E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Diseño y entrenamiento con Matlab de un controlador neuroborroso como parte de un sistema de ayuda en la toma de decisiones para el uso de bombas de insulina

Grado en Ingeniería Eléctrica y Electrónica

Trabajo Fin de Grado

Mikel Marquina Ausejo

María José Pérez-Ilzarbe

Pamplona, 28-1-2016

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Índice

Agradecimientos

Mediante estas líneas me gustaría agradecer a todas las personas que me han apoyado y ayudado durante estos años de carrera, así como a la realización del Trabajo Fin de Grado.

En primer lugar, quisiera agradecer a mi tutora, María José Pérez-Ilzarbe, toda la ayuda prestada y todo el tiempo que me ha dedicado para la realización de este proyecto. También quisiera agradecer a José Basilio Galván Herrera todos los consejos que me ha dado.

Recordar también a todos mis compañeros que me han ayudado durante todos estos años de carrera y por supuesto, a mis amigos de toda la vida.

Y por último agradecer a toda mi familia todo el apoyo que me ha dado.

Resumen y palabras clave

Resumen

En un proyecto fin de carrera realizado previamente en el área de ingeniería de sistemas y automática, se desarrolló un sistema de ayuda en la toma de decisiones para el uso de bombas de insulina. Una parte significativa de dicho sistema está constituido por un controlador borroso. El ajuste de dicho controlador se realizó en el mencionado PFC, con un algoritmo inventado *ad hoc.*

En este trabajo fin de grado se van a estudiar las posibilidades y limitaciones del uso del entrenador de sistemas neuroborrosos de Matlab para realizar el diseño y ajuste de dicho controlador borroso. El objetivo final de la aplicación es que el sistema de decisión sea capaz de adaptarse. Así, partiendo del ajuste realizado para un usuario en determinadas circunstancias, se tratara de que el sistema "aprenda" a adaptarse a nuevas necesidades de este usuario o a las necesidades de otro usuario, a partir de datos que se le suministran de salidas deseadas para entradas dadas.

Abstract

In a previous final degree project made in the area of engineering and automatic systems, a support system was developed to help in the decision making for the use of insulin pumps. A significant part of this system is made up by a fuzzy controller. In said FDP, the adjustment of the controller was performed with an algorithm invented *ad hoc*.

This final project studies the possibilities and limitations of using neurofuzzy Matlab systems for the design and adjustment of the fuzzy controller. The ultimate aim of the application is that the decision system is able to adapt itself. Thus, based on the settings for a user in specific circumstances, the system "learns" to adapt to new requirements of that same user or to requirements of other users, based on data provided by the desired outputs to given inputs.

Laburpena

Sistema eta automatika ingeniaritza arloan aurretik egindako karrera amaierako lan batean, intsulina bonbak erabiltzerakoan erabakiak hartzeko laguntza sistema bat garatu zuten. Sistema horretako zati garrantzitsu bat kontrolagailu lausoa zen. Karrera amaierako lan horretako kontrolatzailearen doikuntza *ad hoc* egin zen.

Gradu amaierako lan honetan, Matlab bidezko sistema neurolausoen entrenamenduaren aukerak eta mugak aztertuko dira, kontrolatzaile horren diseinu eta doikuntza egiteko. Erabakiak hartzeko sistemak moldatzeko ahalmena izatea da aplikazio honen helburua. Erabiltzaile jakin bati doitutako sistema bat, bat-bateko egoerak oinarritzat hartuz, sistemak beste baldintzetara edota beste pertsona bati moldatzen ikasteko ahalmena izan behar du, sarrera zehatz batzuei irteerako datuak emanez.

Lista de palabras clave

Matlab

Controlador Fuzzy

Neuroborroso

Diabetes

ANFIS

Bombas de insulina

Objetivos

Este proyecto nace de la necesidad de crear un sistema automático para controlar el nivel de glucosa para personas con diabetes tipo 1, sin que sea necesario ser un usuario experto, de modo que un usuario sin experiencia y con ayuda del médico, sea capaz de suministrársela.

Por ello el sistema, desde un ajuste inicial, tiene que ser capaz de amoldarse a las necesidades de un nuevo usuario. Este ajuste inicial corresponde a un usuario experto el cual propone unos valores para unas entradas. Puede no ser válido para todos los usuarios, ya sea por las características del usuario (edad, biología etc.) o porque con el tiempo, en el caso de adolescentes, las necesidades del usuario respecto a la insulina cambian. Para ello, el usuario deberá ir metiendo nuevos datos cuando note que la respuesta de la salida de la válvula no se adecua a sus nuevas necesidades.

Para ello, se usará el programa MATLAB por su capacidad para trabajar con sistemas neuroborrosos, por lo que un requisito esencial es familiarizarse con dichos sistemas y aprender su programación.

Por último, se desea analizar las características, limitaciones, errores etc. del programa MATLAB para su uso en este proyecto.

1.- Introducción

1.1.- La diabetes

Según la International Diabetes Federation, “La diabetes es una afección crónica que se desencadena cuando el organismo pierde su capacidad de producir suficiente insulina o de utilizarla con eficacia” (1).

Cuando ingerimos alimentos, estos se transforman en glucosa. La glucosa es la que proporciona la energía necesaria para que las células realicen sus funciones. Pero para que la glucosa penetre en las células, se necesita traspasar la pared celular o barreras celulares y en ello es necesario la intervención de la insulina.

La glucosa que discurre por el torrente sanguíneo, estimula las células Beta, que son las encargadas de liberar insulina. Estas células se encuentran en el extremo del páncreas.

La insulina es la que se ocupa de abrir el camino por la pared celular para que la glucosa penetre en las células.

En caso de que no haya insulina o los receptores encargados del paso de la glucosa no funcionen, la persona sufrirá de carencia de nutrientes además de hiperglucemia, debido a que circula un exceso de glucosa por la sangre, lo cual causa daños en los tejidos. Con el paso del tiempo esto puede causar complicaciones potencialmente letales.

Existen tres tipos de diabetes:

- Diabetes Mellitus Insulinodependiente DM1 (tipo 1):

  Afecta a personas de cualquier edad pero lo más habitual es que aparezca en niños o jóvenes adultos y suele surgir de forma repentina.

  Es debido a una reacción autoinmune en el que las defensas del organismo atacan las células Beta y estas dejan de producir la insulina.

  Los que sufren esta diabetes se tienen que suministrar insulina mediante inyección o bombas de insulina para poder tener controlado en nivel de glucosa en sangre.

- Diabetes Mellitus No Insulinodependiente DM2 (tipo 2):

  Suele desarrollarse en edades más avanzadas que la tipo 1 aunque también puede aparecer en niños y adolescentes. Es la más común y no se conocen las razones por las que aparece, pero existen factores de riesgo como la obesidad, mala alimentación, edad avanzada, mala nutrición durante el embarazo, lo cual puede provocar la aparición de diabetes en el niño, antecedentes familiares etc.

  En este caso, el cuerpo produce insulina pero puede que no sea suficiente o que el organismo no reaccione a sus efectos.

  Los diabéticos tipo 2 no necesitan suministro diario de insulina.

- Diabetes Mellitus Gestacional (DMG):

  La DMG es un tipo de diabetes que puede surgir en una etapa avanzada del embarazo. Es debido a que el organismo no genera o no utiliza la insulina necesaria requerida durante la gestación.

  Este tipo de diabetes desaparece después del embarazo, aunque puede ocurrir que después de un tiempo se desarrolle diabetes de tipo 2.

En los casos de bebés que la madre haya tenido DMG, son más propensos a la obesidad y existe la posibilidad de que desarrollen diabetes de tipo 2 en edad adulta.

1.2.- Control de glucosa en sangre

La glucemia es la concentración de glucosa libre en sangre y se mide en miligramos por decilitro (mg/dl). Las personas que no tienen diabetes, la glucosa sanguínea preprandial está en un rango de 70-130 mg/dl. Después de comer, la glucosa en sangre aumenta de proporción y lo normal es que a las 2 horas, no pase de 180mg/dl.

Se considera valores normales a una concentración en sangre de 100 mg/dl en ayunas y se le llama normoglucemia. A valores por encima de 180 mg/dl se le llama hiperglucemia y es debido a que no se tiene suficiente insulina en el cuerpo, mientras que valores inferiores a 70 mg/dl es debido a que la glucosa se agota demasiado rápido, porque la glucosa se libera con demasiada lentitud o porque se libera demasiada insulina y se denomina hipoglucemia.

Las personas con el páncreas sano son capaces de mantener los niveles de glucosa en niveles óptimos mediante el control interno del organismo, produciendo mayor o menor cantidad de insulina durante y después de la ingesta de alimentos.

Como se ha mencionado anteriormente, los diabéticos tipo uno se deben tratar mediante administración de insulina exógena. Existen dos métodos para la administración de insulina; mediante inyección o mediante bomba de insulina.

En el suministro de dosis múltiples vía inyección se deben usar dos tipos de insulinas; una de acción rápida y otra prolongada en el tiempo. Para periodos interprandriales, es decir, entre comidas, se debe asegurar un nivel mínimo de insulina mediante el uso de insulinas lentas, mientras que para las comidas se debe suministrar de acción rápida. Adicionalmente se tiene que llevar control de la alimentación, especialmente de los carbohidratos ingeridos ya que si son muchos se producen hiperglucemias y si son pocos surge hipoglucemia. Además se debe tener en cuenta factores que alteran la velocidad de la digestión como pueden ser la actividad física, grasas ingeridas etc.

El uso de las bombas de insulina, además de su reducido tamaño, es lo que más se asemeja al control fisiológico que realiza el cuerpo, dando por ello unos buenos resultados. Este sistema administra continuamente, antes, durante la ingesta y mientras la digestión, hormonas a través del catéter.

El objetivo final para el suministro de insulina mediante bombas de insulina sería el control de las variables mediante el uso de sensores, con lo cual se podría administrar la insulina automáticamente a lo largo del día. A esto se le llama páncreas artificial ya que hace la función del páncreas. Desgraciadamente existen muchas dificultades que solventar, como por ejemplo; la medición precisa de glucosas en sangre, el control de las obstrucciones y perdidas en el catéter, el tiempo entre la administración subcutánea y la llegada a sangre, diferente en cada momento del día y de las persona etc. Debido a todos estos problemas, es necesaria la interacción entre la bomba de insulina y el propio usuario.

En estado preprandial, la bomba debe suministrar una dosis continua basal y durante y después de la ingesta, la glucosa aumenta, por lo tanto se requiere más insulina. La insulina a suministrar se divide en dos; una parte se suministra de forma continuada (Bolus C) durante un tiempo y la otra parte se suministra en una sola dosis (Bolus N).

La decisión de la cantidad de unidades de insulina a administrar, el reparto entre Bolus C y Bolus N y el tiempo de suministro de Bolus C, depende de muchos factores (relativos a la cantidad/carbohidratos, a las grasas, velocidad de digestión, franja horaria etc.), por lo que puede ser una tarea complicada.

Por ello, muchos usuarios optan únicamente por suministrarse insulinas de efecto rápido (Bolus N) y ver cómo evoluciona y después corregir si es necesario. Lo que ocurre es que se dan retrasos desde la administración subcutánea y el momento en el que se alcanza la eficiencia máxima en la metabolización de los carbohidratos, por lo que es muy difícil controlarlo mediante este método y el riesgo de hipo o hiperglucemia aumenta.

La decisión de la forma de suministro de la insulina está atada a muchos factores, lo que hace que los usuarios deban ser expertos. Para facilitar el uso de las bombas de insulina sin ser un usuario experimentado, se va a realizar un control llamado borroso, el cual, con unos datos pedidos por consola, sea capaz de tomar las decisiones y adecuarlas a las necesidades del usuario..

1.3.- Lógica borrosa

La lógica borrosa o difusa es el tratamiento de información poco precisa o abstracta, en la cual, la precisión no es esencial. Se asemeja a la metodología que usa el ser humano habitualmente para la toma de decisiones. Un ejemplo puede ser la temperatura; es una sensación y se podría clasificar como caliente templado o frio. Otro ejemplo puede ser la velocidad, la altura etc.

La lógica binaria trata los datos como 1 o 0, lo cual se puede interpretar como un SI absoluto o un NO absoluto. Un ejemplo puede ser el mencionado anteriormente, la temperatura. Si el valor de entrada o dato está por debajo de un valor definido, la lógica binaria dará un 0, mientras que si está por encima, dará un 1 o viceversa.

En la lógica borrosa, existen grados intermedios. Por ejemplo, en el caso de la temperatura se podrían definir tres categorías; caliente, templado y frio. A cada categoría se le asignaría un **Conjunto Borroso.** Si se define que 20 grados es templado y que 16 ya pasa a ser frio, una temperatura intermedia como pudieran ser 17 grados, no es ni templada ni fría, por lo que pertenece a los dos conjuntos borrosos (templado y frio).

Los conjuntos borrosos se caracterizan matemáticamente mediante **funciones de pertenencia**. Los conjuntos borrosos están definidos en un universo, siendo en este caso la temperatura. Las funciones de pertenencia de los conjuntos borrosos están acotadas entre 0 y 1, 1 es un 100% de pertenencia (20 grados para el conjunto templado) y 0 un 0% de pertenencia (10 grados para el conjunto templado).

A continuación se muestra un ejemplo de funciones de pertenencia para el universo temperatura (se denominará T).

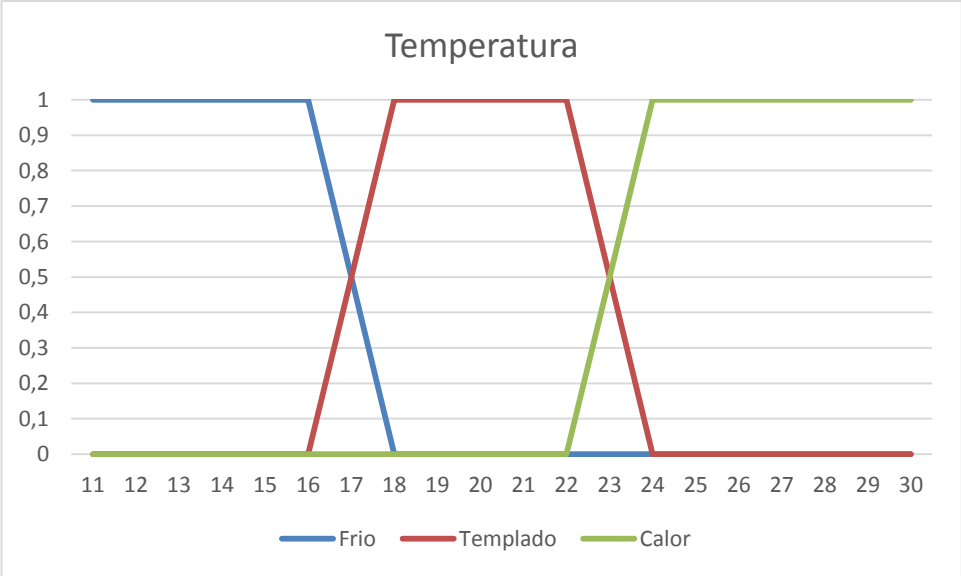


Figura 1.1; Funciones de pertenencia del universo Temperatura

Como se observa en la imagen, 16 grados pertenece exclusivamente al conjunto frio ($T_{(16)}$=1), con un grado de pertenencia de 1, mientras que 17 grados tiene un grado de pertenencia de 0.5 al conjunto frio ($T_{(17)}$=0.5), 0.5 al de templado ($T_{(17)}$=0.5) y un 0% al conjunto calor. Como se diría coloquialmente, no hace frio ni calor. 20 grados tiene un grado de pertenencia de 1 a la función de pertenencia templada y a partir de los 24 grados en adelante, el grado de pertenencia al conjunto calor es de 1.

Las funciones de pertenencia pueden ser de diferentes tipos. Las más comunes son las triangulares y trapezoidales (caso anterior). Existen más tipos de funciones de pertenencia como las Gamma, Gaussianas, Sigmoidal y Pseudo-exponencial. La elección del tipo y el número de funciones de pertenencia queda a elección del diseñador. Cuantas más funciones de pertenencia se definan, más compleja puede ser la relación entradas-salidas que se aprenda. No tienen por qué ser todas las funciones simétricas e iguales entre sí, además que en un punto, la suma no tienen por qué ser 1.

El modo de trabajar de los sistemas borrosos es igual que lo haría una persona. Mediante un sistema simple de reglas, el sistema saca una salida la cual es el resultado de dichas reglas. Siguiendo con el ejemplo de la temperatura, las reglas podrían ser:

- SI hace frio, ENTONCES pon el calentador a más temperatura.
- SI se está templado, ENTONCES mantén la temperatura.
- SI hace calor, ENTONCES baja la temperatura del calentador.

En este ejemplo únicamente se ha empleado una entrada, la temperatura, pero pueden ser varias las entradas a un sistema borroso, como por ejemplo la humedad o cualquier otra entrada que tenga que ser tomada en cuenta para obtener la salida deseada.

Las entradas al sistema borroso pueden ser borrosas o numéricas. Las entradas borrosas son las que no son precisas como por ejemplo, decir que hace frio, no especifica la temperatura exacta. Decir que hacen 17° es una entrada precisa o numérica. El controlador borroso puede dar salidas tanto numéricas como borrosas, independientemente de qué tipo sea la entrada. En este proyecto sólo se va a trabajar con entradas numéricas. Por otro lado, el

sistema neuroborroso de Matlab que se va a analizar en este proyecto sólo admite ese tipo de entradas.

Las principales operaciones lógicas que se necesita definir para los conjuntos borrosos son el Y lógico, el O lógico y el NO lógico.

Las operaciones lógicas se pueden definir de varias maneras pero solo se explicarán las más utilizadas:

Primer modo:

El O lógico, que equivale matemáticamente a la unión de dos conjuntos, se calcula como el máximo donde coinciden ambas.

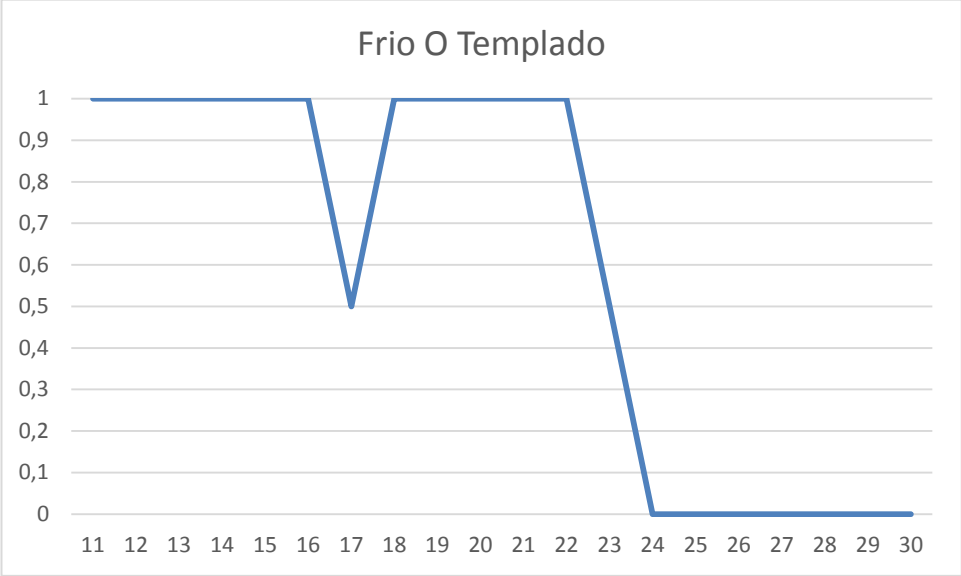


Figura 1.2

El Y lógico, que equivale matemáticamente a la intersección de dos conjuntos, se calcula como el mínimo donde coinciden ambas.

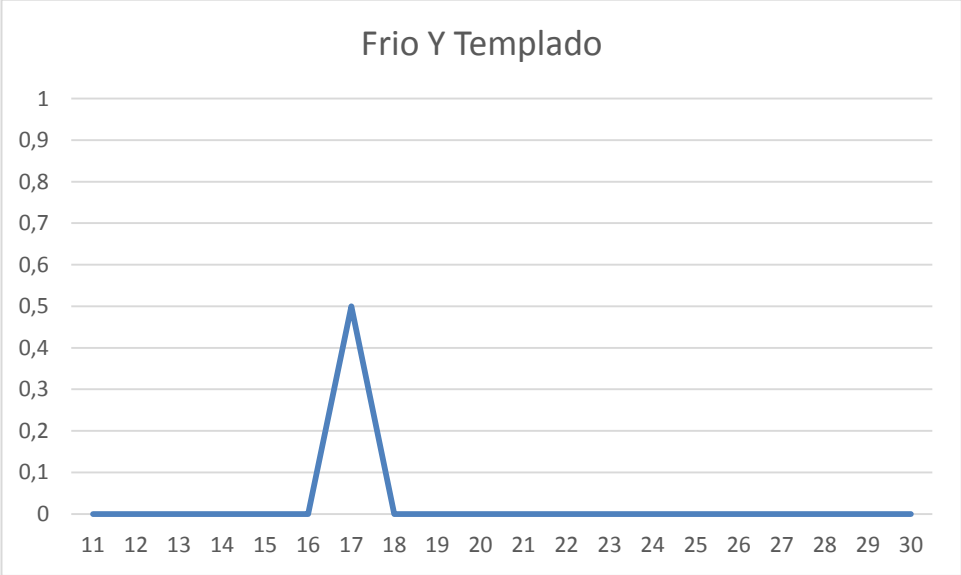


Figura 1.3

Por último, el NO lógico equivale matemáticamente al complementario del conjunto.

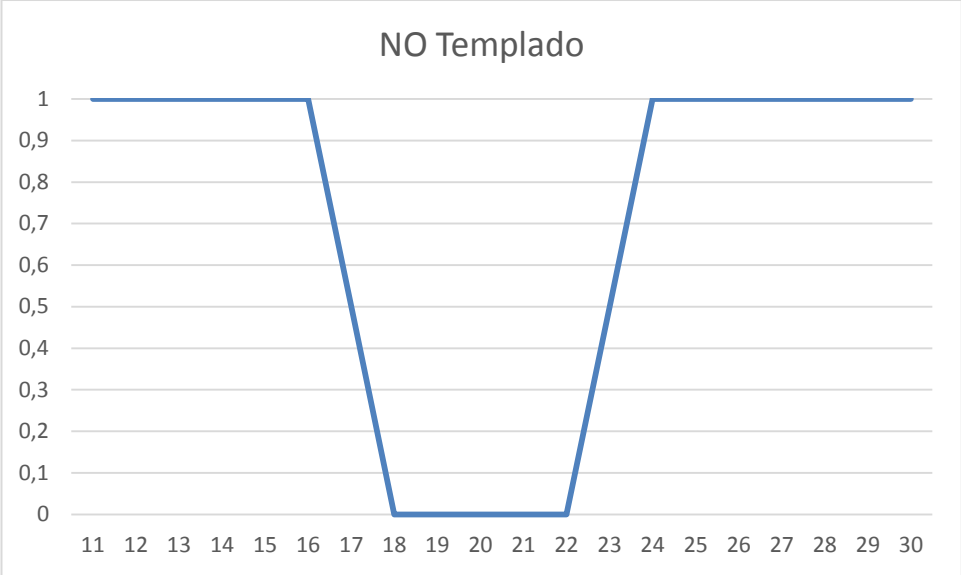


Figura 1.4

Segundo modo:

Para el Y lógico o intersección se usa el producto de las funciones.

$(A \cap B)(x) = A(x)B(x)$

Para el O lógico o unión se usa la suma probabilista.

$(A \cup B)(x) = A(x) + B(x) - A(x)B(x)$

Como se puede ver, mediante el primer modo no existe interacción entre los dos conjuntos para el resultado ya que para la unión solo depende del máximo en el punto al igual que en la intersección que solo depende del mínimo.

Sin embargo, mediante el segundo método el resultado sí que existe interacción entre los dos conjuntos o funciones para el resultado.

1.4.- Controladores Borrosos

Los controladores que se van a estudiar son los llamados **Mamdani** y **Sugeno** con entrada numérica y salida numérica.

El sistema está compuesto por tres grandes bloque; el **emborronamiento**, el **bloque de inferencia borrosa** y el **bloque de desemborronamiento**.

Mamdani

En el sistema borroso de Mamdani, las entradas van a ser numéricas, los antecedentes de las reglas son borrosas y los consecuentes son borrosos.

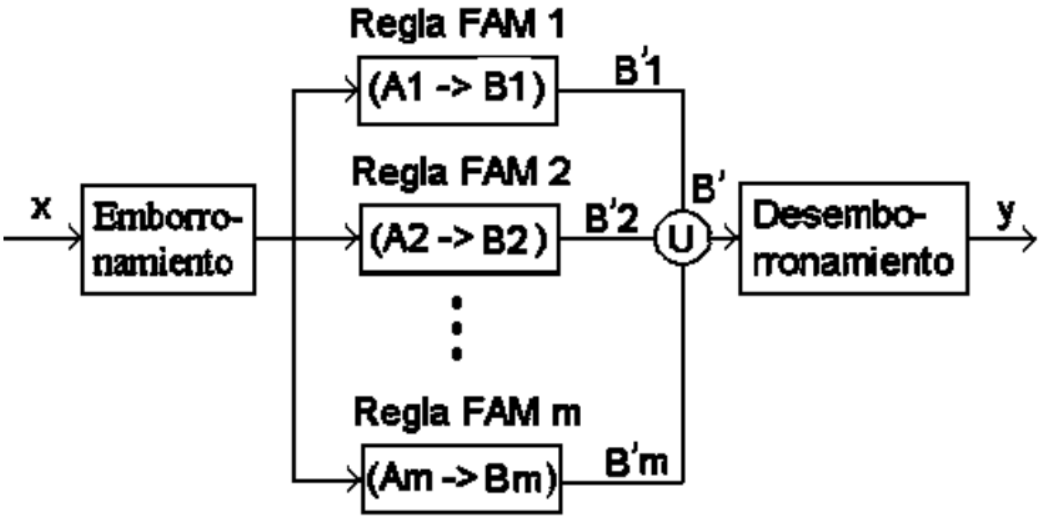


Figura 1.5;Partes del Controlador Borroso

Emborronamiento

Es el encargado de transformar las entradas en información tratable como entrada al sistema borroso.

Una entrada numérica puede ser representada como un conjunto borroso cuya función de pertenencia tiene el valor 1 en el punto de la entrada (x_i) y a todo lo demás se le asigna el valor 0. Este tipo de conjunto se llama singleton.

$$A'(x) = \begin{cases} 1 & \text{si} \quad x = x_i \\ 0 & \text{si} \quad x \neq x_i \end{cases}$$

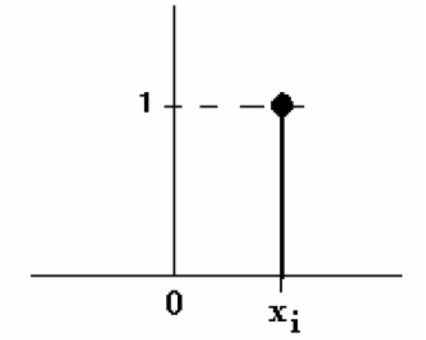


Figura 1.6

Grado de coincidencia entre conjuntos de entrada y antecedentes:

Para el cálculo de coincidencia entre conjuntos se utiliza la "**medida de posibilidad**" $\lambda=\Pi(A'/A)$, que es el valor máximo de la intersección entre entrada y antecedente. Para el caso que nos interesa, entrada numérica, cualquier definición de intersección nos da como **grado de coincidencia** el **grado de pertenencia** del valor numérico de entrada al conjunto borroso antecedente.

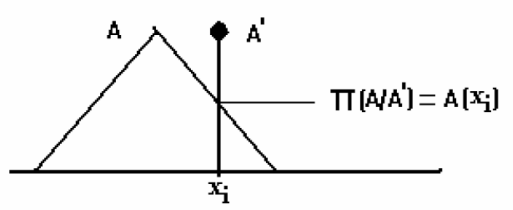


Figura 1.7

Bloque de inferencia borrosa

De este bloque salen, en un primer paso, tantos conjuntos borrosos como reglas tenga el controlador. En un segundo paso se obtiene un solo conjunto de salida a partir de los de salida de las reglas.

Primeramente, para cada regla, mediante los grados de coincidencia entre cada entrada y el antecedente correspondiente obtenidos como se ha explicado, se obtiene **un nivel de activación de la regla (λ)**. Esto se puede hacer de varias formas, las más utilizadas son las del mínimo (para dos entradas λ=min(λ1,λ2)) y el producto (para dos entradas λ=λ1λ2)). Con este nivel de activación se obtiene el conjunto borroso de salida de la regla, para lo que nuevamente se pueden utilizar diversos métodos, de los que los más utilizados son el mínimo y el producto.

Para sistemas con dos antecedentes A y B

Mínimo

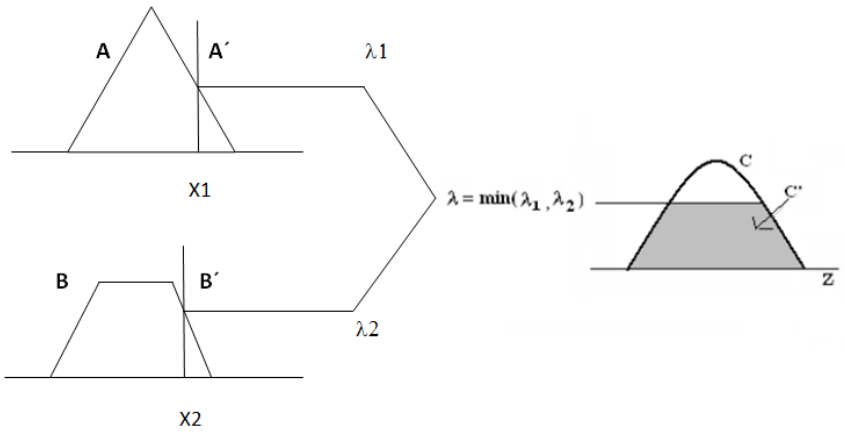


Figura 1.8

Mediante mínimos, la función se corta por el grado de coincidencia más pequeño (mínimo).

Producto

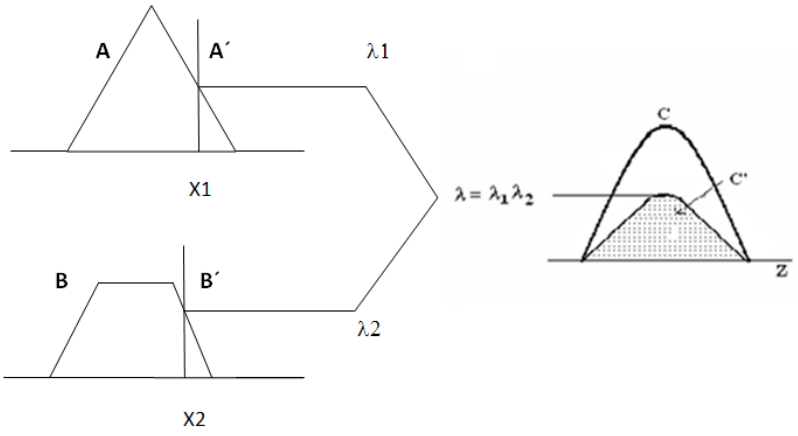


Figura 1.9

Mediante producto, la función se multiplica por los dos grados de coincidencia obtenidos.

Después de obtener los conjuntos de salida de cada regla, se debe obtener un único conjunto borroso de salida. Esto se puede hacer de diversas formas. La que se utiliza en este trabajo es la de la suma.

Suma de conjuntos de salida

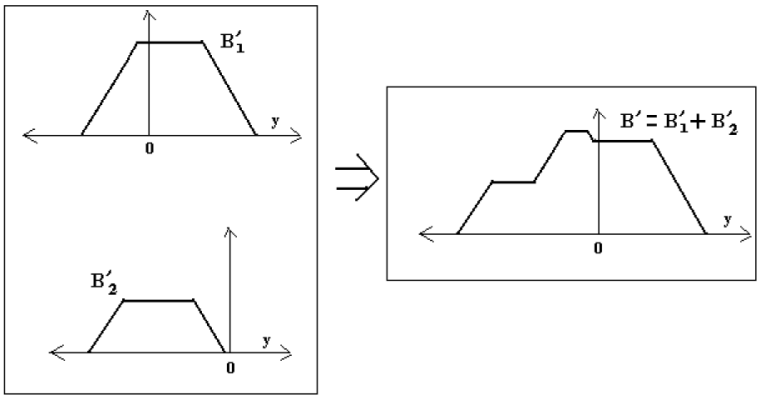


Figura 1.10

Desemborronamiento

De este bloque se obtiene un valor numérico. Existen diversos métodos para calcular esta salida; el más sencillo es el método de la media de los máximos, que se realiza sumando todos los valores de los máximos y sacando su media

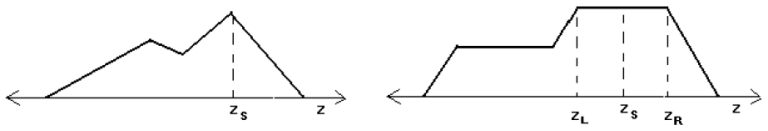


Figura 1.11

Este método de desemborronamiento no es muy eficaz ya que solo tiene en cuenta los máximos y no el total del conjunto. Es muyo mejor, por ello más utilizado, y es el que se va a utilizar aquí, el método del centro de gravedad.

Método del centro de gravedad

Como bien indica su nombre, con este método se obtiene el centro de gravedad de todo el conjunto. Se puede demostrar que, si se utiliza la suma para la agregación, el centro de gravedad del conjunto agregado se puede calcular a partir de las áreas y los centros de gravedad de los conjuntos de salida de cada regla así:

$$Zs = \frac{C1xA1 + C2xA2 + C3xA3 + \cdots + CixAi}{A1 + A2 + A3 + \cdots + Ai}$$

Ci=centroide (centro de gravedad) del conjunto de salida de la regla i-ésima

Ai=área del conjunto de salida de la regla i-ésima

$$Zs = \frac{\Sigma CixAi}{\Sigma Ai}$$

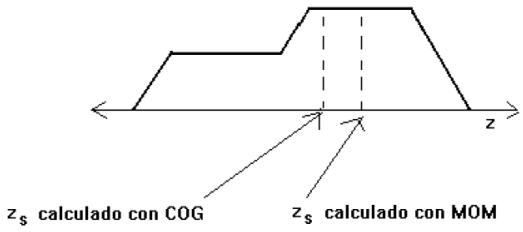


Figura 1.12

Sugeno

En el sistema borroso de Sugeno, las entradas son numéricas, los antecedentes de las reglas son borrosos y los consecuentes son funciones.

SI (antecedente 1) Y/O (antecedente 2), entonces z=f(entrada1, entrada2)

Ejemplo

SI X1 es positivo Y X2 es positivo, entonces y=0.75X1 + 0.25X2 – 0.3

Puede que la salida en lugar de ser una función sea una constate, como ocurre en este TFG, ya que las salidas son los datos “correctos” de salida que ha proporcionado el usuario experto para diferentes entradas de grasas y velocidad.

Como se ha dicho en el inicio, las entradas de Sugeno son numéricas y los antecedentes son borrosos, por tanto el grado de coincidencia entre la entrada y el antecedente es el grado de pertenencia del valor de la entrada al conjunto antecedente.

Con los grados de coincidencia se obtienen, como se ha explicado, los niveles de activación. Para obtener la salida numérica se debe realizar el método de la media ponderada, que consiste en ponderar las salidas de cada regla (calculadas evaluando las funciones de salida en los valores de las entradas) con los niveles de activación de las reglas:

$$Z1 = \lambda 1 \; x \; f1(x_1, x_2, \dots x_n)$$

$$Z2 = \lambda 2 \; x \; f2(x_1, x_2, \dots x_n)$$

$$En\ general: Zi = \lambda i \; x \; fi(x_1, x_2, \dots x_n)$$

$$Zs = \frac{Z1 + Z2 + \cdots + Zi}{\lambda 1 + \lambda 2 + \cdots + \lambda i}$$

X_1=entrada numérica 1

X_2=entrada numérica 2

x_n=entrada numérica n

λi=nivel de activación de la regla i

$fi(x_1, x_2, \dots x_n)$ =función del consecuente i en función de las entradas

Zs=salida numérica del controlador

$$Zs = \frac{\Sigma Zi}{\Sigma \lambda i}$$

El sistema borroso más conocido es el de Mamdani. Sin embargo, el sistema de aprendizaje automático (NEUROBORROSO) que tiene Matlab es sólo para el modelo de Sugeno. Sin embargo, se puede demostrar que, un Sugeno con salidas constantes es equivalente a un Mamdani con agregación con la suma e inferencia con el producto.

1.5.- Aprendizaje

En este TFG se parte de un primer controlador que se ajustó “a mano” para reproducir las relaciones entradas-salidas deseadas que nos proporcionó un paciente diabético usuario experto de bomba de insulina (sujeto1).

El trabajo que se va a presentar consiste en utilizar un método de aprendizaje para sistemas borrosos que tiene MATLAB para tratar de que el controlador sea capaz de reajustar sus parámetros de acuerdo con las necesidades de otro sujeto.

El problema de determinar un modelo matemático para un sistema desconocido mediante observación de los pares de datos entradas y es un problema de identificación. El propósito de los sistemas de identificación son múltiples:

- predecir el comportamiento de los sistemas

-explicar la interacción y relación entre las entradas y las salidas del sistema.

-diseñar un controlador basado en el modelo del sistema.

En primer lugar se debe escoger la estructura del modelo, es decir, se ha de ser capaz de representar el sistema mediante una función parametrizada $y = f(u;\ \theta)$ donde y es la salida del modelo, u es el vector de entradas y θ es el vector parámetros.

En este TFG se dispone de una estructura conocida para el modelo a identificar, correspondiente al modelo con el que se consiguió reproducir el comportamiento del sujeto 1 en el PFC de cuyos resultados partimos. Lo que hay que hacer es aplicar técnicas de optimización para determinar el vector θ de modo que el modelo resultante $y = f(u;\ \theta)$ describa el sistema apropiadamente, es decir, que ante las entradas de los datos de aprendizaje el modelo proporcione las salidas deseadas.

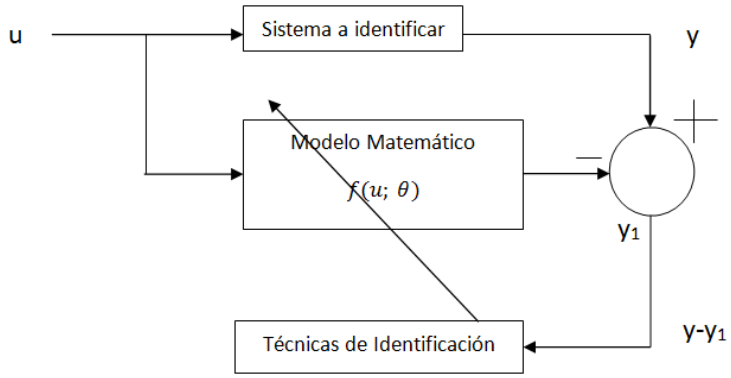


Figura 1.13

El diagrama de bloques de la figura representa el modo de trabajar para la identificación del sistema. U es la entrada e y es la salida deseada. y_1 es la salida del modelo matemático el cual se compara con la salida deseada y mediante el bloque de técnicas de identificación, se cambia el valor de θ, modificando así el modelo matemático. La resta y - y_1 es el error e irá disminuyendo con reiteradas iteraciones hasta que llegue un momento en que la diferencia entre y e y_1 sea cero.

Por lo General el modelo matemático no se acopla al sistema a identificar en un único paso; es necesario ”entrenar” más de una vez hasta que el modelo matemático satisfaga las necesidades.

El estudio teórico del algoritmo matemático que utiliza Matlab para minimizar el error de identificación no es objeto del presente proyecto. El trabajo se va a centrar aprender a utilizarlo correctamente hasta conseguir resolver el problema que nos planteamos, así como en detectar las capacidades de dicho algoritmo, sus fallos, debilidades o carencias, y las posibilidades de mejora.

1.6.- Sistema de decisión para ayuda en el uso de bombas de insulina

Los factores que afectan a decisión de la cantidad y forma de administración de la insulina se basa principalmente en las cantidades de carbohidratos que se van a ingerir, la velocidad de digestión de los carbohidratos y las cantidades de grasa. Aparte, existe otro factor que también influye en la decisión como la previsión del ejercicio que se va a realizar posteriormente.

A continuación se va a explicar el diagrama de bloque que representa el sistema implementado en el PFC del que se parte.

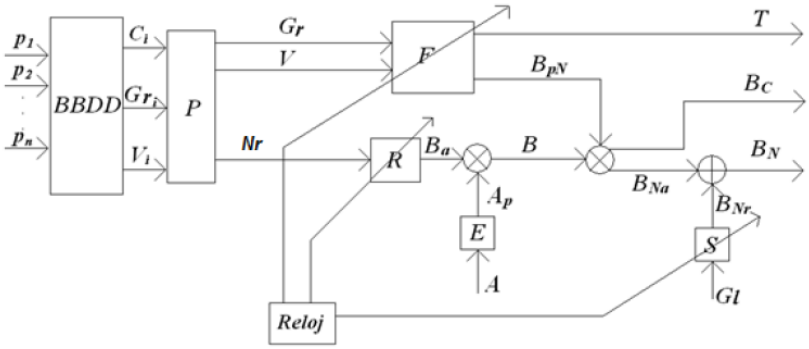


Figura 1.14; Esquema de sistema de decisión

Inicialmente se introduce la comida que se va a ingerir y la cantidad en gramos (P_1, P_2, …, P_n). Estas entradas van al **bloque BBDD** el cual traduce en cantidad de carbohidratos (C_i), grasas (Gr_i) y velocidad de digestión (V_i) de los carbohidratos para cada plato, incluidas la bebidas. Estos valores salen escalados de 1 a 9. Para la velocidad de digestión de los carbohidratos, el valor 1 indica "muy rápido" y el 9 "muy lento", para grasas el valor 1 indica "alimentos sin grasas" y el 9 indica "alimentos con mucha grasa". Este bloque contiene una amplia base de datos para poder tratar.

Al **bloque P** entran los valores de grasas (Gr_i), velocidad de digestión de los carbohidratos (V_i) y la cantidad de carbohidratos de cada plato (C_i). Mediante unos cálculos, a la salida se obtienen 3 únicos valores; la cantidad total de carbohidratos (Nr), un valor ponderado de las grasa en gramos (Gr) y el valor de la velocidad de digestión ponderado (V).

$$C = \sum_{i=1}^{n} C_i \;;\; Gr = \frac{\sum_{i=1}^{n} p_i\, Gr_i}{\sum_{i=1}^{n} p_i} \;;\; V = \frac{\sum_{i=1}^{n} C_i\, V_i}{C}$$

El **bloque reloj** ajusta los valores de salida en función de la hora en la que se consumen los alimento. Para las franjas horarias, desayuno, almuerzo-comida y merienda-cena, este bloque se encarga de proporcionar los parámetros de ratio insulina/carbohidratos (RC) y el factor de sensibilidad a la insulina (S) apropiado para dichas franjas.

El **bloque R** proporciona las unidades de insulina (Ba) necesarias en relación a los carbohidratos (Nr) obtenidos en el bloque P. Para una actividad prevista (A) baja y en normoglucemia, las unidades de insulina a suministrar (Ba) tiene una proporcionalidad con las raciones de carbohidratos que se van a ingerir, lo cual se implementa en este bloque. Esta proporcionalidad es el antes mencionado ratio insulina/carbohidratos (RC). Además, las unidades de insulina (Ba) cambian en función de la actividad prevista y de la franja horaria en la que se encuentre el usuario, por lo que el bloque reloj tiene que inferir en la salida de este bloque. Hay que destacar que este ratio insulina/carbohidratos es diferente para cada persona.

$$Ba = RC\ x\ Nr$$

El **bloque E** es el encargado de modificar los Bolus totales (Ba) en función de la actividad prevista (A). El subíndice a indica que está pendiente de modificación por otros factores. La

actividad (A) está escalada de 1 (actividad mínima) a 9 (actividad extrema). De este modo, cuanta más actividad (A) se vaya a realizar, más se reducen las unidades de insulina (Ba) a administrar. Esta reducción se realiza de forma lineal; con una actividad prevista de 2 no se realizan modificaciones y con una actividad prevista de 8 se reduce en un 40%. A continuación se detallará la reducción (Ap), siempre que la actividad prevista (A) sea mayor a 2:

$$Ap = (A - 2)x6.67$$

Una vez obtenido el factor de reducción en función de la actividad prevista (Ap), se debe aplicar a los Bolus totales (Ba) calculados en el bloque R. El cálculo de los Bolus totales después de reajuste (B) se realiza mediante la siguiente fórmula:

$$B = \left(1 - \frac{Ba}{100}\right) xBa$$

El **bloque F** realiza las estimaciones del porcentaje de Bolus tipo N (Bpn) sobre el total de Bolus (B) y el tiempo de suministro de los Bolus tipo C (T). Hay que tener en cuenta que las variables T y Bpn varían para cada individuo y con el ritmo circadiano.

Estas dos variables guardan relación con la naturaleza del alimento que se va a consumir, es decir, con los parámetros grasas (Gr) y velocidad (V) y son independientes de la cantidad de comida, es decir, de los carbohidratos (Nr). Por ello, las entradas al bloque Fuzzy (bloque F) son Gr y V y no Nr. Mediante los datos proporcionados por los usuarios, se ha determinado que el tiempo de suministro de los Bolus tipo C (T) está en la franja 60-300.

Para cada franja horaria se han propuesto tres estimadores, implementados por el bloque reloj. En la franja desayuno solo se administran Bolus tipo N (Bolus totales=Bolus tipo N), por lo que no tiene sentido hablar de T. Para la franja almuerzo/comida, y merienda/cena se realizarán dos controladores Fuzzy para cada franja ya que, como se ha mencionado anteriormente, los valores cambian en función del ritmo circadiano. Estos controladores proporcionarán las salidas para una actividad prevista baja y en normoglucemia.

La salida BpN es en % por ello, se debe multiplicar con los Bolus Totales después de haber sido reajustados (B), y de esa multiplicación se obtienen los Bolus de tipo N previos a reajuste por el bloque S (BNa).

El **bloque S** actúa sobre BNa en función del nivel de glucemia en sangre (Gl). Para realizar el cálculo, el bloque S considera el factor de sensibilidad a la insulina (S), el cual informa de cuantos mg/dl desciende la glucemia en sangre (Gl) por unidad de insulina. Si supera los 100mg/dl se deberá suministrar mayor número de Bolus para procesar toda la glucosa excedente, mientras que si es inferior, se debe reducir la cantidad de insulina. Esto se realiza sobre BNa para que sea efectivo en poco tiempo.

$$B_{N_r} = \frac{Gl - 100}{S} \quad ; \quad B_N = B_{Na} + B_{Nr}$$

1.7.- Controlador neuroborroso

Este es el bloque fundamental de decisión del sistema. En las siguientes líneas se explicará el proceso realizado en un TFC anterior para el diseño del controlador.

Como se ha comentado en el apartado de sistema de decisión, las entradas al bloque F son grasas (Gr) y velocidad de digestión (V) y las salidas son el tiempo (T) para la insulina de efecto lento (Bolus C) y la proporción de Bolus N sobre los Bolus totales, previo a correcciones.

Respecto al número de controladores, caben dos posibilidades; realizar uno con los mismos antecedentes de entrada o realizar dos controladores con diferentes antecedentes de entrada. La programación de un solo controlador resulta más laboriosa y además, según se ha comprobado, con dos controladores, uno para T y otro para BN, con los mismos antecedentes para cada uno, da unos buenos resultados, además de que la programación es más sencilla.

Figura 1.15; Un controlador para dos salidas

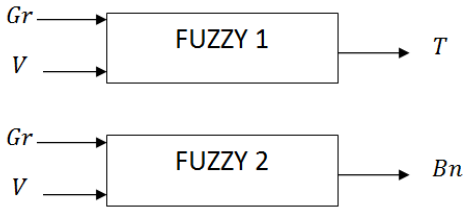


Figura 1.16; Un controlador para cada salida

Los datos proporcionados por un usuario experto para normoglucemia son los siguientes.

Comida	Raciones	Velocidad	Grasas	Ejercicio	Bolus N	% Bolus N	Bolus C	% Bolus C	Velocidad
D	5	7	3	5	5,5	100,00	0	0,00	0
M	5,5	4	6	6	6,5	61,90	4	38,10	180
M	5	3	2	3	3	33,33	6	66,67	180
M	5	3	4	3	3	33,33	6	66,67	210
M	5	3	7	3	3	30,00	7	70,00	240
T	5	5	5	1	5	50,00	5	50,00	210
M	5	6	2	4	7	70,00	3	30,00	180
T	4	7	2	1	5	71,43	2	28,57	180
M	5	5	4	3	5	55,56	4	44,44	210
M	5	5	7	3	5	50,00	5	50,00	240
M	5	8	2	3	7	77,78	2	22,22	120
M	8	5	3	2	8	61,54	5	38,46	210
T	5	7	2	1	6	70,59	2,5	29,41	180
M	4,5	6	2	4	6	75,00	2	25,00	120
T	6	5	5	2	6	57,14	4,5	42,86	210
M	4,5	6	4	3	5,5	64,71	3	35,29	180
M	4	7	4	3	6	66,67	3	33,33	180
M	11	5	2	2	12	66,67	6	33,33	240
M	5	8	4	3	7	77,78	2	22,22	180
M	5	8	7	3	6	60,00	4	40,00	240

Tabla 1.1; Datos de UE1

En las siguientes tablas se muestran los datos de UE1 escalados de 1 a 9 en forma matricial y descompuestas en función de la franja horaria.

Bolus N Franja almuerzo/comida.

		Velocidad								
	BN	1	2	3	4	5	6	7	8	9
Grasas	1									
	2			33,33		66,67	72,5		77,78	
	3					61,54				
	4			33,33		55,56	64,71	66,67	77,78	
	5									
	6				61,9					
	7			30		50			60	
	8									
	9									

Tabla 1.2

Bolus N franja merienda/cena.

		Velocidad								
	BN	1	2	3	4	5	6	7	8	9
Grasas	1									
	2							71,01		
	3									
	4									
	5					53,57				
	6									
	7									
	8									
	9									

Tabla 1.3

Tiempo franja horaria almuerzo/comida.

		Velocidad								
	T	1	2	3	4	5	6	7	8	9
Grasas	1									
	2			180		240	150		120	
	3					210				
	4			210		210	180	180	180,00	
	5									
	6				180					
	7			240		240			240	
	8									
	9									

Tabla 1.4

Tiempo franja horaria merienda/cena.

Grasas \ Velocidad	1	2	3	4	5	6	7	8	9
1									
2							180		
3									
4									
5					210				
6									
7									
8									
9									

Tabla 1.5

Tomando los valores centrales, se realizó una tabla para cada salida.

Grasas \ Velocidad	3	5	8
2	3	5	7
4	3	5	7
7	3	5	6

Tabla 1.6

Grasas \ Velocidad	3	5	8
2	6	4	2
4	6	1	2
7	7	5	3

Tabla 1.7

Grasas \ Velocidad	3	5	8
2	33,33	55,56	77,78
4	33,33	55,56	77,76
7	30	50	60

Tabla 1.8

Grasas \ Velocidad	3	5	8
2	180	180	120
4	210	210	180
7	240	240	240

Tabla 1.9

Después, se realizó un controlador borroso de tipo Mamdani para obtener un matriz 9x9 de los datos que faltan. Para ellos, se definieron 3 antecedentes para cada entrada; velocidad de digestión alta, media y baja y grasas alta, media y baja. Por lo tanto, el controlador se compuso de 9 reglas. Los consecuentes se implementaron de forma que los centroides fueran los valores de tiempo (T) y de las grasas (Gr) proporcionados por el usuario experto.

Una vez realizado el controlador, se revisó si los valores obtenidos por este controlador coincidían con los del usuario y resultó que eran válidos, por lo que se dio por bueno.

A continuación, se realizó una matriz 9x9 para las dos salidas. Esta matriz abarcaba todas las combinaciones posibles. Con ella, se realizó otro controlador borroso de 9 conjuntos borrosos triangulares para la velocidad y otros 9 conjuntos borrosos para las grasas, es decir, 81 reglas. Este programa se le dio al usuario que había proporcionado los datos y se comprobó que las decisiones que tomaba el usuario y el controlador eran similares, por lo tanto, se comprobó que era capaz de calcular decisiones válidas.

Observando el controlador, se advirtió que tenía los mismos consecuentes en varias de sus regla, por ello, se vio la posibilidad de diseñar un sistema borroso más sencillo con menos categorías en para los universos de entrada. Se definieron tres antecedentes borrosos trapezoidales para cada variable de entrada, es decir, un sistema con únicamente 9 reglas.

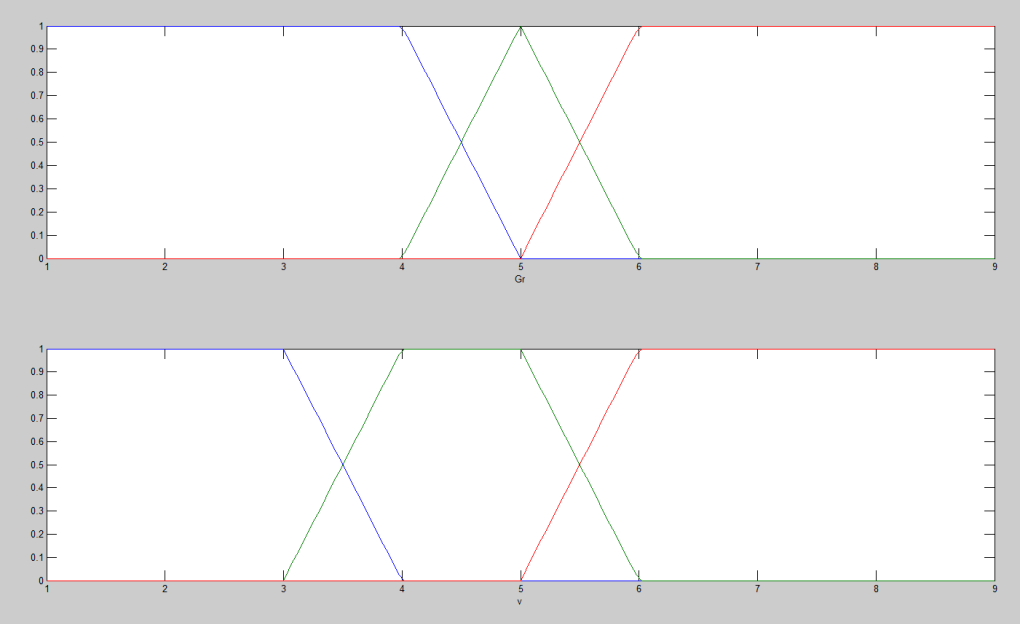


Figura 1.17

Para Bolus N se decidió mantener los 9 consecuentes, aunque 2 de ellos coincidían, pero para el universo de salida del tiempo, se dejaron 6 consecuentes diferentes, aunque cuatro tomaban valores iguales dos a dos.

Por último se comprobó que este nuevo sistema funcionaba igual de bien que el sistema de 81 reglas.

En el proyecto actual, se desea ensayar con la capacidad de aprendizaje de sistema neuroborroso de Matlab y ajustar los controladores borrosos para T y % Bolus N, ante unas entradas dadas. Se han diseñado controladores de Tipo Sugeno, ya que el sistema

neuroborroso de Matlab sólo admite este tipo de sistema. Además, los consecuentes proporcionados por el usuario son numéricos, que es como Sugeno puede trabajar.

2.- Cuerpo

2.1.- Introducción a ejercicios y comandos

En este apartado se explicarán los comandos fundamentales necesarios para el aprendizaje y ajuste de los controladores que se usaron en este TFG.

El comando de Matlab para generar un sistema borroso de inferencia (FIS o Fuzzy Inference System) es el comando "genfis1". Este comando genera un FIS a partir de la matriz de datos de entradas-salidas con la que se desea trabajar. Para eso, se tienen que introducir las características deseadas del FIS, como tipo de funciones de pertenencia, cuantas MF, tipo de salidas deseadas (constantes, funciones etc.) y como se ha mencionado antes, los datos con los que se quiere generar el FIS. El comando "genfis1" sólo utiliza estos últimos datos para definir el rango de las entradas, pero no realiza el entrenamiento. El sistema borroso que devuelve será el sistema inicial para el entrenamiento posterior.

fismat = genfis1(data,numMFs,inmftype,outmftype)

También se puede dejar a Matlab que genere el sistema borroso de inferencia FIS por defecto. Para ello, únicamente se debe introducir los datos de trabajo.

fismat = genfis1(data)

Este comando, inicialmente genera una salida de ceros, es decir, que no tiene en cuenta los datos de salida, y por ello es necesario entrenar el sistema para que sea capaz de dar una respuesta deseada ante las entradas. Existen otros comandos como "genfis2" y "genfis3" que la salida generada no es cero, para lo cual realiza un cálculo con la matriz de datos y obtiene unos valores de salida que pueden no ser los adecuados. Por ello, también se debe realizar iteraciones de entrenamiento.

Se ha optado por el comando "genfis1" porque es más sencillo de usar.

Para realizar el entrenamiento se usa el comando "anfis".

[fis,error,stepsize,chkFis,chkErr] = anfis (trnData,initFis,trnOpt,dispOpt,chkData,optMethod)

Lo más importante a introducir para realizar el entrenamiento son los datos con los que se quiere entrenar (trnData), el FIS sobre el que se va a entrenar (initFis) y el número de iteraciones a realizar.

Los datos con los que se realiza el entrenamiento se expresan en forma de matriz. Si el sistema tiene N entradas, la matriz deberá tener N+1 columnas, siendo las primeras N columnas las entradas y la última columna será la salida a esas entradas.

El FIS resultante será el "fis" y el error se almacenará en "error".

Una vez realizada la introducción del método de ejecución de aprendizaje, se expondrán los ejercicios realizados en este Trabajo Fin de Grado.

2.2.- Ejercicio 2 entradas

Después de haber estudiado la teoría y realizado los ejercicios de la guía "Fuzzy Logic Toolbox", se ha planteado realizar un ejercicio que sea de dos entradas y una salida para aprender a programar sistemas de dos entradas ya que los ejercicios propuestos del libro eran de una entrada únicamente.

El ejercicio propuesto consta de dos entradas y una salida, y consiste en ajustar 200 datos de la siguiente función de dos variables:

$$x_1 \epsilon [-1,1]$$

$$x_2 \epsilon [-1,1]$$

$$y = sinc({x_1}^2 + {x_2}^2)$$

Se ha elegido que el número de funciones de pertenencia sea de 5 para cada entrada y de forma gaussiana. Mediante el comando "genfis1" se deja que Matlab distribuya automáticamente las funciones de pertenencia.

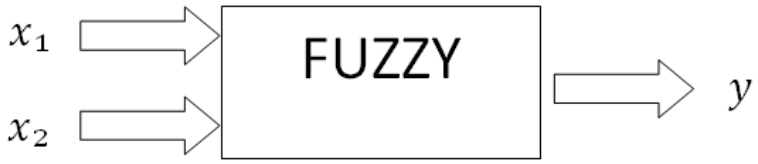


Figura 2.1; Esquema ejercicio

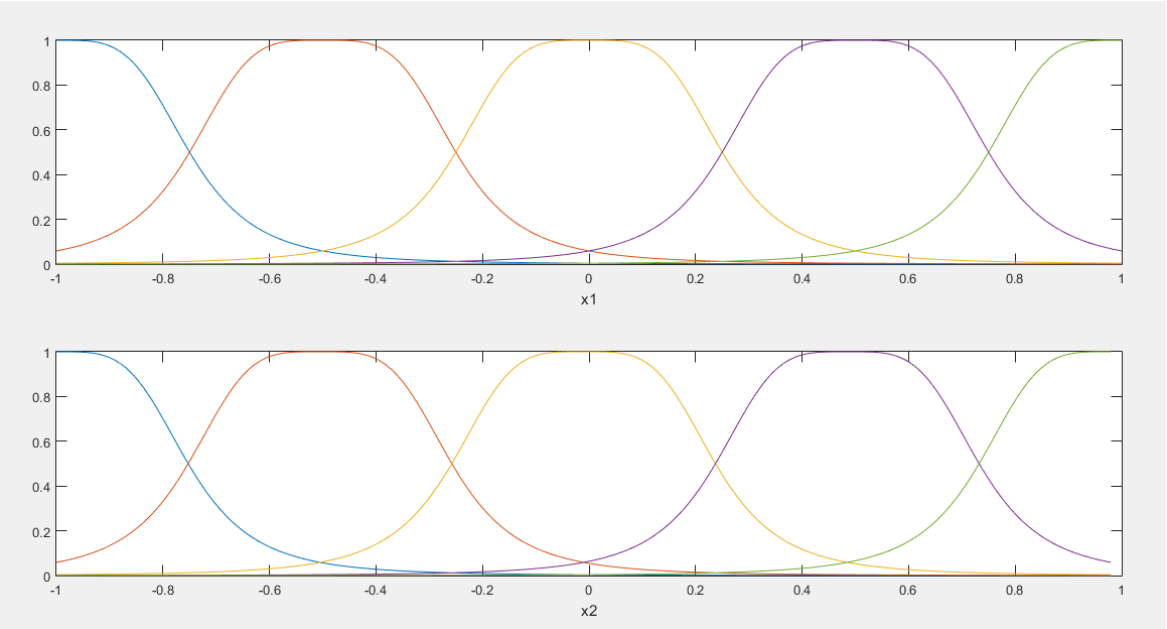


Figura 2.2; MF aleatorias antes de entrenamiento

Se ve que Matlab realiza por defecto la distribución de forma equidistante.

Después, se realiza el aprendizaje tomando como FIS base el generado con el comando "genfis1" y con los datos de entrenamiento x_1, x_2 e y. En el aprendizaje se modifica los parámetros y adapta las funciones de pertenencia preestablecidas por Matlab de modo que satisfaga de manera más eficiente la relación entre las entradas y las salidas. Las funciones de pertenencia de los conjuntos antecedentes evolucionan como se ve en la siguiente figura.

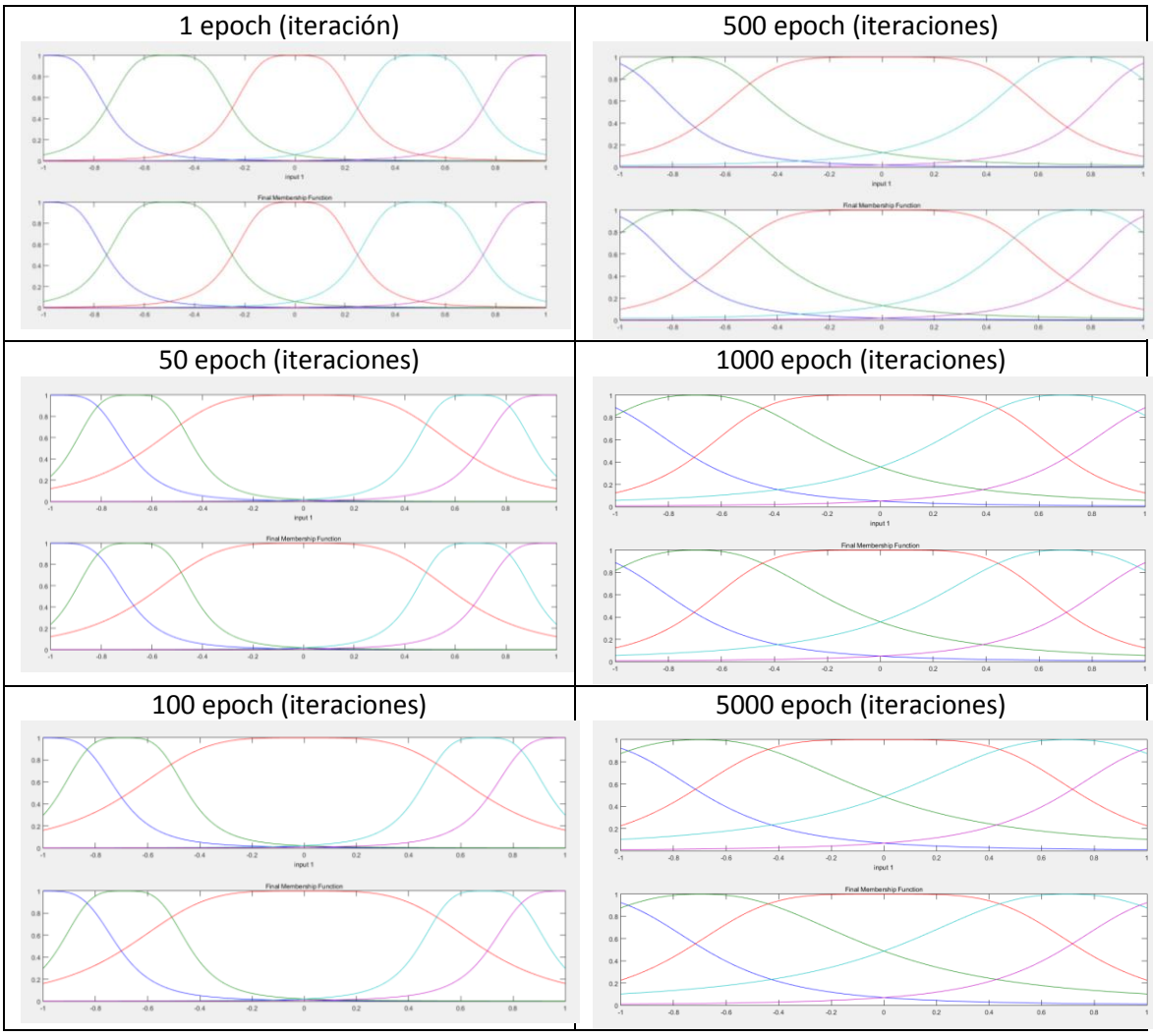


Figura 2.3; Evolución de MF

Error

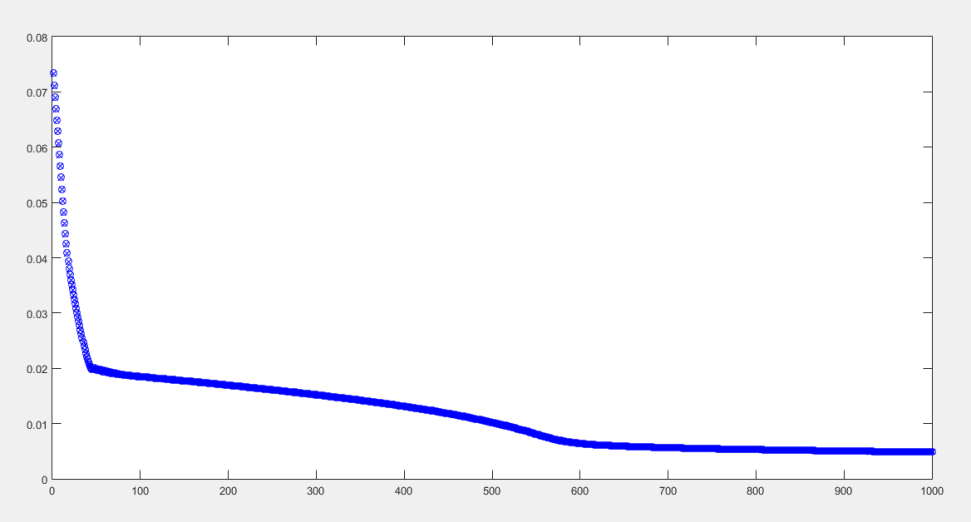
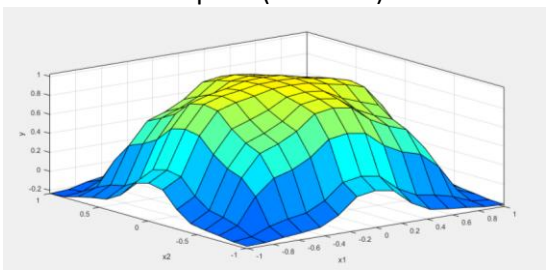
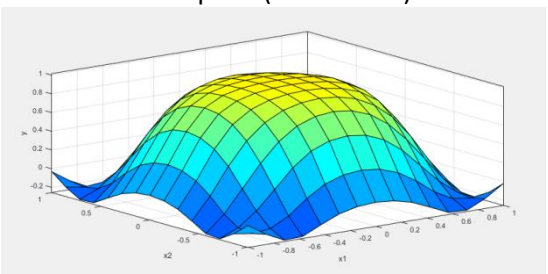
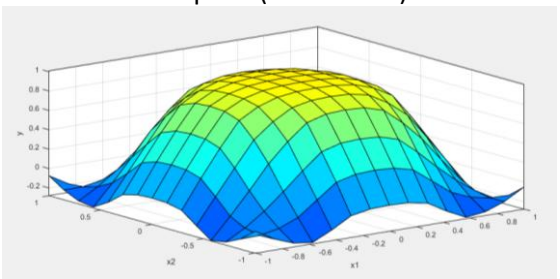
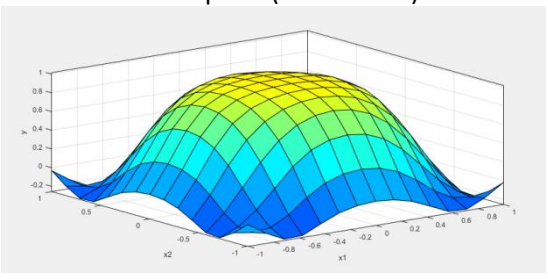


Figura 2.4; Evolución del error

Como se puede ver, el error entre los datos de entrenamiento y de chequeo disminuye cuantos más entrenos se realizan. En este ejercicio alrededor de las 1000 iteraciones se estanca el error de entrenamiento. También se puede ver cómo se modifica la superficie que representa la relación entradas-salida cuantas más iteraciones se realizan.

Superficie

1 epoch (iteración)	500 epoch (iteraciones)
	
50 epoch (iteraciones)	1000 epoch (iteraciones)
	

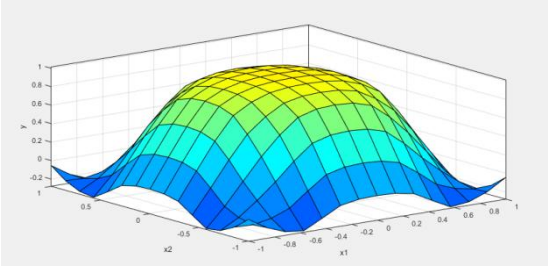
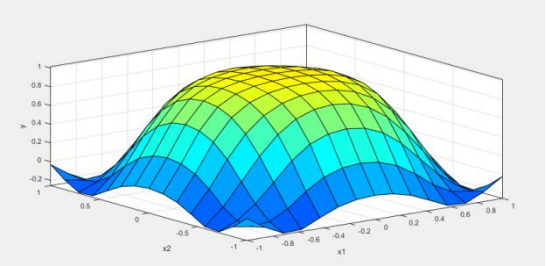
100 epoch (iteraciones)	5000 epoch (iteraciones)
	

Figura 2.5; Evolución de las superficies

En las siguientes imágenes se muestra el antes y después de las funciones de pertenencia de los antecedentes y los consecuentes de todas las reglas.

1 epoch

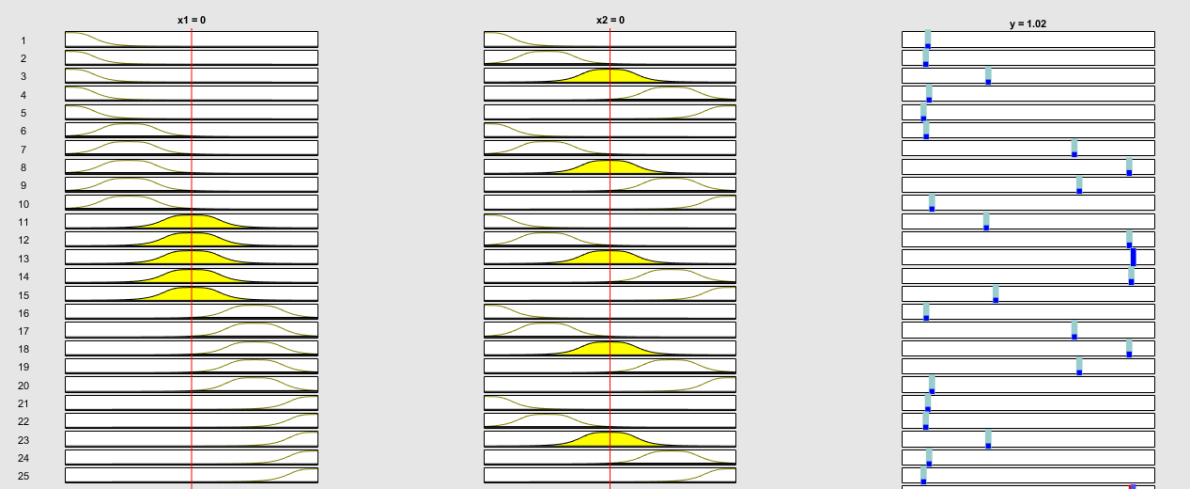


Figura 2.6

5000 epoch

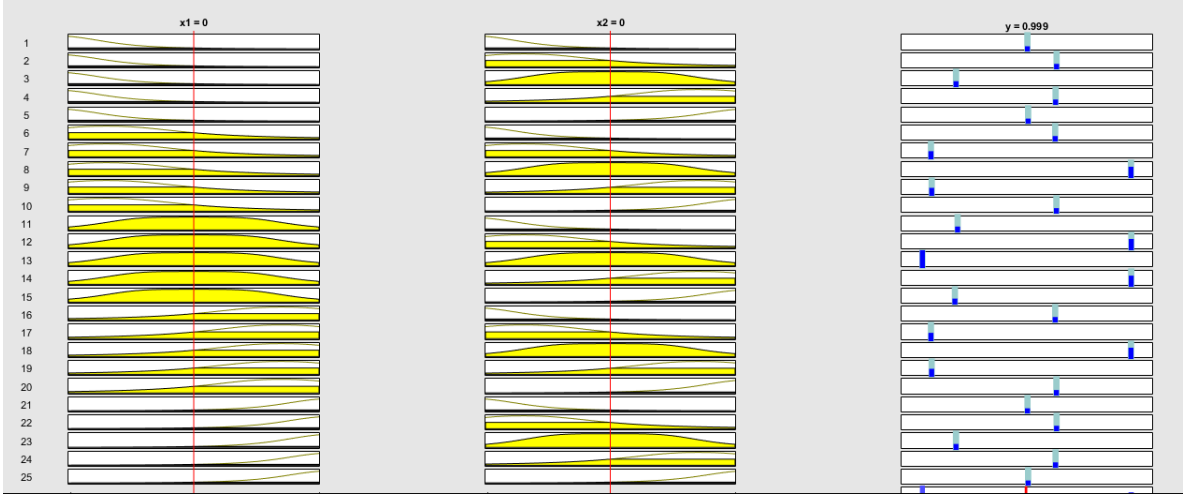


Figura 2.7

2.3.- Ajuste de controlador mediante datos de Usuario Experto 1 (UE1)

En este primer ejercicio vinculado directamente al proyecto, se desea comparar el controlador obtenido en un proyecto anterior que se construyó "a mano" a partir de los datos de un usuario y un controlador realizado mediante los datos de usuario aplicando los algoritmos de aprendizaje de Matlab. Para ello, se han introducido los valores correspondientes a una actividad prevista baja y en la franja horaria almuerzo/comida que proporcionó el usuario experto en una matriz y se ha comenzado por definir las funciones de pertenencia igual que en el proyecto anterior; de forma trapezoidal, tres conjuntos borrosos para las grasas (alta, media y baja) y otros tres para la velocidad de digestión; 9 reglas en total. Esto se define con el comando "genfis1" con el que se genera un sistema borroso de inferencia para que sea entrenado con los datos de la matriz. En el controlador del proyecto anterior, el grado "Medio" de las grasas se fijó como triangular, pero para dar más libertad al aprendizaje, se ha definido como trapezoidal, de modo que pueda modificarlo más fácilmente. Los output o salidas a generar se han definido como constantes, que es como el usuario desea recibir los datos de salida. Respecto a la distribución de las MF, se ha dejado que Matlab las ponga por defecto.

A continuación se muestran los datos relativos a una actividad prevista baja y para la franja almuerzo/comida en normoglucemia proporcionados por el usuario experto 1 (UE1).

	Raciones	Velocidad	Grasas	Ejercicio	Bolus N	% Bolus N	Bolus C	% Bolus C	Velocidad
M	5	3	2	3	3	33,33	6	66,67	180
M	5	3	4	3	3	33,33	6	66,67	210
M	5	6	2	4	7	70,00	3	30,00	180
M	5	5	4	3	5	55,56	4	44,44	210
M	5	5	7	3	5	50,00	5	50,00	240
M	5	8	2	3	7	77,78	2	22,22	120
M	8	5	3	2	8	61,54	5	38,46	210
M	4,5	6	4	3	5,5	64,71	3	35,29	180
M	4	7	4	3	6	66,67	3	33,33	180
M	11	5	2	2	12	66,67	6	33,33	240
M	5	8	4	3	7	77,78	2	22,22	180
M	5	8	7	3	6	60,00	4	40,00	240

Tabla 2.1

Como se ha dicho en el apartado de Controlador neuroborroso, se han realizado dos controladores, uno para el tiempo de suministro de los Bolus C (T) y otro para los Bolus N.

Tiempo de suministro T

Como se ha mencionado antes, después de introducir la matriz de datos, se dejó que Matlab generase las funciones de pertenencia por defecto. Las únicas características que se fijaron fueron que debía estar compuesto por 3 MFs para cada uno de los dos universos de entradas y que estos debían ser de forma trapezoidal.

Antes de entrenar con los datos de UE1, el programa ha organizado las MF de forma equidistante, pero se puede ver que estas no abarcan todo el rango que se esperaba. En el

caso de las grasas, el rango va desde 2 hasta 7 y en el caso de la velocidad va de 3 a 8. Esto ocurre porque no se han introducido datos que estén fuera de los dichos rangos.

Después de realizar la optimización, el programa ha cambiado la distribución inicial de las MF y lo ha adaptado de forma que el error entre los datos de entrenamiento y los datos de salida del controlador resultante descienda y se estabilice.

En estas imágenes se muestran las funciones de pertenencia generadas previo al entrenamiento y después de 100 iteraciones.

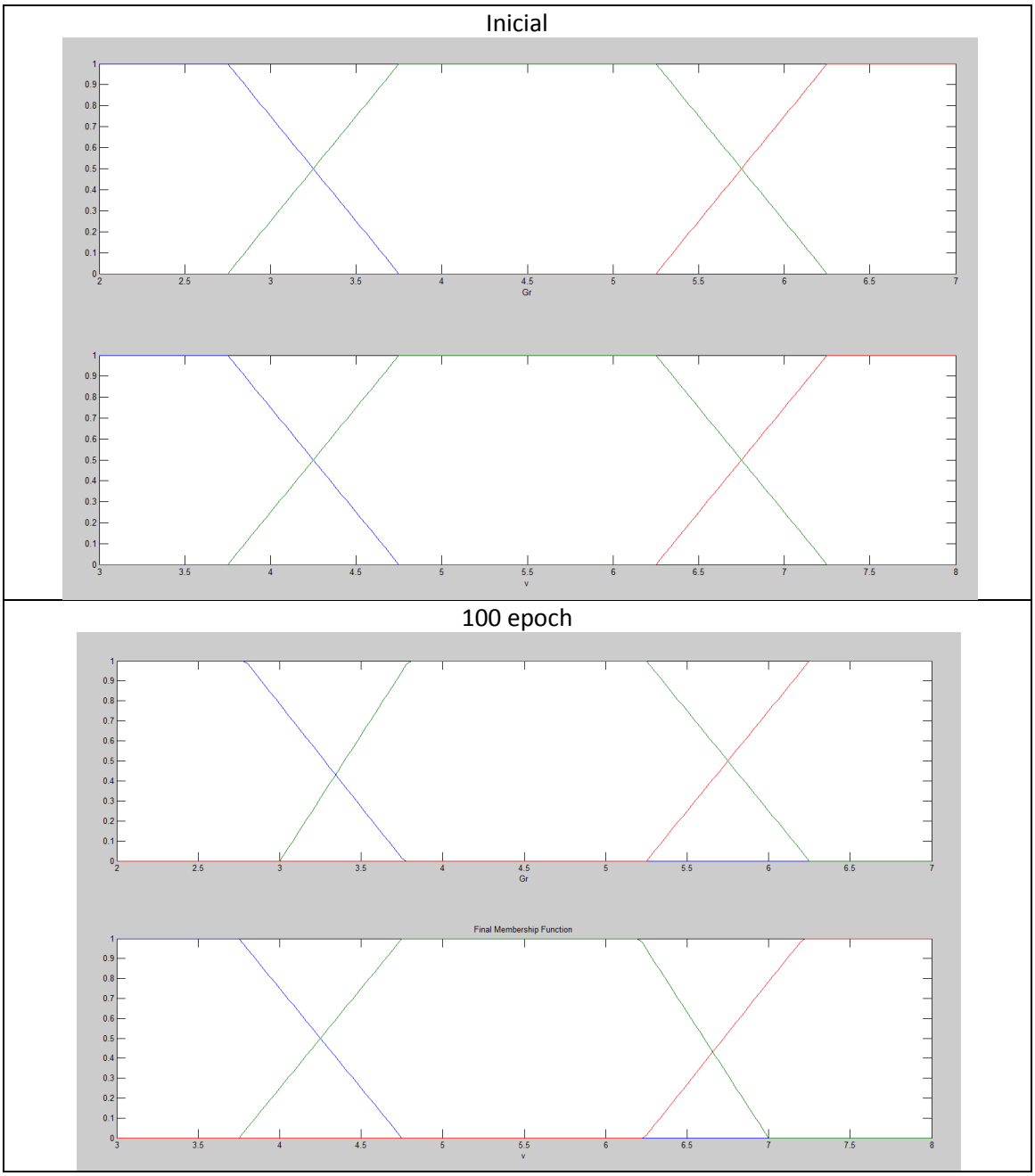


Figura 2.8

La imagen inferior muestra la evolución del error en función de las iteraciones realizadas.

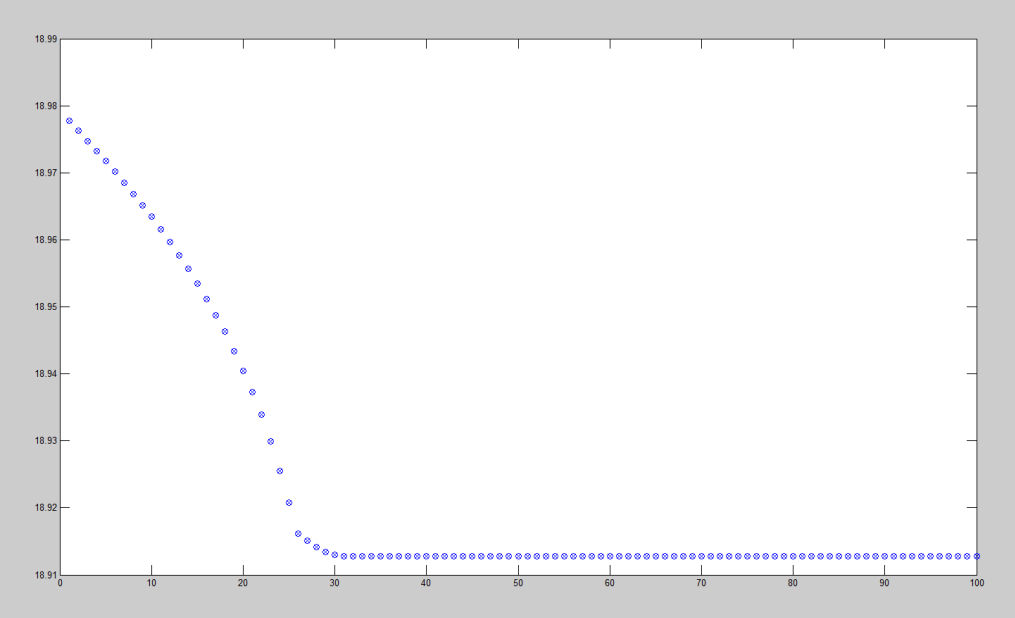


Figura 2.9

El error de la superficie inicial generada con el comando "genfis1" respecto a los datos de entrada es de 201.7615. Como se ha comentado en el apartado 1.7 relativo al controlador neuroborroso, los consecuentes generados con este comando son de valor cero. Cuando se aplica el aprendizaje con el comando anfis, en la iteración cero, la superficie se transforma en un plano paralelo a una altura de T=180.

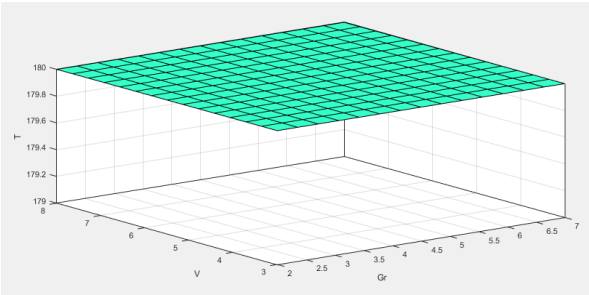


Figura 2.10; Iteración cero

En esta iteración, el error disminuye a 40.7620. En la primera iteración el error vuelve a disminuir a 18.9778 y después de 33 entrenamientos se estabiliza en 18.9128.

En la siguiente tabla se muestran los resultados de los consecuentes que se obtienen con el programa de aprendizaje de Matlab.

T		V		
		Baja	Media	Alta
Gr	Baja	179.9998	199.9999	119.9999
	Media	209.9998	194.9999	179.9999
	Alta	239.9998	239.9998	239.9998

Tabla 2.2

La tabla 2.3 muestra los valores que se obtuvieron en el anterior TFC.

T		V		
		Baja	Media	Alta
Gr	Baja	180	180	150
	Media	210	210	180
	Alta	240	240	210

Tabla 2.3

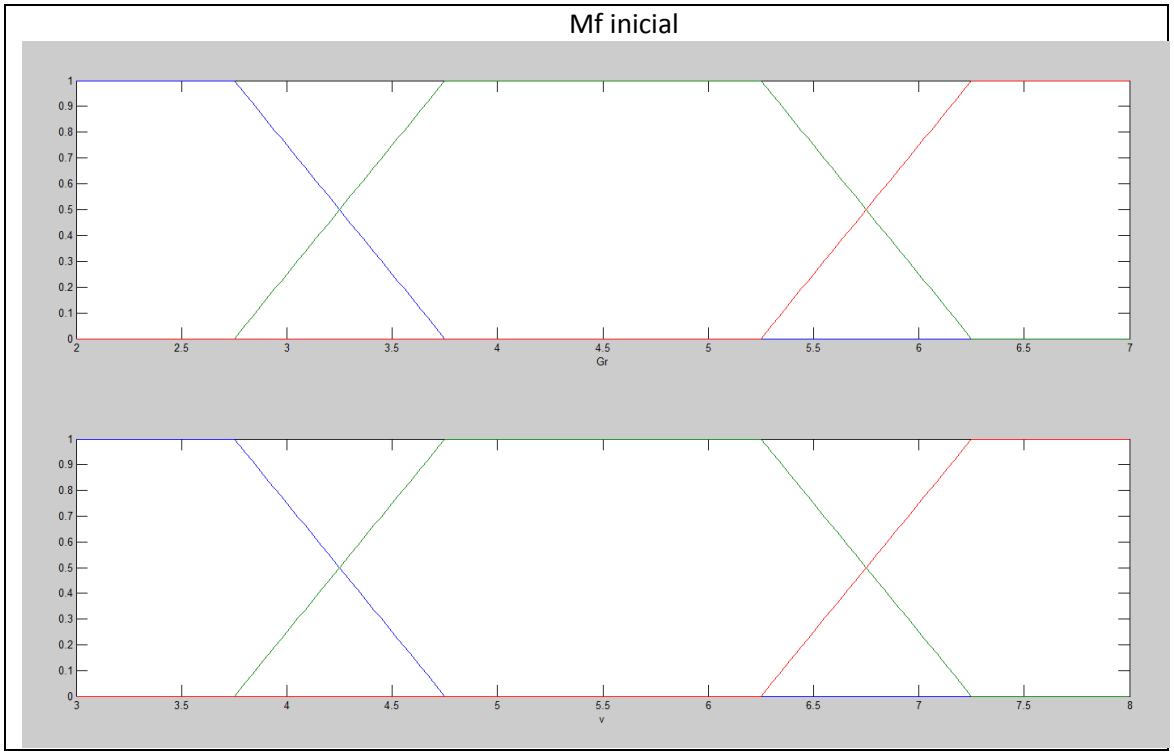
Comparando las tablas de las reglas del TFC anterior y la obtenida en el presente proyecto, se puede ver que Matlab después de modificar las MFs de los antecedentes, ha calculado otros consecuentes diferentes al TFC previo.

Estas diferencias podrían ser debidas a que en el proyecto fin de carrera anterior, el ajuste fue manual y datos próximos que diferían mucho con los de alrededor se despreciaron o se modificaron. En este proyecto se han cogido todos los datos disponibles y se han sometido a entrenamiento, sin analizarlos previamente. Esta puede ser la causa de la diferencia entre ambos controladores.

Bolus N:

Del mismo modo que antes, se ha dejado que Matlab genere por defecto las funciones de pertenencia pero fijando en 3 las funciones de pertenencia para cada universo de entradas y de forma trapezoidal.

Como ocurre con el tiempo de suministro T, hay valores que quedan fuera del universo "grasas", que como se ha mencionado antes, debería ir de 1 a 9, pero al no haber introducido valores que abarquen todo el rango, Matlab acota a los valores que se le han introducido a los que dispone. Lo mismo ocurre con el universo "velocidad".



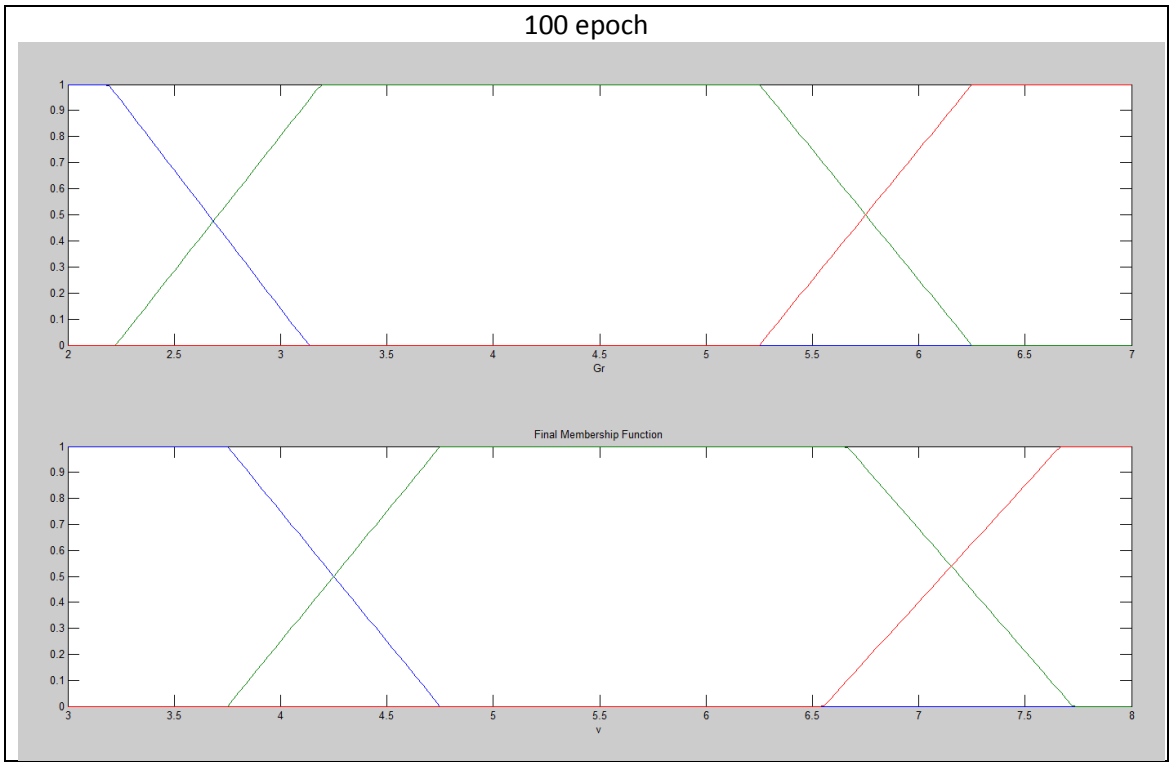


Figura 2.11

Además, como ocurre con el controlador de T, las funciones de pertenecía se modifican en la optimización, ajustándose así mejor forma a los datos de entrenamiento.

En la siguiente imagen se muestra la evolución del error en función de las iteraciones realizadas.

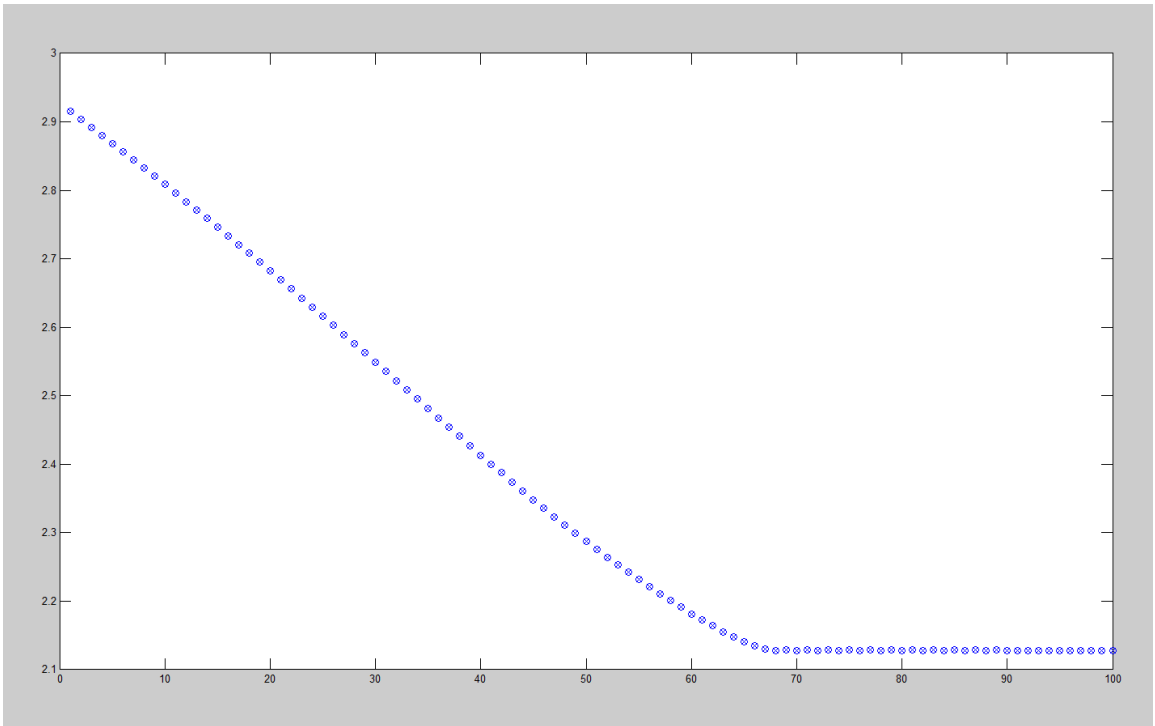


Figura 2.12

Al generar el FIS con el comando “genfis1”, el cual saca todos los valores a cero, el error es de 59.8233. Después de someterlo a entrenamiento con los datos del usuario UE1, en la iteración cero se obtiene una superficie plana con valor T= 53.89. En esta iteración, el error es de 16.3235.

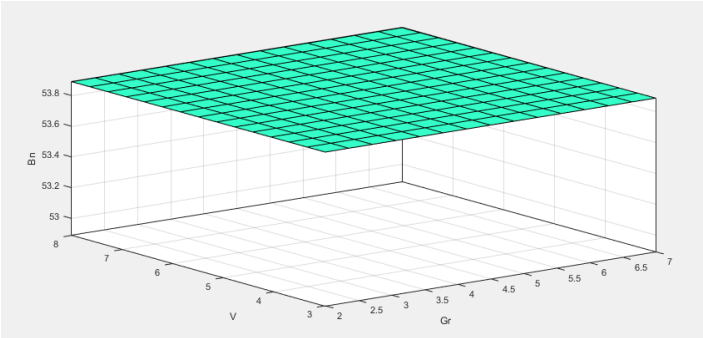


Figura 2.13; Iteración cero

Como muestra la figura 2.11 en la primera iteración del entrenamiento, el error resultante es de 2.9155 después de 75 entrenamientos, el error se estabiliza en 2.1278.

Estos son los resultados obtenidos después de 100 iteraciones:

Bolus N		V		
		Baja	Media	Alta
Gr	Baja	33.3000	69.5850	77.7799
	Media	33.3000	60.1350	77.7799
	Alta	30.0000	49.9999	59.9999

Tabla 2.4

Y estos los del TFC anterior:

Bolus N		V		
		Baja	Media	Alta
Gr	Baja	33.3	56	77.8
	Media	30	50	66.7
	Alta	30	44.4	60

Tabla 2.5

Al igual que ocurre en el controlador de tiempos de suministro de Bolus C (T), después de que se hayan modificado las funciones de pertenencia mediante la optimización, el controlador obtenido en este TFG cambia los consecuentes de las reglas con respecto al controlador realizado en el TFC previo.

En las siguientes imágenes se muestran las superficies obtenidas en el actual TFG después de la optimización.

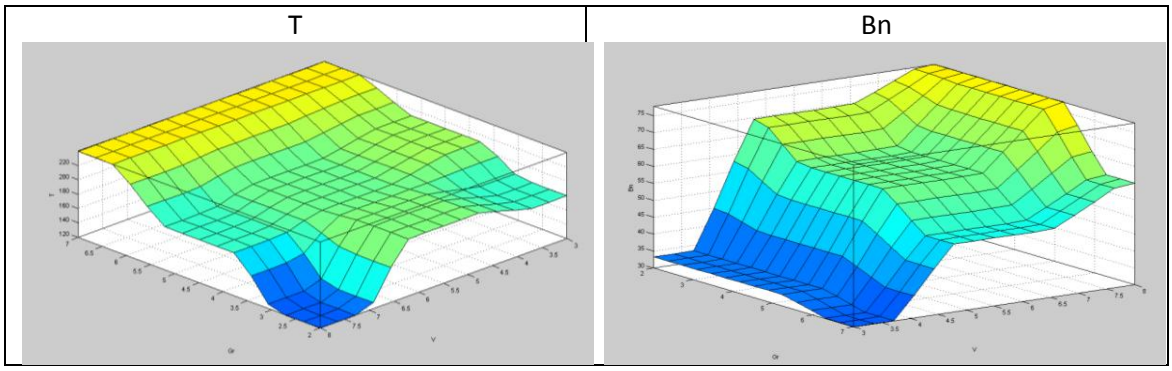


Figura 2.14

Como ocurría en las imágenes que mostraban las funciones de pertenencia, las superficies tampoco abarcan todo el rango correspondiente a cada universo. En ambos controladores, el universo "Grasas" va desde 2 a 7 y el universo "Velocidad" va desde 3 a 8. Como se ha mencionado antes, esto sucede porque no se disponen de datos para todo el universo de entradas. Si nos fijamos en las tablas 1.2 y 1.4 proporcionados por el usuario experto 1 del apartado 1.7 relativo al Controlador Neuroborroso, se aprecia que solo se dispone de datos para entradas del centro de la tabla. Por lo tanto, Matlab excluye todos los valores que estén fuera del rango de los datos de entrenamiento.

2.3.1.- Modificación de las funciones de pertenencia

Ahora se va a comparar el controlador generado que excluye los datos de los extremos, es decir, el modelo en el que el universo "Grasas" va desde 2 a 7 y el universo "Velocidad" que va desde 3 a 8, con uno que ha sido modificado de modo que ambos universos estén definidos entre 1 y 9, ya que los valores de las entradas van a estar en ese rango.

Para realizar el controlador modificado, se comienza del mismo modo que antes, generando un FIS con el comando "genfis1". Este genera unas MFs iniciales repartidas de forma equidistante, en las cuales se excluyen los valores que no existen datos.

Una vez realizado el FIS, se definen los rangos de las entradas Gr y V de forma que estén entre 1 y 9. Después se fuerza a la función de pertenencia del conjunto "Baja" a que incluya en su rango los valores excluidos que debieran pertenecer a este conjunto. Del mismo modo, se fuerza a que la función de pertenencia del conjunto "Alta" incluya en su rango los valores excluidos que debieran pertenecer a este conjunto.

Las figuras que vienen a continuación muestran las funciones de pertenencia resultantes de la modificación explicada en el anterior párrafo.

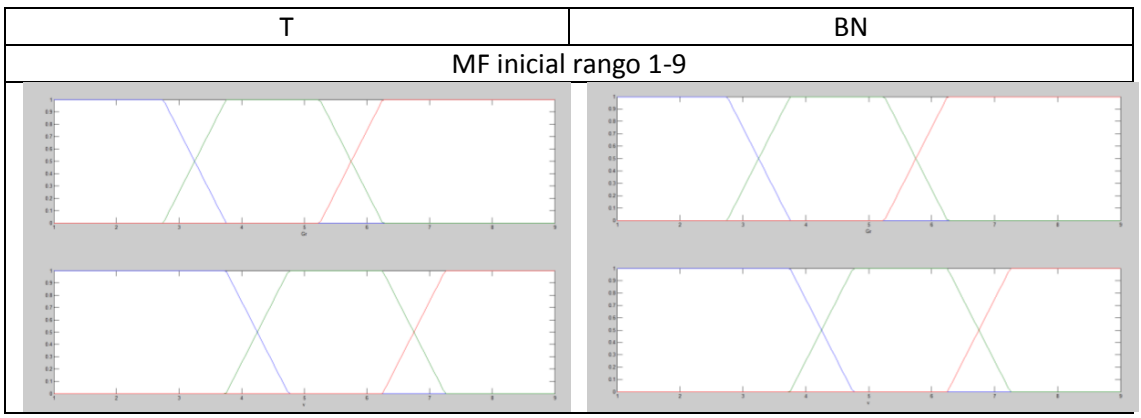


Figura 2.15

Una vez modificadas las funciones de pertenencia, resulta que al realizar la optimización, el programa vuelve a definir el rango de las grasas entre 2 y 7 y el de la velocidad de 3 a 8. Entonces, para solucionar esto, se vuelve a realizar el proceso anterior de expansión de los universos de entradas para que estén definidos de 1 a 9.

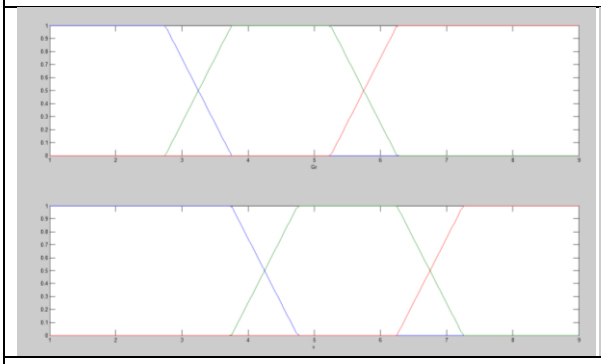
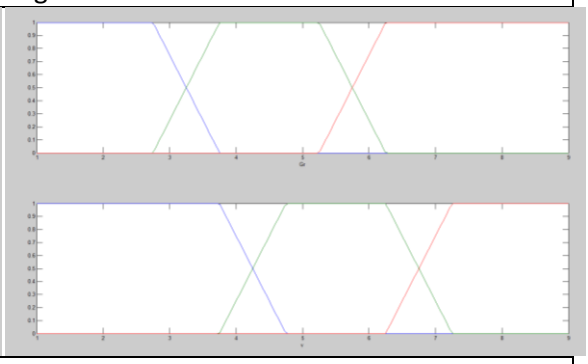
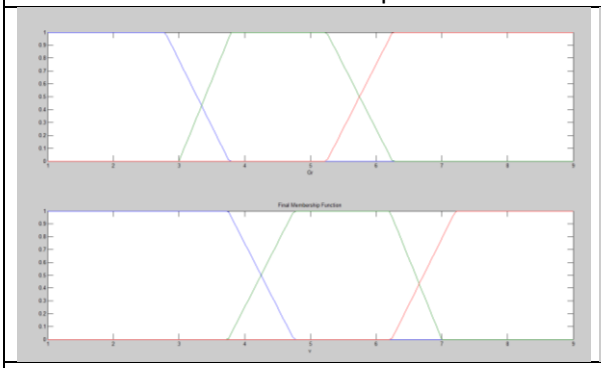
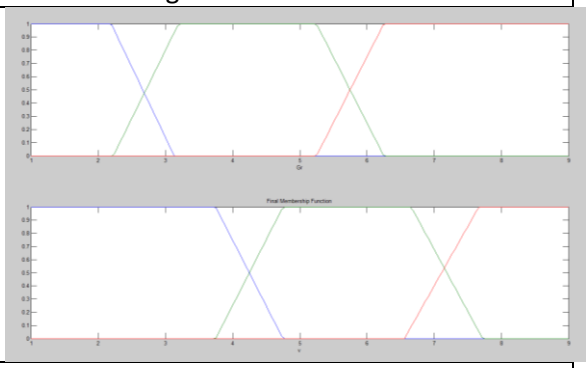
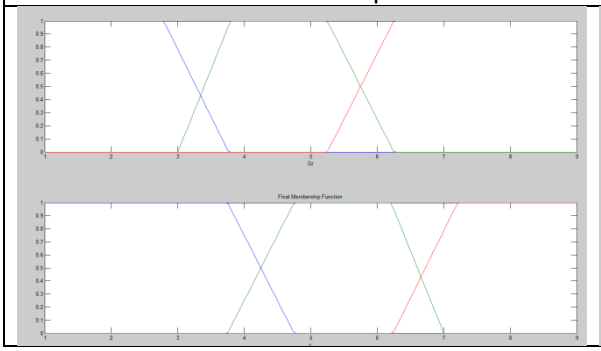
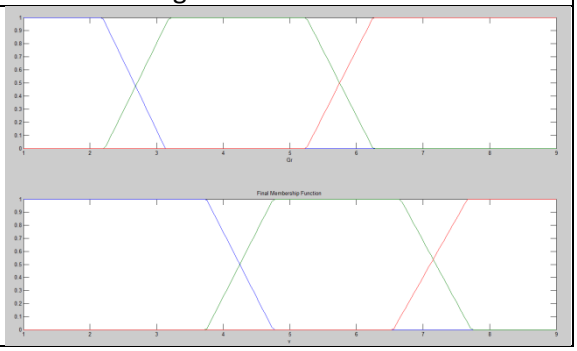
T	BN
MF inicial rango 1-9	
	
Después de 100 iteraciones con rango 1-9	
	
Después de 5000 iteraciones con rango 1-9	
	

Figura 2.16

Para finalizar, se quiere ver si existen diferencias entre el modelo que se obtiene de Matlab el cual no abarca todo el rango de entradas y el modelo que ha sido modificado de modo que sí que tiene definido todo el espectro de entradas posibles, es decir, que ambos universos estén definidos entre 1 y 9.

La siguiente tabla muestra los valores de salida que se obtienen en el modelo sin modificar y con el modificado.

T	FIS sin modificar	FIS modificadas
output= evalfis([1 1],fismat1T)	=180	=180
output= evalfis([1 9],fismat1T)	=120	=120
output= evalfis([9 1],fismat1T)	=180	=240
output= evalfis([9 9],fismat1T)	=180	=240

Tabla 2.6

Bn	FIS sin modificar	FIS modificadas
output= evalfis([1 1],fismat1Bn)	=53.89	=33.3
output= evalfis([1 9],fismat1Bn)	=77.7799	=77.7799
output= evalfis([9 1],fismat1Bn)	=53.89	=30
output= evalfis([9 9],fismat1Bn)	=53.89	=59.9999

Tabla 2.7

Realizando pruebas de evaluación de los puntos que ha obviado el modelo sin modificar, se ve que no siempre realiza bien la tarea de extrapolación, siendo correctas las que se obtienen con el modelo modificado. Por ello, se ve necesario realizar las modificaciones antes descritas en las funciones de pertenencia.

A continuación se muestran las superficies que se obtienen con el modelo sin modificar y con el modelo modificado.

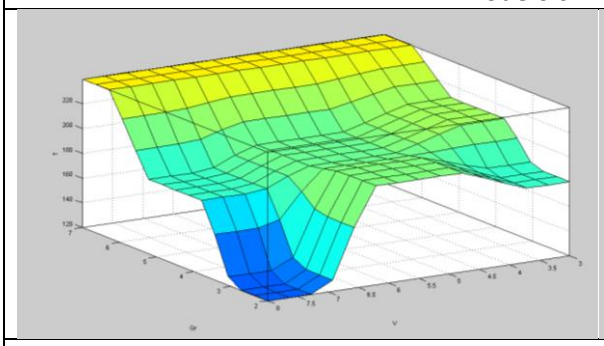
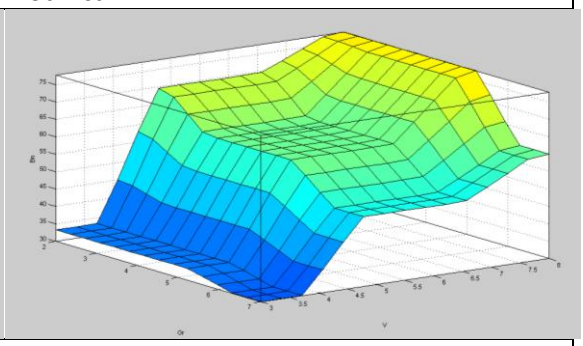
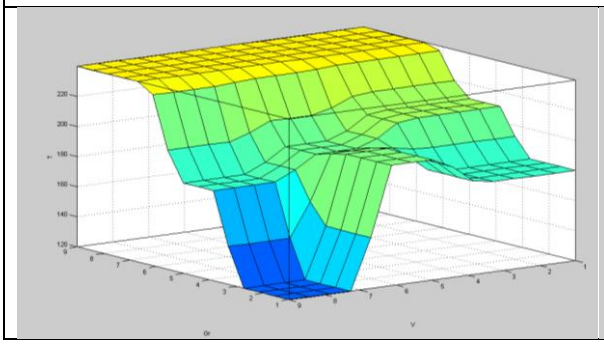
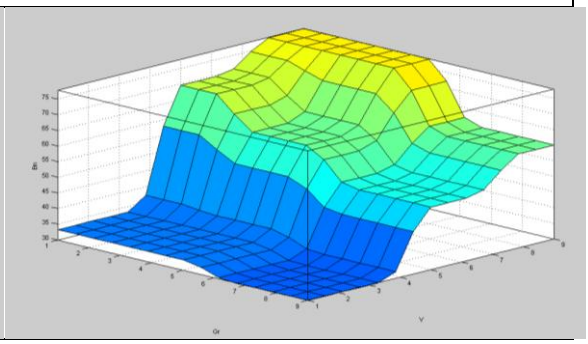
T	BN
Modelo sin modificar	
	
Modelo modificado	
	

Figura 2.17

Observando las figuras de las superficies de la parte superior, se puede ver que lo único que ha cambiado es que en los puntos donde no existen datos, Matlab ha expandido la superficie de los puntos aledaños.

A continuación se comprueba cómo se comporta con los datos disponibles, si en lugar de realizar un sistema de tres antecedentes borrosos para Gr y otros tres para V, el sistema pasa a ser de 5x5.

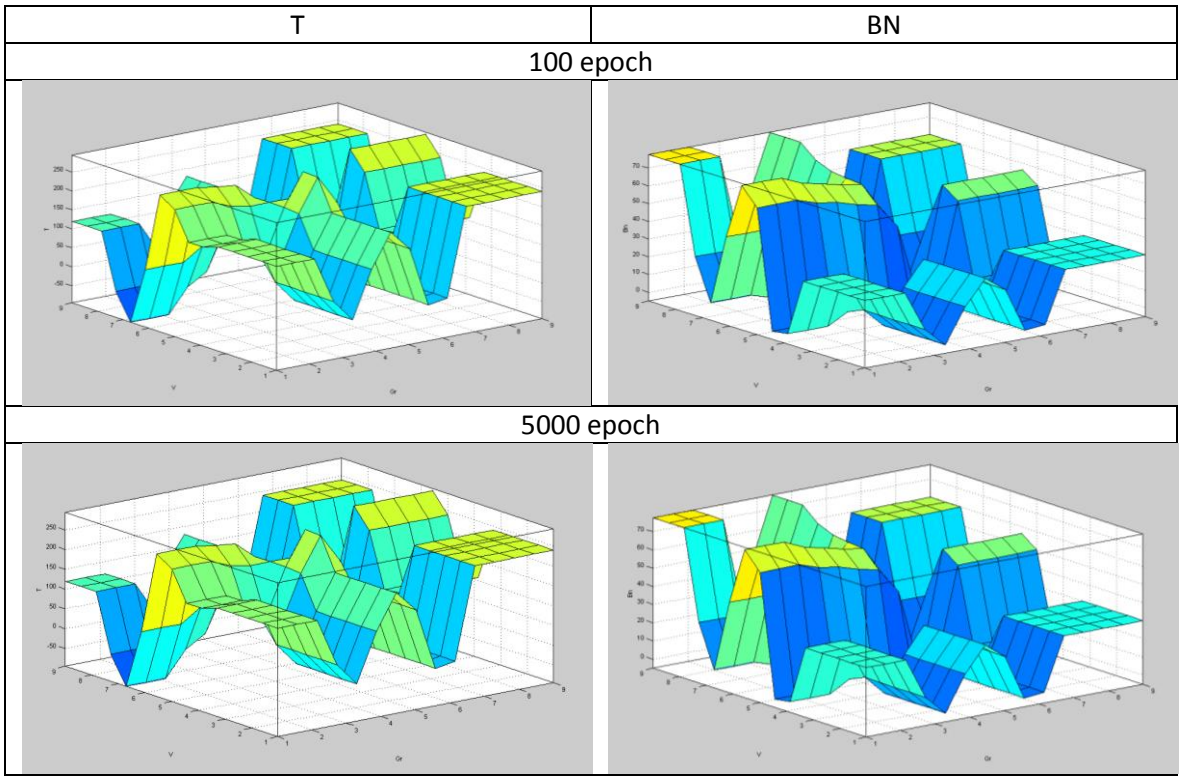


Figura 2.18

Se puede apreciar en las superficies que Matlab no realiza bien el ajuste porque le faltan muchos datos. En lugar de extrapolar los valores que no se le han dado se queda en su valor inicial, generado por el comando genfis1, que asigna valores de salida iniciales de cero. No es cuestión de cuantas iteraciones se hagan, ya que con 100 y 5000, ni la superficie ni el error cambian, sino de la cantidad de datos que se tiene para el ajuste del sistema.

2.3.2.- Conclusiones

El ajuste del controlador tanto de BN y de T realizado en el trabajo fin de carrera previo es muy parecido al obtenido con aprendizaje desde los datos del usuario experto 1.

Se ve la necesidad de modificar las funciones de pertenencia generadas por defecto ya que, si no se modifican, para las entradas que estén fuera del rango no se obtendrán salidas satisfactorias. Lo ideal es que se tengan datos representativos para las diferentes situaciones.

En caso de faltar un conjunto de datos próximos, para sistemas borrosos de inferencia generados mediante “genfis1”, Matlab no extrapola con valores adyacentes a estos y los deja en cero.

2.4.- Entrenamiento desde UE1 a usuario nuevo

Mediante este ejercicio se desea estudiar el comportamiento de MATLAB cuando se tiene un ajuste inicial válido para una persona y se quiere que se adapte a otro usuario nuevo introduciendo datos de la segunda persona.

Esto se ha realizado entrenando con datos del nuevo usuario UE2, tomando como FIS base el FIS de UE1.

Para emular el segundo usuario, se dispone de una tabla de datos para las situaciones más representativas para T y Bn.

T		V		
		Baja	Media	Alta
Gr	Baja	120	120	60
	Media	165	165	105
	Alta	210	210	150

Tabla 2.8; Usuario Experto 2

Bolus N		V		
		Baja	Media	Alta
Gr	Baja	70	70	70
	Media	65	70	65
	Alta	60	75	80

Tabla 2.9; Usuario Experto 2

Con estos datos se han generado dos controladores borrosos de 9 reglas cada uno; un controlador para T y el otro para Bn. En este apartado, estos controladores se usarán para emular los datos que introduce el usuario 2 mediante el uso del comando evalfis[(entrada1, entrada2),FIS UE2].

Del mismo modo, también se dispone de la tabla de los consecuentes del usuario 1. Se ha decidido usar la tabla de consecuentes que se obtuvo en el TFC porque tiene el visto bueno del usuario experto 1.

Los datos del usuario experto 1 se muestran en las tablas 2.10 y 2.11**.**

Bolus N		V		
		Baja	Media	Alta
Gr	Baja	33.3	56	77.8
	Media	33.3	50	66.7
	Alta	30	44.4	60

Tabla 2.10; Usuario Experto 1

T		V		
		Baja	Media	Alta
Gr	Baja	180	180	150
	Media	210	210	180
	Alta	240	240	210

Tabla 2.11; Usuario Experto 1

Al igual que con la tabla de UE2, se han realizado un par de controladores para T y Bn de 9 reglas cada uno con los datos de UE1.

Una vez realizados los diseño de los controladores, se generan los datos del usuario experto 2 mediante el comando evalfis y se inicia el entrenamiento con estos datos nuevos de UE2, tomando como FIS base (sistema borroso de inferencia) el sistema borroso del usuario1.

Inicialmente, se propuso definir la MF triangular perteneciente a grasas como un trapecio con los dos puntos superiores en un mismo punto [4 5 5 6] con la intención de dar libertad al programa para modificarlo a su antojo.

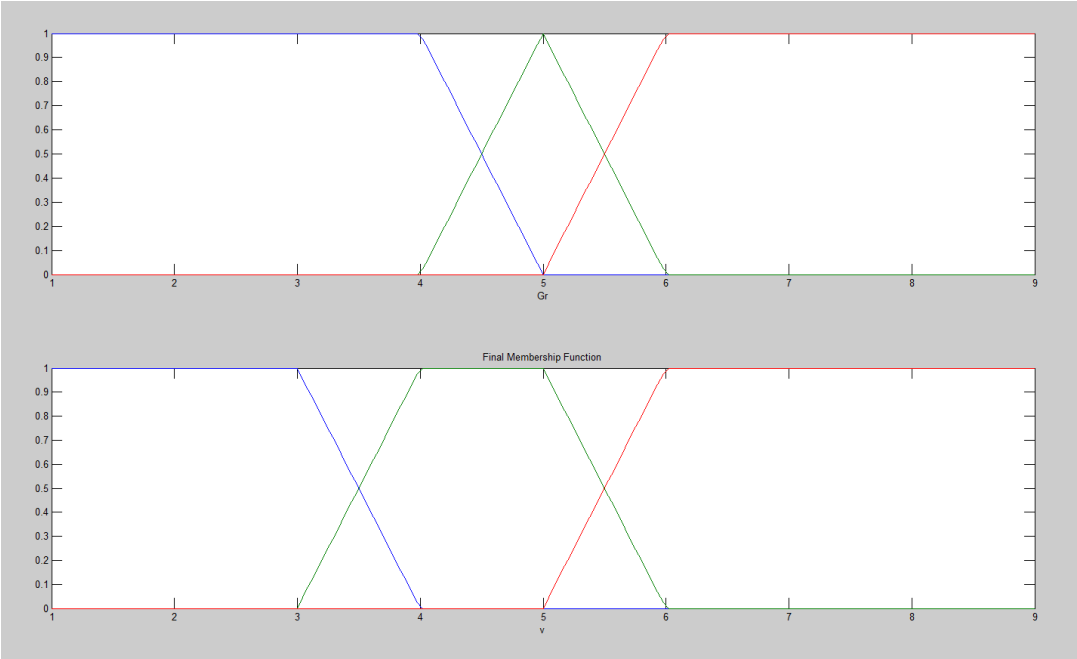


Figura 2.19

Como resultado se obtuvo que en el controlador de T, Matlab superponía el punto c sobre el b y no podía continuar con el aprendizaje.

a = 4.001613, b = 5.000001, c = 5.000000, d = 5.997504

Error using anfismex

Illegal parameters in fisTrapezoidMf() --> b > c

Por ello, en dicho controlador, el conjunto perteneciente a grasas “Medio” se definió como triangular y se realizó el posterior entrenamiento para ver su evolución.

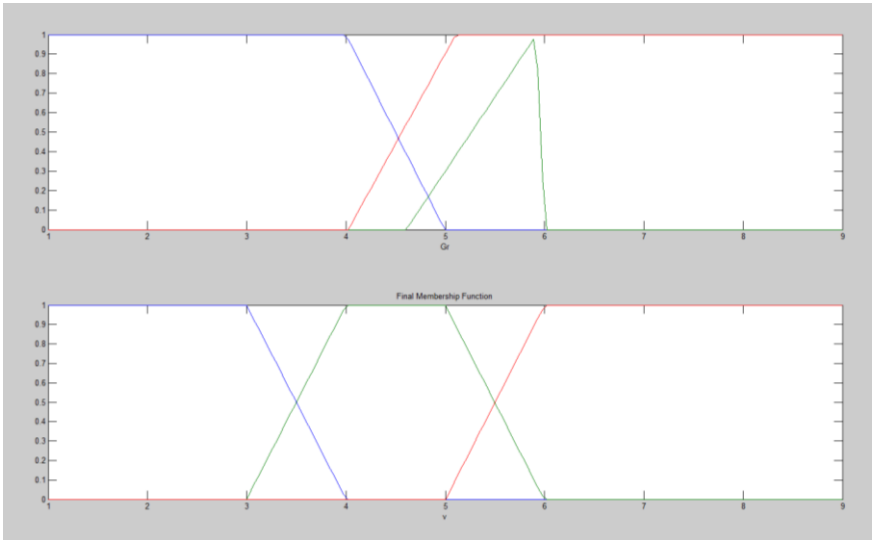


Figura 2.20; MF despues de 1000 iteraciones

En las imágenes superiores se ve como, para el sistema neuroborroso de T, después de 1000 iteraciones evoluciona a un sistema 2x3.

Por el contrario, el controlador de Bn es capaz de modificar correctamente las funciones de pertenencia con un trapecio con los puntos superiores en el mismo punto. Este sistema se transformaba en un sistema 2x3.

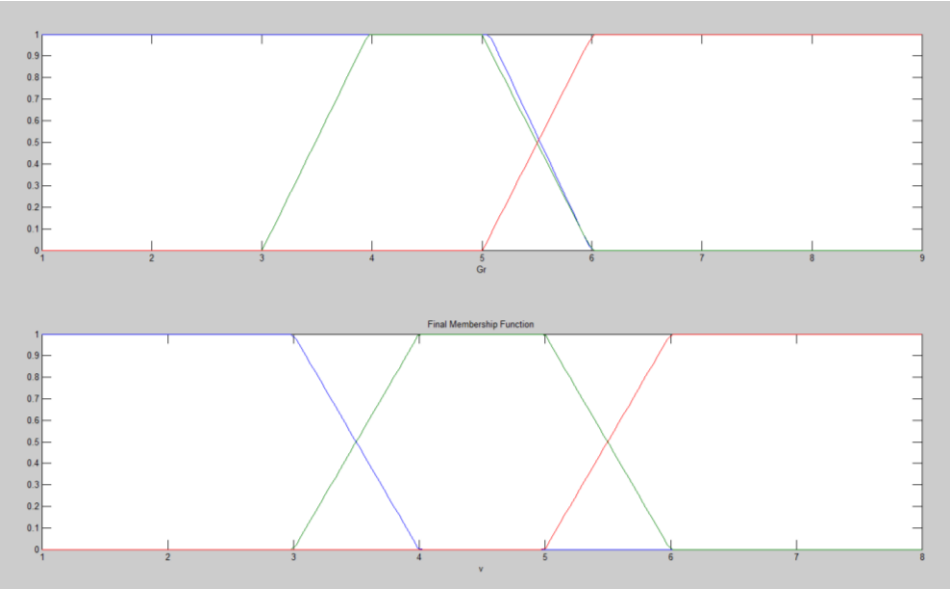


Figura 2.21; 500 iteraciones controlador BN

Ocurre lo mismo si en lugar de un trapecio, se define como triangulo.

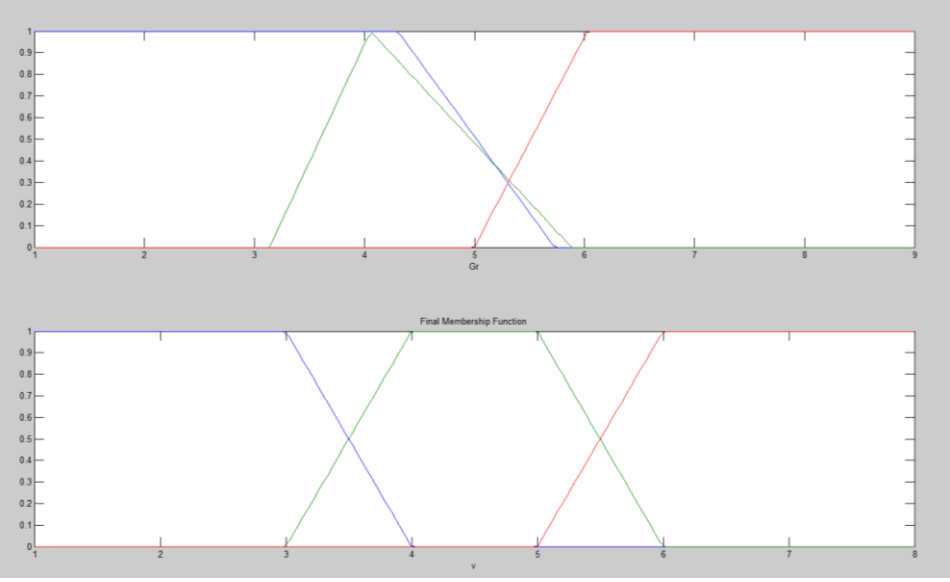


Figura 2.22

Para dar más libertad al programa, se volvieron a definir los conjuntos como trapecios pero se distribuyeron de forma equidistante.

En los siguientes experimentos se quiere ver cuantos datos se necesita y ver cómo evoluciona el sistema. Esto se ha realizado de dos modos diferentes:

- Realizar el aprendizaje únicamente con 9 datos nuevos representativos del UE2.
- Realizar el aprendizaje con datos nuevos generados aleatoriamente del UE2.

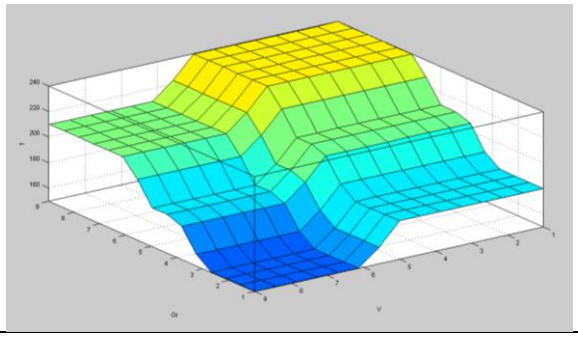
2.4.1..- Resultados para controlador de tiempo de suministro de Bolus C (T)

2.4.1.1..- Aprendizaje con 9 datos representativas

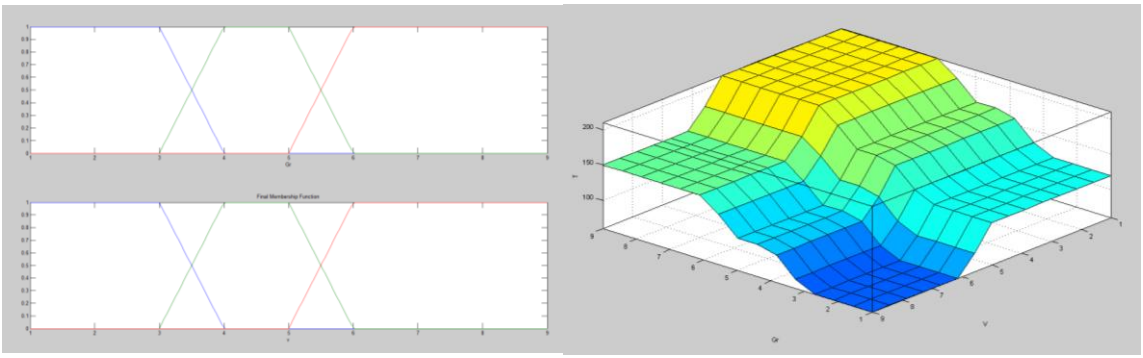
En este ejercicio se quiere ver si el programa es capaz de mover los parámetros de modo que dé una respuesta satisfactoria entrenando únicamente con 9 puntos representativos de los datos del usuario 2.

En las siguientes imágenes se pude ver la evolución de las superficies desde la del FIS inicial correspondiente al usuario 1 hasta la obtenida con los 9 datos del usuario 2 después de entrenar con ellos.

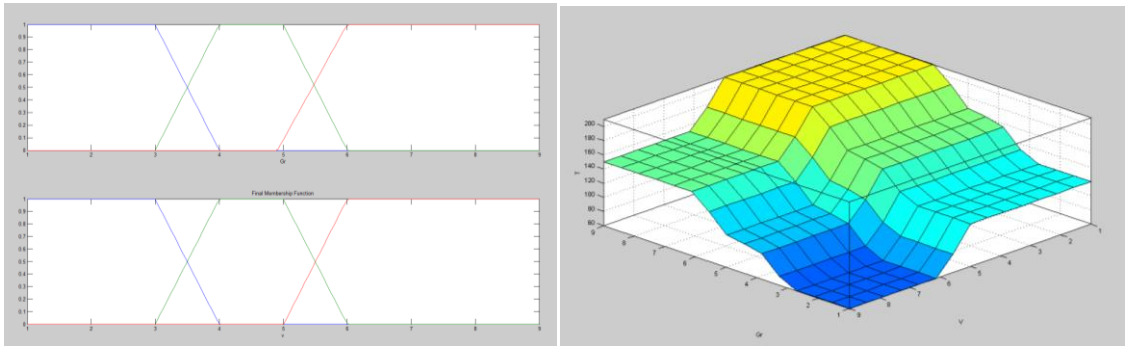
Surface inicial UE1



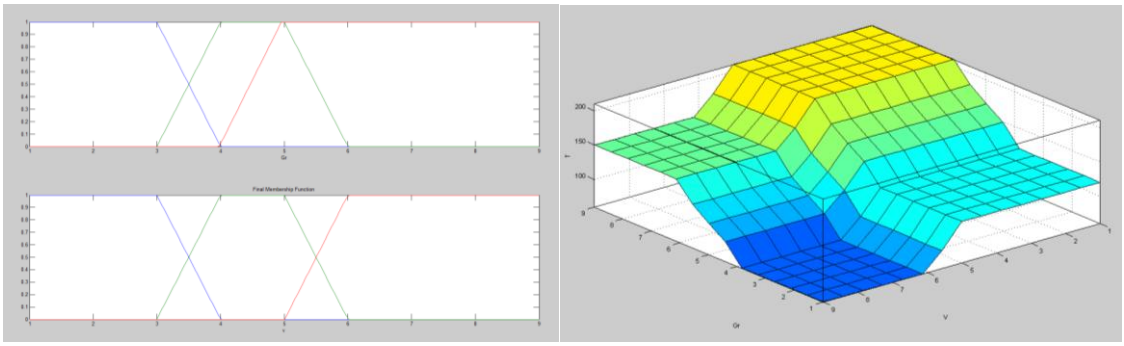
Una iteración



10 iteraciones



100 iteraciones



UE2 T

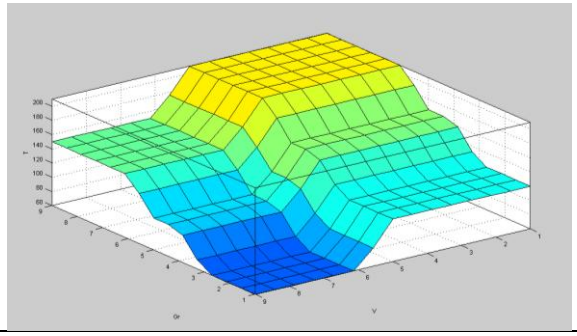


Figura 2.23

Como se aprecia en las imágenes, en una sola iteración el sistema ha movido todos los puntos y se ha transformado en la superficie del usuario 2. Respecto a las funciones de pertenencia, hasta que no se realizan 100 iteraciones, el sistema apenas mueve dichas superficies.

A continuación se muestran las tablas de los consecuentes para cada iteración.

T		V		
		Baja	Media	Alta
Gr	Baja	119.9999	119.9999	59.9999
	Media	164.9998	164.9998	104.9999
	Alta	209.9998	209.9998	149.9998

Tabla 2.11; 1 iteración

Los consecuentes para 10 iteraciones en adelante no se modifican.

En la siguiente figura se muestra la evolución del error en función del número de iteraciones.

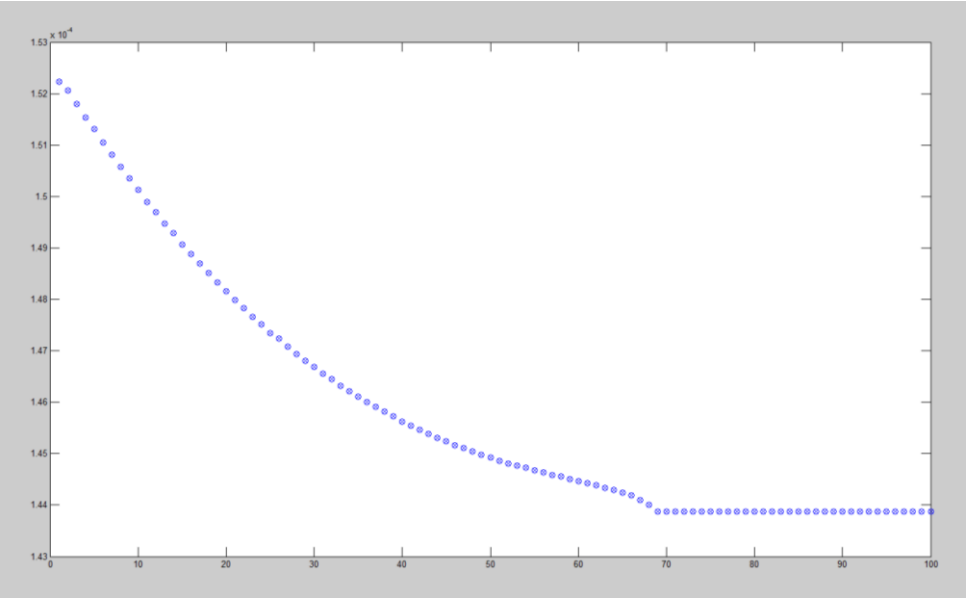


Figura 2.24

1 iteración	trnErrT = 1.5223e-04
10 iteraciones	trnErrT = 1.0e-03 *0.1501
100 iteraciones	trnErrT = 1.0e-03 *0.1439

Tabla 2.12

Desde un inicio, el error es muy pequeño. Esto significa que con nueve puntos, el sistema se ajusta al nuevo usuario a la perfección.

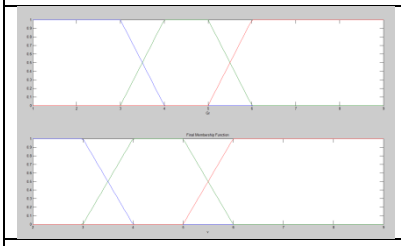
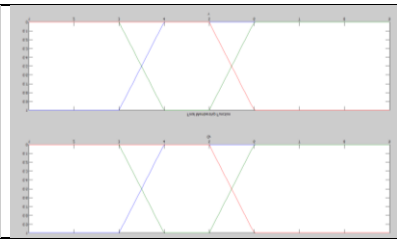
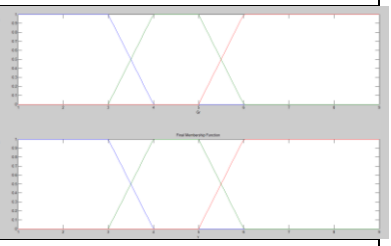
Comentarios

En una sola iteración y con solamente 9 puntos representativos, el sistema se ajusta al nuevo usuario. Esto ocurre porque en el aprendizaje se tienen muchos parámetros libres para poder mover 9 puntos. Por el contrario, cuantos más puntos se tengan, más difícil es mover los puntos de modo que el sistema quede ajustado.

2.4.1.2.-Aprendizaje con datos nuevos generados aleatoriamente

En este ejercicio se desea analizar el número de datos de entrenamiento con los que el sistema es capaz de llegar al nuevo usuario.

En las imágenes inferiores se pueden ver la evolución de las funciones de pertenencia para diferentes números de datos y en diferentes iteraciones.

20 datos de UE2	50 datos de UE2	1000 datos de UE2
1 iteración		
		
10 iteraciones		

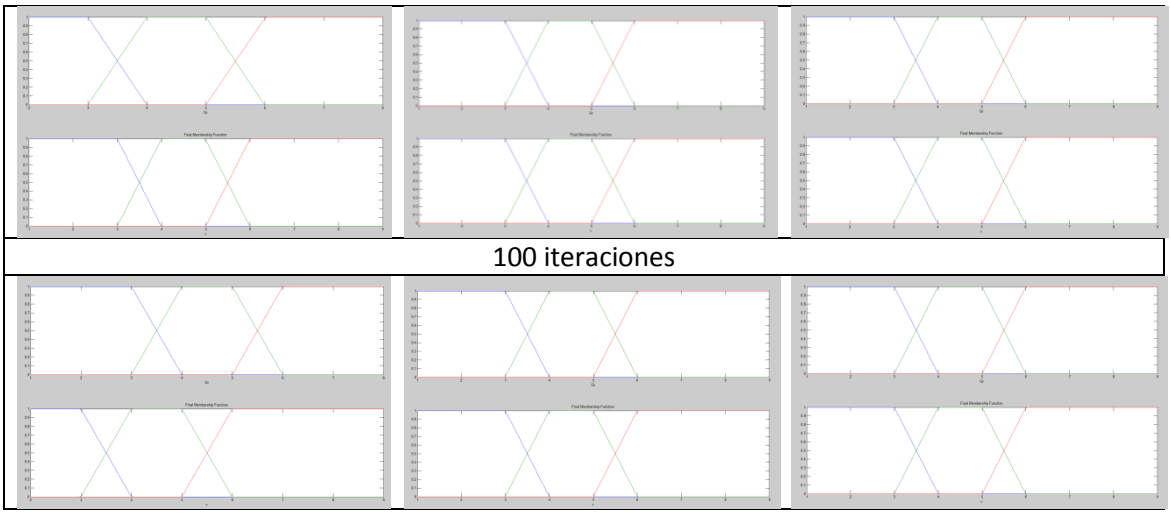
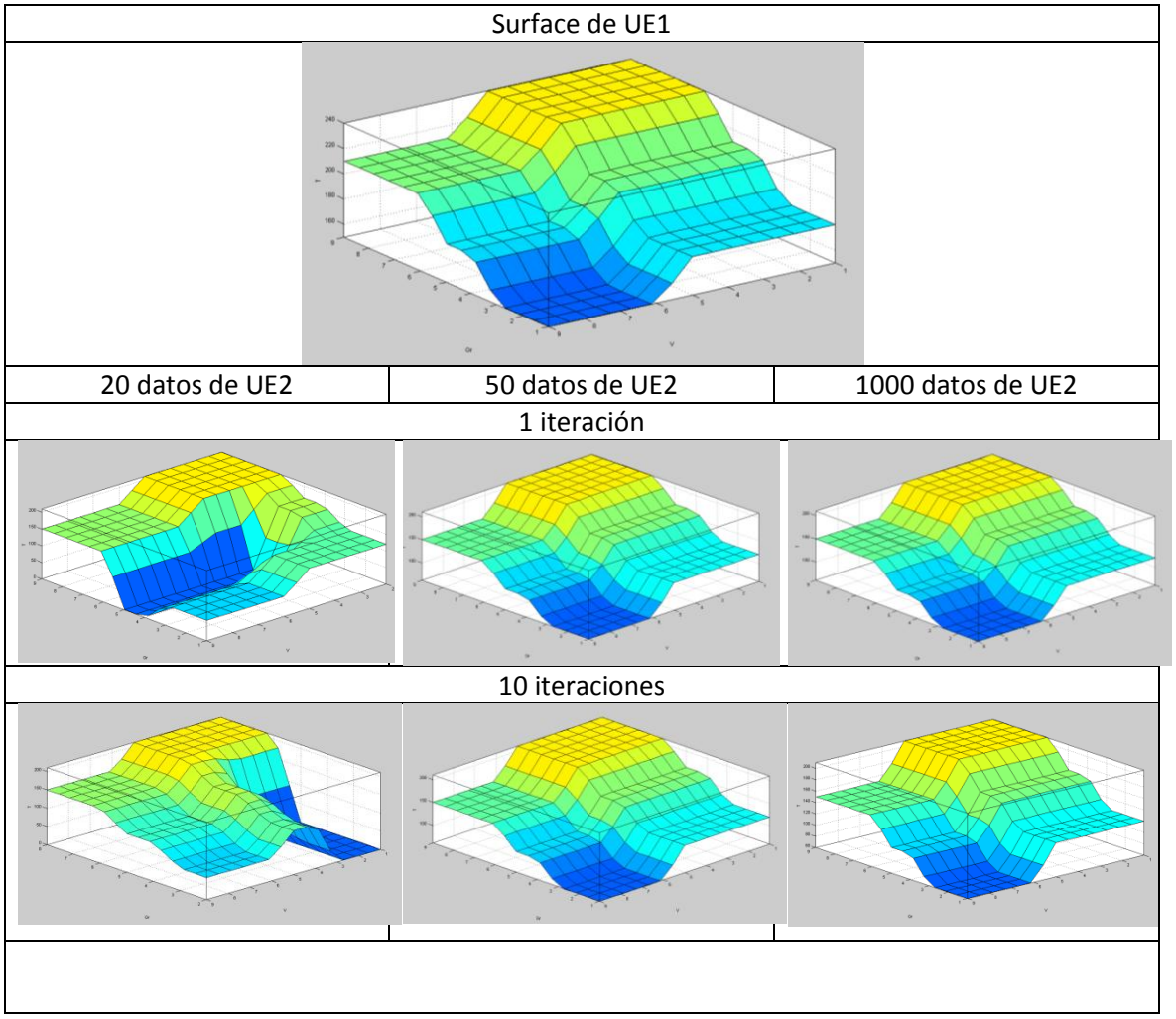


Figura 2.25

En lo que respecta a las funciones de pertenencia, se ve que el sistema no cambia ni la distribución ni la forma de las MFs.

A continuación se analiza la evolución de las superficies con el número de iteraciones con diferentes cifras de datos de entrenamiento.



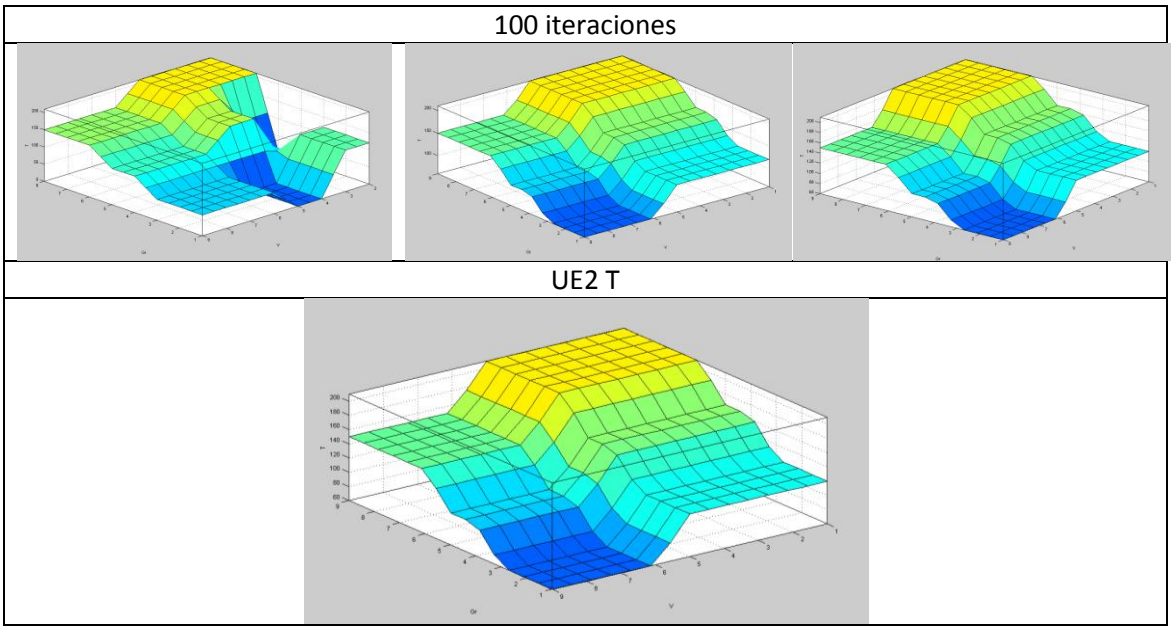


Figura 2.26

Como en el caso de 9 datos representativos, el sistema es capaz de evolucionar en una sola iteración a UE2. Para que el sistema sea capaz de transformarse, debe de haber una representación de datos de cada zona. Por ejemplo, con 20 datos, se puede ver que existen zonas donde la superficie va a cero. Esto es debido a que en esa zona no existían datos con los que realizar el entrenamiento.

En caso de haber suficientes datos representativos, da igual la cantidad de datos con los que se entrene, ya que Matlab es capaz de mover todos los puntos en una sola iteración.

En la siguiente tabla se muestra el error para los diferentes números de datos y a diferentes iteraciones.

20 datos de UE2	50 datos de UE2	1000 datos de UE2
1 iteración		
trnErrT = 5.9888e-05	trnErrT = 3.0153e-05	trnErrT = 1.4656e-06
10 iteraciones		
trnErrT = 0.0001	trnErrT = 0	trnErrT = 0

Tabla 2.13

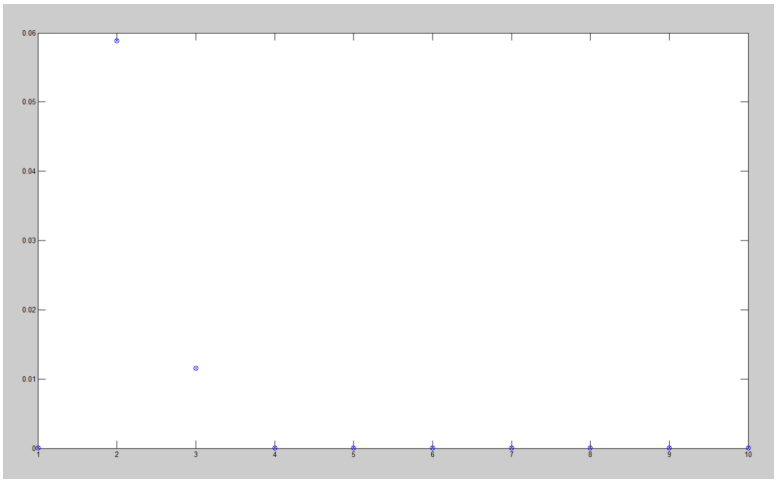


Figura 2.27

Estos son los consecuentes que se obtienen con la primera iteración y 50 datos.

T	V			
Gr		Baja	Media	Alta
	Baja	120	120	60
	Media	165	165	105
	Alta	210	210	150

Tabla 2.14

El error que se genera con respecto a los datos de entrenamiento es mínimo, por lo que se puede decir que el sistema se ajusta muy bien al usuario 2.

Comentarios

Si se dispone de suficientes datos con los que se abarque todas las zonas de los universos de entradas, con una sola iteración el sistema se ajusta con un error despreciable al usuario nuevo.

En caso de que falten datos, el sistema no aprende correctamente y no se ajusta al nuevo usuario.

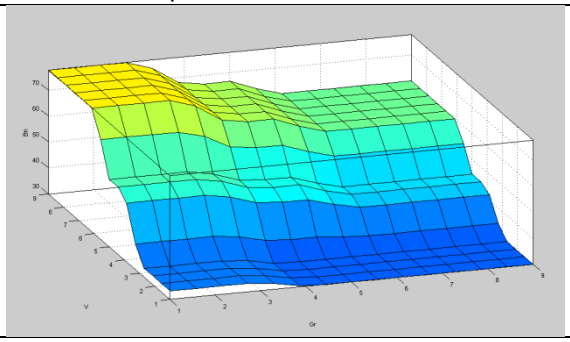
2.4.2.- Resultados para controlador de Bolus N (Bn)

2.4.2.1.- Aprendizaje con 9 datos representativos

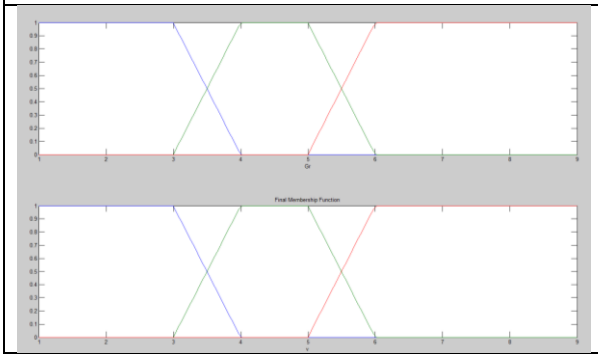
Al igual que en el ejercicio anterior se quiere ver si el programa es capaz de mover los parámetros de modo que dé una respuesta satisfactoria entrenando únicamente con 9 puntos representativos de los datos del usuario 2.

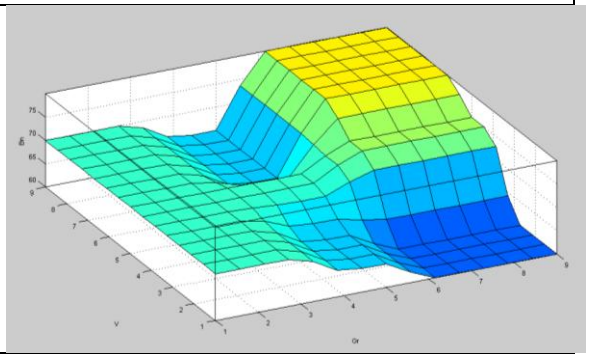
En las siguientes imágenes se pude ver la evolución de las superficies desde la del FIS inicial correspondiente al usuario 1 hasta la obtenida con los 9 datos del usuario 2 después de entrenar con ellos.

Superficie inicial de UE1



Una iteración





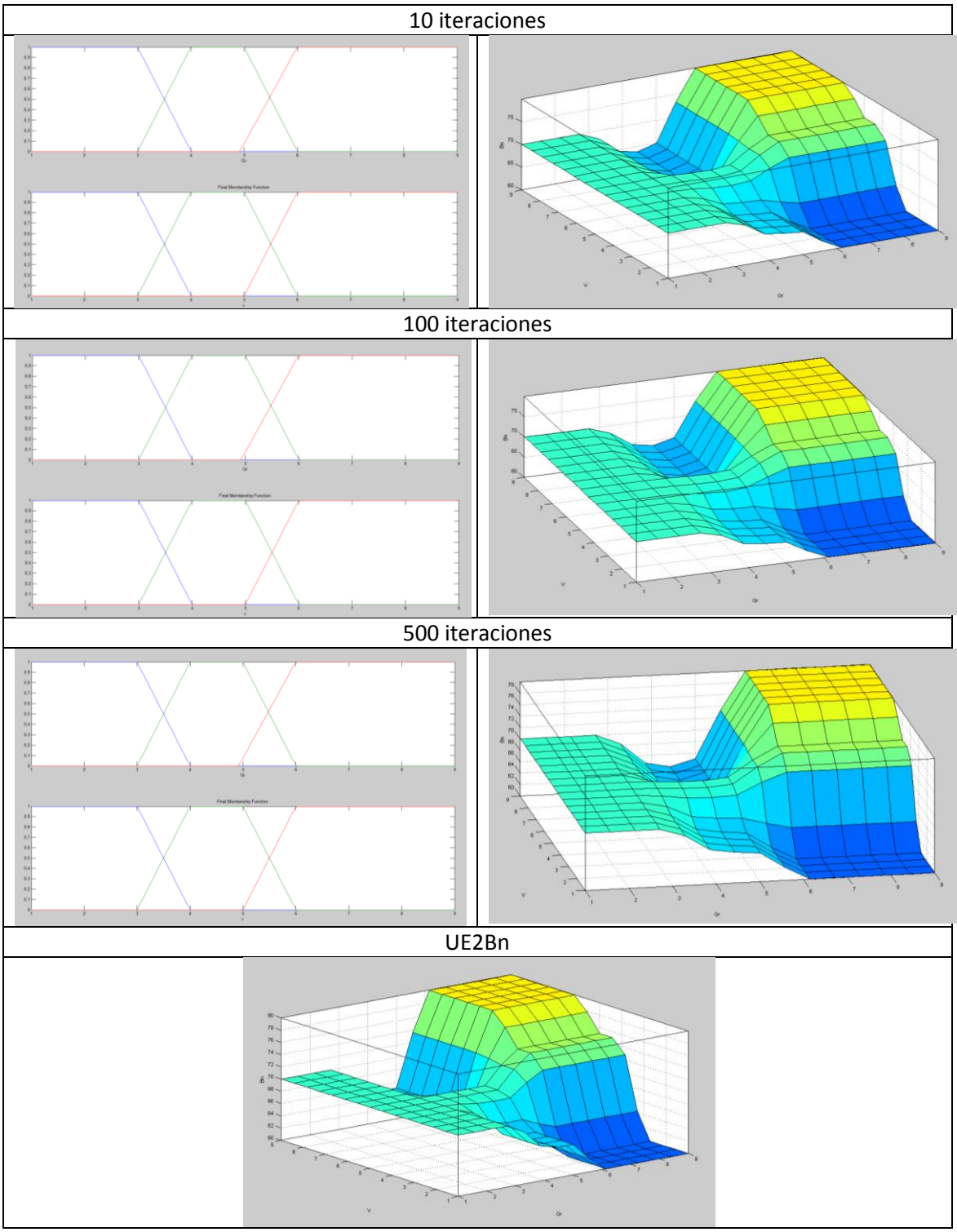


Figura 2.28

Tal y como ocurría con el controlador de T, en una sola iteración el sistema se ha ajustado al usuario 2.

Tabla de consecuentes

Bn		V		
		Baja	Media	Alta
Gr	Baja	69.9999	69.9999	69.9999
	Media	64.9999	69.9999	64.9999
	Alta	59.9999	74.9999	79.9999

Tabla 2.15; 1 iteración

Bn		V		
		Baja	Media	Alta
Gr	Baja	69.9999	69.9999	69.9999
	Media	65.3838	69.6166	63.8483
	Alta	59.9999	74.9999	79.9999

Tabla 2.16; 10 iteraciones

Con 100 y 500 iteraciones, el valor de los consecuentes es el mismo que con 10 iteraciones.

Error

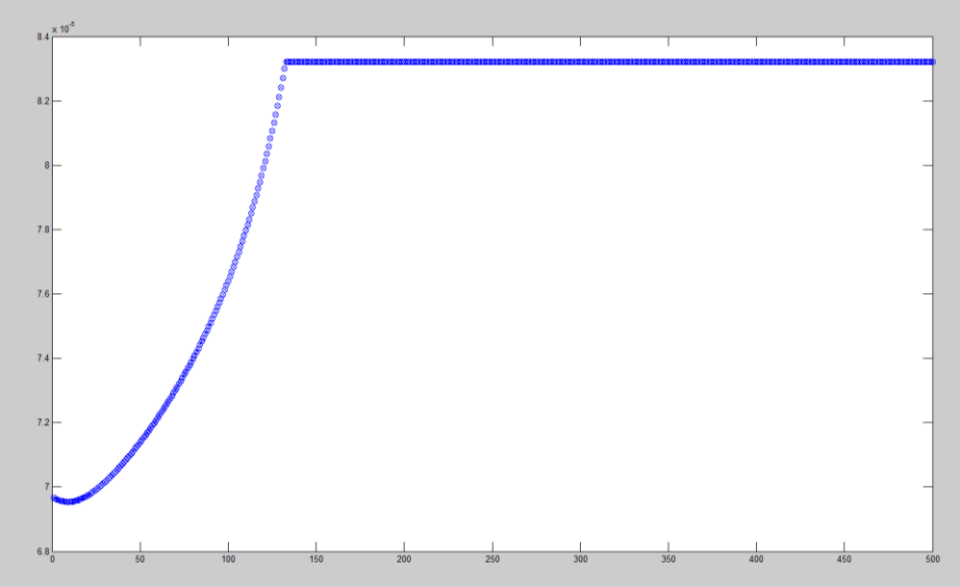


Figura 2.29

1 iteración	trnErrBn = 6.9662e-05
10 iteraciones	trnErrBn = 1.0e-04 * 0.6954
100 iteraciones	trnErrBn = 1.0e-04 *0.7641
500 iteraciones	trnErrBn = 1.0e-04 *0.8320

Tabla 2.17

En la tabla de los consecuentes se ve que en la primera iteración, el error es menor que en la de 10 iteraciones. Esto se ve también en la figura superior del error y en la tabla 2.17.

Comentarios

Nuevamente, con sólo 9 datos el programa de Matlab baja mucho en la primera iteración. De hecho, las siguientes iteraciones son, en todo caso perniciosas, porque el error sube aunque muy poco.

2.4.2.2.- Aprendizaje con datos nuevos generados aleatoriamente

En este ejercicio se desea analizar el número de datos de entrenamiento con los que el sistema es capaz de llegar al nuevo usuario.

En las siguientes imágenes se pueden ver la evolución de las funciones de pertenencia para diferentes números de datos y en diferentes iteraciones.

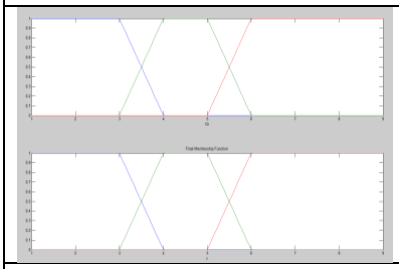
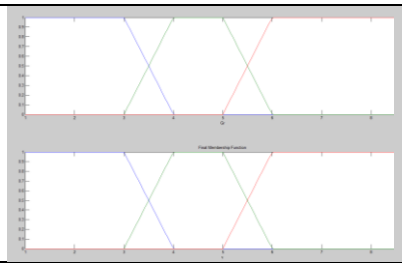
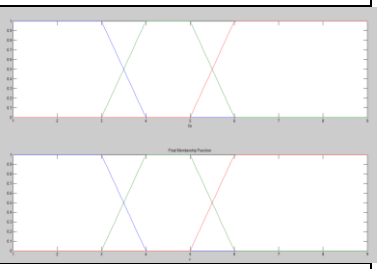
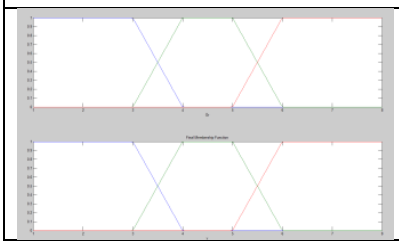
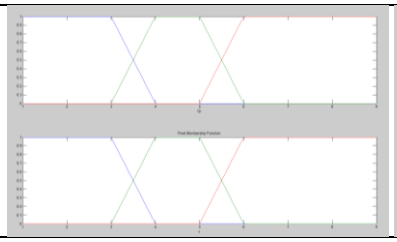
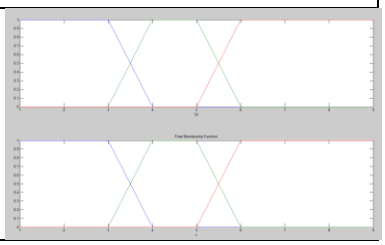
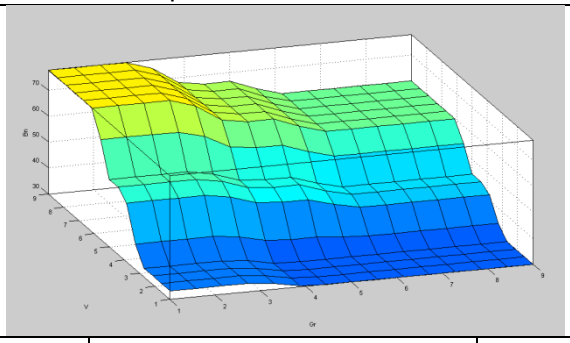
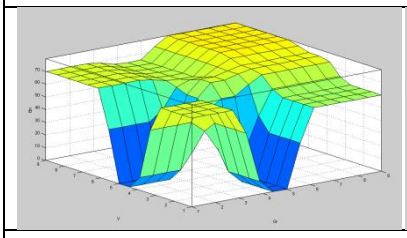
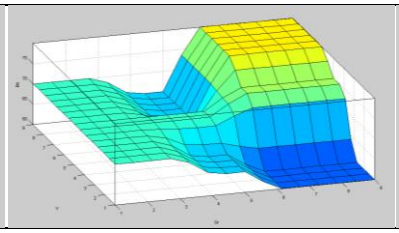
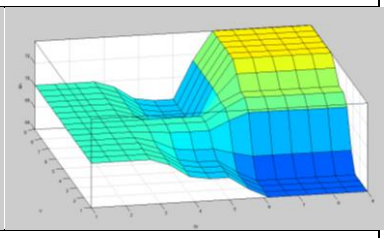
20 datos de UE2	50 datos de UE2	1000 datos de UE2
1 iteración		
		
100 iteraciones		
		

Figura 2.30

Se ve que por muchas iteraciones que se hagan, el programa no mueve las MFs.

Con las siguientes imágenes se muestra la evolución de las superficies a diferentes iteraciones y con distinto número de datos de entrenamiento.

Superficie inicial de UE1		
		
20 datos de UE2	50 datos de UE2	1000 datos de UE2
1 iteración		
		

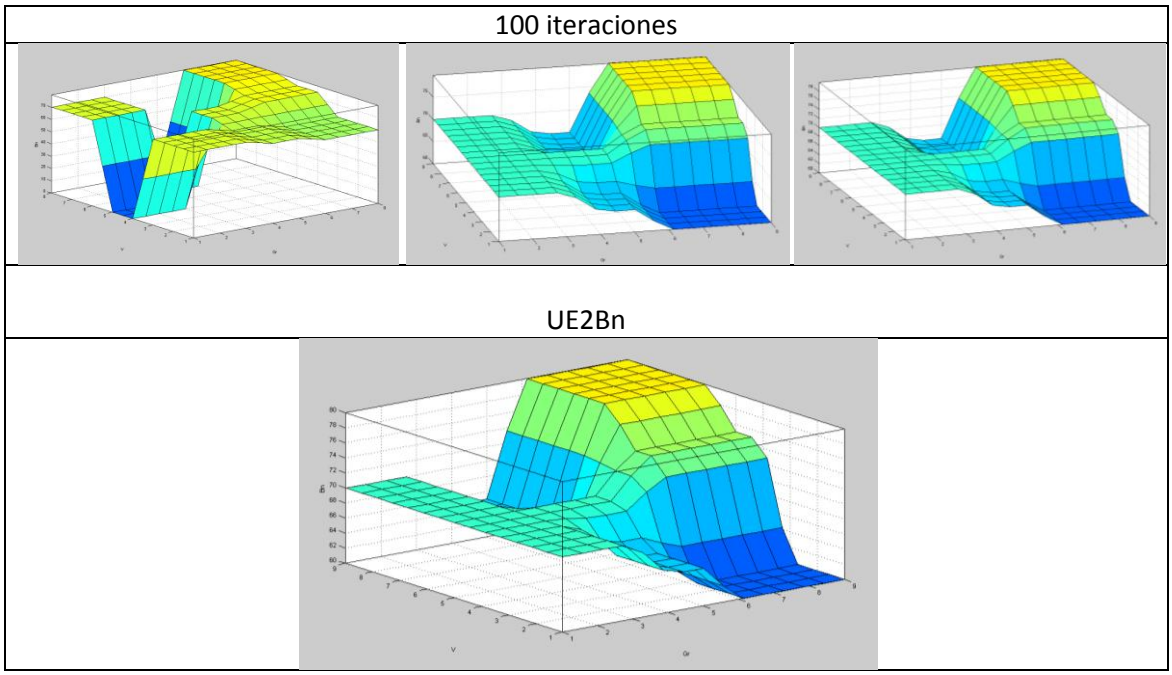


Figura 2.31

Al igual que ocurre en los anteriores ejercicios, Matlab se adapta perfectamente al usuario 2.

Error

20 datos de UE2	50 datos de UE2	1000 datos de UE2
1 iteración		
trnErrBn = 2.7605e-05	trnErrBn = 1.3864e-05	trnErrBn = 6.6265e-07
100 iteraciones		
trnErrBn = 2.3774e-05	trnErrBn =0	trnErrBn =0

Tabla 2.18

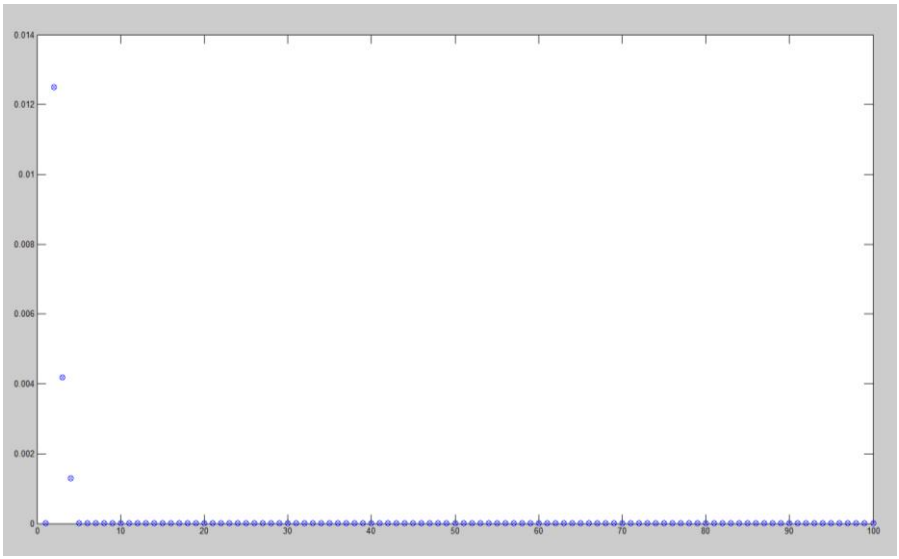


Figura 2.32

Esta es la tabla que se obtiene con la primera iteración y 50 datos.

Bn	V			
Gr		Baja	Media	Alta
	Baja	70	70	70
	Media	65	70	65
	Alta	60	75	80

Tabla 2.19

Comentarios

Como ocurre en los anteriores ejercicios, en una sola iteración el sistema mueve todos los puntos para transformarse a condiciones del nuevo usuario, aunque sean completamente distintos.

En caso de no tener datos suficientes, el programa no se ajusta bien al nuevo usuario.

2.4.3.- Conclusiones

Partiendo de un controlador previamente diseñado, con muy pocos datos y en una sola iteración, Matlab realiza la conversión de UE1 a UE2 con un error mínimo, independientemente de lo diferente que sean entre ellos.

Con muchos datos de entrenamiento Matlab no tiene ninguna dificultad de mover todos los puntos y adaptarse a las condiciones del nuevo usuario en una sola iteración.

Se necesitan datos representativos que abarquen todo el rango de entradas, si no, no se realiza bien el ajuste.

2.5.- Transformación de UE1 a UE2

En esta última prueba, se desea estudiar cuantos datos son necesarios añadir del nuevo usuario en la matriz de datos a entrenar, para satisfacer los requerimientos de insulina del nuevo usuario.

Al inicio, los únicos datos de los que se dispone en la matriz de datos son los del usuario 1. Para comenzar, se genera un FIS con el comando “genfis1” y se definen las características de las funciones de pertenencia. Después, con los datos de la matriz de datos se entrena y obtiene un sistema borroso de inferencia nuevo. Como los únicos datos con los que se ha entrenado son los del usuario 1, es obvio que los resultados que se obtienen son validos únicamente para este usuario.

Mientras tanto, mediante un FIS del usuario 2 y con el comando “evalfis” se han generado datos del usuario 2.

Para ver cómo evoluciona el sistema desde el FIS inicial generado con los datos de UE1 hacia el usuario dos, se van introduciendo datos del UE2 poco a poco en la matriz de datos. Por lo tanto, esta matriz contendrá datos del usuario 1 y del usuario 2.

Una vez introducidos los datos de UE2, esta matriz de datos se usa como datos de entrenamiento. Entonces se vuelve a entrenar tomando como FIS de base el FIS generado que únicamente era válido para el usuario 1 y los datos a entrenar serán los de la matriz de datos, compuesta por datos del UE1 y los datos introducidos del UE2. Este nuevo FIS será una mezcla entre UE1 y UE2.

Cada vez que se entrena el sistema, se le introducen poco a poco nuevos datos del nuevo usuario a la matriz de datos y se vuelve a entrenar tomando esta matriz como datos de entrenamiento y como FIS base, el FIS anterior de entrenamiento. Es decir, el FIS que se

genera se sobrescribe cada vez que se entrena con los nuevos datos que se le introducen del nuevo usuario.

En este trabajo, se ha decidido introducir datos del UE2 en la matriz de datos de diez en diez. Dicho de otra manera, cada vez que se realiza el entrenamiento, se le introducen a la matriz de datos 10 datos nuevos del usuario 2.

Otra parte de este apartado ha sido estudiar la evolución del sistema mediante la metodología explicada en los párrafos anteriores, pero, en lugar de únicamente añadir datos del usuario 2, en caso de que coincidan las entradas de los datos del usuario nuevo y los existentes previamente en la matriz de datos, estos serán sustituidos por los consecuentes que se introduzcan del nuevo usuario. En otras palabras, se sustituyen los datos viejos de la matriz de datos por los nuevos del usuario 2. De igual modo, cada vez que se entrena el sistema, se sobrescribe el anterior y se le vuelven a introducir y/o sustituir los nuevos datos.

2.5.1.- Aprendizaje sin sustitución

Como se ha dicho, se trata de ver la evolución del sistema cuando se le añaden nuevos datos del nuevo usuario a los existentes y se entrena con esta matriz de datos. Los nuevos datos y los viejos conviven en la misma matriz y por ello, se quiere ver cómo evoluciona.

Respecto al número de iteraciones, realizando pruebas se ha comprobado que a partir de 100 iteraciones, el error se estabiliza, por lo que se ha realizado la prueba con 150 iteraciones.

En las siguientes hojas se mostraran y analizaran los resultados obtenidos para los dos controladores T y Bn.

2.5.1.1.- Controlador de tiempo de suministro de Bolus C (T)

En esta prueba se trata de averiguar cuantos datos se deben introducir para que el sistema evolucione desde el usuario inicial UE1 hasta que el sistema se adapte a las necesidades del nuevo usuario 2. Para ello, como se ha explicado entes, se ha partido de unos datos válidos para el usuario 1 y se han ido añadiendo datos del UE2 cada 10 y se ha realizado el aprendizaje.

En un principio, parece más lógico que, cuantos más datos de UE1 se dispongan, más tardará el sistema en transformarse. Por ello, se ha realizado la prueba de dos formas diferentes; en una se disponen de 16 datos, suficientes para que al inicio Matlab genere el FIS al completo, es decir, sin ningún antecedente en cero, y otro sistema donde se tengan 100 datos de UE1, donde, a priori, cueste más que el sistema evolucione.

En la siguiente tabla se muestra cómo cambian los consecuentes cuando se le introducen datos cada 10.

Partiendo desde un FIS generado con 16 datos representativos de UE1				Partiendo desde FIS generado con 100 datos de UE1
0 nuevos datos (UE1)				
T	V			
Gr		Baja	Media	Alta
	Baja	180	180	77.8
	Media	210	210	180
	Alta	240	240	210

10 nuevos datos

T	V				T	V		
Gr		Baja	Media	Alta	Gr		Baja	Media	Alta
	Baja	175.3907	172.8510	86.2222		Baja	172.5692	150.7618	145.1355
	Media	203.6196	221.9294	164.7121		Media	209.7561	210.8505	179.5589
	Alta	239.9998	232.7777	205.2915		Alta	239.0981	244.0205	201.0622

20 nuevos datos

T	V				T	V		
Gr		Baja	Media	Alta	Gr		Baja	Media	Alta
	Baja	154.2369	144.8469	118.0424		Baja	161.8129	146.1596	140.7734
	Media	241.8179	209.4471	154.0717		Media	210.1348	205.6249	179.9122
	Alta	222.0000	169.9999	209.9998		Alta	238.7084	236.8431	197.7113

30 nuevos datos

T	V				T	V		
Gr		Baja	Media	Alta	Gr		Baja	Media	Alta
	Baja	142.9001	73.4496	118.3318		Baja	161.5336	148.3242	130.0600
	Media	226.5052	172.8044	180.8070		Media	210.3814	200.5133	178.7284
	Alta	244.4699	154.2910	170.0162		Alta	237.8336	237.4949	193.6494

40 nuevos datos

T	V				T	V		
Gr		Baja	Media	Alta	Gr		Baja	Media	Alta
	Baja	143.2112	65.7183	119.3162		Baja	158.5714	162.8571	117.2727
	Media	223.2207	158.4960	180.3809		Media	198.4289	205.1792	162.3957
	Alta	226.3168	149.9168	170.0205		Alta	235.3089	231.1883	200.2553

50 nuevos datos

T	V				T	V		
Gr		Baja	Media	Alta	Gr		Baja	Media	Alta
	Baja	141.2091	66.4079	119.8754		Baja	153.7500	160.0000	112.5000
	Media	221.3248	160.8375	180.1061		Media	203.2030	206.0589	158.2834
	Alta	228.3797	150.3868	170.0189		Alta	233.4570	231.0870	201.6107

60 nuevos datos

T	V				T	V		
Gr		Baja	Media	Alta	Gr		Baja	Media	Alta
	Baja	139.0659	64.5026	119.4785		Baja	148.4210	160.0000	110.4000
	Media	221.5251	159.6721	170.2181		Media	202.4395	205.2866	155.9116
	Alta	222.5859	151.0572	170.0164		Alta	232.0662	230.6976	194.7425

70 nuevos datos

T	V				T	V		
Gr		Baja	Media	Alta	Gr		Baja	Media	Alta
	Baja	137.6259	67.1594	120.0115		Baja	147.0000	160.0000	110.4000
	Media	217.6259	158.0777	170.0025		Media	199.4814	205.2087	153.1772
	Alta	219.8048	152.1752	170.0038		Alta	230.8634	229.0245	191.6805

80 nuevos datos

T	V				T	V		
Gr		Baja	Media	Alta	Gr		Baja	Media	Alta
	Baja	136.8042	66.7289	120.0000		Baja	147.0000	153.3333	106.6667
	Media	216.8311	157.2020	169.9999		Media	192.7079	200.7954	152.8825
	Alta	221.0032	152.7774	169.9999		Alta	231.6601	227.9057	189.5998

90 nuevos datos

T	V				T	V		
Gr		Baja	Media	Alta	Gr		Baja	Media	Alta
	Baja	135.3796	69.0961	114.2857		Baja	147.0000	148.5714	103.4483
	Media	217.6326	156.5097	164.2857		Media	190.3910	200.6368	150.9542
	Alta	221.2243	152.4493	169.9999		Alta	231.7117	226.5976	189.6218

100 nuevos datos

T	V			
Gr		Baja	Media	Alta
	Baja	134.0231	69.9947	114.2857
	Media	218.3093	155.6554	164.2857
	Alta	221.1080	153.4176	169.9999

T	V			
Gr		Baja	Media	Alta
	Baja	145.7143	148.5714	103.4483
	Media	188.3381	196.7314	151.7613
	Alta	229.7565	225.3740	181.0494

200 nuevos datos

T	V			
Gr		Baja	Media	Alta
	Baja	130.7718	67.7715	114.5454
	Media	217.5520	154.6073	160.9091
	Alta	217.6300	152.7863	169.9999

T	V			
Gr		Baja	Media	Alta
	Baja	140.0000	143.0769	87.3913
	Media	182.7755	179.4638	131.3997
	Alta	220.0097	220.4386	176.0317

300 nuevos datos

T	V			
Gr		Baja	Media	Alta
	Baja	128.7646	65.6763	101.9512
	Media	217.4424	155.9178	156.8292
	Alta	217.2303	154.7420	162.0000

T	V			
Gr		Baja	Media	Alta
	Baja	133.5000	137.6471	82.1053
	Media	177.5391	176.8336	127.7749
	Alta	217.8727	217.5073	165.6914

Usuario 2

T	V			
Gr		Baja	Media	Alta
	Baja	120	120	60
	Media	165	165	105
	Alta	210	210	150

Tabla 2.20

Como era de esperar, se aprecia que el sistema evoluciona hacia UE2, pero incluso con 300 datos añadidos, el error parece considerable.

En las siguientes figuras se muestra la evolución del error para los dos casos mencionados. En el eje vertical se muestra el error y en eje horizontal se muestran el número de datos introducidos con un factor de 0.1.

Sistema de 16 datos de UE1

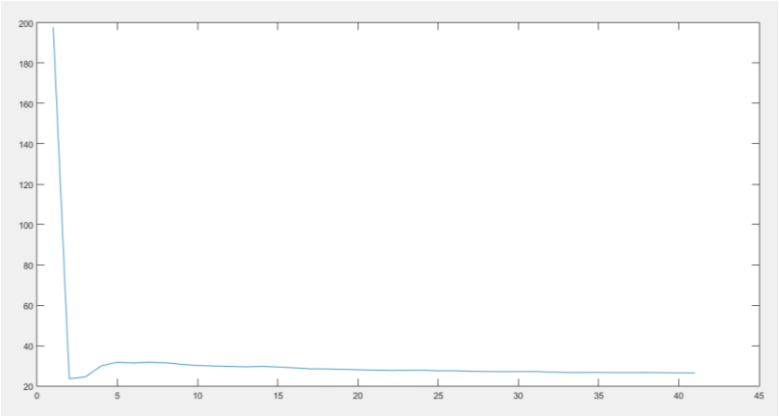


Figura 2.33

Como se puede ver, al añadir los primeros 10 valores de UE2, el error disminuye de forma contundente. De hecho, en la segunda iteración, cuando se añaden los primeros valores de UE2, el error es el mínimo e igual a **14.2938**. Después aumenta y vuelve a descender, finalizando a los 400 valores introducido con un valor de **26.4060.**

Sistema de 100 datos de UE1

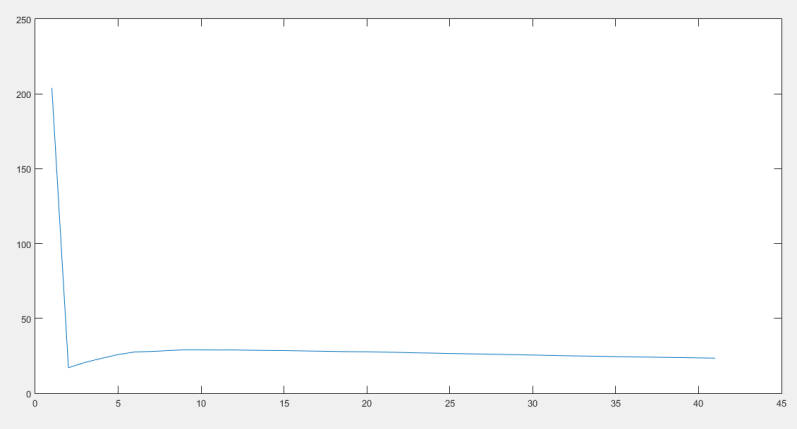
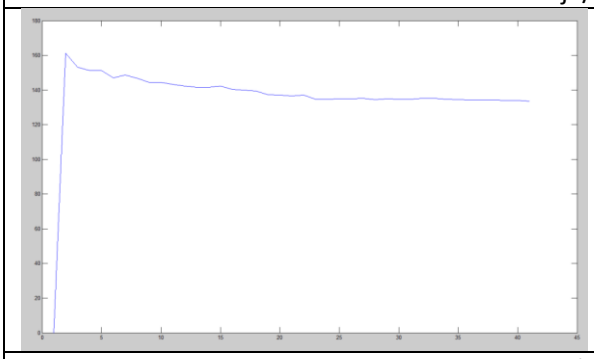
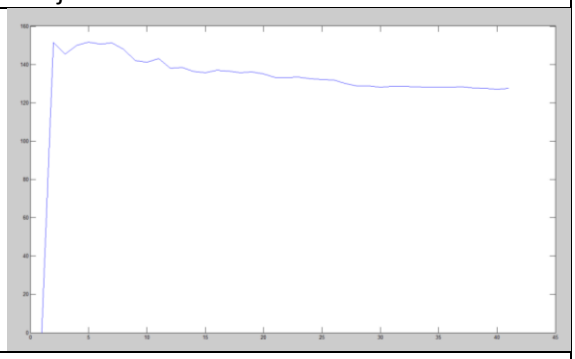
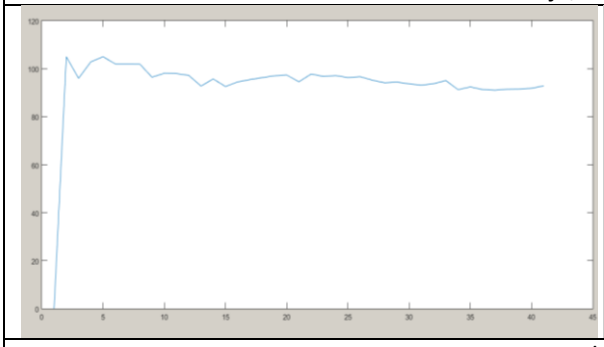
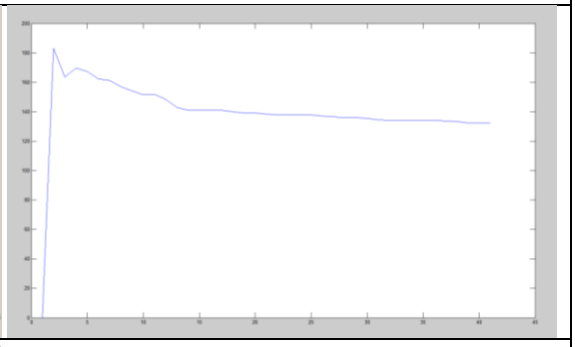
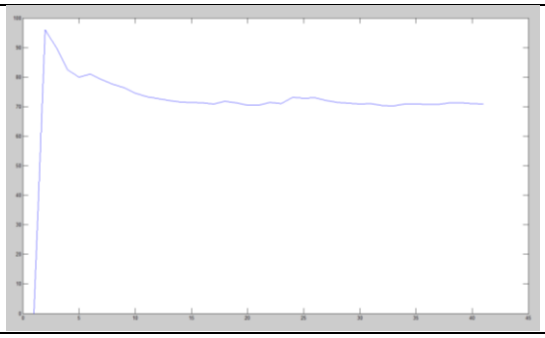
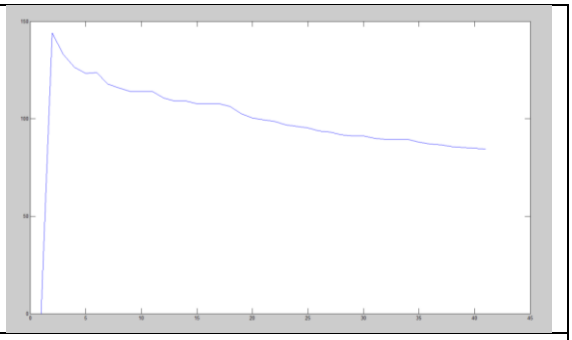


Figura 2.34

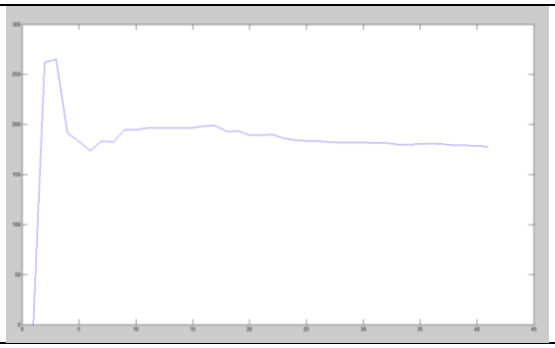
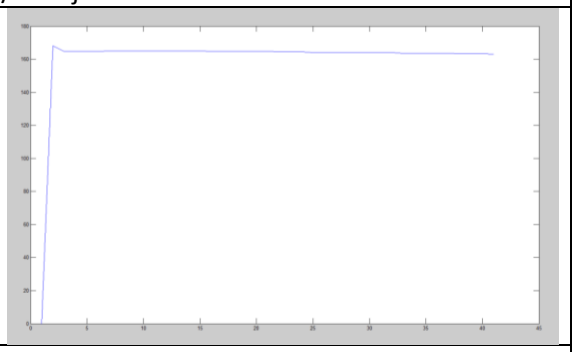
Al igual que el caso anterior, el error mínimo ocurre cuando se introducen los primeros 10 valores y vale **18.5097**. Luego asciende y vuelve a descender, finalizando con un error de **23.4488**.

En las siguientes gráficas se puede ver la evolución de los valores de los consecuentes de las reglas conforme va avanzando el entrenamiento, para el caso de 16 datos representativos de inicio de UE1 y 100 datos de UE1, introduciendo datos de UE2 cada 10.

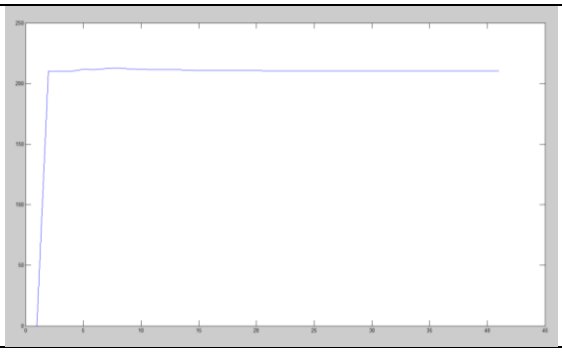
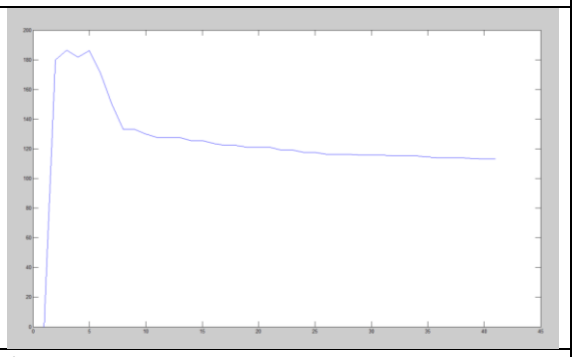
16 datos representativos	100 datos aleatorios
Gr baja/ V baja	
	
Gr baja/ V media	
	
Gr baja/ V alta	

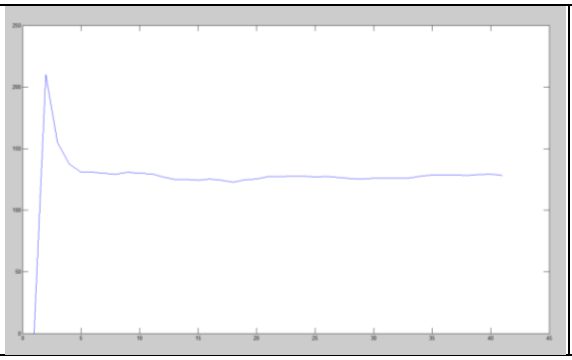
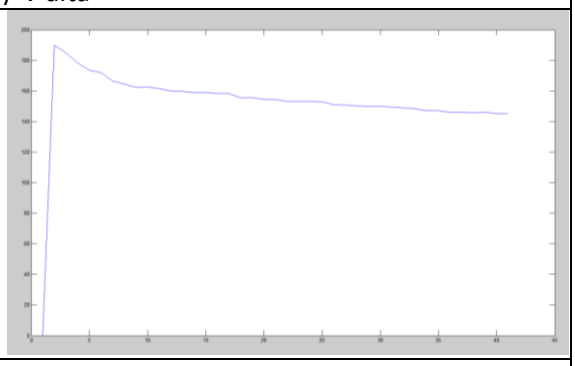
Gr media / V baja

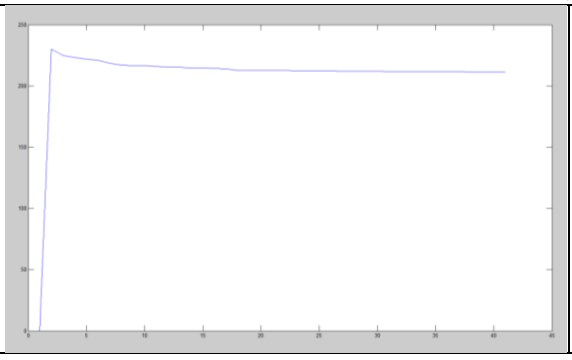
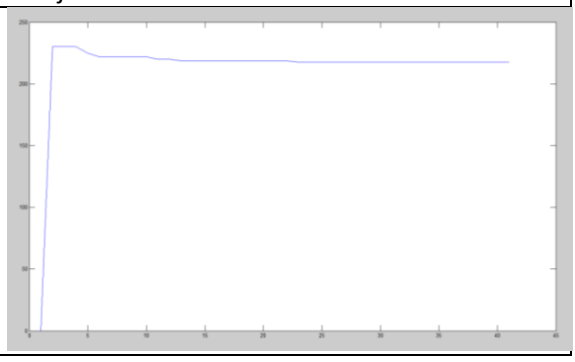
Gr media / V media

Gr media / V alta

Gr alta / V baja

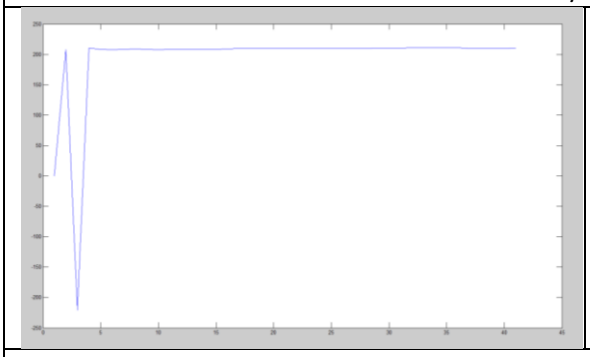
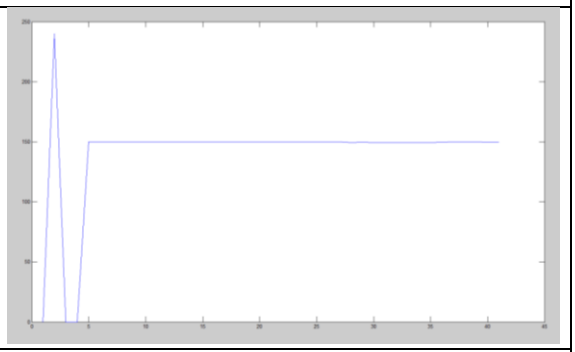
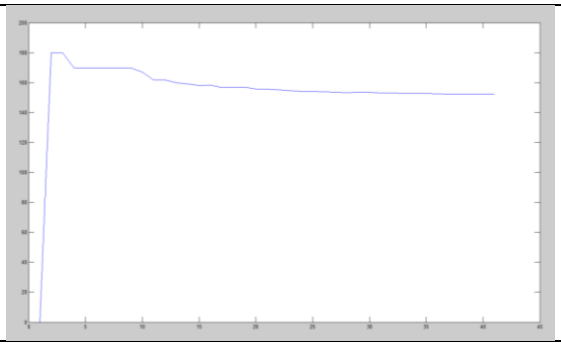
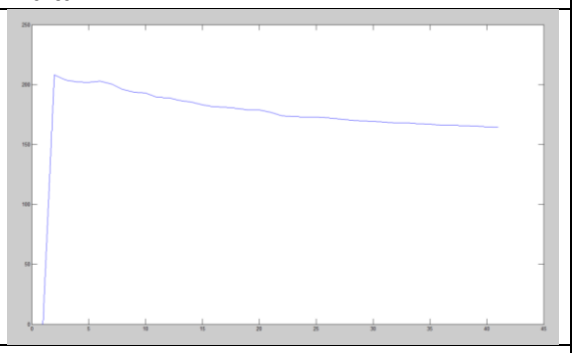
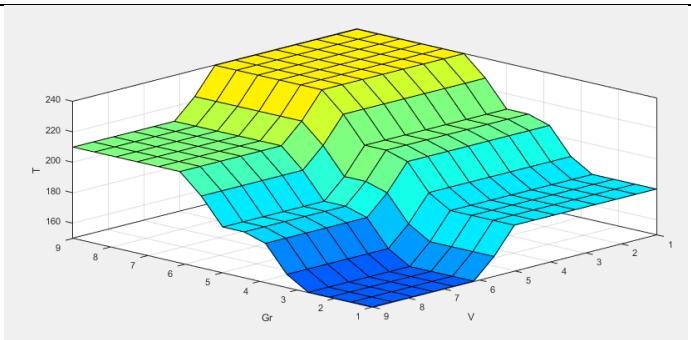
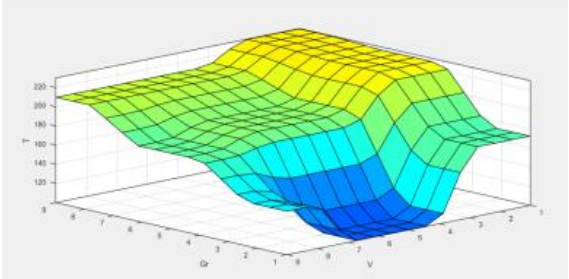
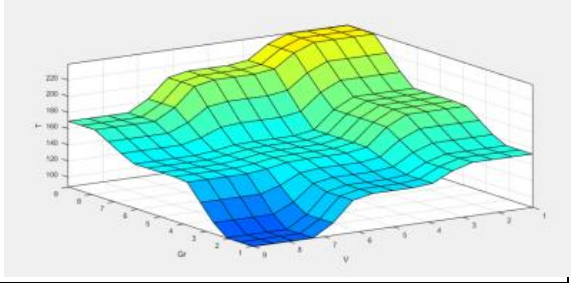
Gr alta / V media	
	
Gr alta / V alta	
	

Figura 2.35

Como se puede apreciar, en la mayoría de los casos, la variación de consecuente se estanca aproximadamente cuando se introducen 100 valores.

Las siguientes imágenes muestran como se modifican las superficies con la introducción de nuevos datos de UE2.

Inicialmente 16 datos de UE1	
	
10 datos nuevos introducidos	**30 datos nuevos introducidos**
	

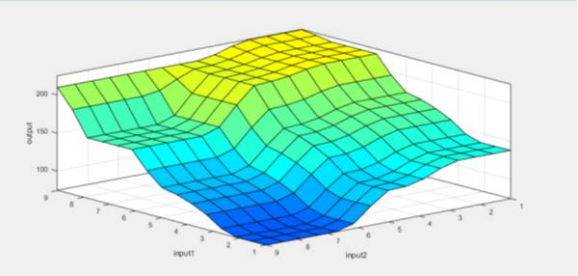
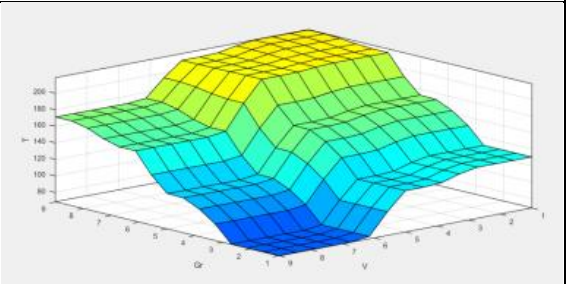
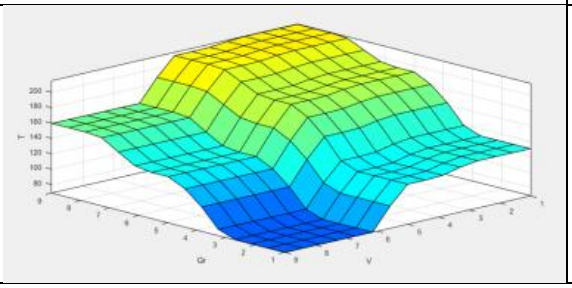
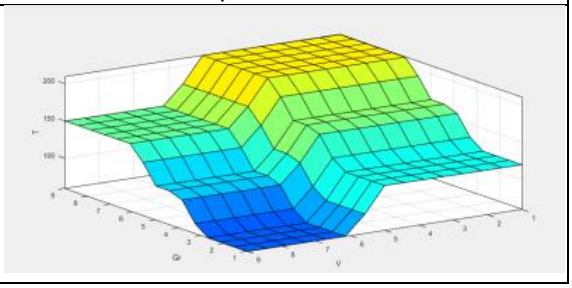
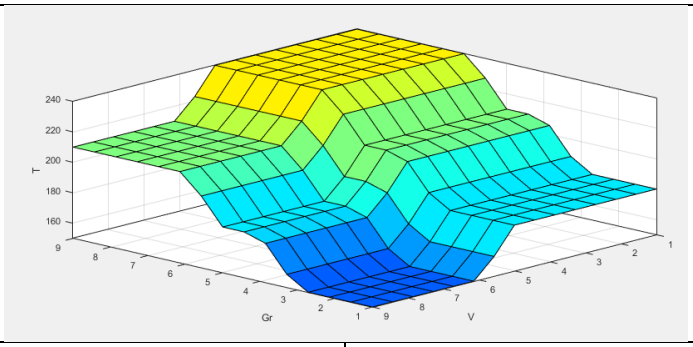
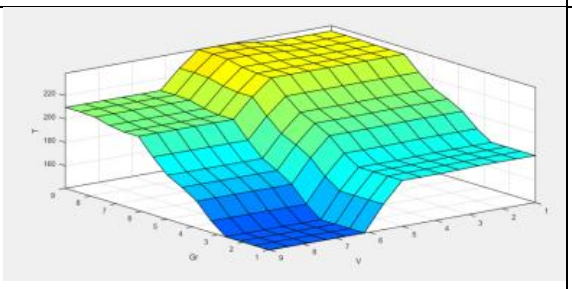
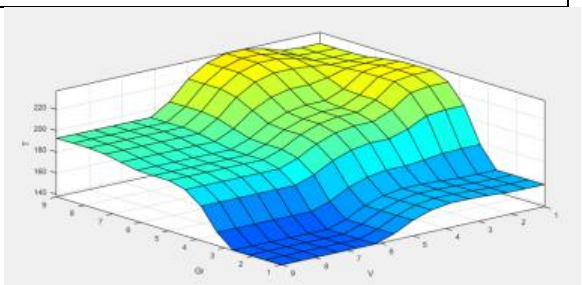
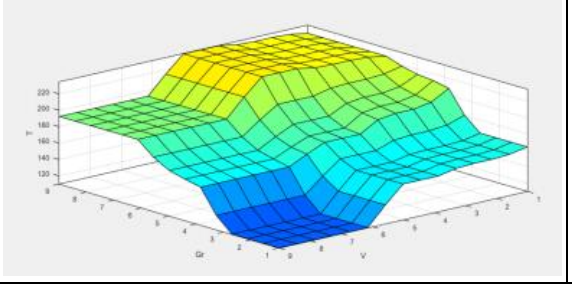
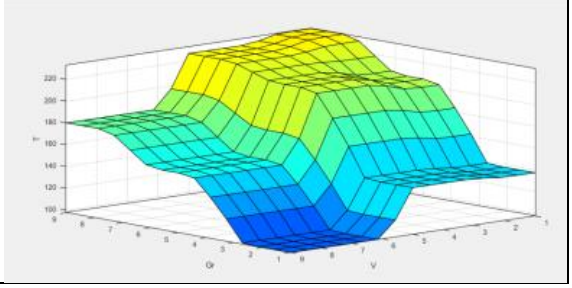
50 datos nuevos introducidos	100 datos nuevos introducidos
	
200 datos nuevos introducidos	**Superficie UE2**
	

Figura 2.36

Inicialmente 100 datos de UE1	
	
10 datos nuevos introducidos	**30 datos nuevos introducidos**
	
50 datos nuevos introducidos	**100 datos nuevos introducidos**
	

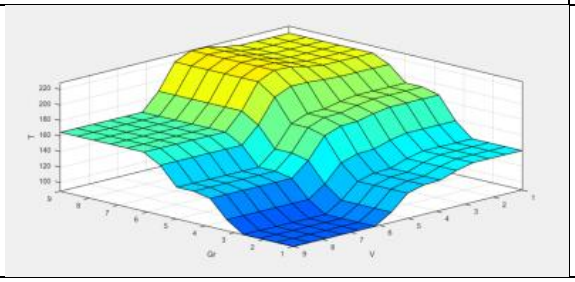
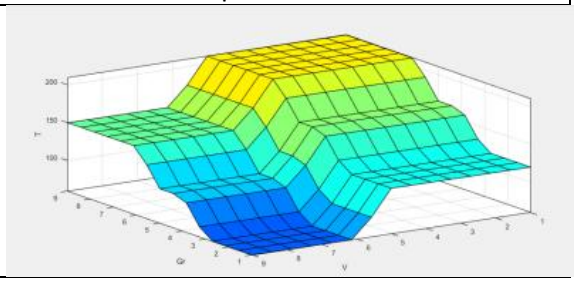
200 datos nuevos introducidos	Superficie UE2
	

Figura 2.37

Analizando las superficies, no se puede afirmar que el sistema con 16 datos de inicio de UE1 se modifique más rápido que el sistema que comenzaba con 100 datos de UE1. Lo que sí que se puede afirmar, es que en ambos casos, se necesitan muchos datos nuevos para que se la superficie se transforme en la superficie de UE2.

2.5.1.2.- Controlador de Bolus N

Al igual que con el controlador de tiempo de suministro (T), se trata de averiguar cuantos datos se deben introducir para que el sistema evolucione desde el usuario inicial UE1 hasta que el sistema se adapte a las necesidades del nuevo usuario 2. Para ello, como se ha explicado entes, se ha partido de unos datos válidos para el usuario 1 y se han ido añadiendo datos del UE2 cada 10 y se ha realizado el aprendizaje.

En un principio, parece más lógico que, cuantos más datos de UE1 se dispongan, más tardará el sistema en transformarse. Por ello, se ha realizado la prueba de dos formas diferentes; en una se disponen de 16 datos, suficientes para que al inicio Matlab genere el FIS al completo, es decir, sin ningún antecedente en cero, y otro sistema donde se tengan 100 datos de UE1, donde, a priori, cueste más que el sistema evolucione.

En la siguiente tabla se muestra cómo cambian los consecuentes cuando se le introducen datos cada 10.

Partiendo desde un FIS generado con 16 datos representativos de UE1	Partiendo desde FIS generado con 100 datos

0 nuevos datos

Bn	V			
Gr		Baja	Media	Alta
	Baja	33.3	56	77.8
	Media	30	50	66.7
	Alta	30	44.4	60

10 nuevos datos

Bn	V			
Gr		Baja	Media	Alta
	Baja	39.7522	75.9556	75.7637
	Media	26.0000	65.6860	62.3324
	Alta	26.9592	73.9009	59.9999

Bn	V			
Gr		Baja	Media	Alta
	Baja	37.8581	60.5241	78.4544
	Media	33.5513	52.7325	66.4412
	Alta	31.8770	45.2588	63.4757

20 nuevos datos

Bn	V			
Gr		Baja	Media	Alta
	Baja	49.5118	73.6950	75.6479
	Media	25.3588	66.7009	65.0676
	Alta	36.3959	74.5499	59.9999

Bn	V			
Gr		Baja	Media	Alta
	Baja	38.3579	63.7007	77.6330
	Media	34.9232	61.4532	66.0535
	Alta	32.4581	48.1603	64.2834

30 nuevos datos

Bn	V			
Gr		Baja	Media	Alta
	Baja	54.6661	72.1142	73.9000
	Media	38.8335	69.0051	72.6000
	Alta	46.7811	74.8624	59.9999

Bn	V			
Gr		Baja	Media	Alta
	Baja	41.6749	62.8053	76.3052
	Media	39.0848	60.2229	65.7974
	Alta	36.8253	47.4734	64.3689

40 nuevos datos

Bn	V			
Gr		Baja	Media	Alta
	Baja	57.7667	71.5600	73.9028
	Media	45.9429	68.2923	71.9544
	Alta	48.0000	75.3846	59.9815

Bn	V			
Gr		Baja	Media	Alta
	Baja	42.6239	62.0111	76.3421
	Media	38.0554	61.0937	65.7697
	Alta	37.3680	49.7861	64.3245

50 nuevos datos

Bn	V			
Gr		Baja	Media	Alta
	Baja	57.7667	71.5600	73.9028
	Media	49.0667	68.7667	71.9538
	Alta	50.0000	75.6667	59.9811

Bn	V			
Gr		Baja	Media	Alta
	Baja	45.7259	61.8968	76.8365
	Media	38.1101	62.2131	66.1779
	Alta	38.3566	52.4815	66.8795

60 nuevos datos

Bn	V			
Gr		Baja	Media	Alta
	Baja	57.7667	71.1143	73.9033
	Media	50.6600	69.0348	71.9535
	Alta	51.4286	75.9375	59.9808

Bn	V			
Gr		Baja	Media	Alta
	Baja	48.2071	63.0882	76.1772
	Media	38.2377	62.4861	66.0542
	Alta	40.6184	53.5662	68.5270

70 nuevos datos

Bn	V			
Gr		Baja	Media	Alta
	Baja	57.7667	71.1143	73.9033
	Media	51.9636	69.3120	70.5615
	Alta	54.5454	76.3889	59.9803

Bn	V			
Gr		Baja	Media	Alta
	Baja	50.2385	64.1154	75.7706
	Media	38.1942	62.9149	66.3342
	Alta	43.3314	54.6290	68.8421

80 nuevos datos

Bn	V			
Gr		Baja	Media	Alta
	Baja	60.8250	70.8667	73.9036
	Media	51.9636	69.3857	70.5614
	Alta	55.7143	76.3889	59.9803

Bn	V			
Gr		Baja	Media	Alta
	Baja	53.9437	64.5141	75.4929
	Media	35.2850	63.0875	65.8355
	Alta	45.6218	56.3858	68.5494

90 nuevos datos

Bn	V			
Gr		Baja	Media	Alta
	Baja	61.8444	70.8667	73.9036
	Media	53.4667	69.7677	70.5609
	Alta	56.2500	76.9048	59.9796

Bn	V			
Gr		Baja	Media	Alta
	Baja	54.8882	65.0668	75.2610
	Media	37.3457	63.8383	65.6212
	Alta	45.8642	57.0687	69.0117

100 nuevos datos

Bn	V			
Gr		Baja	Media	Alta
	Baja	63.3273	70.7800	72.6022
	Media	53.9692	69.3471	70.5614
	Alta	56.4706	76.8182	59.9797

Bn	V			
Gr		Baja	Media	Alta
	Baja	56.4789	65.9148	75.3532
	Media	37.3457	65.1362	65.4214
	Alta	45.8642	57.7864	69.8579

200 nuevos datos

Bn	V			
Gr		Baja	Media	Alta
	Baja	65.6823	70.3900	72.6026
	Media	60.5724	70.2507	72.1356
	Alta	57.5000	77.7778	59.9786

Bn	V			
Gr		Baja	Media	Alta
	Baja	61.1414	67.8734	73.6766
	Media	52.9279	66.8205	64.4678
	Alta	49.1216	63.4857	72.0126

300 nuevos datos									
Bn	V				Bn	V			
Gr		Baja	Media	Alta	Gr		Baja	Media	Alta
	Baja	66.5048	70.2294	72.6274		Baja	63.4128	68.0727	72.5517
	Media	61.9913	70.1255	72.2493		Media	55.0328	67.8092	63.9023
	Alta	58.2353	77.8462	73.2811		Alta	51.7601	64.9265	74.8762

Usuario 2				
Bn	V			
Gr		Baja	Media	Alta
	Baja	70	70	70
	Media	65	70	65
	Alta	60	75	80

Tabla 2.21

Analizando la primera tabla de consecuentes y la final, queda patente que en ambos casos, el sistema evoluciona hacia UE2.

En las siguientes figuras se muestra la evolución del error para los dos casos mencionados. En el eje vertical se muestra el error y en eje horizontal se muestran el número de datos introducidos con un factor de 0.1.

Sistema de 16 datos de UE1

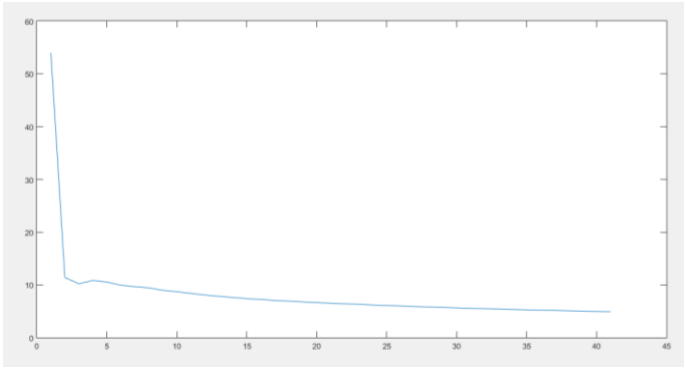


Figura 2.38

Al introducir los primeros 10 datos, el error cae drásticamente y sigue descendiendo cada vez que se le introducen nuevos datos. Al final, se puede ver que el error más o menos se estanca y a los 400 datos introducidos, el error vale **4.9110**.

Sistema de 100 datos de UE1

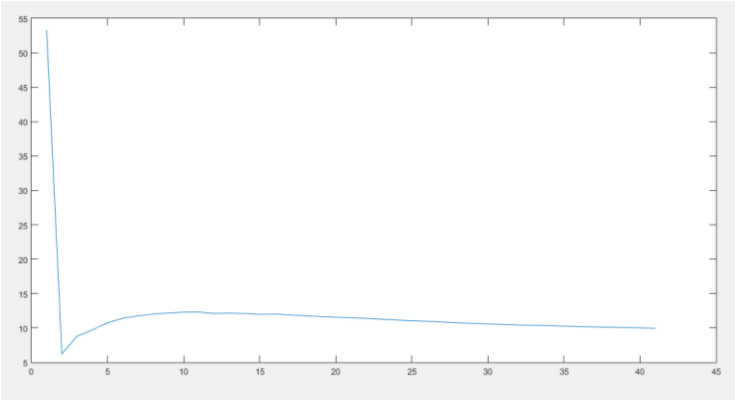
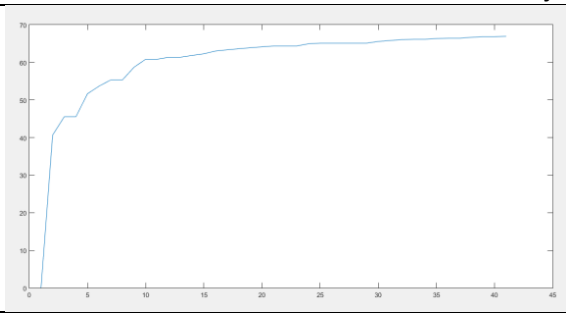
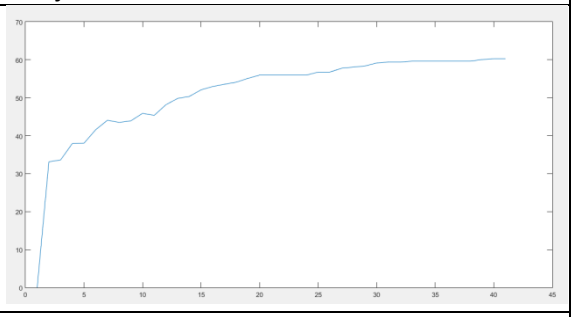
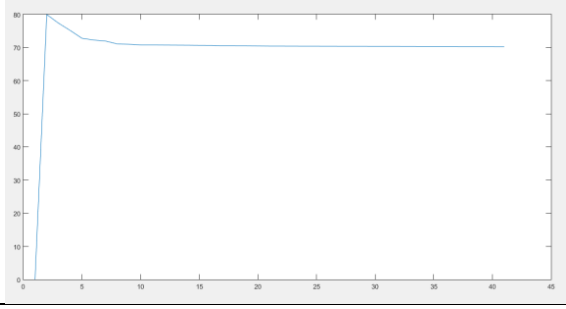
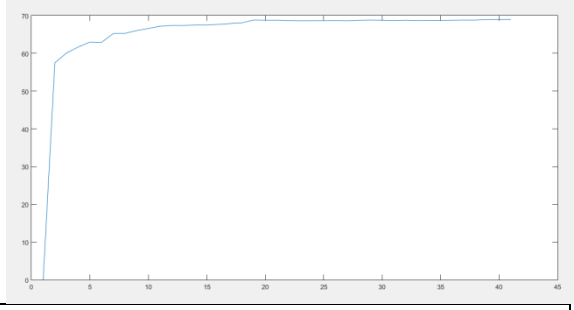
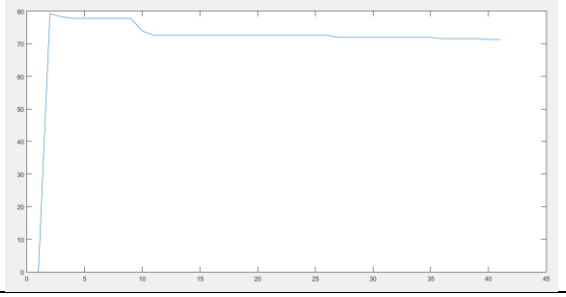
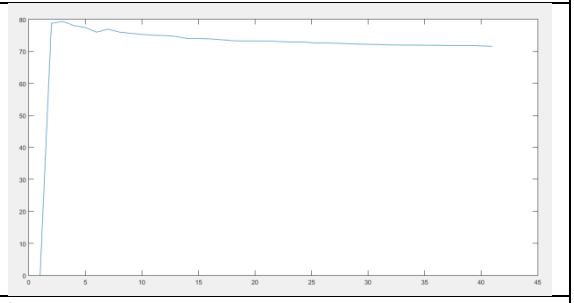
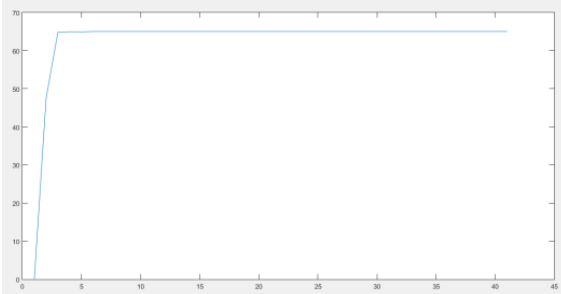
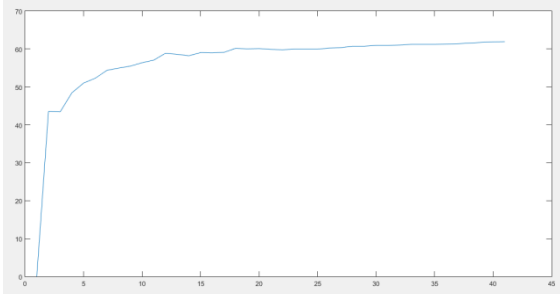


Figura 2.39

Como se ve en la figura superior, al añadir los primeros 10 valores de UE2, el error disminuye de forma drástica. De hecho, en la segunda iteración, cuando se añaden los primeros valores de UE2, el error es el mínimo e igual a **6.1861**. Después aumenta y vuelve a descender, finalizando a los 400 valores introducido con un valor de **9.9610.**

En las siguientes gráficas se puede ver la evolución de los valores de los consecuentes de las reglas conforme va avanzando el entrenamiento, para el caso de 16 datos representativos de inicio de UE1 y 100 datos de UE1, introduciendo datos de UE2 cada 10.

16 datos representativos	100 datos aleatorios
Gr baja/ V baja	
	
Gr baja/ V media	
	
Gr baja/ V alta	
	
Gr media / V baja	
	
Gr media / V media	

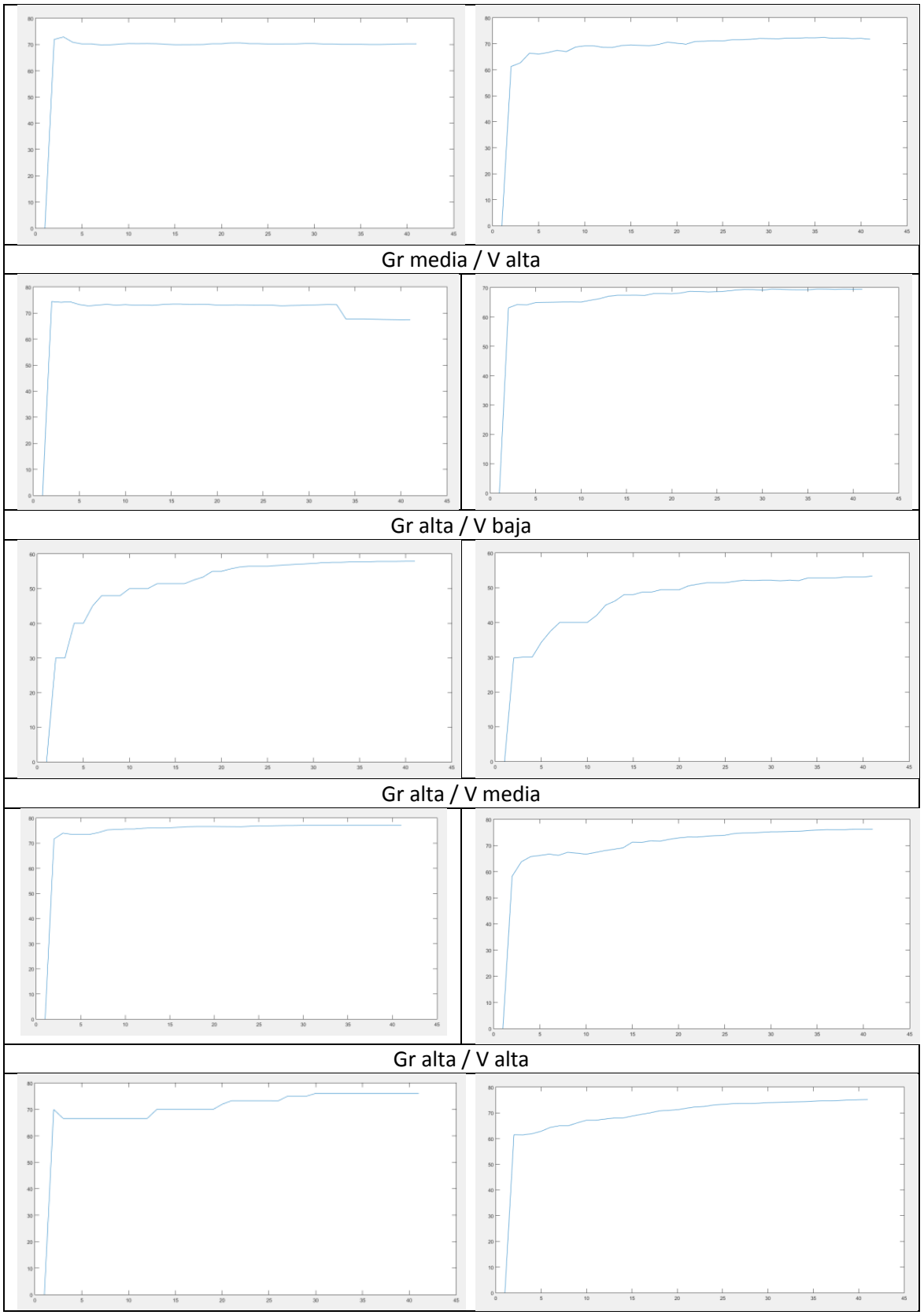


Figura 2.40

Por lo general, los consecuentes se estancan cuando se introducen 150 datos nuevos.

Las siguientes imágenes muestran como se modifican las superficies con la introducción de nuevos datos de UE2.

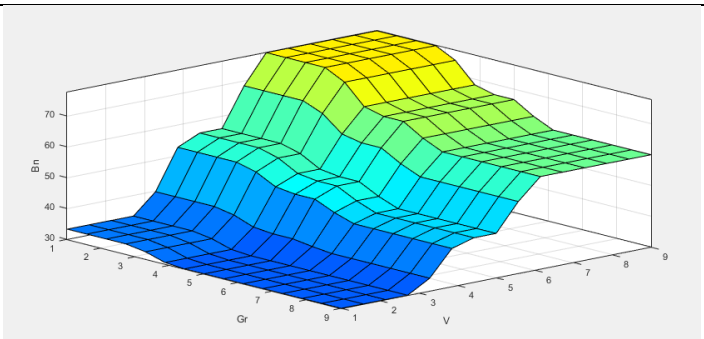
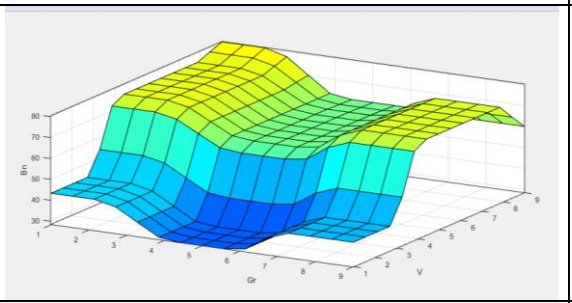
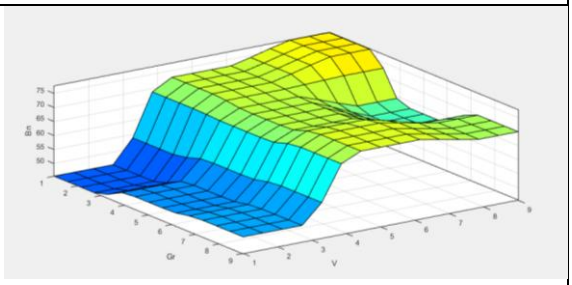
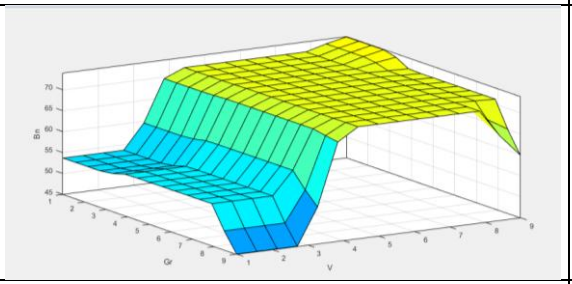
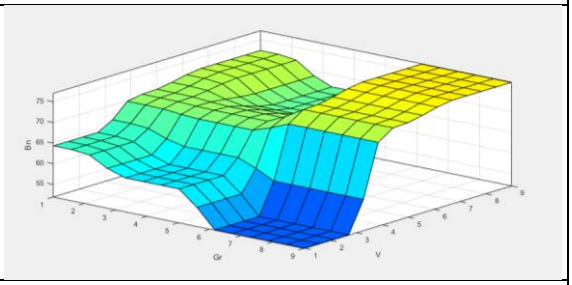
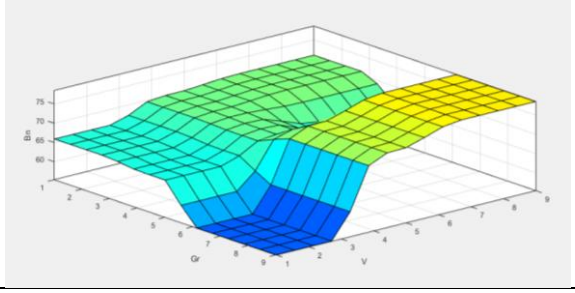
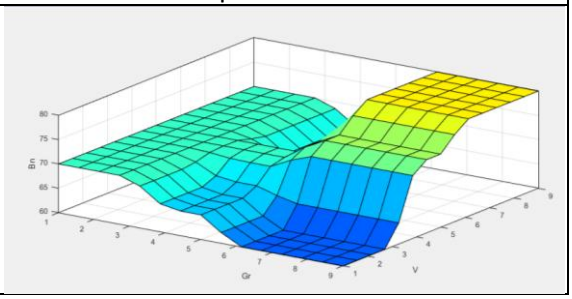
Inicialmente 16 datos de UE1	
	
10 datos nuevos introducidos	30 datos nuevos introducidos
	
50 datos nuevos introducidos	100 datos nuevos introducidos
	
200 datos nuevos introducidos	Superficie UE2
	

Figura 2.41

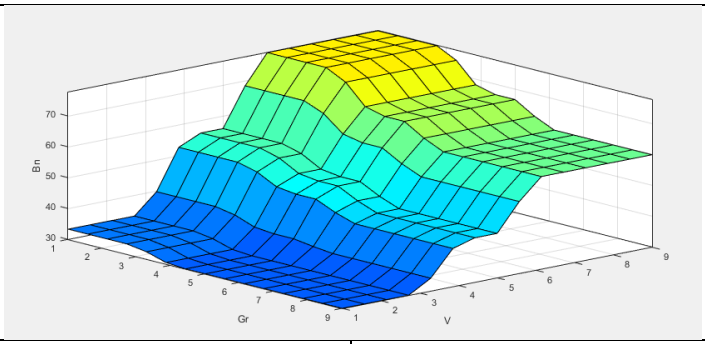
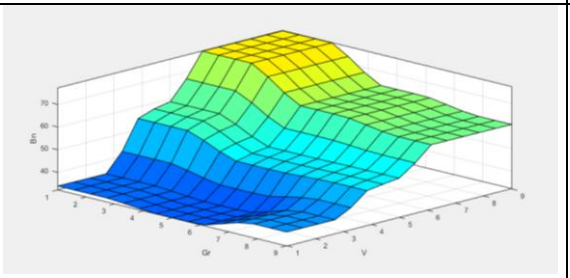
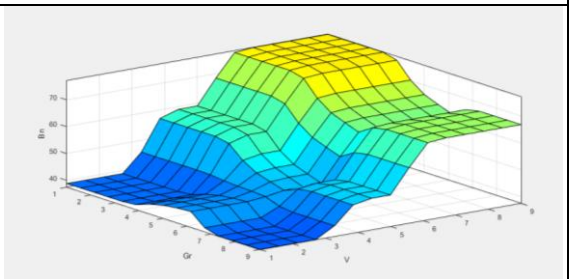
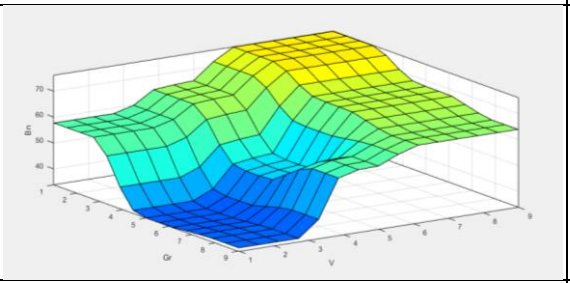
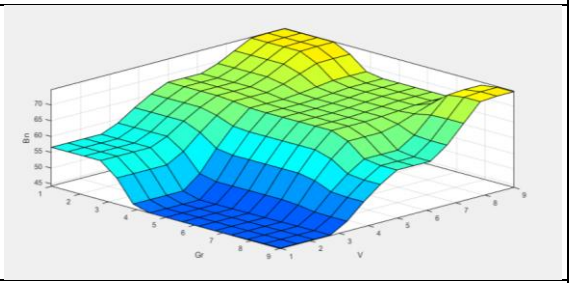
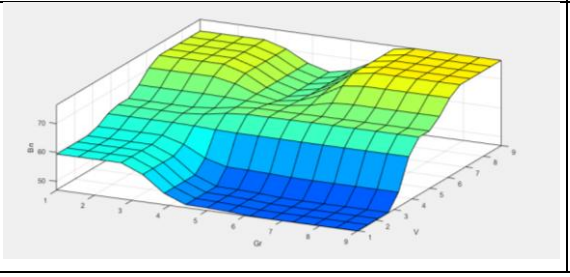
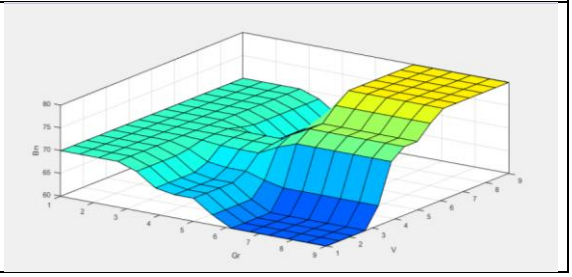
<table>
<tr><td colspan="2">Inicialmente 100 datos de UE1</td></tr>
<tr><td colspan="2">  </td></tr>
<tr><td>10 datos nuevos introducidos</td><td>30 datos nuevos introducidos</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>50 datos nuevos introducidos</td><td>100 datos nuevos introducidos</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>200 datos nuevos introducidos</td><td>Superficie UE2</td></tr>
<tr><td>  </td><td>  </td></tr>
</table>

Figura 2.42

A diferencia de antes, en este caso sí que se aprecia que las superficies evolucionan a la superficie de UE2 con menos datos introducidos cuando al inicio se tienen 16 datos de UE1.

2.5.1.3.- Comentarios

Algo que ocurre en todos los casos es que se necesita gran cantidad de datos para llegar a las exigencias del usuario nuevo. Observando las superficies se aprecia que con 200 datos del nuevo usuario, las salidas que se obtendrían serían validas para este nuevo usuario.

Como era de esperar, el controlador que inicia con menos datos de UE1, evoluciona antes al nuevo usuario, aunque ambos llegan a evolucionar al nuevo usuario.

Cuantas más iteraciones se realizan y cuantos más datos se manejan, más comúnmente ocurre que los puntos de los trapecios se solapen y el programa se pare.

2.5.2.- Aprendizaje mediante sustitución

En este ejercicio se desea estudiar cuantos datos son necesarios añadir del nuevo usuario en la matriz de datos a entrenar, para satisfacer los requerimientos de insulina del nuevo usuario mediante sustitución.

Inicialmente los datos de se dispone en la matriz de datos son los del usuario 1. Al igual que se ha hecho en el aprendizaje sin sustitución, esta matriz de datos va a estar compuesta unas veces por 16 datos significativos y otras veces por 100 datos aleatorios. El objetivo es ver con cuál de los dos controladores, el sistema evoluciona antes.

Para comenzar, se genera un FIS con el comando “genfis1” y se definen las características de las funciones de pertenencia. Después, con los datos de la matriz de datos se entrena y obtiene un sistema borroso de inferencia nuevo.

Mientras tanto, mediante un FIS del usuario 2 y con el comando “evalfis” se han generado datos del usuario 2.

Para ver cómo evoluciona el sistema desde el FIS inicial generado con los datos de UE1 hacia el usuario dos, se van introduciendo datos del UE2 poco a poco en la matriz de datos. En caso de que las entradas coincidan, se sustituirá el consecuente de la matriz por el nuevo.

Una vez introducidos los datos de UE2, esta matriz de datos se usa como datos de entrenamiento. Entonces se vuelve a entrenar tomando como FIS de base el FIS generado que únicamente era válido para el usuario 1 y los datos a entrenar serán los de la matriz de datos.

Cada vez que se entrena el sistema, se le introducen poco a poco nuevos datos del nuevo usuario a la matriz de datos y se vuelve a entrenar tomando esta matriz como datos de entrenamiento y como FIS base, el FIS anterior de entrenamiento. Es decir, el FIS que se genera se sobrescribe cada vez que se entrena con los nuevos datos que se le introducen del nuevo usuario.

Como en el apartado de aprendizaje sin sustitución, los nuevos datos se introducirán de 10 en 10 y se realizaran 150 iteraciones, ya que el error se estabiliza en 100 iteraciones.

2.5.2.1.- Controlador de tiempo de suministro de Bolus C (T)

Como se ha mencionado antes, el objetivo es ver la evolución del sistema cuando se tienen 16 datos representativos de UE1 o 100 datos aleatorios del usuario 1. Para ver esta transformación, se irán introduciendo los datos del usuario 2 cada 10 y en caso de que se repitan las entradas de los datos nuevos y viejos, los viejos serán sustituidos por los nuevos. Después se procederá al entrenamiento con estos datos.

En la siguiente tabla se muestra cómo cambian los consecuentes según se van introduciendo y sustituyendo los datos.

Partiendo desde un FIS generado con 16 datos representativos de UE1	Partiendo desde FIS generado con 100 datos

0 nuevos datos

T	V			
Gr		Baja	Media	Alta
	Baja	180	180	77.8
	Media	210	210	180
	Alta	240	240	210

10 nuevos datos

T	V			
Gr		Baja	Media	Alta
	Baja	168.0000	120.0000	149.9999
	Media	168.8671	103.1028	423.6014
	Alta	232.0799	199.5833	180.0014

T	V			
Gr		Baja	Media	Alta
		173.9137	168.0950	142.6114
		203.4713	204.9922	179.1632
		241.5802	229.9623	199.1459

20 nuevos datos

T	V			
Gr		Baja	Media	Alta
	Baja	155.0493	81.1805	149.9999
	Media	165.0622	77.0744	105.7731
	Alta	229.9512	168.1903	189.9999

T	V			
Gr		Baja	Media	Alta
	Baja	180.0000	165.7915	133.8164
	Media	202.5000	201.6576	165.6083
	Alta	231.8182	232.9192	196.7634

30 nuevos datos

T	V			
Gr		Baja	Media	Alta
	Baja	151.4291	71.4958	120.0000
	Media	166.1610	100.0035	103.7310
	Alta	227.0208	169.7535	180.0000

T	V			
Gr		Baja	Media	Alta
	Baja	168.0000	164.1295	125.9293
	Media	197.1428	188.7728	159.7252
	Alta	231.8182	230.2326	192.9079

40 nuevos datos

T	V			
Gr		Baja	Media	Alta
	Baja	142.9706	65.3760	120.0000
	Media	163.0585	106.7564	0
	Alta	227.6655	163.1817	180.0000

T	V			
Gr		Baja	Media	Alta
	Baja	168.0000	160.3973	118.4147
	Media	197.1428	184.2978	159.9019
	Alta	231.8182	230.4606	186.4899

50 nuevos datos

T	V			
Gr		Baja	Media	Alta
	Baja	139.6067	68.6008	120.0000
	Media	162.8718	106.8347	0
	Alta	226.7683	157.7300	174.0000

T	V			
Gr		Baja	Media	Alta
		160.0000	159.1717	116.5354
		193.1250	183.6492	152.0298
		231.8182	228.7899	183.8292

60 nuevos datos

T	V			
Gr		Baja	Media	Alta
	Baja	137.9494	70.8959	120.0000
	Media	166.2536	106.0295	100.7530
	Alta	222.7390	157.0640	167.1428

T	V			
Gr		Baja	Media	Alta
	Baja	160.0000	155.0294	118.8049
	Media	187.5000	183.4731	146.9439
	Alta	231.8182	228.2031	181.8987

70 nuevos datos

T	V			
Gr		Baja	Media	Alta
	Baja	138.0232	72.1604	124.9597
	Media	168.3637	104.3144	103.7743
	Alta	222.5795	156.3425	163.3333

T	V			
Gr		Baja	Media	Alta
	Baja	154.2857	153.9386	118.0964
	Media	185.4545	183.5041	144.0622
	Alta	230.0000	225.1939	180.4091

80 nuevos datos

T	V		
Gr		Baja	Media	Alta
	Baja	136.9456	69.2662	124.8993
	Media	170.1620	104.9430	103.1044
	Alta	222.7271	159.6502	163.3333

T	V		
Gr		Baja	Media	Alta
	Baja	146.6667	154.0018	115.7663
	Media	185.4545	183.5416	144.3314
	Alta	228.4615	223.9894	176.4772

90 nuevos datos

T	V		
Gr		Baja	Media	Alta
	Baja	132.1018	71.8412	124.9659
	Media	172.3489	106.2132	103.8688
	Alta	220.9468	159.6533	163.3333

T	V		
Gr		Baja	Media	Alta
	Baja	146.6667	154.2671	110.3544
	Media	183.7500	182.2789	140.2559
	Alta	227.1428	222.8460	174.5606

100 nuevos datos

T	V		
Gr		Baja	Media	Alta
	Baja	131.1433	72.2153	124.8201
	Media	177.9067	109.4691	102.5220
	Alta	219.0696	159.2564	160.9091

T	V		
Gr		Baja	Media	Alta
	Baja	144.0000	153.6134	104.5227
	Media	182.3077	181.1167	139.6071
	Alta	226.0000	222.0245	172.3875

200 nuevos datos

T	V		
Gr		Baja	Media	Alta
	Baja	125.3602	120.1114	69.5588
	Media	187.6689	175.9454	119.0943
	Alta	217.7643	214.5426	159.2112

T	V		
Gr		Baja	Media	Alta
	Baja	130.9091	142.4977	92.9960
	Media	180.5555	176.0169	125.5151
	Alta	219.2045	218.7461	164.6846

300 nuevos datos

T	V		
Gr		Baja	Media	Alta
	Baja	121.2640	119.2280	65.2963
	Media	170.9316	172.9590	110.8975
	Alta	218.1211	214.6732	156.8406

T	V		
Gr		Baja	Media	Alta
	Baja	132.0000	138.7255	78.0408
	Media	176.2500	172.9863	119.5137
	Alta	217.7419	216.3197	159.9413

Usuario 2

T	V		
Gr		Baja	Media	Alta
	Baja	120	120	60
	Media	165	165	105
	Alta	210	210	150

Tabla 2.22

Se ve que ambos controladores tienden a los consecuentes de UE2. Como era de esperar, el controlador con menos datos de UE1 se transforma más rápidamente, ya que tiene menos valores que sustituir que el controlador con 100 datos de UE1.

En las siguientes figuras se muestra la evolución del error para los dos casos mencionados. En el eje vertical se muestra el error y en eje horizontal se muestran el número de datos introducidos con un factor de 0.1.

Sistema de 16 datos de UE1

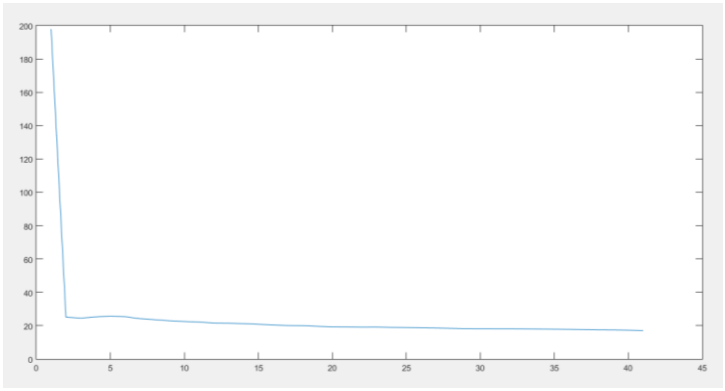


Figura 2.43

Como en los casos anteriores, al introducir los primeros valores de UE2, el error se reduce rápidamente. Después se va reduciendo poco a poco. Error final= **17.1069**.

Sistema de 100 datos de UE1

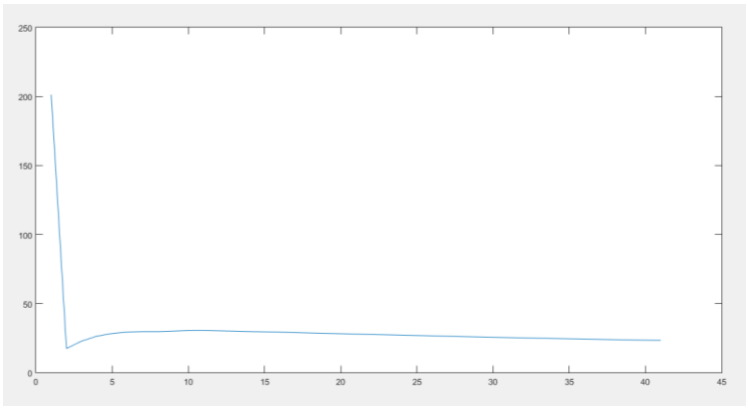
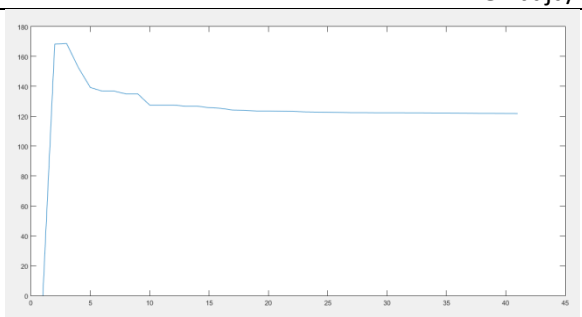
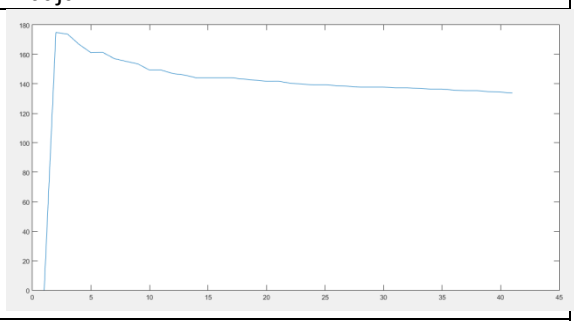


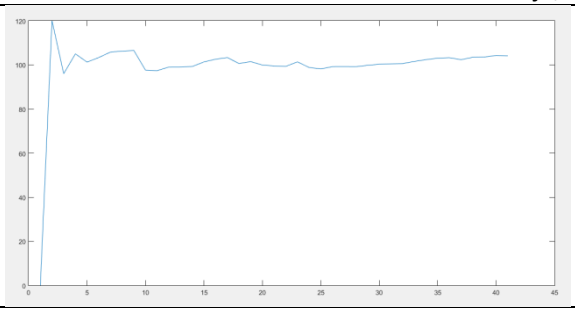
Figura 2.44

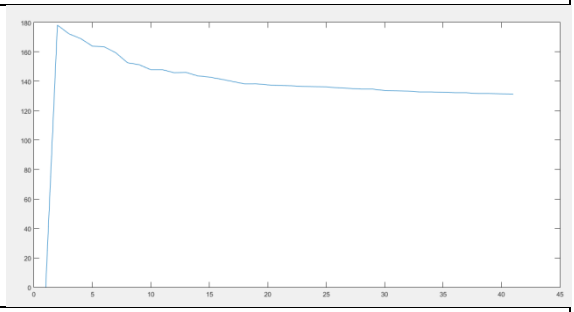
Como en el caso anterior, al inicio del entrenamiento el error se ve reducido mucho, pero en las siguientes veces que se introducen datos nuevos apenas mejora. Con 400 datos introducidos el error es de **23.2846**.

En las siguientes figuras se representa la evolución de los consecuentes en función del número de datos que se introducen del nuevo usuario (UE2) para el aprendizaje.

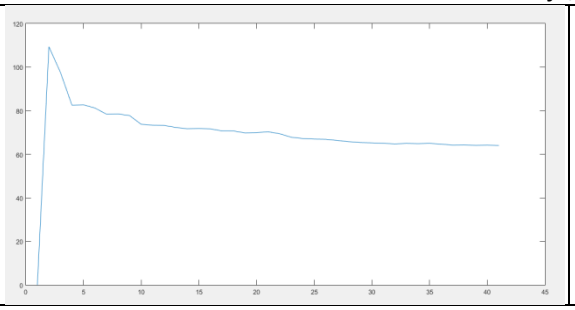
16 datos representativos	100 datos aleatorios
Gr baja/ V baja	
	

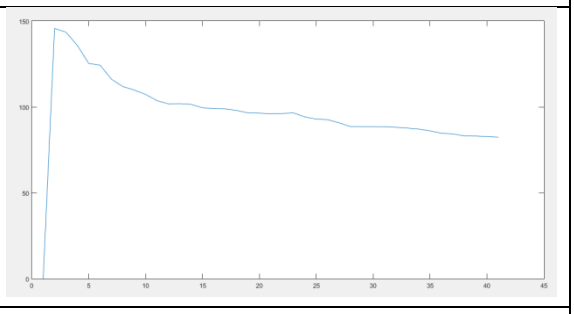
Gr baja/ V media



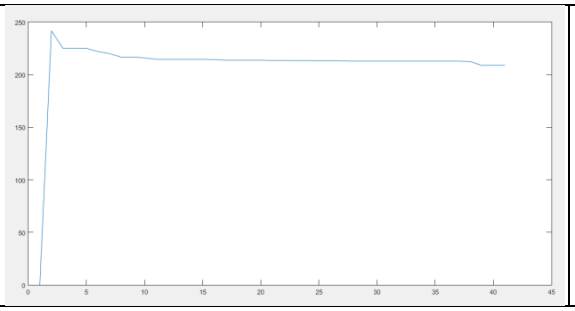


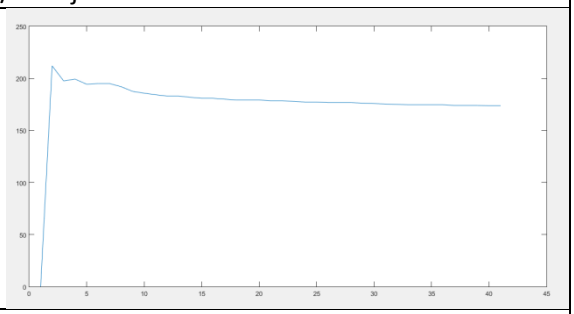
Gr baja/ V alta



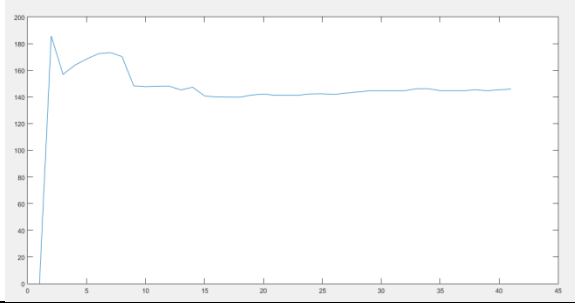


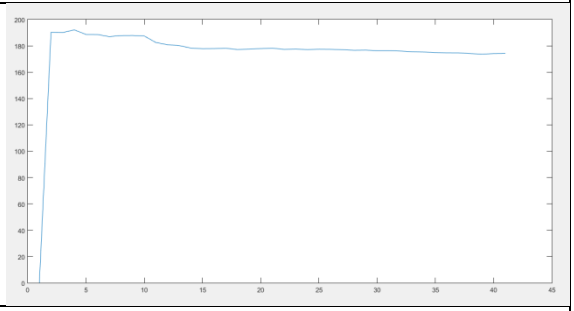
Gr media / V baja



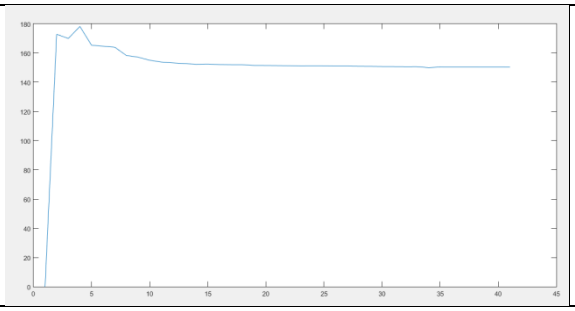


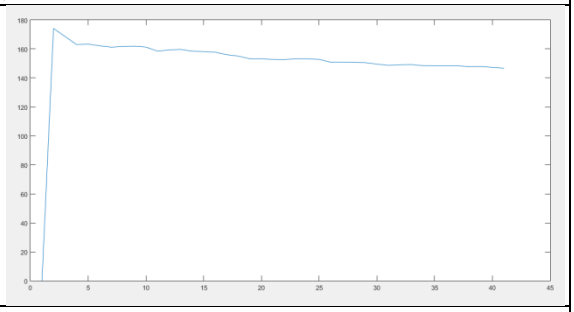
Gr media / V media





Gr media / V alta





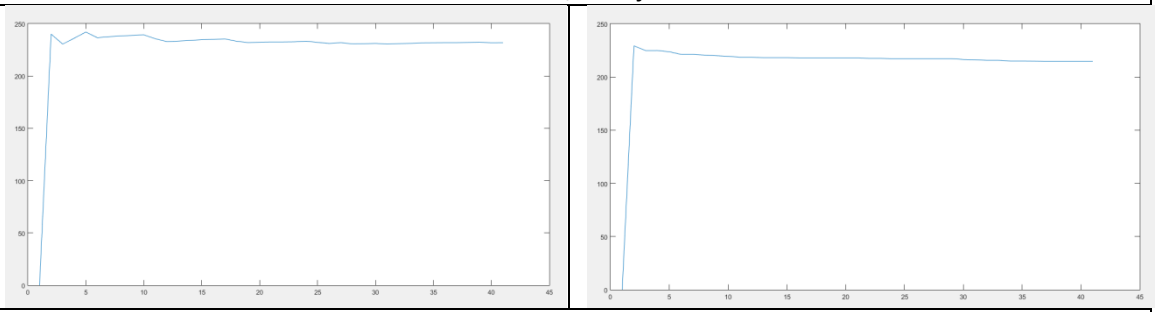
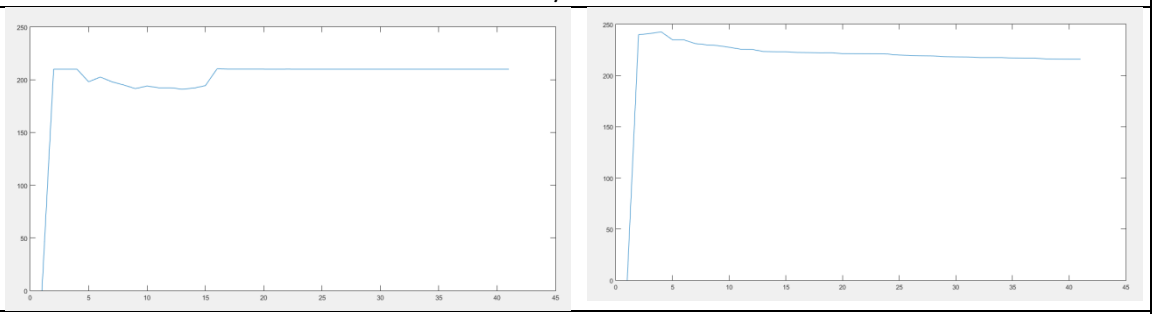
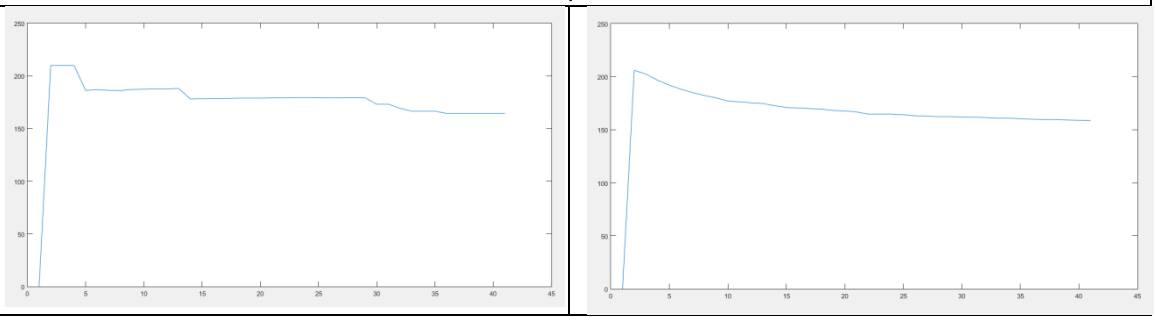
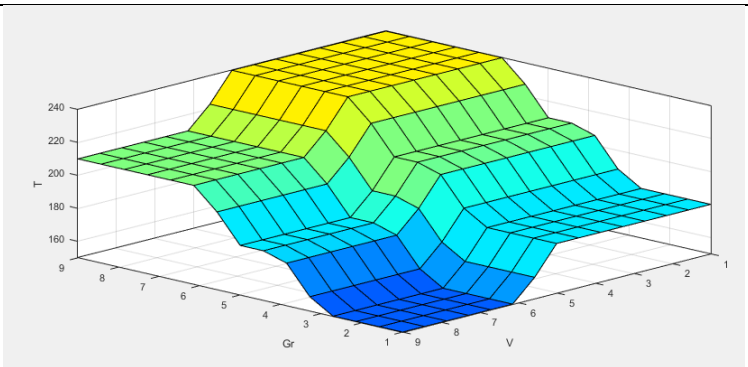
Gr alta / V baja

Gr alta / V media

Gr alta / V alta


Figura 2.45

En general, a partir de cuándo se introducen los 100 primeros valores de UE2, los consecuentes no varían en exceso.

A continuación se mostrará la evolución de las superficies para el controlador con 16 datos iniciales de UE1.

Inicialmente 16 datos de UE1


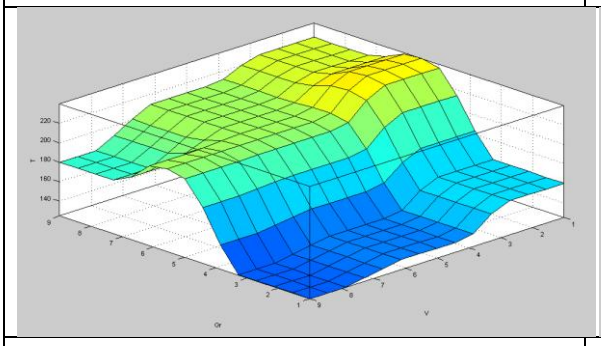
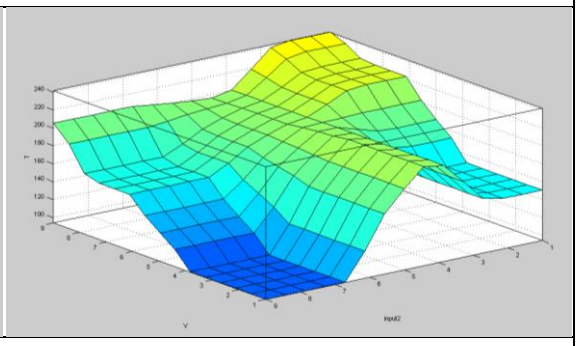
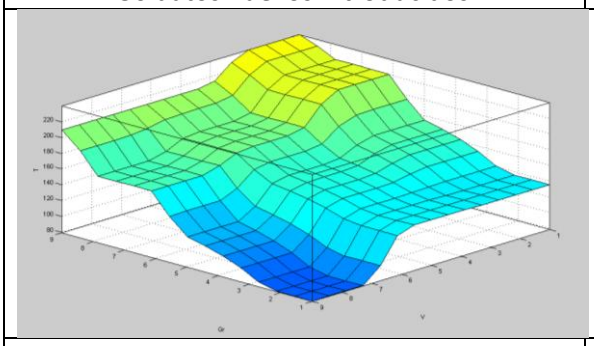
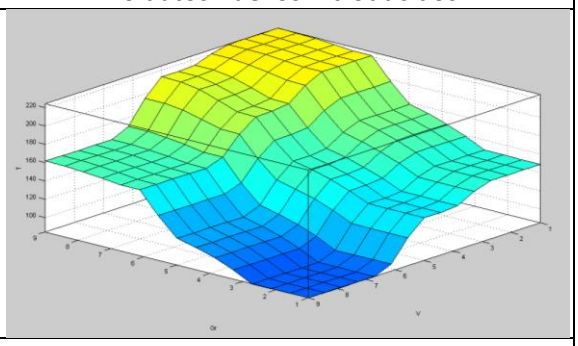
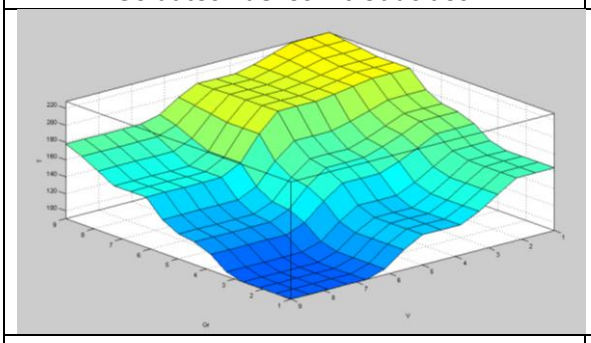
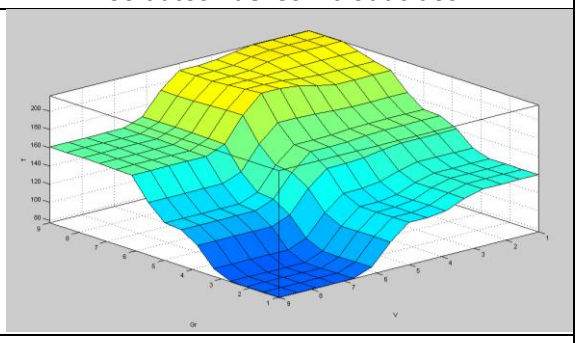
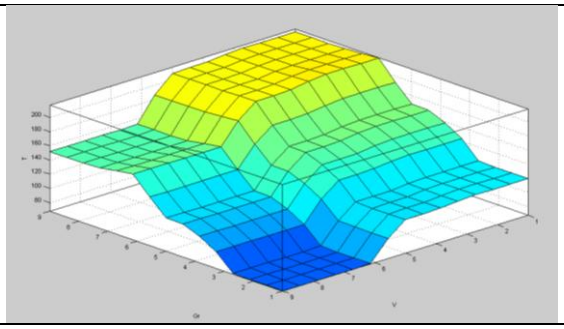
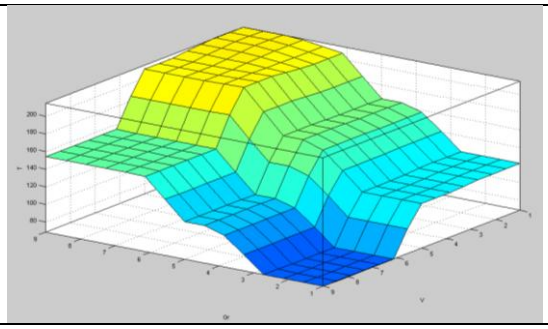
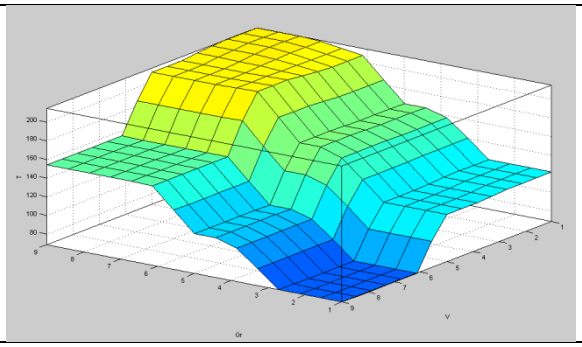
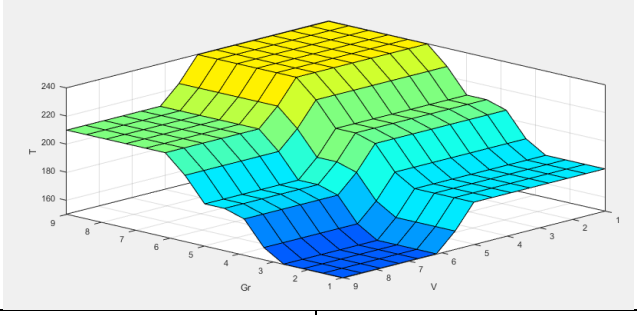
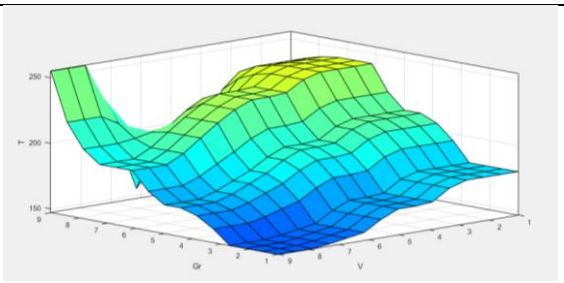
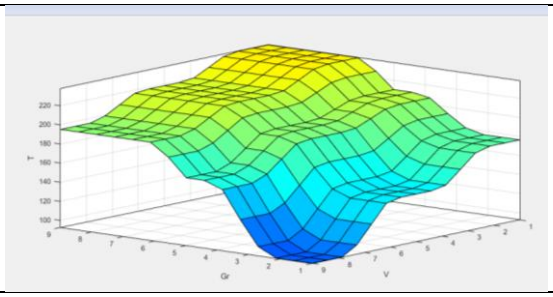
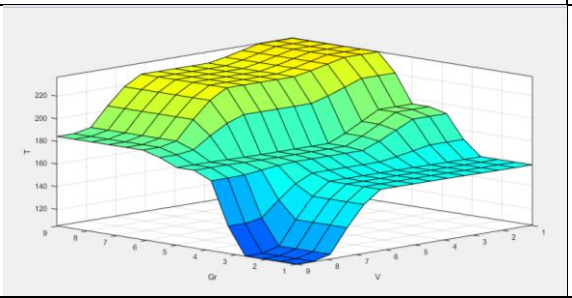
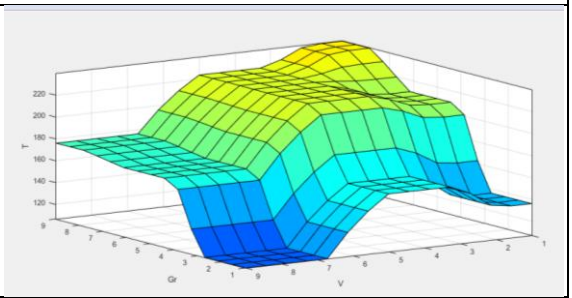
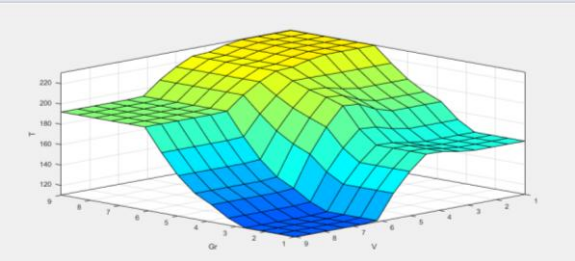
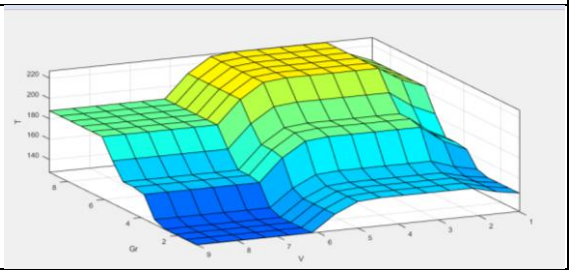
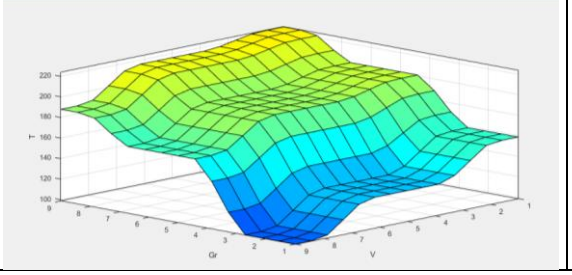
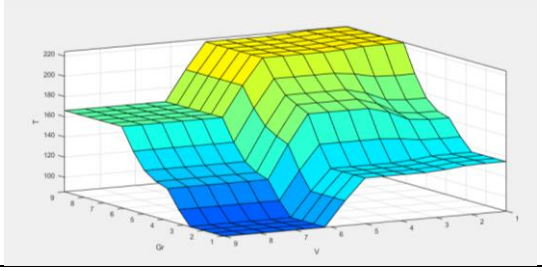
10 datos nuevos introducidos	20 datos nuevos introducidos
	
30 datos nuevos introducidos	40 datos nuevos introducidos
	
50 datos nuevos introducidos	60 datos nuevos introducidos
	
100 datos nuevos introducidos	200 datos nuevos introducidos
	
UE2 T	
	

Figura 2.46

En este caso, se puede decir que con 60 datos nuevos, el sistema ya se parece al del usuario 2. A partir de 60 datos, las superficies que se generan son prácticamente iguales a la del usuario 2.

Dicho esto, se analizará la evolución de las superficies para el controlador con 100 datos iniciales de UE1.

<table>
<tr><td colspan="2">Inicialmente 100 datos de UE1</td></tr>
<tr><td colspan="2">  </td></tr>
<tr><td>10 datos nuevos introducidos</td><td>20 datos nuevos introducidos</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>30 datos nuevos introducidos</td><td>40 datos nuevos introducidos</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>50 datos nuevos introducidos</td><td>60 datos nuevos introducidos</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>100 datos nuevos introducidos</td><td>200 datos nuevos introducidos</td></tr>
<tr><td>  </td><td>  </td></tr>
</table>

UE2 T

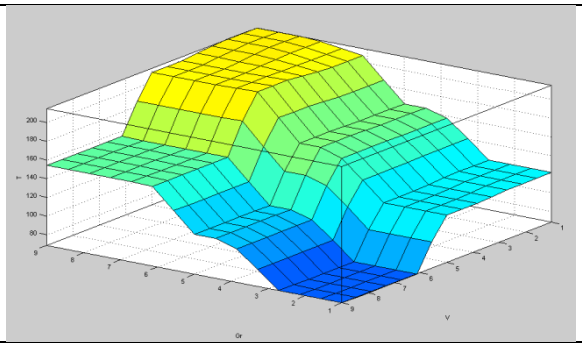


Figura 2.47

En comparación con el controlador de 16 datos de UE1, se ve que con 100 datos tarda mucho más en transformarse en la superficie de UE2. Esto ocurre porque, para pasar de UE1 a UE2, se necesita sustituir todos los valores, y puede que alguno no se sustituya. Los universos "grasas" y "velocidad" están acotados entre 1 y 9, por lo tanto existen 81 datos diferentes posibles. Si se introducen 100 valores aleatorios de UE2, es muy probable que algún dato sea repetido y que no se sustituyan muchos de los datos pertenecientes a UE1. Por eso se necesitan más datos de UE2.

2.5.2.2.- Controlador de Bolus N

Tal y como se ha hecho con el controlador de T, se quiere ver la evolución del sistema de dos controladores; uno con 16 datos de UE1 y otro con 100 datos de UE1. Para visualizar la transformación, se irán introduciendo los datos del usuario 2 cada 10 y en caso de que se repitan las entradas de los datos nuevos y viejos, los viejos serán sustituidos por los nuevos. Después se procederá al entrenamiento con estos datos.

En la siguiente tabla se muestra cómo cambian los consecuentes según se van introduciendo y sustituyendo los datos.

Partiendo desde un FIS generado con 16 datos representativos de UE1					Partiendo desde FIS generado con 100 datos				
0 nuevos datos									
BN	V								
Gr		Baja	Media	Alta					
	Baja	33.3	56	77.8					
	Media	30	50	66.7					
	Alta	30	44.4	60					
10 nuevos datos									
BN	V				BN	V			
Gr		Baja	Media	Alta	Gr		Baja	Media	Alta
	Baja	35.0575	75.1434	80.4123		Baja	32.4447	69.3766	78.5151
	Media	52.7639	66.6295	64.8928		Media	32.2918	60.5646	63.7270
	Alta	30.0000	67.5000	59.9999		Alta	31.5288	51.7828	66.1429
20 nuevos datos									
BN	V				BN	V			
Gr		Baja	Media	Alta	Gr		Baja	Media	Alta
	Baja	33.3000	69.4571	77.8152		Baja	43.5883	74.0790	76.6534
	Media	55.6917	72.7143	68.1917		Media	36.9078	67.4579	62.8941
	Alta	39.9728	59.9779	59.9857		Alta	31.8353	61.7912	65.6095

30 nuevos datos

BN	V				BN	V		
Gr		Baja	Media	Alta	Gr		Baja	Media	Alta
	Baja	33.3000	69.2746	77.8155		Baja	46.6633	72.7136	74.0254
	Media	57.2200	73.6271	68.1901		Media	35.4886	69.4463	63.3422
	Alta	39.9701	59.9763	59.9857		Alta	31.4380	68.6103	65.8333

40 nuevos datos

BN	V				BN	V		
Gr		Baja	Media	Alta	Gr		Baja	Media	Alta
	Baja	54.9179	69.1071	77.8158		Baja	48.3804	71.3012	73.3210
	Media	58.9387	74.4643	68.1885		Media	36.0910	69.8505	65.6375
	Alta	39.9672	59.9749	59.9857		Alta	31.4280	68.4216	65.8333

50 nuevos datos

BN	V				BN	V		
Gr		Baja	Media	Alta	Gr		Baja	Media	Alta
	Baja	54.8846	68.9140	77.8162		Baja	51.4841	71.2475	73.1550
	Media	59.2385	73.9819	70.3934		Media	37.7283	71.5909	65.6375
	Alta	39.9666	69.9931	59.9636		Alta	32.2120	69.2735	66.9231

60 nuevos datos

BN	V				BN	V		
Gr		Baja	Media	Alta	Gr		Baja	Media	Alta
	Baja	54.8019	69.0776	77.8159		Baja	51.3141	71.4242	73.0048
	Media	59.9828	73.5578	70.3942		Media	40.0447	71.9110	65.6375
	Alta	39.9653	69.9938	59.9636		Alta	32.8564	68.0585	67.8571

70 nuevos datos

BN	V				BN	V		
Gr		Baja	Media	Alta	Gr		Baja	Media	Alta
	Baja	54.8019	69.3461	77.8154		Baja	51.2647	72.6074	72.4208
	Media	59.9828	72.6154	70.3959		Media	39.6246	74.1160	67.2333
	Alta	44.9740	69.9954	59.9636		Alta	34.3963	71.0846	68.6667

80 nuevos datos

BN	V				BN	V		
Gr		Baja	Media	Alta	Gr		Baja	Media	Alta
	Baja	54.7425	69.5087	77.8151		Baja	52.0519	72.1590	72.2346
	Media	60.5173	72.4566	70.3962		Media	40.0999	74.4296	68.5100
	Alta	44.9731	71.6653	59.9605		Alta	34.3776	72.8909	70.5556

90 nuevos datos

BN	V				BN	V		
Gr		Baja	Media	Alta	Gr		Baja	Media	Alta
	Baja	54.7124	69.4478	77.8152		Baja	51.9429	72.5972	72.2346
	Media	60.7879	72.5153	70.3961		Media	42.9824	74.1066	69.5545
	Alta	47.9778	71.6652	59.9605		Alta	35.0552	73.0736	71.9048

100 nuevos datos

BN	V				BN	V		
Gr		Baja	Media	Alta	Gr		Baja	Media	Alta
	Baja	54.7010	69.5485	76.4243		Baja	53.6201	72.0181	72.2346
	Media	60.8910	72.0569	69.1358		Media	46.0519	72.1109	69.5545
	Alta	47.9776	71.6660	59.9627		Alta	37.1442	72.8584	71.9048

200 nuevos datos

BN	V				BN	V		
Gr		Baja	Media	Alta	Gr		Baja	Media	Alta
		68.7286	70.0315	70.8122		Baja	57.2962	65.9208	73.3312
		65.0000	69.9253	64.6681		Media	52.2468	65.2012	68.9928
		56.2500	74.9226	78.7116		Alta	58.9292	60.7029	73.4398

300 nuevos datos									
BN	V				BN	V			
Gr		Baja	Media	Alta	Gr		Baja	Media	Alta
		70.2941	70.0378	70.4857		Baja	60.6532	71.0082	72.4963
		65.0000	69.9621	64.8179		Media	62.6745	63.5472	63.9943
		57.8571	74.9501	78.8381		Alta	54.3739	73.4264	76.4597

Usuario 2				
BN	V			
Gr		Baja	Media	Alta
	Baja	70	70	70
	Media	65	70	65
	Alta	60	75	80

Tabla 2.23

Como ocurre con el controlador de T, el sistema tiende a evolucionar a UE2, pero el sistema con 100 datos iniciales de UE1 necesita más datos para llegar a UE2. Con 200 datos introducidos/sustituidos, en el caso de 16 datos iniciales de UE1, los consecuentes son prácticamente los mismos, mientras que el sistema con 100 datos iniciales de UE1 sus consecuentes se alejan de los consecuentes de UE2.

A continuación se analizará la evolución del error en función de los datos que se introducen.

Sistema de 16 datos de UE1

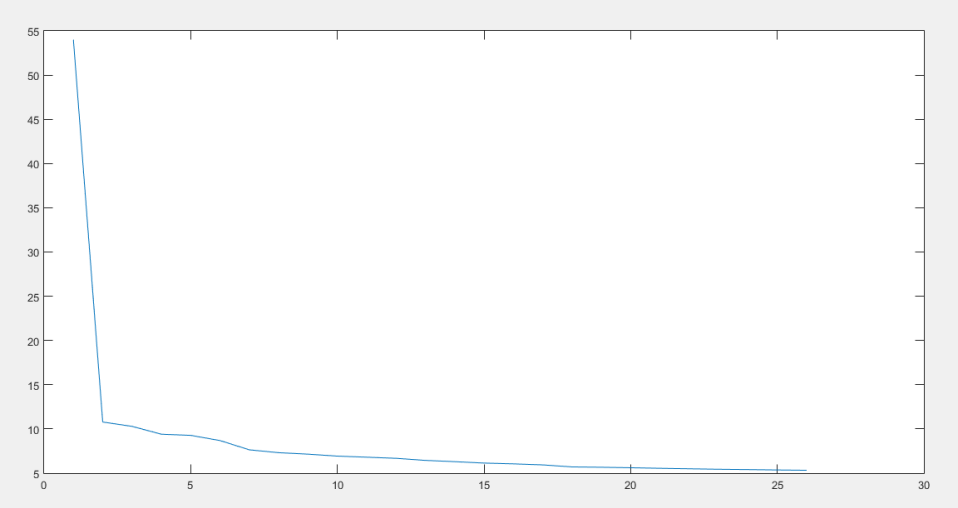


Figura 2.48

En la gráfica superior se ve cómo se va reduciendo el erro hasta que, después de haber introducido 250 datos nuevos, se estanca en **5.3276**.

Sistema de 100 datos de UE1

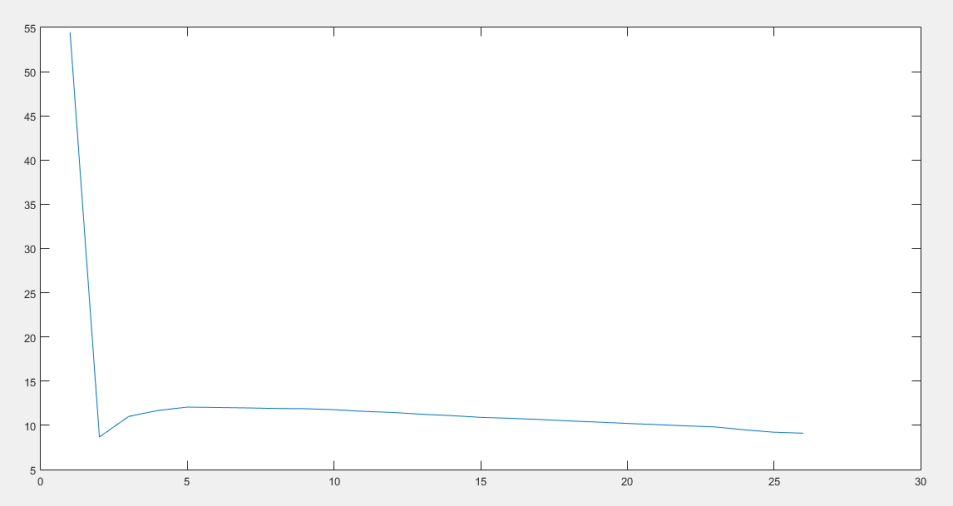
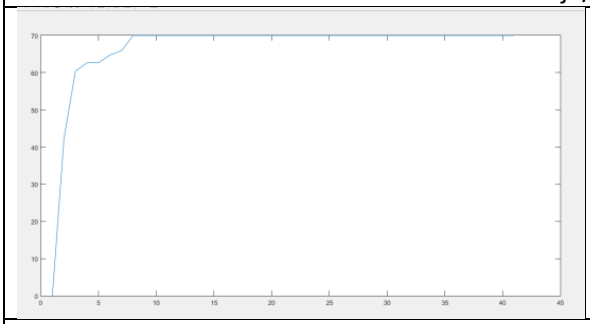
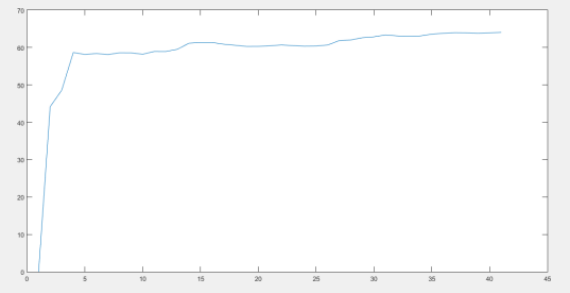
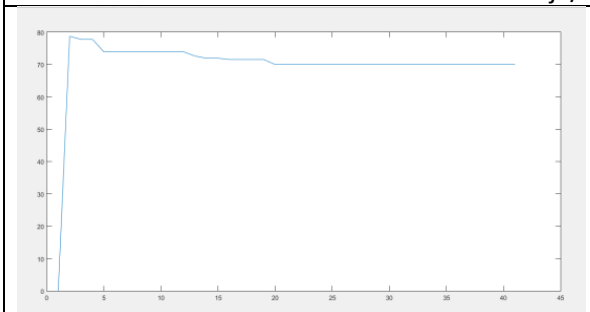
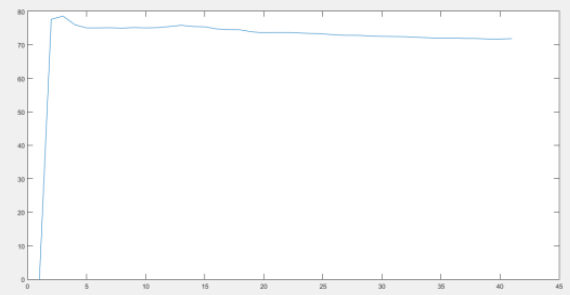


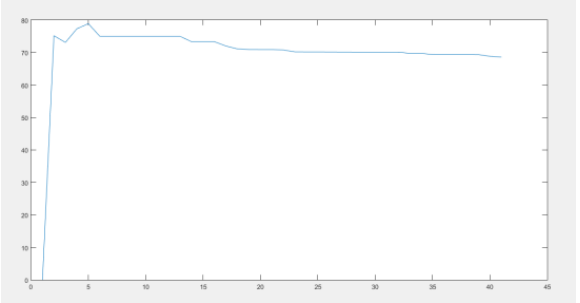
Figura 2.49

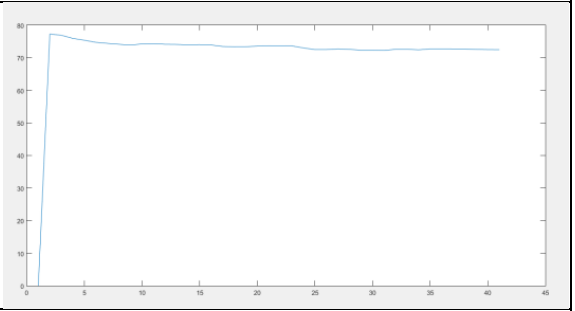
En la primera vez que se introducen los datos, el error se reduce mucho, siendo casi el mínimo. Después vuelve a aumentar pero seguidamente vuelve a disminuir poco a poco. Cuando se han añadido 250 datos, el valor del error es de **9.1077**.

En las siguientes figuras se representa la evolución de los consecuentes en función del número de datos que se introducen del nuevo usuario (UE2) para el aprendizaje.

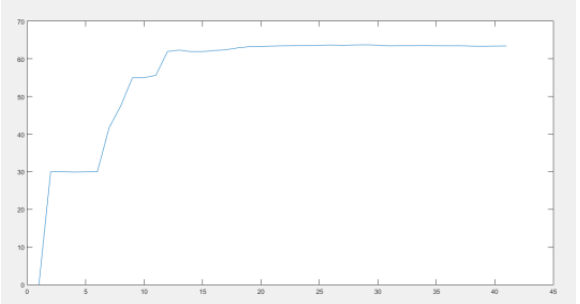
16 datos representativos	100 datos aleatorios
Gr baja/ V baja	
	
Gr baja/ V media	
	

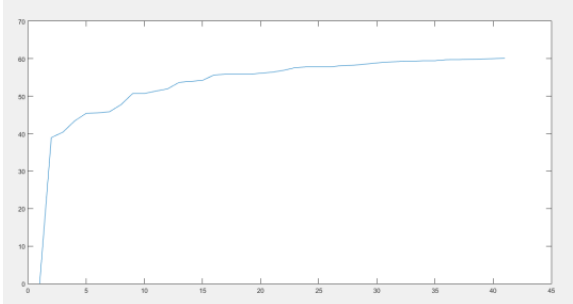
Gr baja/ V alta



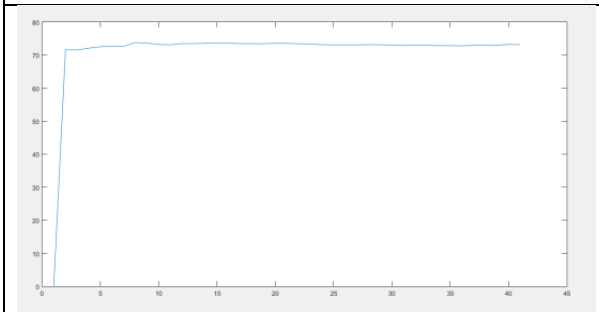


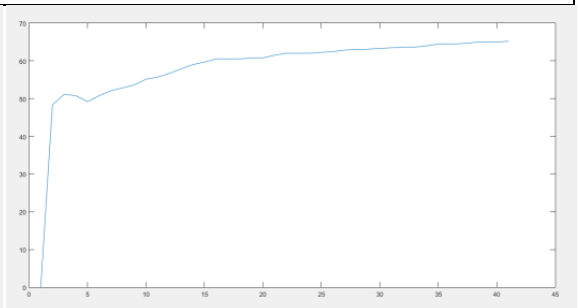
Gr media / V baja



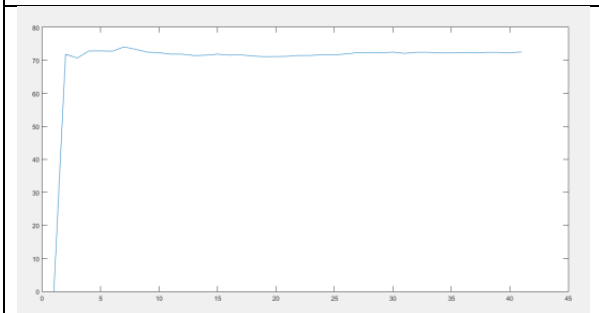


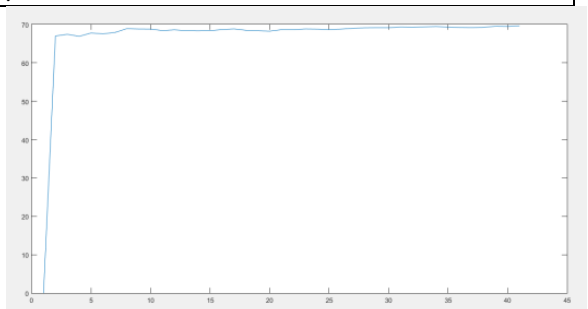
Gr media / V media



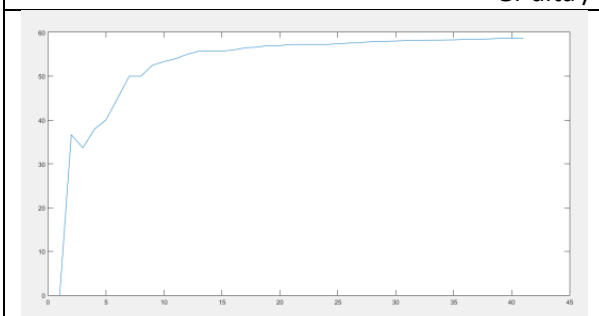


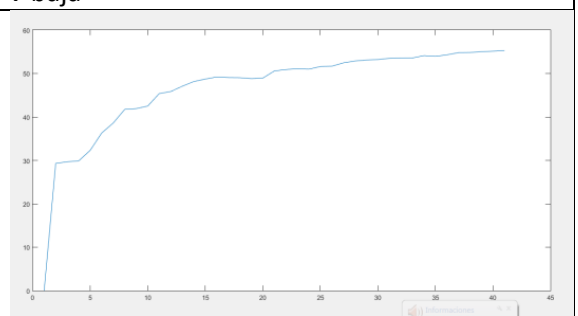
Gr media / V alta





Gr alta / V baja





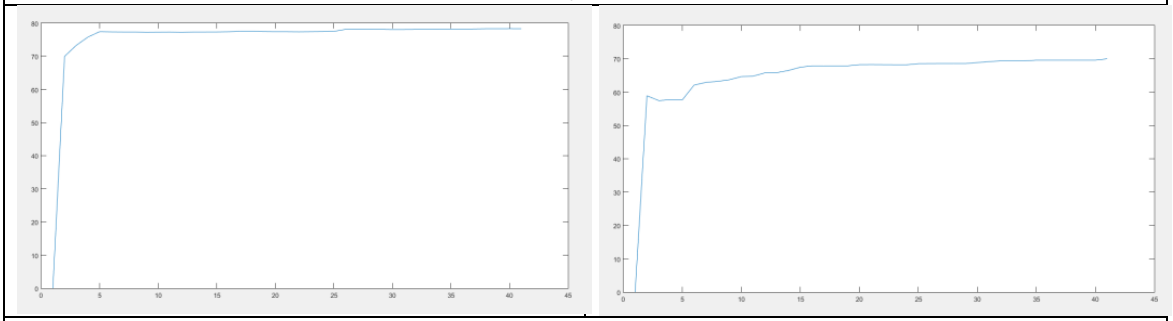
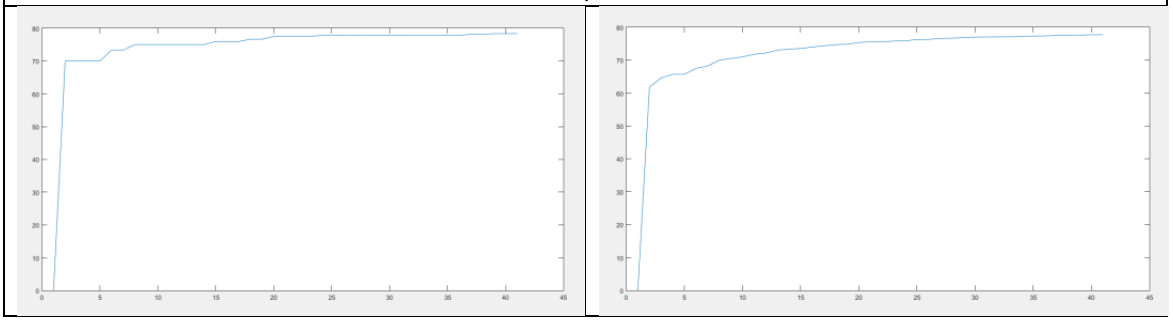
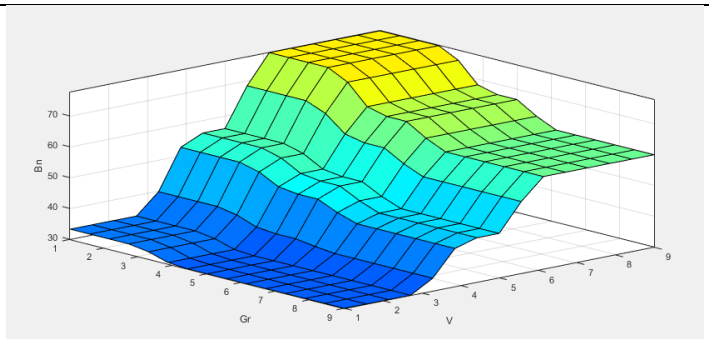
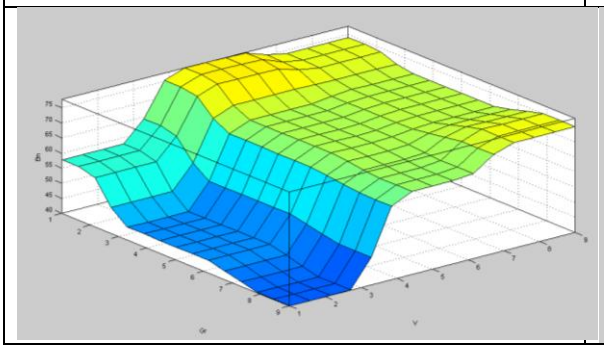
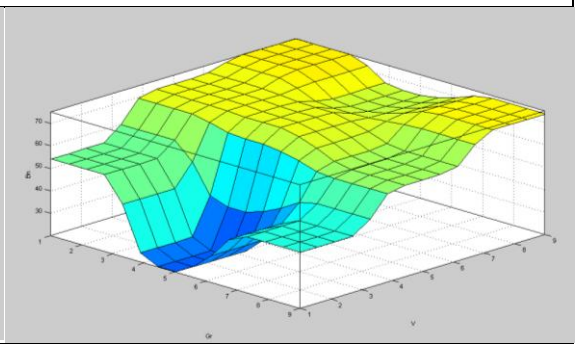
Gr alta / V media

Gr alta / V alta


Figura 2.50

En la figura superior se ve como la mayoría de los consecuentes se estabilizan cuando se introducen 150 datos nuevos del usuario 2.

En las siguientes figuras se puede ver como se modifican las superficies mediante el aprendizaje con nuevos datos y sustituyéndolos por los anteriores de UE1.

Inicialmente 16 datos de UE1	
	
10 datos nuevos introducidos	20 datos nuevos introducidos
	

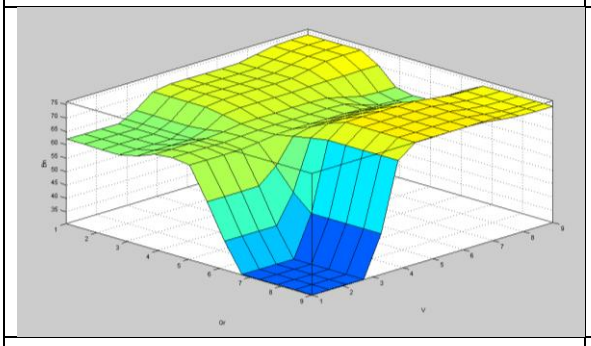
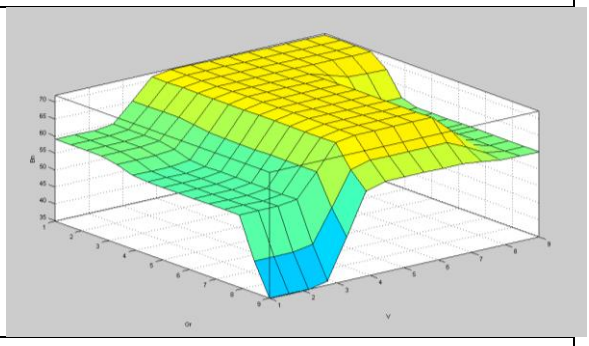
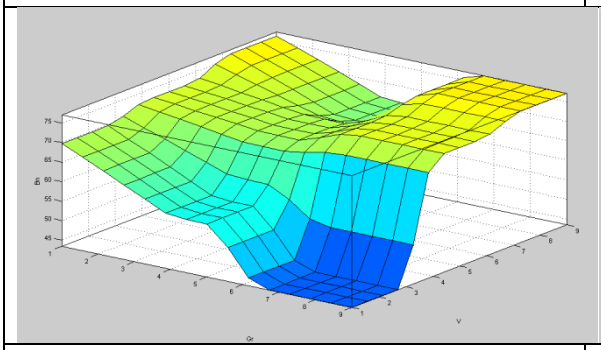
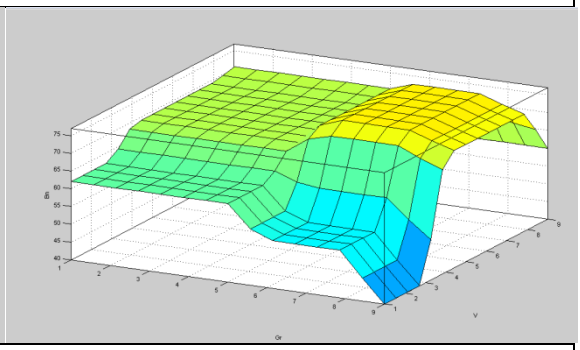
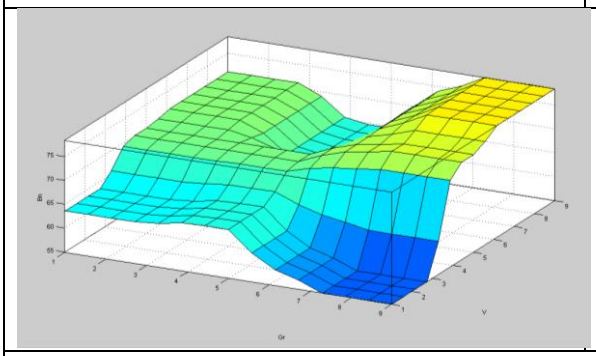
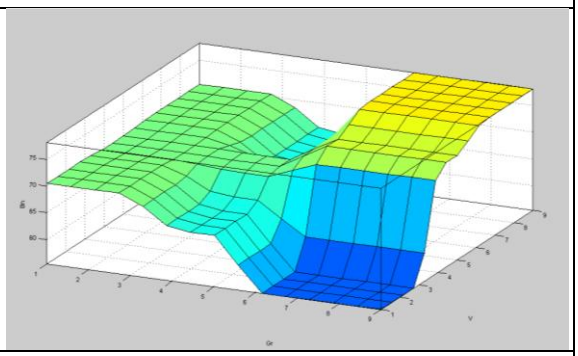
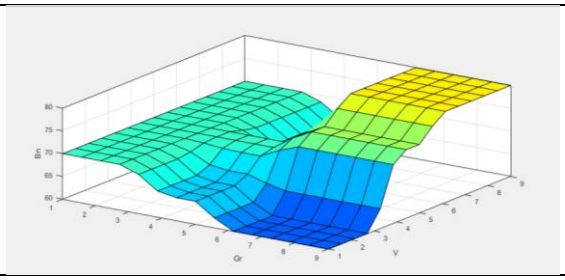
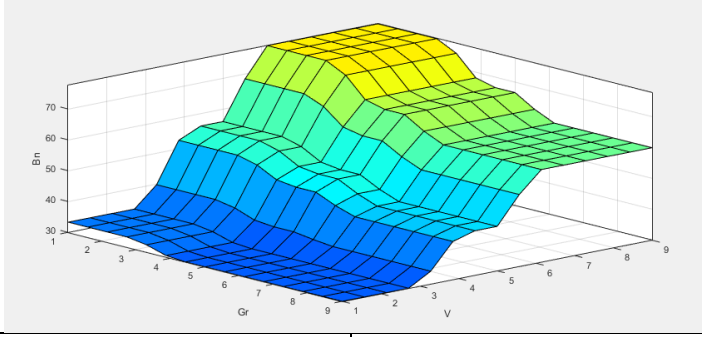
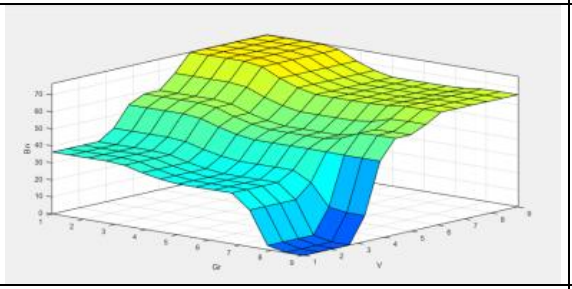
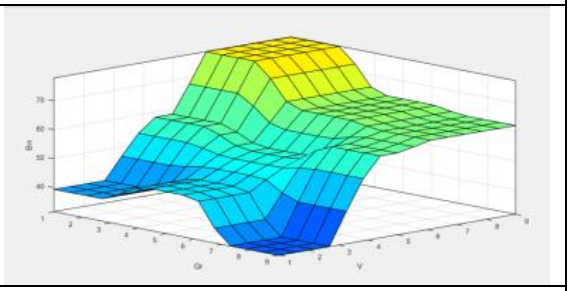
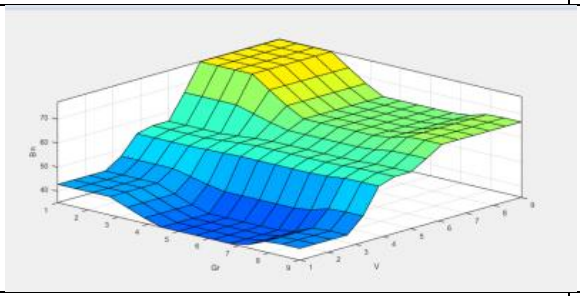
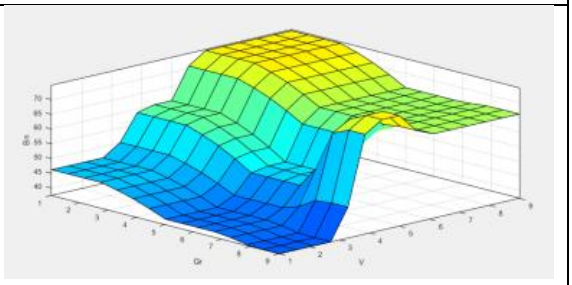
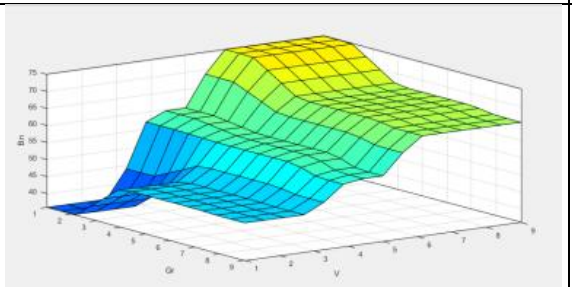
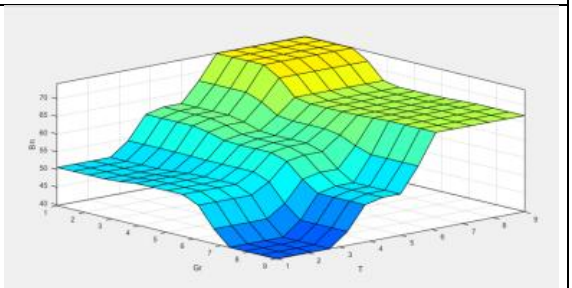
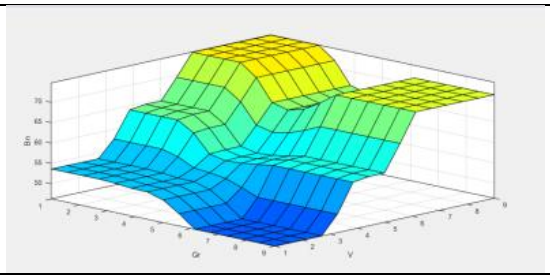
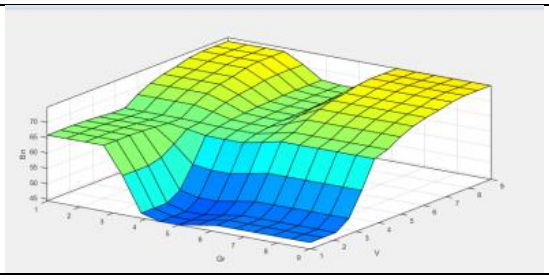
30 datos nuevos introducidos	40 datos nuevos introducidos
	
50 datos nuevos introducidos	60 datos nuevos introducidos
	
100 datos nuevos introducidos	200 datos nuevos introducidos
	
UE2 Bn	
	

Figura 2.51

Inicialmente 100 datos de UE1	
	
10 datos nuevos introducidos	20 datos nuevos introducidos
	
30 datos nuevos introducidos	40 datos nuevos introducidos
	
50 datos nuevos introducidos	60 datos nuevos introducidos
	
100 datos nuevos introducidos	200 datos nuevos introducidos
	

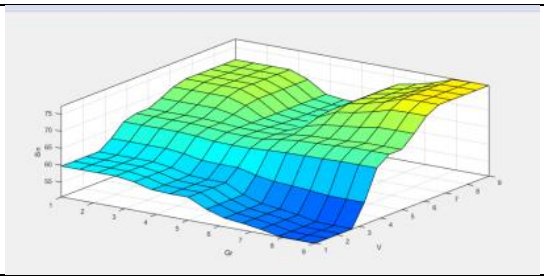
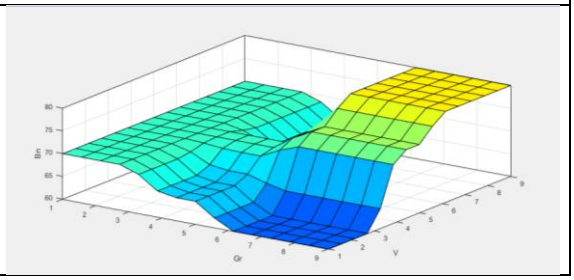
300 datos nuevos introducidos	UE2 Bn
	

Figura 2.52

En este controlador se ve mejor que al tener más datos de UE1, tarda más en transformarse en UE2 porque la superficie de UE1 y la de UE2 son muy diferentes. Como se ha comentado antes, esto es debido a que puede que no se sustituyan todos los datos de UE1 y se quede alguno, infiriendo así en la transformación.

2.5.2.3.- Comentarios

En un inicio, mediante el método de aprendizaje sin sustitución, el resultado una mezcla entre los controladores de UE1 y UE2.

Se necesita gran cantidad de datos. Para llegar a las exigencias del usuario nuevo, pero se puede ver que se va aproximando. Con el que menos datos se necesita para la transformación es con 16 datos de UE1 de inicio. Como ya se ha comentado, en la sustitución simplemente con que se sustituyan esos valores, el sistema tiende al usuario 2, mientras que sin sustitución, se necesitan más datos para "ignorar" los datos del usuario 1.

Cuantas más iteraciones se realicen y más datos se manejen, más comúnmente ocurre que los puntos de los trapecios se solapen y el programa se pare.

2.5.2.4.- Sustitución de datos

La decisión de sustitución de valores antiguos por valores nuevos introducidos por un nuevo usuario fue un punto que generó muchas dudas. No hay duda que para que el sistema se ajuste al nuevo usuario es necesario que este introduzca los datos que son válidos para él.

La cuestión era como asegurarse que el usuario introducía valores correctos y así, modificar el sistema a sus necesidades.

Una idea fue hacer lo siguiente;

Desde pantalla introducir platos.

El sistema evalúa y propone un valor de T y Bolus N para esa situación.

Pregunta si el usuario está de acuerdo.

Si lo está, no se introducen nuevos datos al sistema.

Si no está de acuerdo, el programa pide por pantalla que introduzca los valores de T y Bolus N deseados y los guarda en matriz de datos.

En el caso de introducir un conjunto de datos que llevan a resultados no satisfactorios para el usuario, se decidió que los datos introducidos por usuario fueran almacenados para poder borrarlos en el futuro.

2.5.3.- Conclusiones

El método con el que mejor se realiza el ajuste es mediante sustitución, ya que sin sustituir se necesitan más datos para llegar al usuario destino.

3.- Conclusiones generales

El programa es capaz de hacer evolucionar el controlador hasta otro controlador independientemente de cuáles sean sus condiciones iniciales.

Se ha visto que el programa de entrenamiento de sistemas neuroborrosos de Matlab puede ser utilizado para ajustar con éxito el controlador borroso contenido en el sistema de decisión para ayuda de usuarios de insulina a las necesidades de un usuario nuevo.

Respecto a los datos necesarios, el sistema es capaz de hacer evolucionar el controlador si se dispone de suficientes datos representativos.

En cuanto al número de iteraciones de aprendizaje, se ha visto que se puede conseguir un buen ajuste del sistema con muy pocas iteraciones.

Cuando se entrena con muchos datos o con muchas iteraciones, el programa se para en medio del aprendizaje debido a que ocurre un error con los antecedentes trapezoidales. A veces incluso se para cuando se trabaja con pocos datos.

Aunque no es propiamente un fallo, es incomodo que partiendo de un controlador con un determinado rango de existencia de las funciones de pertenencia, el método de entrenamiento de Matlab reduce el rango en caso de no tener datos de entrenamiento que abarque todo el universo, y no exista una opción para forzar a no hacerlo.

4.- Bibliografía

1. The MathWorks, Inc. *Fuzzy Logic Toolbox User´s Guide* [en línea], 2ª ed. Revisada. Versión 2.2.22. Natick, MA, (E.E.U.U.)TheMathWorks, Inc. 1995-2015 [Revisado: Septiembre 2015], [Consulta: Septiembre 2015]. [368 diapositivas]. Disponible en: http://cn.mathworks.com/help/pdf_doc/fuzzy/fuzzy.pdf
2. Harris M, Zimmet P. *Classification of diabetes mellitus and other categories of glucose intolerance*. In Alberti K, Zimmet P, Defronzo R, editors. International Textbook of Diabetes Mellitus. SecondEdition. Chichester: John Wiley and SonsLtd; 1997. p9-23.
3. Jang, J.-S. R., Sun, C.-T. y Mizutani, E. *Neuro-Fuzzy and Soft Computing: a computational approach to learning and machine intelligence*. Prentice Hall (de. Lit.); E.E.U.U. Pearson Education 1997; ISBN 0-13-261066-3; P95-97.
4. Recarte Mendiburu, L. (2013) . Desarrollo de un Sistema de Decisión basado en Lógica Borrosa para el uso de Bombas de Insulina. [Figura 1.14]
5. UNED. Facultad de Ciencias. *Nutrición y Dietética*[en línea] 2015 [Consulta: Noviembre 2015]. Disponible en: http://www.uned.es/pea-nutricion-y-dietetica-I/guia/enfermedades/diabetes/index.htm

5.- Anexos

Ejercicio 2.3

```
%datos de comida
V=[3 3 3 6 5 5 8 5 6 7 5 8 8]'
Gr=[2 4 7 2 4 7 2 3 4 4 2 4 7]'
%Bn=[33.3 33.3 30 72.5 55.56 50 77.78 61.54 64.71 66.67 66.67 77.78
60]'
T=[180 210 240 150 210 240 120 210 180 180 240 180 240]'


data_T=[Gr V T]
trn_T=data_T
chk_T=data_T

numfs1=3;
numfs2=3;
nummfs=[numfs1 numfs2]
mftypein='trapmf'
mftypeout='constant'%el valor de salida tiene que ser una constante

%Se genera un FIS inicial
fismatT = genfis1(data_T,nummfs,mftypein,mftypeout)
%Se amplían las funciones de pertenencia
fismatT.input(1).range=[1 9];
fismatT.input(2).range=[1 9];

fismatT.input(1).mf(1).params(1:2)=1;
fismatT.input(1).mf(numfs1).params(3:4)=9;

fismatT.input(2).mf(1).params(1:2)=1;
fismatT.input(2).mf(numfs2).params(3:4)=9;

%Se entrena el fismatT con los datos de trn_T
numepochs=500 %número de iteraciones
[fismat1T, trnErrT, steperrorT, chkfis_T, chkErrT]=anfis(trn_T,
fismatT, numepochs, NaN, chk_T)

%Se amplían las funciones de pertenencia
fismat1T.input(1).range=[1 9];
fismat1T.input(2).range=[1 9];

fismat1T.input(1).mf(1).params(1:2)=1;
fismat1T.input(1).mf(numfs1).params(3:4)=9;

fismat1T.input(2).mf(1).params(1:2)=1;
fismat1T.input(2).mf(numfs2).params(3:4)=9;

%calcular salidas en los datos de entrada de entrenamiento
Tout_trn=evalfis([Gr V],fismat1T)
[T Tout_trn]

T_trnRMSE=norm(Tout_trn-T)/sqrt(length(Tout_trn))

matriz_reglasT=zeros(numfs1, numfs2)
for k=1:numfs1*numfs2%matriz de 9 componentes
    %indice del consecuente de la regla k (1, 2 3 ...9)
    indice=fismat1T.rule(k).consequent;
    valor_consecuente=fismat1T.output.mf(indice).params;
```

```
    %fismat1.rule(k).antecedent es la posicion donde se va a escribir
%el consecuente constante
    %de la regla k
    %fismat1.rule(k).antecedent es un vector (1 1 para el primer valor,
%1 2 para el segundo etc)
    %el cual se ocupa de situar en su posicion de la matriz el valor
%de salida
    %fismat1T.rule(k).antecedent(1) coge el valor de la 1ª columna %
(marca la fila)
    %fismat1T.rule(k).antecedent(2) coge el valor de la 2ª columna del
%vector (marca la columna)


matriz_reglasT(fismat1T.rule(k).antecedent(1),fismat1T.rule(k).anteced
ent(2))=valor_consecuente
    pause
end
```

Ejercicio 2.4

```
%fismat1T directamente de Sujeto1T
fismatT=readfis('Sujeto1T.fis')

fismatUE2T=readfis('Sujeto2T.fis')
%generar entradas aleatorias de UE2
entradas=round(1+rand(100,2)*8);

%generar entradas representativas de UE2
% cont=1
% for i=1:4:9
%     for j=1:4:9
%         entradas(cont,:)=[i j];
%         cont=cont+1;
%     end
% end
salidasUE2T=evalfis(entradas,fismatUE2T);


%crear una matriz con los datos del usuario 2
data_ue2=[entradas salidasUE2T];
chk_T=data_ue2;

%Entrenamiento
numepochs=1000;
[fismat1T, trnErrT, steperrorT, chkfis_T, chkErrT]=anfis(data_ue2,
fismatT, numepochs, NaN, chk_T);

%calcular salidas en los datos de entrada de entrenamiento
Tout_trn=evalfis(entradas,fismat1T);
[salidasUE2T Tout_trn];

matriz_reglasT=zeros(numfs1,numfs2)
for k=1:numfs1*numfs2%matriz de 9 componentes
    %indice del consecuente de la regla k (1, 2 3 ...9)
    indice=fismat1T.rule(k).consequent;
    valor_consecuente=fismat1T.output.mf(indice).params;
    %fismat1.rule(k).antecedent es la posicion donde se va a escribir
%el consecuente constante
    %de la regla k
```

```
    %fismat1.rule(k).antecedent es un vector (1 1 para el primer valor,
%1 2 para el segundo etc)
    %el cual se ocupa de situar en su posicion de la matriz el valor
%de salida
    %fismat1T.rule(k).antecedent(1) coge el valor de la 1ª columna %
(marca la fila)
    %fismat1T.rule(k).antecedent(2) coge el valor de la 2ª columna del
%vector(marca la columna)
matriz_reglasT(fismat1T.rule(k).antecedent(1),fismat1T.rule(k).anteced
ent(2))=valor_consecuente
    %pause
End
```

Ejercicio 2.5

2.5.1.- Sin sustitución

```
%GENERAR 100 DATOS DEL USUARIO 1
%generar entradas
% entradasUE1T=round(1+rand(100,2)*8)
%en lugar de generar 100 datos, se generan 16 de UE1, para ver cuantos
%datos de UE2 se necesitan
valores=[1 3 6 9]
cont=1
for i=1:4
    for j=1:4
       entradasUE1T(cont,:)=[valores(i) valores(j)]
       cont=cont+1
    end
end

salidasUE1T=evalfis(entradasUE1T,fismatUE1T)
%crear una matriz con los datos del usuario 1
data_UE1T=[entradasUE1T salidasUE1T]


%GENERACIÓN DEL FISMAT DESDE DATOS
numfs1=3;
numfs2=3;
nummfs=[numfs1 numfs2]
mftypein='trapmf'
mftypeout='constant'%el valor de salida tiene que ser una constante
fismatUE1T = genfis1(data_UE1T,nummfs,mftypein,mftypeout)

%CREAR 100 DATOS DE UE2
fismatUE2T=readfis('Sujeto2T.fis')
%generar entradas
entradasUE2T=round(1+rand(100,2)*8);
salidasUE2T=evalfis(entradasUE2T,fismatUE2T);
%crear una matriz con los datos del usuario 2
data_UE2T=[entradasUE2T salidasUE2T];


incremento=10;
for a=1:length(entradasUE2T)/incremento+1

    if a==1
        data_trn=data_UE1T;
        fismat1T=fismatUE1T;
    else
```

```
        %selecciona las filas 11-20, 21-30...
        b=a*incremento-(2*incremento-1)
        c=b+(incremento-1)

        %data=datos de UE2 desde la fila b a la c
        data=data_UE2T(b:c,:)
        data_trn=[data_trn;data]
        chk_T=data_trn
        numepochs=200
        [fismat1T, trnErrT, steperrorT, chkfis_T,
chkErrT]=anfis(data_trn, fismatUE1T, numepochs, NaN, chk_T)
    end



    %Error de fismatT respecto a datos
    Tout_trn=evalfis(data_trn(:,1:2),fismat1T);


    T_trnRMSE=norm(Tout_trn-data_trn(:,3))/sqrt(length(Tout_trn))

    matriz_reglasT=zeros(numfs1,numfs2)
    for k=1:numfs1*numfs2%matriz de 9 componentes
        %indice del consecuente de la regla k (1, 2 3 ...9)
        indice=fismat1T.rule(k).consequent;
        valor_consecuente=fismat1T.output.mf(indice).params;
        %fismat1.rule(k).antecedent es la posicion donde se va a
%escribir el consecuente constante
        %de la regla k
        %fismat1.rule(k).antecedent es un vector (1 1 para el primer
%valor, 1 2 para el segundo etc)
        %el cual se ocupa de situar en su posicion de la matriz el
%valor de salida
        %fismat1T.rule(k).antecedent(1) coge el valor de la 1ª columna
% (marca la fila)
        %fismat1T.rule(k).antecedent(2) coge el valor de la 2ª columna
del vector(marca la columna)


matriz_reglasT(fismat1T.rule(k).antecedent(1),fismat1T.rule(k).anteced
ent(2))=valor_consecuente

fismatUE1T=fismat1T
end
```

Ejercicio 2.5

2.5.2.- Con sustitución

```
%GENERAR 100 DATOS DEL USUARIO 1
fismatUE1T=readfis('Sujeto1T.fis')
%generar entradas
entradasUE1T=round(1+rand(100,2)*8)

%en lugar de generar 100 datos, se generan 16 de UE1, para ver cuantos
%datos de UE2 se necesitan
% valores=[1 3 6 9]
```

```
% cont=1
% for i=1:4
%     for j=1:4
%        entradasUE1T(cont,:)=[valores(i) valores(j)]
%        cont=cont+1
%     end
% end

salidasUE1T=evalfis(entradasUE1T,fismatUE1T)
%crear una matriz con los datos del usuario 1
data_UE1T=[entradasUE1T salidasUE1T]


%GENERACIÓN DEL FISMAT DESDE DATOS
numfs1=3;
numfs2=3;
nummfs=[numfs1 numfs2]
mftypein='trapmf'
mftypeout='constant'%el valor de salida tiene que ser una constante
fismatUE1T = genfis1(data_UE1T,nummfs,mftypein,mftypeout)

%CREAR 100 DATOS DE UE2
fismatUE2T=readfis('Sujeto2T.fis')
%fismatUE2T=readfis('F:\Proyecto\5DatosCada10\Sujeto2T.fis');
%generar entradas
entradasUE2T=round(1+rand(200,2)*8);
salidasUE2T=evalfis(entradasUE2T,fismatUE2T);


salidasUE2T=evalfis(entradasUE2T,fismatUE2T)
%crear una matriz con los datos del usuario 2
data_UE2T=[entradasUE2T salidasUE2T]

incremento=200;
for a=1:length(entradasUE2T)/incremento+1

    if a==1
        data_trn=data_UE1T;
        fismat1T=fismatUE1T;
    else
        %selecciona las filas 11-20, 21-30...
        b=a*incremento-(2*incremento-1)
        c=b+(incremento-1)


        %data=datos de UE2 desde la fila b a la c
        data=data_UE2T(b:c,:)

        %Sustituir valores viejos por nuevos
        for i=1:length(data)
            val1=data(i,1)
            val2=data(i,2)
            val3=data(i,3)

            [filA, colA]=ind2sub(size(data_trn(:,1)),val1)
            [filB, colB]=ind2sub(size(data_trn(:,2)),val2)
            if filA==filB
                data_trn(filA,3)=val3
            end
```

```
        end


        data_trn=[data_trn;data]
        chk_T=data_trn
        numepochs=200
        [fismat1T, trnErrT, steperrorT, chkfis_T,
chkErrT]=anfis(data_trn, fismatUE1T, numepochs, NaN, chk_T)

    end


     Tout_trn=evalfis(data_trn(:,1:2),fismat1T);


    T_trnRMSE=norm(Tout_trn-data_trn(:,3))/sqrt(length(Tout_trn))

    matriz_reglasT=zeros(numfs1,numfs2)
    for k=1:numfs1*numfs2%matriz de 9 componentes
        %indice del consecuente de la regla k (1, 2 3 ...9)
        indice=fismat1T.rule(k).consequent;
        valor_consecuente=fismat1T.output.mf(indice).params;
        %fismat1.rule(k).antecedent es la posicion donde se va a
%escribir el consecuente constante
        %de la regla k
        %fismat1.rule(k).antecedent es un vector (1 1 para el primer
%valor, 1 2 para el segundo etc)
        %el cual se ocupa de situar en su posicion de la matriz el
valor de salida
        %fismat1T.rule(k).antecedent(1) coge el valor de la 1ª columna
% (marca la fila)
        %fismat1T.rule(k).antecedent(2) coge el valor de la 2ª columna
%del vector(marca la columna)


matriz_reglasT(fismat1T.rule(k).antecedent(1),fismat1T.rule(k).anteced
ent(2))=valor_consecuente
    end

fismatUE1T=fismat1T
end
```