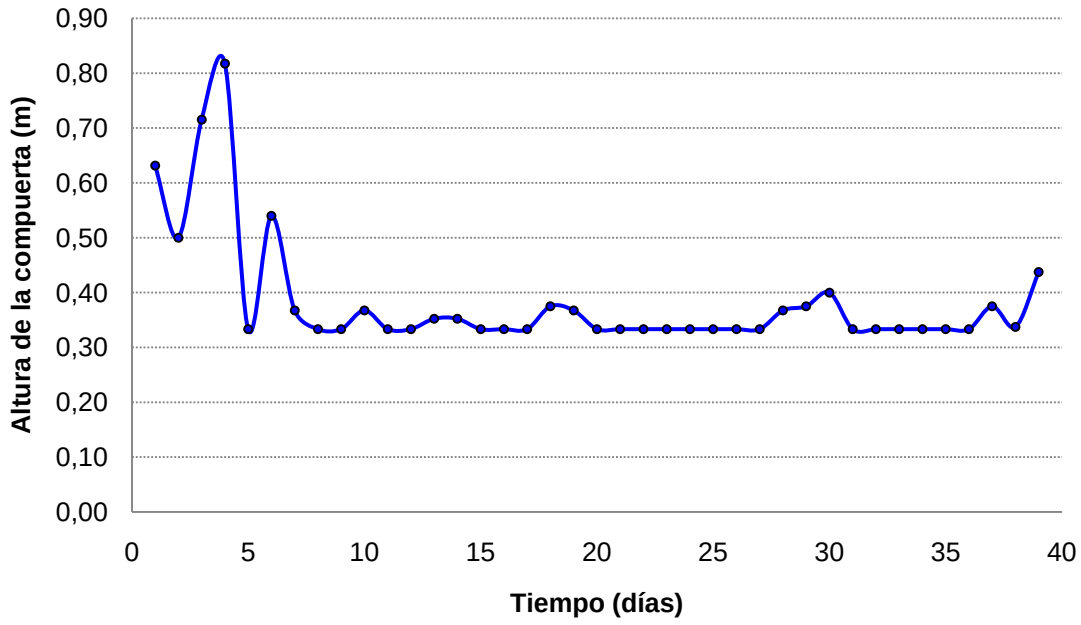


Figura 4.5. *Representación gráfica de la variación de la altura de la compuerta frente al tiempo (MAB B)*

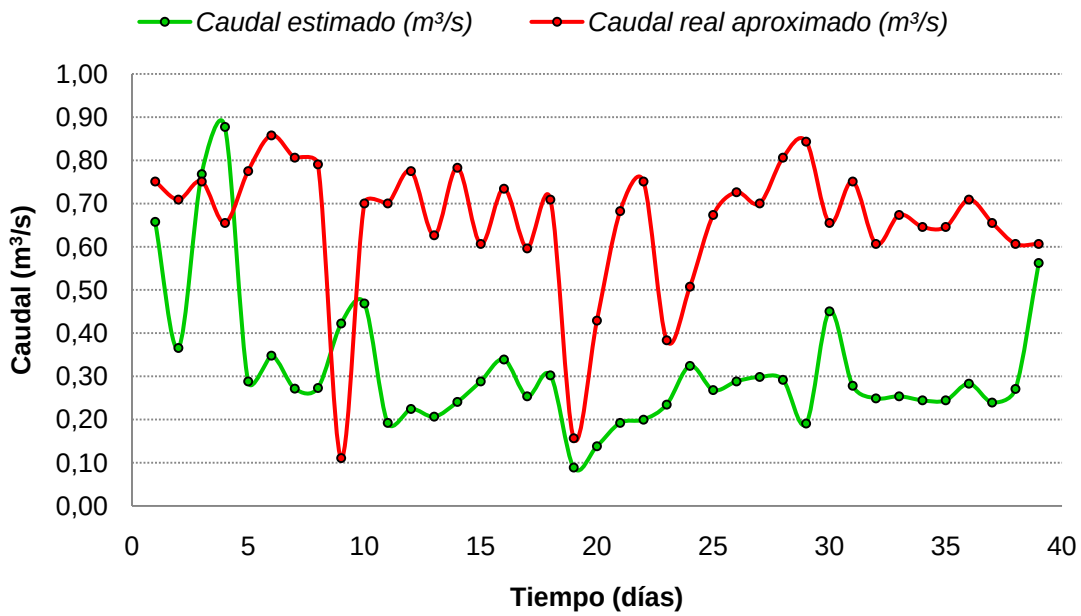


Figura 4.6. *Representación gráfica de la variación del caudal frente al tiempo (MAB B)*

4.3. Resultados obtenidos al utilizar una memoria asociativa borrosa C más restrictiva que la inicial

		Nivel de amonio				
		MB	**B**	**M**	**A**	**MA**
Temperatura	**MB**	*M*	*A*	*A*	*MA*	*MA*
	B	*B*	*M*	*A*	*MA*	*MA*
	M	*MB*	*B*	*M*	*A*	*MA*
	A	*B*	*M*	*A*	*MA*	*MA*
	MA	*M*	*A*	*A*	*MA*	*MA*

Figura 4.7. *Memoria asociativa borrosa C*

Al utilizar las bases de reglas borrosas expuestas en la memoria asociativa borrosa C (*figura 4.7*), la aplicación informática desarrollada proporciona la relación de resultados de la altura de la compuerta y el caudal que pasa a través de la misma presentada en la *tabla 4.3*.

Fecha	Altura de la compuerta (m)	Caudal estimado (m^3/s)	Caudal real aproximado (m^3/s)
21/04/08	1.265	1.374	0.751
22/04/08	0.833	0.870	0.709
23/04/08	1.287	1.430	0.751
24/04/08	1.373	1.514	0.655
25/04/08	0.686	0.736	0.775
26/04/08	1.173	1.261	0.858
27/04/08	0.942	1.038	0.806
28/04/08	0.694	0.738	0.791
29/04/08	0.571	0.543	0.111
30/04/08	0.830	0.839	0.700
01/05/08	0.618	0.618	0.700
02/05/08	0.458	0.415	0.775
03/05/08	0.686	0.706	0.626
04/05/08	0.686	0.719	0.783
05/05/08	0.458	0.573	0.607
06/05/08	0.458	0.076	0.735
07/05/08	0.694	0.728	0.596
08/05/08	0.708	0.782	0.709
09/05/08	0.830	0.712	0.157
10/05/08	0.371	0.429	0.429
11/05/08	0.630	0.630	0.683
12/05/08	0.694	0.702	0.751
13/05/08	0.618	0.634	0.384
14/05/08	0.458	0.658	0.507
15/05/08	0.458	0.530	0.674
16/05/08	0.457	0.574	0.726
17/05/08	0.618	0.317	0.700
18/05/08	0.942	1.052	0.806
19/05/08	0.708	0.742	0.843
20/05/08	0.866	0.885	0.655
21/05/08	0.458	0.552	0.751
22/05/08	0.458	0.481	0.607
23/05/08	0.618	0.643	0.674
24/05/08	0.618	0.638	0.646
25/05/08	0.618	0.638	0.646
26/05/08	0.458	0.563	0.709
27/05/08	0.708	0.755	0.655
28/05/08	0.671	0.712	0.607
29/05/08	0.771	0.827	0.607

Tabla 4.3. *Resultados obtenidos utilizando la memoria asociativa borrosa C*

En la *figura 4.8* se muestra la representación gráfica de la variación de la altura de la compuerta a lo largo del tiempo, mientras que en la *figura 4.9* se

muestra la representación gráfica de la variación del caudal, tanto estimado como real aproximado, a lo largo del tiempo.

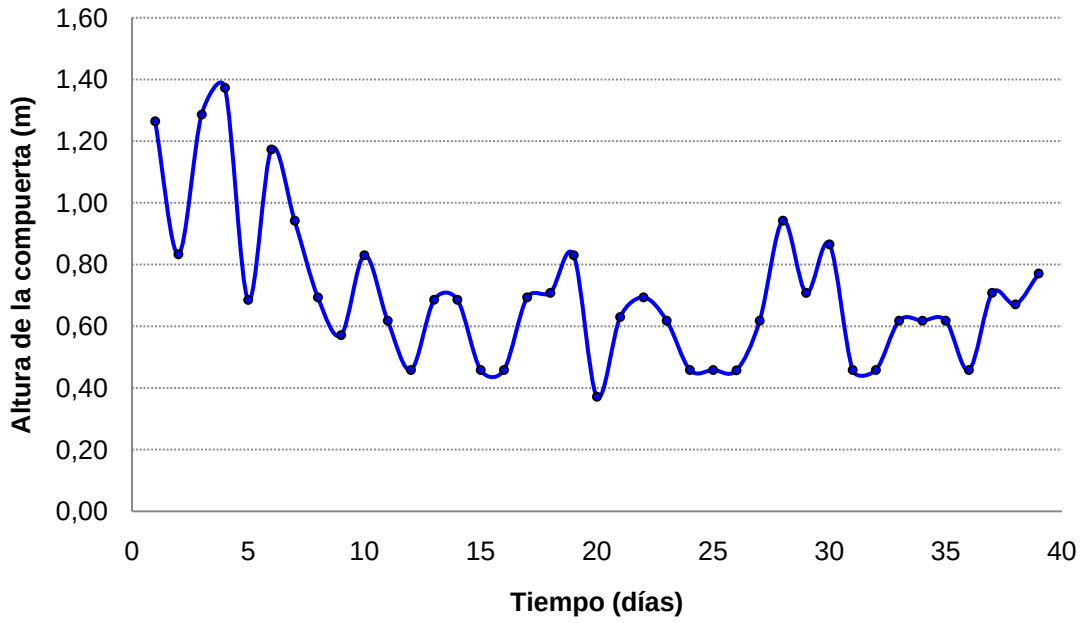


Figura 4.8. *Representación gráfica de la variación de la altura de la compuerta frente al tiempo (MAB C)*

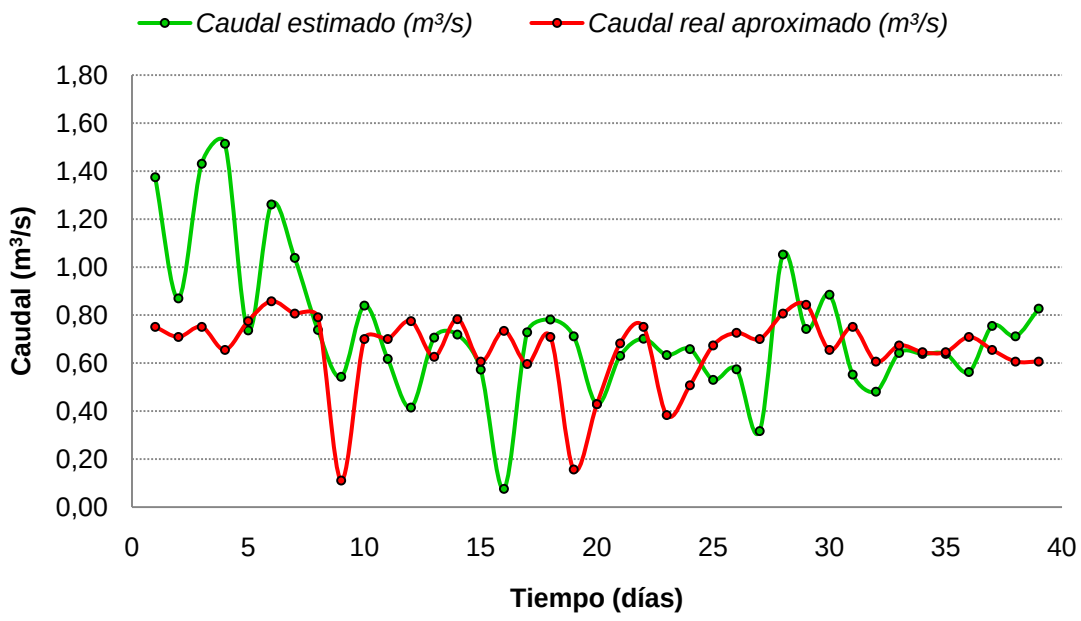


Figura 4.9. *Representación gráfica de la variación del caudal frente al tiempo (MAB C)*

Finalmente, en la *figura 4.10* se muestra la representación gráfica de la variación de los caudales obtenidos al aplicar las reglas borrosas de las memorias asociativas A, B y C y del caudal real aproximado a lo largo del tiempo.

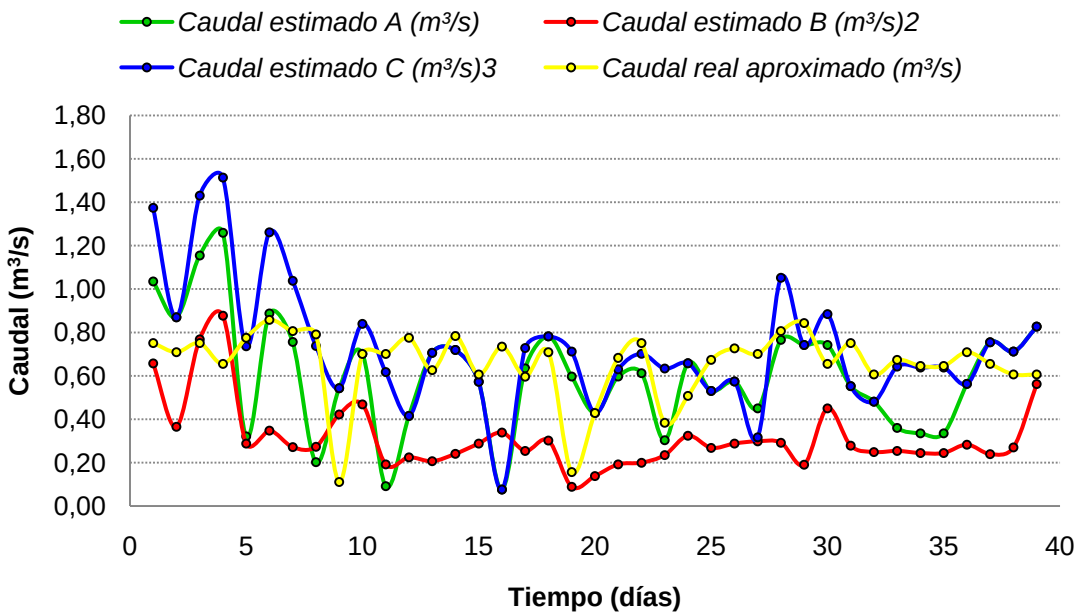


Figura 4.10. *Representación gráfica de la variación de los distintos caudales frente al tiempo*

Simulaciones	RMSE (m^3/s)
Simulación A	0.288
Simulación B	0.404
Simulación C	0.304

Tabla 4.4. *Valores de la magnitud de evaluación RMSE de cada simulación respecto a los valores reales aproximados*

Simulación	Media	Desviación estándar	t	Grados de libertad	Nivel de significación (p)
A	0.606	0.257	-1.137	38	0.262
B*	0.317	0.160	-10.079	38	< 0.001
C	0.734	0.289	1.625	38	0.112
Real	0.658	0.159	-	-	-

(*) Diferencias significativas

Tabla 4.5. *Resultados del test t de Student para la comparación de cada simulación con los valores reales aproximados*

La comparación de la variación de caudales en el tiempo para cada una de las memorias asociativas borrosas probadas con respecto a los valores reales aproximados se ha realizado mediante el cálculo de la magnitud de evaluación RMSE (*tabla 4.4*) y del test t de Student (*tabla 4.5*).

5. DISCUSIÓN

5. DISCUSIÓN

El sistema de control borroso desarrollado para la regulación de compuertas de entrada a balsas de producción acuícola semi-intensiva, utilizando como variables de entrada la temperatura y el nivel de amonio del agua, se constituye como una herramienta de apoyo fundamental en la planificación de la circulación del agua en este tipo de instalaciones. De este modo, esta toma de decisiones, que normalmente suele realizarse basándose en la experiencia de los técnicos responsables, puede ser complementada con el modelo desarrollado en este trabajo.

Se han evaluado tres conjuntos de reglas de decisión diferentes (o tres memorias asociativas distintas) para la regulación de una compuerta de entrada de agua con los datos diarios reales de temperatura y nivel de amonio de una de las balsas de la empresa "*Langostinos de Huelva*". La regulación de caudal más aproximada a la situación real es la correspondiente a la memoria asociativa borrosa A, que se constituye como el conjunto intermedio de reglas de decisión entre las memorias B y C (B = conjunto de reglas menos restrictivo que A; C = conjunto de reglas más restrictivo que A; para niveles altos de temperatura y nivel de amonio, la compuerta se abrirá más con el conjunto de reglas A que con el B; para niveles altos de temperatura y nivel de amonio, la compuerta se abrirá menos con el conjunto de reglas C que con el A).

Así, los resultados muestran diferencias significativas entre las medias de caudal obtenidas mediante el sistema borroso con el conjunto de reglas B y los valores reales aproximados (p < 0.001). Sin embargo, no se presentan diferencias significativas entre las medias de caudal del sistema borroso con los conjuntos de reglas A (p = 0.262) y C (p = 0.112) y los valores reales aproximados. Siendo menores las diferencias en el sistema borroso con la memoria asociativa borrosa A. Esto coincide con los resultados obtenidos mediante la evaluación de la magnitud RMSE, cuyo valor también es menor para la memoria asociativa borrosa A ($RMSE_{A \rightarrow Real\ aproximado}$ = 0.288 m^3/s < $RMSE_{C \rightarrow Real\ aproximado}$ = 0.304 m^3/s < $RMSE_{B \rightarrow Real\ aproximado}$ = 0.404 m^3/s).

Con respecto a la variación de la altura de la compuerta con el tiempo, el mayor rango de variaciones se presenta con la memoria asociativa C (entre 0.371 y 1.373 m), lo cual es lógico, ya que esta situación es la más restrictiva. También se presentan variaciones significativas en la altura de la compuerta con la memoria asociativa A (entre 0.371 y 1.151 m), que son algo menores que en la situación anterior. La memoria asociativa B, la cual se corresponde con la situación menos restrictiva, presenta una menor variación (entre 0.333 y 0.817 m), manteniéndose los valores de la altura de la compuerta en 0.3 y 0.4 metros la mayor parte del tiempo. Esta falta de variación explica el peor

comportamiento de la memoria asociativa B frente a la situación real aproximada.

Las diferencias encontradas entre el sistema borroso y la situación real aproximada pueden deberse a la necesidad de incorporación en el modelo de otras variables de entrada o de control, como pueden ser la turbidez, el color o el oxígeno del agua, que en la situación real de "*Langostinos de Huelva*" son otros parámetros utilizados en la toma de decisiones por parte de los técnicos responsables.

Por otra parte, también sería recomendable incorporar variaciones en la geometría de los conjuntos borrosos. Si bien en este trabajo se han conseguido buenos resultados considerando conjuntos borrosos triangulares con distancias homogéneas entre sus centroides, en trabajos de otros autores se han conseguido mejorías importantes como la optimación de la geometría de estos conjuntos borrosos (Pulido-Calvo y Gutiérrez-Estrada, 2009).

Asimismo, habría que mencionar que los caudales reales se han obtenido mediante cálculos aproximados utilizando la expresión experimental para desagües por orificios sumergidos. Esto puede presentar desviaciones con respecto al cálculo de caudales del sistema de control borroso. En trabajos futuros, podría plantearse la medición de los caudales reales exactos mediante métodos de aforo más precisos que podrían implicar la obtención de una ecuación de gasto específica para los tablones con orificios que se utilizan actualmente en la planta de acuicultura en estudio.

En toda modelación heurística es primordial la calidad y cantidad de los datos (Yang *et al.*, 1997). En este sentido, se piensa que una mejor aproximación a la realidad se conseguiría con un mayor número de datos experimentales. Se comprueba que en el conjunto de datos en estudio existen dos valores medidos de lámina de agua que son bastante inferiores al resto, lo que ha supuesto la no consideración de todas las reglas planteadas en la memoria asociativa borrosa.

6. CONCLUSIONES

6. CONCLUSIONES

a) El sistema de control borroso desarrollado para la regulación de la entrada de caudales en balsas de plantas de acuicultura semi-intensiva ha demostrado ser útil como herramienta de apoyo para la toma de decisiones de los técnicos responsables de este tipo de instalaciones.

b) Aunque los resultados obtenidos son satisfactorios, se cree necesaria una optimación del sistema borroso considerando la geometría de los conjuntos borrosos y la memoria asociativa borrosa.

c) La evaluación y fiabilidad del sistema de control borroso debería ser analizada con la incorporación de otras variables de entrada o de control, como la turbidez, el color o la concentración de oxígeno del agua, que normalmente son parámetros utilizados en la toma de decisiones por parte de los técnicos responsables de este tipo de instalaciones.

d) La metodología propuesta puede ser considerada como de propósito general y, por tanto, podría aplicarse en la regulación de balsas o depósitos de otros sistemas de distribución de agua.

REFERENCIAS

REFERENCIAS

Akter, T., Simonovic, S.P., 2005. Aggregation of fuzzy views of a large number of stakeholders for multi-objective flood management decision-making. *Journal of Environmental Management*, 77(2), 133-143.

Aqil, M., Kita, I., Yano, A., Nishiyama, S., 2006. Prediction of flood abnormalities for improved public safety using a modified adaptive neuro-fuzzy inference system. *Water Science and Technology*, 54(11-12), 11-19.

ASCE Task Committee on Application of Artificial Neural Networks in Hydrology, 2000a. Artificial neural networks in hydrology. I: Preliminary concepts. *Journal of Hydrology Engineering,* 5(2), 115-123.

ASCE Task Committee on Application of Artificial Neural Networks in Hydrology, 2000b. Artificial neural networks in hydrology. II: Hydrologic applications. *Journal of Hydrology Engineering,* 5(2), 124-137.

Bagis, A., 2003. Fuzzy and PD controller based intelligent control of spillway gates of dams. *Journal of Intelligent & Fuzzy Systems*, 14(1), 25-36.

Bagis, A., Karaboga, D., 2004. Artificial neural networks and fuzzy logic based control of spillway gates of dams. *Hydrological Processes*, 18(13), 2485-2501.

Bardossy, A., 1996. The use of fuzzy rules for the description of elements of the hydrological cycle. *Ecological Modelling*, 85(1), 59-65.

Begovich, O., Ruíz, V.M., Georges, D., Besanc, G., 2005. Real-time application of a fuzzy gain scheduling control scheme to a multi-pool open irrigation canal prototype. *Journal of Intelligent & Fuzzy Systems*, 16(3), 189-199.

Biscos, C., Mulholland, M., Le Lann, M-V., Buckley, C.A., Brouckaert, C.J., 2003. Optimal operation of water distribution networks by predictive control using MINLP. *Water SA*, 29(4), 393-403.

Brion, L.M., Mays, L.W., 1991. Methodology for optimal operation of pumping stations in water distribution systems. *Journal of Hydraulic Engineering*, 117(11), 1-19.

Buchleiter, G.W., Heermann, D.F., 1986. Using computers to manage irrigation systems. *Journal of Water Resources Planning and Management*, 112(3), 354-365.

Buchleiter, G.W., Heermann, D.F., 1990. Management of multiple pump stations. *Applied Engineering in Agriculture*, 6(1), 39-44.

Cembrano, G., Wells, G., Quevedo, J., Pérez, R., Argelaguet, R., 2000. Optimal control of a water distribution network in a supervisory control system. *Control Eng. Practice*, 8, 1177-1188.

Chang, N.B., Chen, H.W., Ning, S.K., 2001. Identification of river water quality using the Fuzzy Synthetic Evaluation approach. *Journal of Environmental Management*, 63(3), 293-305.

Chanona, J., Pastor, L., Borrás, L., Seco, A., 2006. Application of a fuzzy algorithm for pH control in a struvite crystallisation reactor. *Water Science and Technology*, 53(12), 161-168.

Charte, F., 1998. *Programación con Visual Basic 6*. ANAYA Multimedia, Madrid.

Chen, W.C., Chang, N.B., Chen, J.C., 2003. Rough set-based hybrid fuzzy-neural controller design for industrial wastewater treatment. *Water Research*, 37(1), 95-107.

Cheng, C.T., 1999. Fuzzy optimal model for the flood control system of the upper and middle reaches of the Yangtze River. *Hydrological Sciences Journal – Journal des Sciences Hydrologiques*, 44(4), 573-582.

Colt, J., Plesha, P., Huguenin, J., 2006. Impact of net positive suction head on the design and operation of seawater pumping systems for use in aquaculture. *Aquacultural Engineering*, 35, 239-257.

Dubrovin, T., Jolma, A., Turunen, E., 2002. Fuzzy model for real-time reservoir operation. *Journal of Water Resources Planning and Management*, 128(1), 66-73.

Ernst, D.H., Bolte, J.P., Nath, S.S., 2000. AquaFarm: simulation and decision support for aquaculture facility design and management planning. *Aquacultural Engineering*, 23,121–179.

FAO. 2000. *Examen mundial de la pesca y la acuicultura*. Organización de las Naciones Unidas para la Agricultura, Roma.

Gutiérrez-Estrada, J.C., De-Pedro-Sanz, E., López-Luque, R., Pulido-Calvo, I., 2005. SEDPA, an expert system for disease diagnosis in eel rearing systems. *Aquacultural Engineering*, 33, 110-125.

Hasebe, M., Nagayama, Y., 2002. Reservoir operation using the neural network and fuzzy systems for dam control and operation support. *Advances in Engineering Software*, 33(5), 245-260.

Jowitt, P.W., Germanopoulos, G., 1992. Optimal pump scheduling in water-supply networks. *Journal of Water Resources Planning and Management*, 118(4), 406-422.

Karaboga, D., Bagis, A., Haktanir, T., 2008. Controlling spillway gates of dams by using fuzzy logic controller with optimum rule number. *Applied Soft Computing*, 8, 232-238.

Kerr, N.M. 1981. Design of equipment and selection of materials-an engineer's assessment. *Aquaculture in heated effluents and recicurlation systems* (ed. K. Tiews), Vol. I, 151-181. Heenemann Verlagsgesellschaft mbH, Berlín.

Lai, E., Lundie, S., Ashbolt, N.J., 2007. A new approach to aid urban water management decision making using trade-off sacrifice modelled by fuzzy logic. *Water Science and Technology*, 56(8), 11-20.

Lee, C.S., Chang, S.P., 2005. Interactive fuzzy optimization for an economic and environmental balance in a river system. *Water Research*, 39(1), 221-231.

Lee, P.G., 2000. Process control and artificial intelligence software for aquaculture. *Aquacultural Engineering*, 23(1-3), 13-36.

Lee, P.G., Lea, R.N., Dohmann, E., Prebilsky, W., Turk, P.E., Ying, H., Whitson, J.L., 2000. Denitrification in aquaculture systems: an example of a fuzzy logic control problem. *Aquacultural Engineering*, 23(1-3), 37-59.

León, C., Martín, S., Elena, J.M., Luque, J., 2000. EXPLORE-Hybrid expert system for water networks management. *Journal of Water Resources Planning and Management*, 126(2), 65-74.

Li, J.B., Huang, G.H., Zeng, G.M., Maqsood, I., Huang, Y., 2007. An integrated fuzzy-stochastic modeling approach for risk assessment of groundwater contamination. *Journal of Environmental Management*, 82(2), 173-188.

Lian, S.T., Marzuki, K., Rubiyah, Y., 1998. Tuning of a neuro-fuzzy controller by genetic algorithms with an application to a coupled-tank liquid-level control system. *Engineering Applications of Artificial Intelligence*, 11(4), 517-529.

Liou, S.M., Lo, S.L., Hu, C.Y., 2003. Application of two-stage fuzzy set theory to river quality evaluation in Taiwan. *Water Research*, 37(6), 1406-1416.

Marsili-Libelli, S., 2004. Fuzzy prediction of the algal blooms in the Orbetello lagoon. *Environmental Modelling Software*, 19, 799-808.

Martín-del-Brío, B., Sanz-Molina, A., 2001. *Redes Neuronales y Sistemas Borrosos*. Ra-Ma, Madrid.

Moradi-Jalal, M., Mariño, M.A., Afshar, A., 2003. Optimal design and operation of irrigation pumping stations. *Journal of Irrigation and Drainage Engineering*, 129(3), 149-154.

Moreno, I., 2006. *Estimación neuroborrosa de la demanda de agua en la zona regable de Fuente Palmera (Córdoba)*. Proyecto Fin de Carrera, Escuela Politécnica Superior, Universidad de Huelva.

Munson, B.R., Young, D.F., Okiishi, T.H., 1999. *Fundamentos de mecánica de fluidos*. Limusa Wiley, México D.F., México.

Murnleitner, E., Becker, T.M., Delgado, A., 2002. State detection and control of overloads in the anaerobic wastewater treatment using fuzzy logic. *Water Research*, 36(1), 201-211.

Nasiri, F., Huang, G., Fuller, N., 2007. Prioritizing groundwater remediation policies: A fuzzy compatibility analysis decision aid. *Journal of Environmental Management*, 82(1), 13-23.

Ning, S.K., Chang, N.B., 2004. Optimal expansion of water quality monitoring network by fuzzy optimization approach. *Environmental Monitoring and Assessment*, 91(1-3), 145-170.

Nitivattananon, V., Sadowski, E.C., Quimpo, R.G., 1996. Optimization of water supply system operation. *Journal of Water Resources Planning and Management*, 122(5), 374-384.

Papandroulakis, N., Markakis, G., Divanach, P., Kentouri, M., 2000. Feeding requirements of sea bream (Sparus aurata) larvae under intensive rearing conditions - Development of a fuzzy logic controller for feeding. *Aquacultural Engineering*, 21(4), 285-299.

Petroutsos, E., 1999. *Visual Basic 6*. ANAYA Multimedia, Madrid.

Planells, P., Carrión, P., Ortega, J.F., Moreno, M.A., Tarjuelo, J.M., 2005. Pumping selection and regulation for water-distribution networks. *Journal of Irrigation and Drainage Engineering*, 131(3), 273-281.

Pulido-Calvo, I., Gutiérrez-Estrada, J.C., 2009. Improved irrigation water demand forecasting using a soft-computing hybrid model. *Biosystems Engineering*, 102(2), 202-218.

Pulido-Calvo, I., Gutiérrez-Estrada, J.C., Asensio-Fernández, R., 2006b. Optimal design of pumping stations of inland intensive fishfarms. *Aquacultural Engineering*, 35, 283-291.

Pulido-Calvo, I., Gutiérrez-Estrada, J.C., Corbacho, J.M., 2008. Pipes size selection of wáter distribution systems of fishfarms. *Aquacultural Engineering*, 39, 43-52.

Pulido-Calvo, I., Gutiérrez-Estrada, J.C., López-Luque, R., Roldán, J., 2006a. Regulating reservoirs in pressurized irrigation water supply systems. *Journal of Water Supply: Research and Technology-AQUA*, 55(5), 367-381.

Pulido-Calvo, I., Portela, M.M., 2007a. Application of neural approaches to one-step daily flow forecasting in Portuguese watersheds. *Journal of Hydrology*, 332, 1–15.

Pulido-Calvo, I., Portela, M.M., 2007b. Aproximaciones neuronales univariantes para la predicción de caudales diarios en cuencas portuguesas. *Ingeniería del Agua,* 14(2), 97-111.

Pulido-Calvo, I., Roldán, J., López-Luque, R., Gutiérrez-Estrada, J.C., 2003. Water delivery system planning considering irrigation simultaneity. *Journal of Irrigation and Drainage Engineering*, 129(4), 247-255.

Tarquin, A.J., Dowdy, J., 1989. Optimal pump operation in water distribution. *Journal of Hydraulic Engineering*, 115(2), 496-501.

Ulanicki, B., Rance, J.P., Davis, D., Chen, S., 1993. Computer-aided optimal pump selection for water distribution networks. *Journal of Water Resources Planning and Management*, 119(5), 542-562.

Yang, C.C., Prasher, S.O., Lacroix, R., Sreekanth, S., Patni, N.K., Masse, L., 1997. Artificial neural network model for subsurface-drained farmlands. *Journal of Irrigation and Drainage Engineering*, 123, 285-292.

Yu, P.S., Chen, S.T., 2005. Updating real-time flood forecasting using a fuzzy rule-based model. *Hydrological Sciences Journal – Journal des Sciences Hydrologiques*, 50(2), 265-278.

Zadeh, L., 1965. Fuzzy sets. *Information and Control*, 8, 338-353.

Zessler, U., Shamir, U., 1989. Optimal operation of water distribution systems. *Journal of Water Resources Planning and Management*, 115, 735-752.

ANEXO

a) Código fuente del formulario "Programa.frm"

b) Código fuente del formulario "FAM.frm"

c) Código fuente del formulario "Variables.frm"

d) Código fuente del formulario "CT.frm"

e) Código fuente del formulario "CNA.frm"

f) Código fuente del formulario "CA.frm"

ANEXO

a) Código fuente del formulario “Programa.frm”

```
' Calcularmin calcula el valor mínimo de la función de pertenencia y
su correspondiente etiqueta.
Sub Calcularmin(x, y, etiqueta)
If x <= 1 Then
    y = 1
    etiqueta = 0
ElseIf x > 1 And x <= 1.5 Then
    y = x - 1
    etiqueta = 1
ElseIf x > 1.5 And x <= 2 Then
    y = 2 - x
    etiqueta = 0
ElseIf x > 2 And x <= 2.5 Then
    y = x - 2
    etiqueta = 2
ElseIf x > 2.5 And x <= 3 Then
    y = 3 - x
    etiqueta = 1
ElseIf x > 3 And x <= 3.5 Then
    y = x - 3
    etiqueta = 3
ElseIf x > 3.5 And x <= 4 Then
    y = 4 - x
    etiqueta = 2
ElseIf x > 4 And x <= 4.5 Then
    y = x - 4
```

```
    etiqueta = 4
ElseIf x > 4.5 And x <= 5 Then
    y = 5 - x
    etiqueta = 3
Else
    y = 1
    etiqueta = 4
End If
End Sub
' Calcularmax calcula el valor máximo de la función de pertenencia y
su correspondiente etiqueta.
Sub Calcularmax(x, y, etiqueta)
If x <= 1 Then
    y = 1
    etiqueta = 0
ElseIf x > 1 And x <= 1.5 Then
    y = 2 - x
    etiqueta = 0
ElseIf x > 1.5 And x <= 2 Then
    y = x - 1
    etiqueta = 1
ElseIf x > 2 And x <= 2.5 Then
    y = 3 - x
    etiqueta = 1
ElseIf x > 2.5 And x <= 3 Then
    y = x - 2
    etiqueta = 2
ElseIf x > 3 And x <= 3.5 Then
    y = 4 - x
    etiqueta = 2
ElseIf x > 3.5 And x <= 4 Then
```

```
    y = x - 3
    etiqueta = 3
ElseIf x > 4 And x <= 4.5 Then
    y = 5 - x
    etiqueta = 3
ElseIf x > 4.5 And x <= 5 Then
    y = x - 4
    etiqueta = 4
Else
    y = 1
    etiqueta = 4
End If
End Sub
' MAB aplica las reglas introducidas en la memoria asociativa borrosa.
Sub MAB(AA, BB, etiqueta)
Dim r As Integer
Dim s As Integer
Dim w As Integer
w = 0
For s = 0 To 4
    For r = 0 To 4
        If AA = r And BB = s Then
            etiqueta = Clasificar(A(w).Caption)
        End If
        w = w + 1
    Next r
Next s
End Sub
' Clasificar cambia las etiquetas según el criterio MB B M A MA por
valores numéricos.
Function Clasificar(VV) As Integer
```

```
If VV = "MB" Then
    Clasificar = 0
ElseIf VV = "B" Then
    Clasificar = 1
ElseIf VV = "M" Then
    Clasificar = 2
ElseIf VV = "A" Then
    Clasificar = 3
ElseIf VV = "MA" Then
    Clasificar = 4
End If
End Function
' Calcularalturacompuerta calcula la altura de la compuerta.
Function Calcularalturacompuerta(etiqueta) As Double
If etiqueta = 0 Then
    Calcularalturacompuerta = altxmin.Caption + ((altxmax.Caption -
altxmin.Caption) / 6)
ElseIf etiqueta = 1 Then
    Calcularalturacompuerta = altxmin.Caption + ((2 * (altxmax.Caption
- altxmin.Caption)) / 6)
ElseIf etiqueta = 2 Then
    Calcularalturacompuerta = altxmin.Caption + ((3 * (altxmax.Caption
- altxmin.Caption)) / 6)
ElseIf etiqueta = 3 Then
    Calcularalturacompuerta = altxmin.Caption + ((4 * (altxmax.Caption
- altxmin.Caption)) / 6)
ElseIf etiqueta = 4 Then
    Calcularalturacompuerta = altxmin.Caption + ((5 * (altxmax.Caption
- altxmin.Caption)) / 6)
End If
End Function
' Subrutina principal.
Private Sub Calcular_Click()
```

```
' Declaración de variables.

Dim i As Integer

Dim k As Integer

Dim z As Integer

Dim m As Integer

Dim n As Integer

Dim aux As Double

Dim numerador As Double

Dim denominador As Double

Dim Temperatura As Double

Dim lbt As Double ' Valor modificado de la temperatura.

Dim yt(1) As Double ' Valores de la función de pertenencia de la
temperatura.

Dim et(1) As Integer ' Etiquetas de la temperatura.

Dim Amonio As Double

Dim lbna As Double ' Valor modificado del nivel de amonio.

Dim yna(1) As Double ' Valores de la función de pertenencia del nivel
de amonio.

Dim ena(1) As Integer ' Etiquetas del nivel de amonio.

Dim Altura As Double

Dim Alturam As Double 'Valor modificado de la altura de la compuerta
resultante.

Dim lba(3) As Double ' Valores de la altura de la compuerta.

Dim lbam(3) As Double ' Valores modificados de la altura de la
compuerta.

Dim ya(3) As Double ' Valores de la función de pertenencia de la
altura de la compuerta.

Dim yaord(3) As Double ' Valores ordenados de la función de
pertenencia de la altura de la compuerta.

Dim ea(3) As Integer ' Etiquetas de la altura de la compuerta.

Dim ay1 As Double

Dim h As Double ' Cociente entre la altura Y1 y la altura de la
compuerta.
```

```
Dim coefA As Double

Dim coefB As Double

Dim coefC As Double

Dim coefD As Double

Dim coefE As Double

Dim coefF As Double

Dim QmodoA As Double

Dim QmodoB As Double

Dim QmodoC As Double

Dim QmodoD As Double

Dim QmodoE As Double

Dim QmodoF As Double

Dim escala As Double

Dim y3A As Double

Dim y3B As Double

Dim y3C As Double

Dim y3D As Double

Dim y3E As Double

Dim y3F As Double

' Inicialización del contador i.

i = 0

' Comprobación de los valores introducidos.

If IsNumeric(tvalor.Text) And tvalor.Text >= tempxmin.Caption And
tvalor.Text <= tempxmax.Caption Then

    Temperatura = tvalor.Text

Else

    MsgBox "Error. El valor introducido para la temperatura no es
válido."

    i = i + 1

End If

If IsNumeric(avalor.Text) And avalor.Text >= amonxmin.Caption And
avalor.Text <= amonxmax.Caption Then
```

```
    Amonio = avalor.Text
Else
    MsgBox "Error. El valor introducido para el nivel de amonio no es válido."
    i = i + 1
End If
If IsNumeric(y1valor.Text) Then
    ay1 = y1valor.Text
Else
    MsgBox "Error. El valor introducido para la altura Y1 no es válido."
    i = i + 1
End If
' Si no hay errores, continuamos el programa.
If i = 0 Then
    ' Modificamos las variables para trabajar más fácilmente con ellas.
    lbt = (6 * (Temperatura - tempxmin.Caption)) / (tempxmax.Caption - tempxmin.Caption)
    lbna = (6 * (Amonio - amonxmin.Caption)) / (amonxmax.Caption - amonxmin.Caption)
    ' Llamamos a las subrutinas Calcularmin y Calcularmax.
    Call Calcularmin(lbt, yt(0), et(0))
    Call Calcularmax(lbt, yt(1), et(1))
    Call Calcularmin(lbna, yna(0), ena(0))
    Call Calcularmax(lbna, yna(1), ena(1))
    ' Llamamos a la subrutina MAB.
    Call MAB(ena(0), et(0), ea(0))
    Call MAB(ena(1), et(0), ea(1))
    Call MAB(ena(0), et(1), ea(2))
    Call MAB(ena(1), et(1), ea(3))
    ' Calculamos el valor de la función de pertenencia para la altura de la compuerta.
```

```
If yt(0) < yna(0) Then
    ya(0) = yt(0)
Else
    ya(0) = yna(0)
End If
If yt(0) < yna(1) Then
    ya(1) = yt(0)
Else
    ya(1) = yna(1)
End If
If yt(1) < yna(0) Then
    ya(2) = yt(1)
Else
    ya(2) = yna(0)
End If
If yt(1) < yna(1) Then
    ya(3) = yt(1)
Else
    ya(3) = yna(1)
End If
' Calculamos los valores para la altura de la compuerta.
lba(0) = Calcularalturacompuerta(ea(0))
lba(1) = Calcularalturacompuerta(ea(1))
lba(2) = Calcularalturacompuerta(ea(2))
lba(3) = Calcularalturacompuerta(ea(3))
' Calculamos el valor final de la altura de la compuerta.
k = 0
numerador = 0
denominador = 0
If et(0) = et(1) And ena(0) = ena(1) Then
```

```
        numerador = ya(0) * lba(0)
        denominador = ya(0)
        k = 1
    ElseIf et(0) = et(1) And ena(0) <> ena(1) Then
        numerador = (ya(0) * lba(0)) + (ya(1) * lba(1))
        denominador = ya(0) + ya(1)
        k = 2
    ElseIf et(0) <> et(1) And ena(0) = ena(1) Then
        numerador = (ya(0) * lba(0)) + (ya(2) * lba(2))
        denominador = ya(0) + ya(2)
        k = 3
    Else
        For m = 0 To 3
            numerador = numerador + (ya(m) * lba(m))
            denominador = denominador + ya(m)
        Next m
        k = 4
    End If
    Altura = numerador / denominador
    ' Calculamos los coeficientes de descarga.
    h = ay1 / Altura
    coefA = (72 - (3 * h)) / 155
    coefB = ((978 / 851) * (h ^ 6)) - ((1593 / 155) * (h ^ 5)) +
((3821 / 205) * (h ^ 4)) + ((2195 / 23) * (h ^ 3)) - ((36633 / 76) *
(h ^ 2)) + ((41876 / 53) * h) - (917 / 2)
    coefC = -((163 / 1677) * (h ^ 6)) + ((1377 / 550) * (h ^ 5)) -
((1015 / 38) * (h ^ 4)) + ((6497 / 43) * (h ^ 3)) - ((14341 / 30) * (h
^ 2)) + ((61783 / 77) * h) - (18981 / 34)
    coefD = -((40 / 38411) * (h ^ 6)) + ((179 / 4281) * (h ^ 5)) -
((659 / 951) * (h ^ 4)) + ((1591 / 262) * (h ^ 3)) - ((1634 / 55) * (h
^ 2)) + ((3315 / 43) * h) - (8692 / 105)
    coefE = -((8 / 368233) * (h ^ 6)) + ((37 / 24584) * (h ^ 5)) -
((371 / 8664) * (h ^ 4)) + ((333 / 520) * (h ^ 3)) - ((2672 / 503) *
(h ^ 2)) + ((2695 / 116) * h) - (5833 / 140)
```

```
    coefF = -((7 / 114036) * (h ^ 6)) + ((59 / 12957) * (h ^ 5)) -
((658 / 4713) * (h ^ 4)) + ((949 / 419) * (h ^ 3)) - ((8143 / 397) *
(h ^ 2)) + ((10530 / 107) * h) - (20522 / 105)

    ' Calculamos los caudales.

    QmodoA = coefA * 0.5 * Altura * Sqr(2 * 9.81 * ay1)

    QmodoB = coefB * 0.5 * Altura * Sqr(2 * 9.81 * ay1)

    QmodoC = coefC * 0.5 * Altura * Sqr(2 * 9.81 * ay1)

    QmodoD = coefD * 0.5 * Altura * Sqr(2 * 9.81 * ay1)

    QmodoE = coefE * 0.5 * Altura * Sqr(2 * 9.81 * ay1)

    QmodoF = coefF * 0.5 * Altura * Sqr(2 * 9.81 * ay1)

    ' Ordenamos los valores de la función de pertenencia de la altura
de la compuerta obtenidos anteriormente de menor a mayor.

    For m = 0 To 3

        yaord(m) = ya(m)

    Next m

    For m = 0 To 2

        For n = m To 3

            If yaord(m) > yaord(n) And m <> n Then

                aux = yaord(m)

                yaord(m) = yaord(n)

                yaord(n) = aux

            End If

        Next n

    Next m

    ' Copiamos los valores obtenidos en la ventana principal.

    tmin.Caption = Round(yt(0), 3)

    tmin.Visible = True

    tmax.Caption = Round(yt(1), 3)

    tmax.Visible = True

    temp.Caption = Round(tvalor.Text, 5)

    temp.Visible = True

    namin.Caption = Round(yna(0), 3)
```

```
namin.Visible = True
namax.Caption = Round(yna(1), 3)
namax.Visible = True
amon.Caption = Round(avalor.Text, 5)
amon.Visible = True
For m = 0 To 3
    alabel(m).Caption = Round(yaord(m), 3)
    alabel(m).Visible = True
Next m
alt.Caption = Round(Altura, 5)
alt.Visible = True
altvalor.Text = Round(Altura, 5)
z = 0
If h >= 0 And h < 2 Then
    q1valor.Text = Round(QmodoA, 5)
    q2valor.Text = "-"
    q3valor.Text = "-"
    q4valor.Text = "-"
    q5valor.Text = "-"
    q6valor.Text = "-"
    z = 1
ElseIf h >= 2 And h < 3 Then
    q1valor.Text = "-"
    q2valor.Text = Round(QmodoB, 5)
    q3valor.Text = "-"
    q4valor.Text = "-"
    q5valor.Text = "-"
    q6valor.Text = "-"
    z = 2
ElseIf h >= 3 And h < 4 Then
```

```
        q1valor.Text = "-"
        q2valor.Text = "-"
        q3valor.Text = Round(QmodoC, 5)
        q4valor.Text = "-"
        q5valor.Text = "-"
        q6valor.Text = "-"
        z = 3
    ElseIf h >= 4 And h <= (58 / 11) Then
        q1valor.Text = "-"
        q2valor.Text = "-"
        q3valor.Text = Round(QmodoC, 5)
        q4valor.Text = Round(QmodoD, 5)
        q5valor.Text = "-"
        q6valor.Text = "-"
        z = 4
    ElseIf h > (58 / 11) And h < 6 Then
        q1valor.Text = "-"
        q2valor.Text = "-"
        q3valor.Text = "-"
        q4valor.Text = Round(QmodoD, 5)
        q5valor.Text = "-"
        q6valor.Text = "-"
        z = 5
    ElseIf h >= 6 And h < 8 Then
        q1valor.Text = "-"
        q2valor.Text = "-"
        q3valor.Text = "-"
        q4valor.Text = Round(QmodoD, 5)
        q5valor.Text = Round(QmodoE, 5)
        q6valor.Text = "-"
```

```
        z = 6
    ElseIf h >= 8 And h <= (97 / 11) Then
        q1valor.Text = "-"
        q2valor.Text = "-"
        q3valor.Text = "-"
        q4valor.Text = Round(QmodoD, 5)
        q5valor.Text = Round(QmodoE, 5)
        q6valor.Text = Round(QmodoF, 5)
        z = 7
    ElseIf h > (97 / 11) And h <= 16 Then
        q1valor.Text = "-"
        q2valor.Text = "-"
        q3valor.Text = "-"
        q4valor.Text = "-"
        q5valor.Text = Round(QmodoE, 5)
        q6valor.Text = Round(QmodoF, 5)
        z = 8
    End If
    ' Dibujamos las líneas de la temperatura.
    lineatmin.X1 = 1000
    lineatmin.X2 = 1000 + CInt(lbt * 500)
    lineatmin.Y1 = 2000 - CInt(yt(0) * 1000)
    lineatmin.Y2 = 2000 - CInt(yt(0) * 1000)
    lineatmin.Visible = True
    cirtmin1.Left = 950
    cirtmin1.Top = 1950 - CInt(yt(0) * 1000)
    cirtmin1.Visible = True
    cirtmin2.Left = 950 + CInt(lbt * 500)
    cirtmin2.Top = 1950 - CInt(yt(0) * 1000)
    cirtmin2.Visible = True
```

```
lineatmax.X1 = 1000
lineatmax.X2 = 1000 + CInt(lbt * 500)
lineatmax.Y1 = 2000 - CInt(yt(1) * 1000)
lineatmax.Y2 = 2000 - CInt(yt(1) * 1000)
lineatmax.Visible = True
cirtmax1.Left = 950
cirtmax1.Top = 1950 - CInt(yt(1) * 1000)
cirtmax1.Visible = True
cirtmax2.Left = 950 + CInt(lbt * 500)
cirtmax2.Top = 1950 - CInt(yt(1) * 1000)
cirtmax2.Visible = True
lineat.X1 = 1000 + CInt(lbt * 500)
lineat.X2 = 1000 + CInt(lbt * 500)
lineat.Y1 = 2000
lineat.Y2 = 2000 - CInt(yt(1) * 1000)
lineat.Visible = True
cirt.Left = 950 + CInt(lbt * 500)
cirt.Top = 1950
cirt.Visible = True
' Dibujamos las líneas del nivel de amonio.
lineanamin.X1 = 1000
lineanamin.X2 = 1000 + CInt(lbna * 500)
lineanamin.Y1 = 2000 - CInt(yna(0) * 1000)
lineanamin.Y2 = 2000 - CInt(yna(0) * 1000)
lineanamin.Visible = True
cirnamin1.Left = 950
cirnamin1.Top = 1950 - CInt(yna(0) * 1000)
cirnamin1.Visible = True
cirnamin2.Left = 950 + CInt(lbna * 500)
cirnamin2.Top = 1950 - CInt(yna(0) * 1000)
```

```
    cirnamin2.Visible = True
    lineanamax.X1 = 1000
    lineanamax.X2 = 1000 + CInt(lbna * 500)
    lineanamax.Y1 = 2000 - CInt(yna(1) * 1000)
    lineanamax.Y2 = 2000 - CInt(yna(1) * 1000)
    lineanamax.Visible = True
    cirnamax1.Left = 950
    cirnamax1.Top = 1950 - CInt(yna(1) * 1000)
    cirnamax1.Visible = True
    cirnamax2.Left = 950 + CInt(lbna * 500)
    cirnamax2.Top = 1950 - CInt(yna(1) * 1000)
    cirnamax2.Visible = True
    lineana.X1 = 1000 + CInt(lbna * 500)
    lineana.X2 = 1000 + CInt(lbna * 500)
    lineana.Y1 = 2000
    lineana.Y2 = 2000 - CInt(yna(1) * 1000)
    lineana.Visible = True
    cirna.Left = 950 + CInt(lbna * 500)
    cirna.Top = 1950
    cirna.Visible = True
    ' Dibujamos las líneas de la altura de la compuerta.
    For m = 0 To 3
        lbam(m) = (6 * (lba(m) - altxmin.Caption)) / (altxmax.Caption
- altxmin.Caption)
    Next m
    Alturam = (6 * (Altura - altxmin.Caption)) / (altxmax.Caption -
altxmin.Caption)
    lineaa1.X1 = 1000
    lineaa1.X2 = 1000 + CInt(lbam(0) * 500)
    lineaa1.Y1 = 2000 - CInt(ya(0) * 1000)
    lineaa1.Y2 = 2000 - CInt(ya(0) * 1000)
```

```
lineaa1.Visible = True
cira11.Left = 950
cira11.Top = 1950 - CInt(ya(0) * 1000)
cira11.Visible = True
cira12.Left = 950 + CInt(lbam(0) * 500)
cira12.Top = 1950 - CInt(ya(0) * 1000)
cira12.Visible = True
lineaa2.X1 = 1000
lineaa2.X2 = 1000 + CInt(lbam(1) * 500)
lineaa2.Y1 = 2000 - CInt(ya(1) * 1000)
lineaa2.Y2 = 2000 - CInt(ya(1) * 1000)
lineaa2.Visible = True
cira21.Left = 950
cira21.Top = 1950 - CInt(ya(1) * 1000)
cira21.Visible = True
cira22.Left = 950 + CInt(lbam(1) * 500)
cira22.Top = 1950 - CInt(ya(1) * 1000)
cira22.Visible = True
lineaa3.X1 = 1000
lineaa3.X2 = 1000 + CInt(lbam(2) * 500)
lineaa3.Y1 = 2000 - CInt(ya(2) * 1000)
lineaa3.Y2 = 2000 - CInt(ya(2) * 1000)
lineaa3.Visible = True
cira31.Left = 950
cira31.Top = 1950 - CInt(ya(2) * 1000)
cira31.Visible = True
cira32.Left = 950 + CInt(lbam(2) * 500)
cira32.Top = 1950 - CInt(ya(2) * 1000)
cira32.Visible = True
lineaa4.X1 = 1000
```

```
lineaa4.X2 = 1000 + CInt(lbam(3) * 500)
lineaa4.Y1 = 2000 - CInt(ya(3) * 1000)
lineaa4.Y2 = 2000 - CInt(ya(3) * 1000)
lineaa4.Visible = True
cira41.Left = 950
cira41.Top = 1950 - CInt(ya(3) * 1000)
cira41.Visible = True
cira42.Left = 950 + CInt(lbam(3) * 500)
cira42.Top = 1950 - CInt(ya(3) * 1000)
cira42.Visible = True
lineaa.X1 = 1000 + CInt(Alturam * 500)
lineaa.X2 = 1000 + CInt(Alturam * 500)
lineaa.Y1 = 1900
lineaa.Y2 = 2100
lineaa.Visible = True
cira.Left = 950 + CInt(Alturam * 500)
cira.Top = 1950
cira.Visible = True
If k = 1 Then
    lineaa2.Visible = False
    cira21.Visible = False
    cira22.Visible = False
    lineaa3.Visible = False
    cira31.Visible = False
    cira32.Visible = False
    lineaa4.Visible = False
    cira41.Visible = False
    cira42.Visible = False
ElseIf k = 2 Then
    lineaa3.Visible = False
```

```
        cira31.Visible = False
        cira32.Visible = False
        lineaa4.Visible = False
        cira41.Visible = False
        cira42.Visible = False
    ElseIf k = 3 Then
        lineaa2.Visible = False
        cira21.Visible = False
        cira22.Visible = False
        lineaa4.Visible = False
        cira41.Visible = False
        cira42.Visible = False
    End If
    ' Dibujamos las compuertas y caudales.
    nivelagua1.Visible = False
    nivelagua2.Visible = False
    lineaqa.Visible = False
    lineaqb.Visible = False
    lineaqc.Visible = False
    lineaqd.Visible = False
    lineaqe.Visible = False
    lineaqf.Visible = False
    escala = 2 / (altxmax.Caption - altxmin.Caption)
    Compuerta.Top = 3500 - CInt((Altura - altxmin.Caption) * 500 *
escala)
    Agua1.Height = CInt((ay1 - altxmin.Caption) * 500 * escala)
    Agua1.Top = 4500 - CInt((ay1 - altxmin.Caption) * 500 * escala)
    lineaay1.Y1 = 4500 - CInt((ay1 - altxmin.Caption) * 500 * escala)
    lineaay1.Y2 = 4500 - CInt((ay1 - altxmin.Caption) * 500 * escala)
    lineaay1.Visible = True
    y3A = (Altura - altxmin.Caption) * escala
```

```
y3B = 2 * (Altura - altxmin.Caption) * escala
y3C = 3 * (Altura - altxmin.Caption) * escala
y3D = 4 * (Altura - altxmin.Caption) * escala
y3E = 6 * (Altura - altxmin.Caption) * escala
y3F = 8 * (Altura - altxmin.Caption) * escala
If z = 1 Then
    lineaqa.Y1 = 4500 - CInt(y3A * 500)
    lineaqa.Y2 = 4500 - CInt(y3A * 500)
    lineaqa.Visible = True
    Agua2.Height = CInt(y3A * 500)
    Agua2.Top = 4500 - CInt(y3A * 500)
ElseIf z = 2 Then
    lineaqb.Y1 = 4500 - CInt(y3B * 500)
    lineaqb.Y2 = 4500 - CInt(y3B * 500)
    lineaqb.Visible = True
    Agua2.Height = CInt(y3B * 500)
    Agua2.Top = 4500 - CInt(y3B * 500)
ElseIf z = 3 Then
    lineaqc.Y1 = 4500 - CInt(y3C * 500)
    lineaqc.Y2 = 4500 - CInt(y3C * 500)
    lineaqc.Visible = True
    Agua2.Height = CInt(y3C * 500)
    Agua2.Top = 4500 - CInt(y3C * 500)
ElseIf z = 4 Then
    lineaqc.Y1 = 4500 - CInt(y3C * 500)
    lineaqc.Y2 = 4500 - CInt(y3C * 500)
    lineaqd.Y1 = 4500 - CInt(y3D * 500)
    lineaqd.Y2 = 4500 - CInt(y3D * 500)
    lineaqc.Visible = True
    lineaqd.Visible = True
```

```
        Agua2.Height = CInt(y3D * 500)
        Agua2.Top = 4500 - CInt(y3D * 500)
    ElseIf z = 5 Then
        lineaqd.Y1 = 4500 - CInt(y3D * 500)
        lineaqd.Y2 = 4500 - CInt(y3D * 500)
        lineaqd.Visible = True
        Agua2.Height = CInt(y3D * 500)
        Agua2.Top = 4500 - CInt(y3D * 500)
    ElseIf z = 6 Then
        lineaqd.Y1 = 4500 - CInt(y3D * 500)
        lineaqd.Y2 = 4500 - CInt(y3D * 500)
        lineaqe.Y1 = 4500 - CInt(y3E * 500)
        lineaqe.Y2 = 4500 - CInt(y3E * 500)
        lineaqd.Visible = True
        lineaqe.Visible = True
        Agua2.Height = CInt(y3E * 500)
        Agua2.Top = 4500 - CInt(y3E * 500)
    ElseIf z = 7 Then
        lineaqd.Y1 = 4500 - CInt(y3D * 500)
        lineaqd.Y2 = 4500 - CInt(y3D * 500)
        lineaqe.Y1 = 4500 - CInt(y3E * 500)
        lineaqe.Y2 = 4500 - CInt(y3E * 500)
        lineaqf.Y1 = 4500 - CInt(y3F * 500)
        lineaqf.Y2 = 4500 - CInt(y3F * 500)
        lineaqd.Visible = True
        lineaqe.Visible = True
        lineaqf.Visible = True
        Agua2.Height = CInt(y3F * 500)
        Agua2.Top = 4500 - CInt(y3F * 500)
    ElseIf z = 8 Then
```

```
        lineaqe.Y1 = 4500 - CInt(y3E * 500)
        lineaqe.Y2 = 4500 - CInt(y3E * 500)
        lineaqf.Y1 = 4500 - CInt(y3F * 500)
        lineaqf.Y2 = 4500 - CInt(y3F * 500)
        lineaqe.Visible = True
        lineaqf.Visible = True
        Agua2.Height = CInt(y3F * 500)
        Agua2.Top = 4500 - CInt(y3F * 500)
    End If
    Agua1.ZOrder vbSendToBack
    Agua2.ZOrder vbSendToBack
    lineaay1.ZOrder vbBringToFront
    lineaqa.ZOrder vbBringToFront
    lineaqb.ZOrder vbBringToFront
    lineaqc.ZOrder vbBringToFront
    lineaqd.ZOrder vbBringToFront
    lineaqe.ZOrder vbBringToFront
    lineaqf.ZOrder vbBringToFront
    Muro.ZOrder vbBringToFront
    Compuerta.ZOrder vbBringToFront
End If
End Sub
' Cambiar la memoria asociativa borrosa.
Private Sub CambiarFAM_Click()
FAM.Show
End Sub
' Cambiar los valores extremos de las variables.
Private Sub CambiarVariables_Click()
Variables.Show
End Sub
```

```
' Barra de desplazamiento vertical.

Private Sub DesplazamientoV_Change()

Formulario.Top = 1000 - DesplazamientoV.Value

End Sub

Private Sub DesplazamientoV_Scroll()

Formulario.Top = 1000 - DesplazamientoV.Value

End Sub

' Barra de desplazamiento horizontal.

Private Sub DesplazamientoH_Change()

Formulario.Left = 1000 - DesplazamientoH.Value

End Sub

Private Sub DesplazamientoH_Scroll()

Formulario.Left = 1000 - DesplazamientoH.Value

End Sub

Private Sub Form_Load()

' Centrar el formulario y establecer los valores mínimo y máximo de
las barras de desplazamiento vertical y horizontal.

If Formulario.Height <= (Screen.Height - DesplazamientoH.Height -
2750) Then

    Formulario.Top = (Screen.Height - Formulario.Height -
DesplazamientoH.Height - 750) / 2

    DesplazamientoV.Enabled = False

Else

    Formulario.Top = 1000

    DesplazamientoV.Min = 0

    DesplazamientoV.Max = 2750 + Formulario.Height +
DesplazamientoH.Height - Screen.Height

End If

If Formulario.Width <= (Screen.Width - DesplazamientoV.Width - 2000)
Then

    Formulario.Left = (Screen.Width - Formulario.Width -
DesplazamientoV.Width) / 2

    DesplazamientoH.Enabled = False
```

```
Else

    Formulario.Left = 1000

    DesplazamientoH.Min = 0

    DesplazamientoH.Max = 2000 + Formulario.Width + DesplazamientoV.Width - Screen.Width

End If

' Colocar la barra de desplazamiento vertical.

DesplazamientoV.Height = Screen.Height - DesplazamientoH.Height - 750

DesplazamientoV.Top = 0

DesplazamientoV.Left = Screen.Width - DesplazamientoV.Width

' Colocar la barra de desplazamiento horizontal.

DesplazamientoH.Width = Screen.Width - DesplazamientoV.Width

DesplazamientoH.Top = Screen.Height - DesplazamientoH.Height - 750

DesplazamientoH.Left = 0

End Sub
```

b) Código fuente del formulario “FAM.frm”

```
Private Sub Aceptar_Click()

Dim j As Integer

Dim w As Integer

j = 0

' Comprobamos si se han rellenado correctamente todas las opciones.

For w = 0 To 24

    If Altura(w).ListIndex <> 0 And Altura(w).ListIndex <> 1 And Altura(w).ListIndex <> 2 And Altura(w).ListIndex <> 3 And Altura(w).ListIndex <> 4 Then

        j = 1

    End If

Next w

' Si se han rellenado correctamente todas las opciones, copiamos en la ventana principal la memoria asociativa borrosa introducida.
```

```
If j = 0 Then
    For w = 0 To 24
        If Altura(w).ListIndex = 0 Then
            Programa.A(w).Caption = "MB"
        ElseIf Altura(w).ListIndex = 1 Then
            Programa.A(w).Caption = "B"
        ElseIf Altura(w).ListIndex = 2 Then
            Programa.A(w).Caption = "M"
        ElseIf Altura(w).ListIndex = 3 Then
            Programa.A(w).Caption = "A"
        ElseIf Altura(w).ListIndex = 4 Then
            Programa.A(w).Caption = "MA"
        End If
    Next w
    FAM.Hide
    Unload FAM
Else
    MsgBox "Error. No ha especificado todas las opciones."
End If
End Sub
Private Sub Cancelar_Click()
FAM.Hide
Unload FAM
End Sub
```

c) Código fuente del formulario “Variables.frm”

```
Private Sub AlturaCompuerta_Click()
CA.Show
End Sub
```

```
Private Sub Cancelar_Click()

Variables.Hide

Unload Variables

End Sub

Private Sub NivelAmonio_Click()

CNA.Show

End Sub

Private Sub Temperatura_Click()

CT.Show

End Sub
```

d) Código fuente del formulario “CT.frm”

```
Private Sub Aceptar_Click()

Dim j As Integer

j = 0

' Comprobación de los valores introducidos. Si son correctos, los
copiamos en la ventana principal.

If IsNumeric(Tmin.Text) Then

    j = j + 1

Else

    MsgBox "Error. El valor mínimo introducido no es válido."

End If

If IsNumeric(Tmax.Text) Then

    j = j + 1

Else

    MsgBox "Error. El valor máximo introducido no es válido."

End If

If j = 2 Then

    If Tmin.Text < Tmax.Text Then
```

```
        Programa.tempxmin.Caption = Tmin.Text

        Programa.tempxmax.Caption = Tmax.Text

        CT.Hide

        Unload CT

    Else

        MsgBox "Error. El valor mínimo introducido no puede ser mayor
o igual que el valor máximo introducido."

    End If

End If

End Sub

Private Sub Cancelar_Click()

CT.Hide

Unload CT

End Sub
```

e) Código fuente del formulario “CNA.frm”

```
Private Sub Aceptar_Click()

Dim j As Integer

j = 0

' Comprobación de los valores introducidos. Si son correctos, los
copiamos en la ventana principal.

If IsNumeric(NAmin.Text) Then

    j = j + 1

Else

    MsgBox "Error. El valor mínimo introducido no es válido."

End If

If IsNumeric(NAmax.Text) Then

    j = j + 1

Else

    MsgBox "Error. El valor máximo introducido no es válido."
```

```
End If

If j = 2 Then

    If NAmin.Text < NAmax.Text Then

        Programa.amonxmin.Caption = NAmin.Text

        Programa.amonxmax.Caption = NAmax.Text

        CNA.Hide

        Unload CNA

    Else

        MsgBox "Error. El valor mínimo introducido no puede ser mayor
o igual que el valor máximo introducido."

    End If

End If

End Sub

Private Sub Cancelar_Click()

CNA.Hide

Unload CNA

End Sub
```

f) Código fuente del formulario “CA.frm”

```
Private Sub Aceptar_Click()

Dim j As Integer

j = 0

' Comprobación de los valores introducidos. Si son correctos, los
copiamos en la ventana principal.

If IsNumeric(Amin.Text) Then

    j = j + 1

Else

    MsgBox "Error. El valor mínimo introducido no es válido."

End If

If IsNumeric(Amax.Text) Then
```

```
    j = j + 1
Else
    MsgBox "Error. El valor máximo introducido no es válido."
End If
If j = 2 Then
    If Amin.Text < Amax.Text Then
        Programa.altxmin.Caption = Amin.Text
        Programa.altxmax.Caption = Amax.Text
        CA.Hide
        Unload CA
    Else
        MsgBox "Error. El valor mínimo introducido no puede ser mayor
o igual que el valor máximo introducido."
    End If
End If
End Sub
Private Sub Cancelar_Click()
CA.Hide
Unload CA
End Sub
```