1 OBJETIVO

El objetivo de este proyecto es. analizar mediante simulaciones por computador el desempeño del esquema de control con aprendizaje propuesto por Kawato en 1987.

2 DESCRIPCIÓN GENERAL

Partiremos de un sistema dinámico dado por un modelo lineal con una serie de limitaciones. Se utilizará una referencia consistente en escalones de distintas amplitudes colocados en distintas zonas de operación. Para ese sistema y esa referencia calcularemos decir brevemente cómo el PID óptimo y los resultados de control que se obtienen con él.

Partiendo de los datos obtenidos se pasará a un esquema de control basado en el controlador de Kawato y trataremos de comprobar que mejoras introduce respecto al PID óptimo.
Para ello se realizan varias tandas de experimentos:

1. Para tratar de encontrar qué parámetros optimizan el comportamiento del controlador de Kawato para este sistema y esa referencia.
2. Utilizar otras referencias para tratar de estudiar que variaciones introduce en los resultados obtenidos.
3. Intentar que los resultados obtenidos puedan tener utilizada práctica.

2.1 APORTACIÓN DE LAS REDES NEURONALES AL CONTROL AUTOMÁTICO:

El cada vez más extendido, uso de redes neuronales ha llevado a que se introduzcan entre las técnicas de control de sistemas dinámicos con aprendizaje. La capacidad de estas redes para generalizar y adaptar de manera eficiente las convierten en buenas candidatas para el control con aprendizaje de sistemas tanto lineales como no lineales.

El objetivo de un controlador con red neuronal directo es generar la señal de control correcta para conducir el sistema dinámico siguiendo la referencia.

El controlador basado en Redes Neuronales debe tener las siguientes características:

1. Estructura flexible para aproximar sistemas no lineales. Esta característica aumenta las prestaciones del controlador.
2. Capacidad de aprendizaje debida a su estructura flexible.

2.1.1 ARQUITECTURA DEL CONTROLADOR:

La figura 1 muestra la arquitectura básica del controlador , que básicamente está dividida entre Observador y Controlador. El controlador está también dividido entre Controlador por realimentación (Feedback) y Controlador por adelanto (Feedforward)

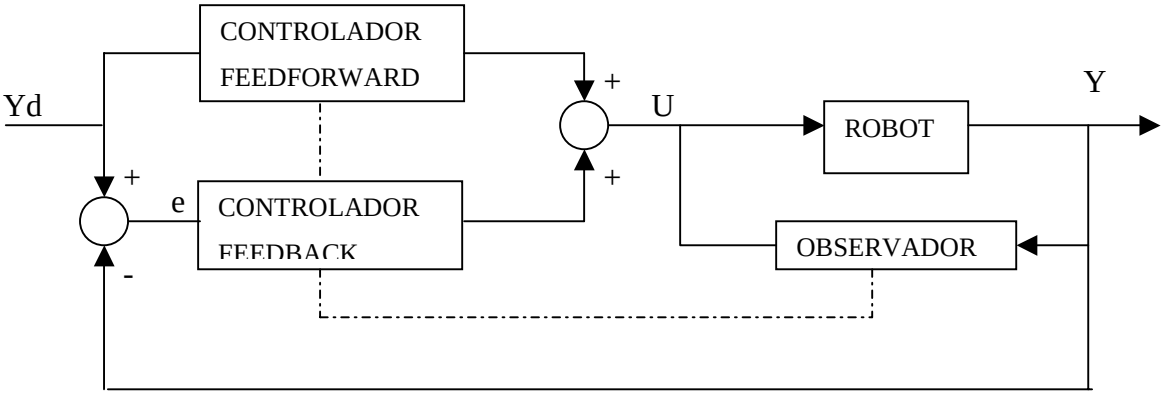
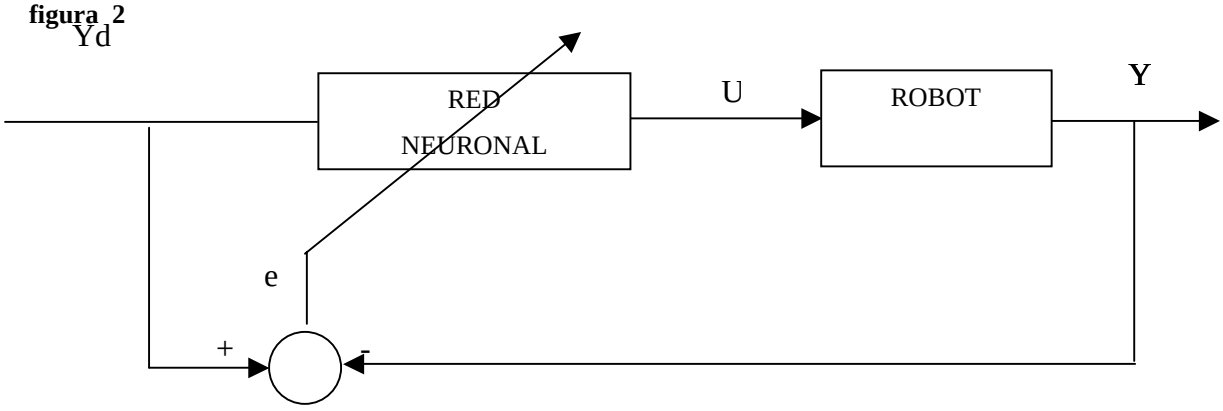


figura 1

El controlador feedforward se encarga de compensar los términos no lineales . El controlador feedback trabaja para eliminar los errores de posición, velocidad y fuerza que el feedforward no puede compensar.
Vamos a mostrar algunas arquitecturas de controlador que pueden realizarse utilizando redes neuronales.

El primer esquema que vamos a considerar consiste en que la red neuronal realice las funciones de controlador y observador. Puede verse en la siguiente figura y se denomina Controlador Feedforward Directo.

figura 2



En este esquema la red neuronal realiza la dinámica inversa de la planta. Su principal ventaja es que se explota al máximo la capacidad de la red neuronal, mientras que su mayor problema reside en que al principio se pierde robustez, debido a que la salida dependerá del valor inicial de pesos de la red neuronal.

El siguiente esquema que vamos a considerar es la red neuronal como observador

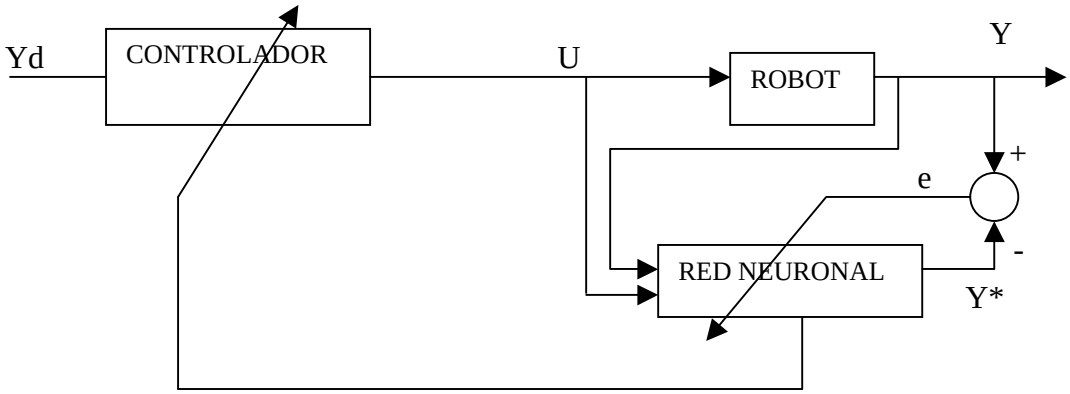


figura 3

En este caso no se utilizan todas las posibilidades se la red , ya que únicamente lleva a cabo la identificación del sistema. Sin embargo la robustez inicial del sistema la mantiene el controlador cuando los pesos de la red tienen los valores iniciales.

En la figura 4 se muestra el controlador feedforward- feedback de Kawato. La red neuronal es entrenada para hacer cero la salida del controlador feedback. Por lo tanto el

control del sistema se transfiere del controlador PD feedback al controlador NN feedforward. Este esquema tiene la ventaja de que el sistema es estable al comienzo del entrenamiento de la red neuronal debido a la robustez del controlador PD. En cualquier caso la regla de entrenamiento de la red neuronal depende del sistema a controlar.

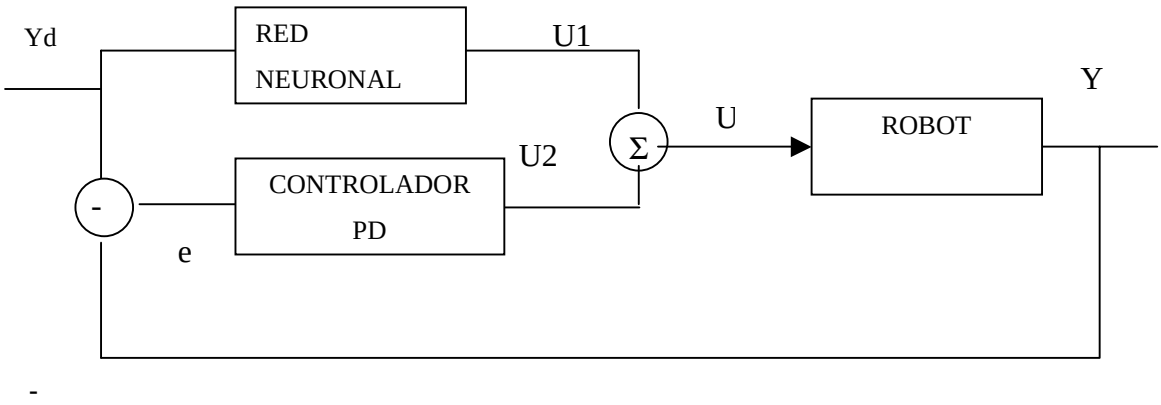


figura 4

Este será el esquema de control objeto de este proyecto. Sobre él trabajaremos para intentar demostrar las mejoras que introduce.

2.1.2 ESTRUCTURA DE LAS REDES DE NEURONAS:

Vamos a ver las estructuras dinámicas de una NN con respecto a un sistema de control en tiempo discreto ya que la NN es normalmente realizada por medio de un computador digital. Hay dos métodos para realizar la dinámica usando NN:

- Usar un bucle de realimentación externo con una función de retraso de tiempo.
- Usar un bucle de realimentación interno con un operador de retraso de tiempo.

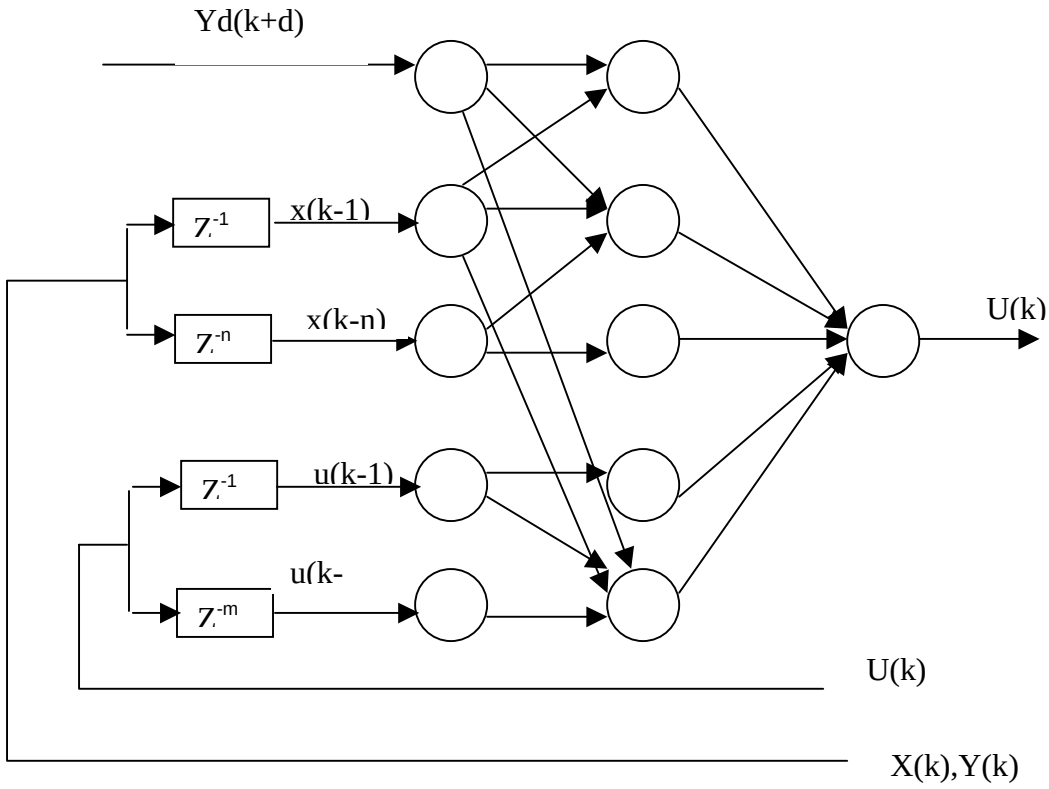


figura 5

Las funciones retraso se realizan usando memorias.

La red neuronal puede seguir la dinámica por realimentación de la información de los estados de la planta y de las señales de control.

La red neuronal es bastante simple y puede utilizarse la regla delta generalizada modificada.

Por otra parte el orden del sistema dinámico está limitado por la información retroalimentada. Por tanto, se pierde parte de la flexibilidad cuando se fija qué información se retroalimenta.

2.2 EL CONTROLADOR DE KAWATO

El esquema de control que vamos a utilizar se muestra en la figura.
Se basa en añadir en paralelo al controlador PID una red neuronal. La salida del controlador consistirá en la suma de la salida del controlador PID más la salida de la red neuronal.

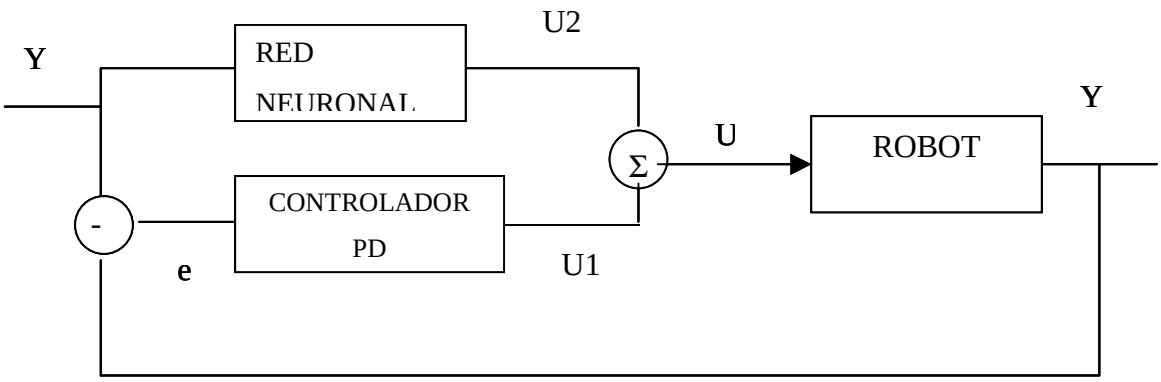


figura 6

La red neuronal tendrá como entradas:

- el error del sistema (diferencia entre la referencia y la salida),
- la referencia correspondiente al instante actual y los instantes posteriores
- la salida del sistema y en el instante actual y los anteriores

- la salida del controlador U en el instante actual y los anteriores

El objetivo de la red será anular la salida del controlador PID a lo largo de los entrenamientos, por tanto el error utilizado en el algoritmo de entrenamiento de la red será:
Error= U1.

El hecho de que la salida del controlador PID se anule será un buen indicativo de que la diferencia entre la salida se va aproximando a la referencia, ya que U1 guarda una estrecha relación con el error cometido en cada instante. Dicho de otra manera, la red deberá anticiparse al PID de manera que la aportación de éste a la salida del controlador se haga innecesaria. A medida que transcurran los entrenamientos la salida de la red irá ganando peso y la del PID irá disminuyendo.

2.3 CONTROLADOR PID BÁSICO

Vamos a describir brevemente cómo son las ecuaciones y el esquema de control de un controlador PID (Proporcional, Integral y Derivativo).
La figura 7 muestra un esquema d control compuesto por

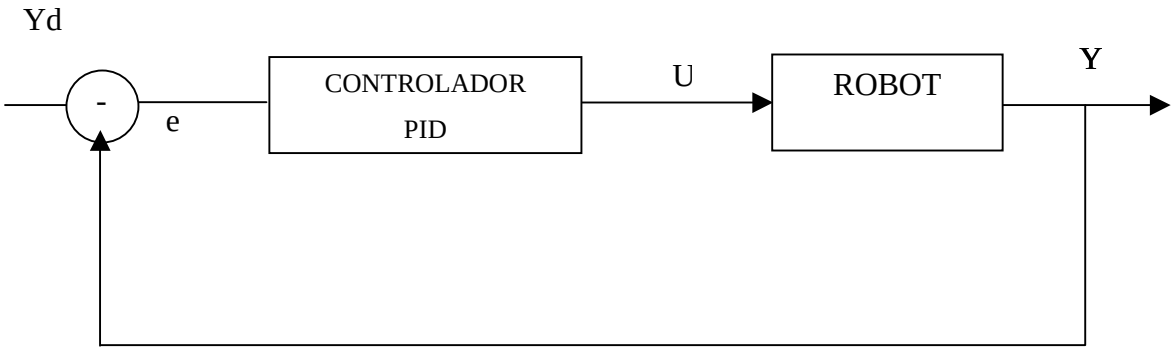


figura 7

$$u(t)=K\left[e(t)+\frac{1}{Ti}\int e(s)ds+Td\frac{de(t)}{dt}\right]$$

siendo *u* la variable de control y *e* la diferencia entre la referencia y la salida.

El término básico en el controlador PID es el proporcional P, que origina una actuación de control correctiva proporcional el error.

El término integral I brinda una corrección proporcional a la integral del error. Esta acción tiene la ventaja de asegurar que en última instancia se aplicará suficiente acción de con-trol para reducir el error de regulación a cero. Sin embargo, la acción integral también tiene un efecto desestabilizador debido al corrimiento de fase agregado.

El término derivativo D da propiedades predictivas a la ac-tuación, generando una acción de control proporcional a la velocidad de cambio del error. Tiende dar más estabilidad al sistema pero suele generar grandes valores en la señal de control.

Las formas estándar de controladores PID:

Proporcional

$$K\,P\,(s) = Kp$$

Proporcional e Integral

$$K\,PI\,(s) = Kp\left(1 + \frac{1}{Ti\,s}\right)$$

Proporcional y Derivativo

$$K\,PD\,(s) = Kp\left(1 + \frac{T\,d\,s}{(1 + td\,s)}\right)$$

Proporcional, Integral y Derivativo

$$\text{K PID (s)} = \text{Kp}\left(1 + \frac{\text{T d} \cdot \text{s}}{(1 + \text{td s})} + \frac{1}{\text{T i} \cdot \text{s}} \right)$$

2.3.1 DISCRETIZACIÓN:

Si utilizamos, por ejemplo la aproximación por operador derivada, siendo *T* el período de muestreo resulta,

$$u_k = K\left[e_k + \frac{T}{Ti}\sum_{j=0}^{k} e_j + \frac{Td}{T}\left(e_k - e_{k-1}\right) \right]$$

si renombramos constantes, podemos rescribirlo como

$$u_k = \left[Kp \cdot e_k + Ki \cdot \sum_{j=0}^{k} e_j + Kd \cdot \left(e_k - e_{k-1}\right) \right]$$

Que es la expresión que hemos utilizado en el desarrollo de este proyecto.

2.4 ESTRUCTURA Y APRENDIZAJE DE LA RED BACKPROPAGATION:

En una red neuronal backpropagation existe una capa de entrada con n neuronas, una capa de salida con m neuronas y al menos una capa oculta de neuronas internas.
Cada neurona de una capa, exceptuando las de entrada, recibe entradas de todas las neuronas de la capa anterior y envía su salida a todas las neuronas de la capa posterior, excepto las de la capa de salida. No hay conexiones hacia atrás ni tampoco entre neuronas de la misma capa.

La aplicación del algoritmo backpropagation tiene dos fases, una hacia delante y otra hacia atrás. Durante la fase hacia delante el patrón es presentado a la red y propagado a través de las capas hasta alcanzar la de salida. Una vez obtenida la salida de la red se inicia la fase hacia atrás. Se obtiene el error comparando la salida obtenida con la deseada. Primero se ajustan los pesos de la última capa proporcionalmente al error calculado. Se pasa a la capa anterior con una retropropagación del error, ajustando los pesos y continuando con el proceso hasta llegar a la primera capa.
De esta manera se van modificando los pesos para cada patrón de aprendizaje, del que conocemos su salida deseada.
A diferencia de otras técnicas la backpropagation requiere el uso de neuronas cuya función de activación sea continua y por tanto derivable. Generalmente se utilizan funciones de tipo sigmoidal.

2.4.1 DIMENSIONAMIENTO DE LA RED:

En cuanto al número de capas, en general son suficientes 3 (entrada,oculta y salida).En algunas aplicaciones puede ser necesario para aumentar la velocidad de aprendizaje utilizar más capas.

El número de neuronas de las capas de entrada y salida viene determinado por la naturaleza de la aplicación. Determinar cuántas neuronas debe haber en la capa oculta es un problema complejo.

El número de neuronas de la capa oculta interviene en la eficiencia de aprendizaje y generalización de la red. No existen reglas que indiquen el número óptimo, en cada problema se debe ensayar con distintos números de neuronas para organizar la representación interna y escoger luego la mejor solución. La idea más utilizada, sobre todo en sistemas simulados consiste en tener el menor número posible de neuronas en la capa oculta, porque cada una supone una mayor carga de procesamiento en caso de simulación software. En un sistema implementado en hardware este problema no es crucial, sin embargo sí influiría el problema de comunicación entre los distintos elementos de la red.

Es posible eliminar neuronas ocultas si la red converge sin problemas, determinando el número final en función del rendimiento global del sistema. Si la red no converge es posible que sea necesario aumentar este número. Por otro lado, examinando los valores de los pesos de las neuronas ocultas periódicamente en la fase de aprendizaje se pueden detectar aquellas cuyos valores cambian muy poco durante el aprendizaje respecto a sus valores iniciales y eliminar aquellas neuronas que apenas participan en el proceso de aprendizaje.

2.4.2 CONSIDERACIONES SOBRE EL ALGORITMO:

El algoritmo backpropagation encuentra un valor mínimo de error local o global mediante la aplicación de pasos descendientes (gradiente descendiente). Cada punto de la superficie de la función de error corresponde a un conjunto de valores de los pesos de la red. Con el gradiente descendiente, siempre que se realiza un cambio en todos los

pesos de la red, se asegura en descenso por la superficie del error hasta encontrar el valle más cercano, lo que puede hacer que el proceso de aprendizaje se detenga en un mínimo local de error.

Por tanto, uno de los problemas uno de los problemas que presenta este algoritmo de entrenamiento de redes multicapa es que busca minimizar la función de error, pudiendo caer en un mínimo local o en algún estacionario, con lo cual no se llega a encontrar el mínimo global de la función del error. Sin embargo hay que tener en cuenta qué no tiene por qué alcanzarse el mínimo global en todas las aplicaciones, sino que puede ser suficiente con un error mínimo preestablecido.

2.4.3 CONTROL DE LA CONVERGENCIA:

En el algoritmo del gradiente decreciente es conveniente avanzar por la superficie de error con incrementos pequeños de los pesos. Esto se debe a que tenemos una información local de la superficie y no se sabe lo lejos o lo cerca que está el punto mínimo. Con incrementos grandes, se corre el riesgo de pasar por encima del punto mínimo sin conseguir estacionarse en él. Con incrementos pequeños, aunque se tarde más en llegar, se evita que ocurra esto.

Elegir un incremento de paso adecuado influye en la velocidad con que converge el algoritmo. Esta velocidad se controla a través de la constante de proporcionalidad o tasa de aprendizaje α . Normalmente α debe ser un número pequeño para que la red llegue a asentarse en una solución. Un valor pequeño de α significa que la red tendrá que hacer un gran número de iteraciones. Si esta constante es muy grande, los cambios de peso son muy grandes, avanzando muy rápidamente por la superficie del error, con el riesgo de saltar el mínimo y estar oscilando a alrededor de él, pero sin poder alcanzarlo.

Lo habitual es aumentar el valor de α a medida que disminuye el error de la red durante la fase de aprendizaje. Así, aceleramos la convergencia, aunque sin llegar nunca a

valores deα demasiado grandes que harían que la red oscilase alejándose demasiado del valor mínimo. Otra forma de incrementar la velocidad de la convergencia es consiste en añadir un término momento compuesto por la suma de una fracción del anterior cambio cuando se calcula el valor del cambio del peso actual. Este término adicional tiende a mantener los cambios de peso en la misma dirección.

Otro aspecto a tener en cuenta es la posibilidad de convergencia hacia alguno de los mínimos locales que pueden existir en la superficie de error del espacio de pesos. El algoritmo de retropropagación, no se asegura en ningún momento que el mínimo que se encuentre sea global. Una vez que la red se asienta en un mínimo, sea local o global, cesa el aprendizaje, aunque el error siga siendo demasiado alto, si se ha alcanzado un mínimo local. En todo caso si la solución es admisible desde el punto de vista del error, no importa si el mínimo es local o global o si se ha detenido en algún momento previo a alcanzar un verdadero mínimo.
En la práctica, si una red deja de aprender antes de llegar a una solución aceptable, se realiza un cambio en el número de neuronas de la capa oculta o los parámetros de aprendizaje o, simplemente se vuelve a empezar con un conjunto distinto de pesos originales y se vuelve a resolver el problema.

2.4.4 INICIALIZACIÓN DE LOS PESOS:

Para una rápida adaptación del sistema sería conveniente inicializar los pesos con una combinación de valores W muy cercano al punto de mínimo error buscado. Pero es imposible, porque a priori no se conoce dónde está el punto mínimo. Lo que se hace es partir de un punto cualquiera del espacio, inicializando los pesos los pesos con valores pequeños y aleatorios. Lo mismo se hace con los términos umbral θ que aparecen en las

ecuaciones de entrada neta a cada neurona. Este valor umbral se puede tratar como un peso más que está conectado a una neurona ficticia de salida siempre 1. La utilización de los términos umbral es opcional, pues en caso de utilizarse, son tratados exactamente igual que el resto de pesos y participan como tal en el proceso de aprendizaje.

2.5 RED NEURONAL UTILIZADA:

Vamos a particularizar la teoría antes resumida para la red utilizada en este proyecto:

Vamos a utilizar un red con una capa oculta:

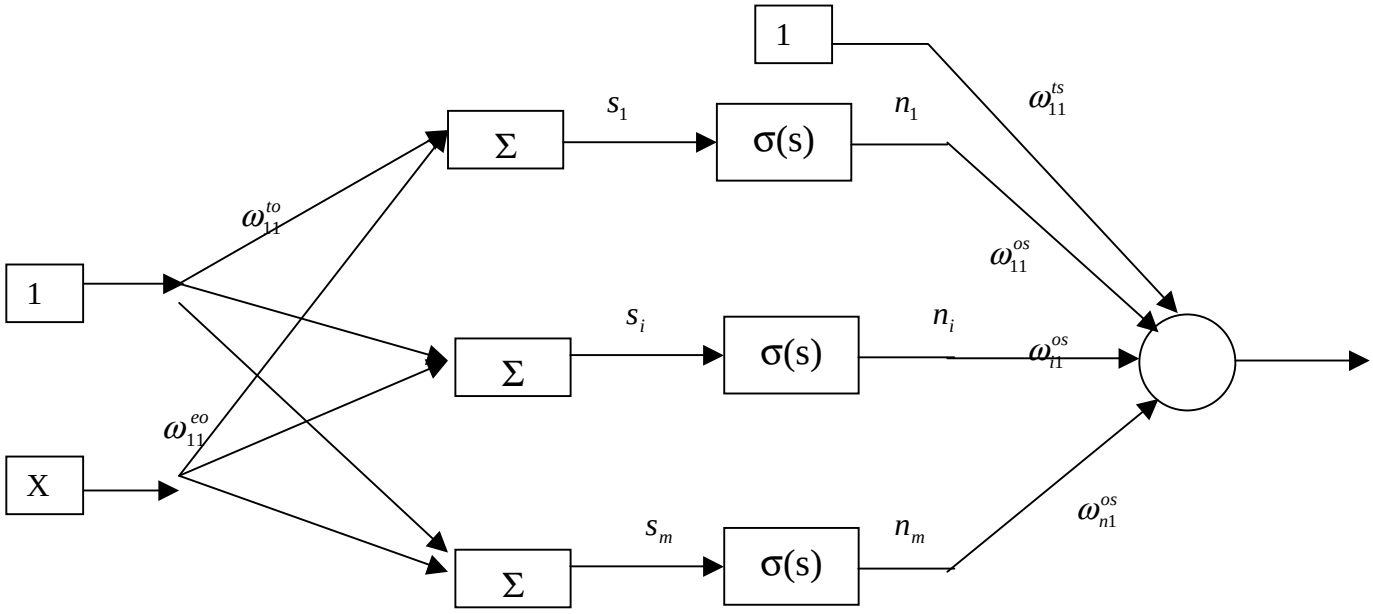


figura 8

La ecuación que tendremos a la salida de la red:

$$y_{red} = \sum \omega_{i1}^{os} \bullet n_i + \omega^{ts}$$

2.5.1 FUNCIÓN DE ACTIVACIÓN

Como ya se ha comentado en el apartado 2.4, la función de activación en con este tipo de algoritmo debe ser continua y por tanto derivable y generalmente se utilizan funciones de tipo sigmoidal.

La que se ha utilizado es la que se muestra a continuación:

$$\sigma(S)=\frac{1-e^{-s}}{1+e^{-s}}$$

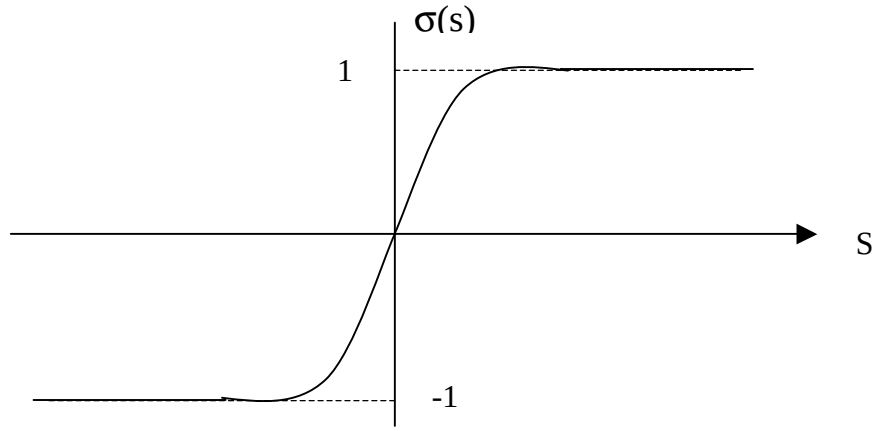


figura 9

vamos a ver cúales van a ser las ecuaciones que se utilizan para actualizar los pesos de la red con el algoritmo descrito:

$\omega \leftarrow \omega + \Delta\omega$ donde

$$\Delta\omega = -\alpha \bullet \frac{1}{2} \bullet \frac{\partial e^2}{\partial \omega} = -\alpha \bullet e \frac{\partial e}{\partial \omega} = \alpha \bullet e \bullet \frac{\partial y_{red}}{\partial \omega}$$

Particularizando para los cuatro tipos de pesos presentes en la figura 8.

- ω_{11}^{ts}

$$\frac{\partial y_{red}}{\partial \omega_{11}^{ts}} = 1$$

$$\omega_{11}^{ts} = \omega_{11}^{ts} + \alpha \bullet e$$

- ω_{i1}^{os}

$$\frac{\partial y_{red}}{\partial \omega_{i1}^{os}} = n_i$$

$$\omega_{i1}^{os} = \omega_{i1}^{os} + \alpha \bullet e \bullet n_i$$

- ω_{1i}^{to}

$$\frac{\partial y_{red}}{\partial \omega_{1i}^{to}} = \omega_{i1}^{os} \bullet \sigma'(s_i)$$

$$\omega_{1i}^{to} = \omega_{1i}^{to} + \alpha \bullet e \bullet \omega_{i1}^{os} \bullet \sigma'(s_i)$$

- ω_{1i}^{eo}

$$\frac{\partial y_{red}}{\partial \omega_{1i}^{eo}} = \omega_{i1}^{os} \bullet \sigma'(s_i) \bullet x$$

$$\omega_{1i}^{eo} = \omega_{1i}^{eo} + \alpha \bullet e \bullet \omega_{i1}^{os} \bullet \sigma'(s_i) \bullet x$$

Vamos a calcular la derivada de la función de activación, necesaria para calcular la actualización de los pesos a lo largo de los entrenamientos.

$$\frac{d\sigma(s)}{ds} = \frac{2e^{-s}}{\left(1+e^{-s}\right)^2}$$

$$e = y_{obj} - y_{red}$$

Estas ecuaciones son las que se han utilizado, pero para facilitar la programación se han compactado los pesos en matrices, de manera que mediante álgebra matricial se actualizan muchos pesos a la vez.
Vamos a describir las matrices utilizadas y la información que contienen.

El vector X representa las variables de entrada hacia la red neuronal:

$$X^* = \begin{bmatrix} 1 \\ y(k) \\ y(k-1) \\ y(k-2) \\ y(k-3) \\ r(k+ant) \\ r(k+ant-1) \\ \dots \\ r(k+1) \\ r(k) \\ r(k-1) \\ U(k) \\ U(k-1) \\ U(k-2) \\ \dots \end{bmatrix}$$

Los pesos de entrada, se representan en la matriz W, que tendrá unas dimensiones :

(Número de n entradas) x (número de neuronas de la capa oculta)

$$W = \begin{bmatrix} W_{11}^{to} & W_{12}^{to} & W_{1nn}^{to} \\ W_{11}^{eo} & W_{12}^{eo} & W_{1nn}^{eo} \\ W_{21}^{eo} & & \\ & & \\ & & \\ W_{nent1}^{eo} & & W_{nentnn}^{eo} \end{bmatrix}$$

Los pesos de salida:

$$W^{os} = \begin{bmatrix} W_1^{os} \\ \\ W_{nn}^{os} \end{bmatrix}$$

Sus dimensiones son :

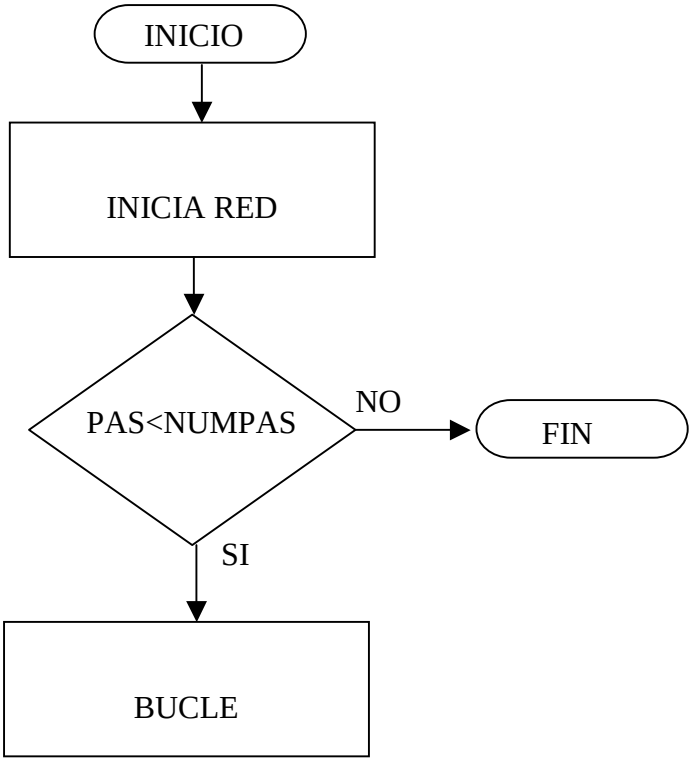
1 x número de neuronas de la capa oculta

El otro peso de salida , W^{ts}, es una variable, ni una matriz.

2.6 DIAGRAMAS DE FLUJO

En este capítulo vamos explicar las principales funciones utilizadas para el desarrollo del proyecto. Para explicar dichas funciones de una manera sencilla y comprensible a simple vista se han utilizado diagramas de flujo.

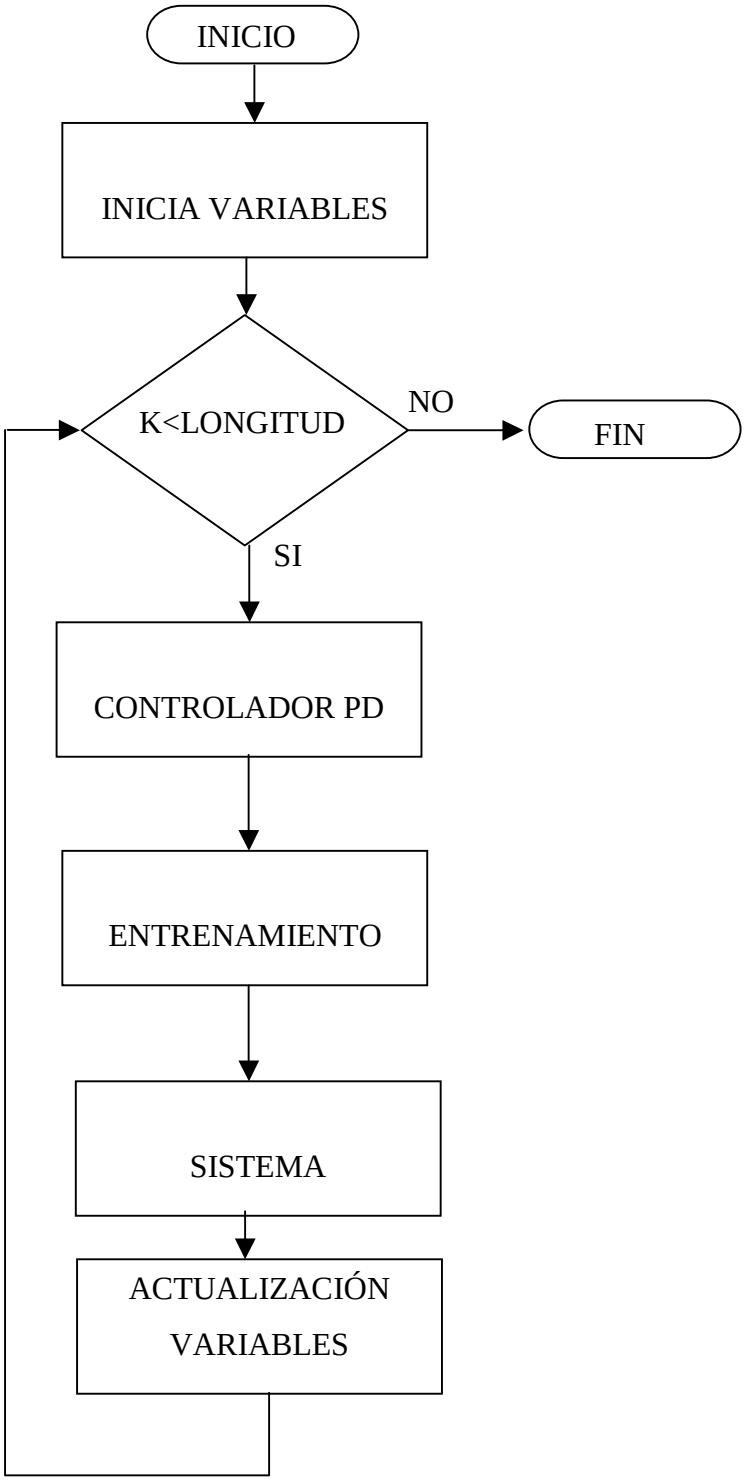
2.6.1 RED NEURONAL



Esta rutina realiza un bucle de manera que la red se sigan realizando pasadas de aprendizaje mientras no se hayan llegado al número de pasadas prefijado (NUMPAS).
En INICIA RED se inicializan tanto las variables que intervienen en la red neuronal como aquellas que sólo necesitan inicializarse al comienzo, como por ejemplo la referencia o la planta.

2.6.2 BUCLE

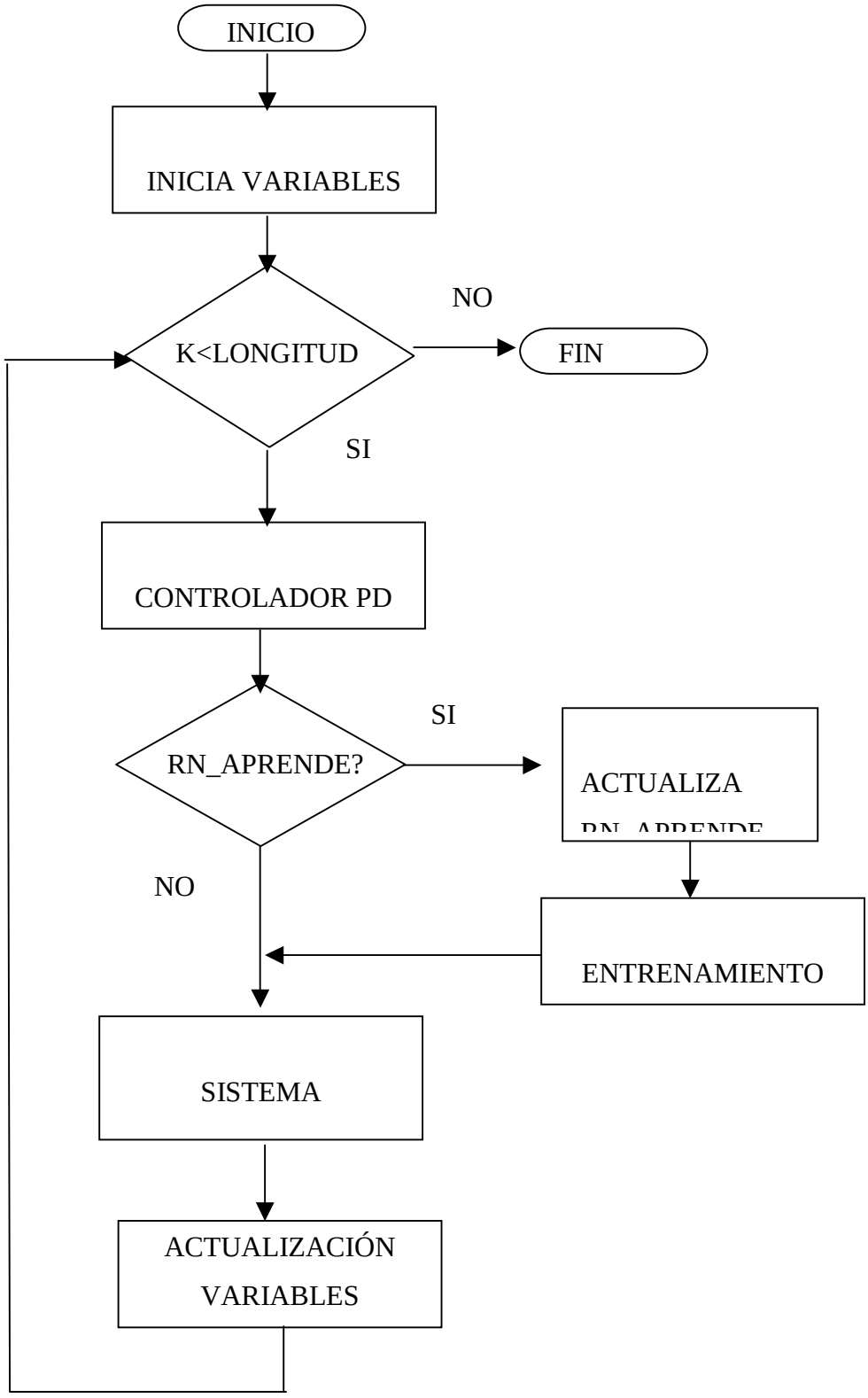
Esta rutina se ejecutará tantas veces como pasadas de entrenamiento haya. En ella se calcula la salida del controlador PD, se entrena la red neuronal y se calcula su salida y con la suma de ambas señales de control se calcula la salida del sistema. Finalmente se actualizan todas las variables.


INICIO
INICIA VARIABLES
K<LONGITUD
NO
FIN
SI
CONTROLADOR PD
ENTRENAMIENTO
SISTEMA
ACTUALIZACIÓN VARIABLES

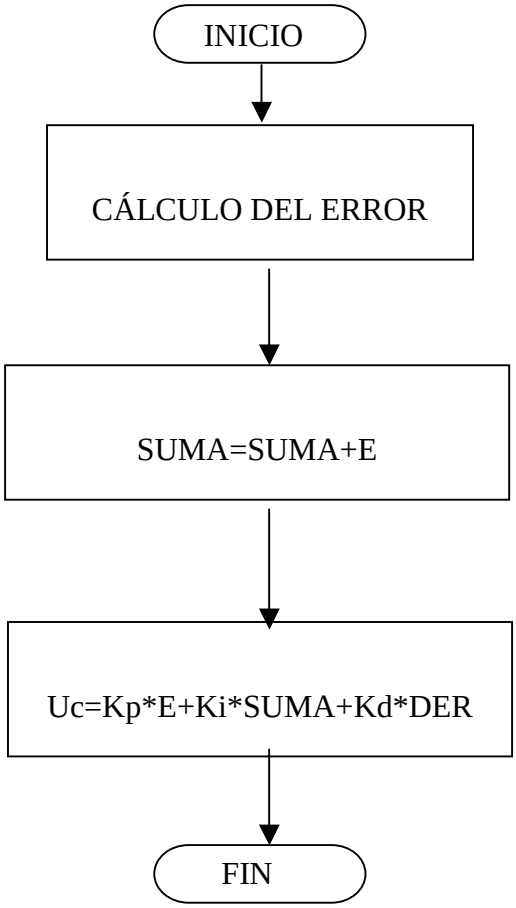
2.6.3 BUCLE_STOP

Esta rutina se utiliza en la tanda de experimentos 4 . Es muy parecida a la función anterior bucle, salvo en el hecho de compara los resultados obtenidos en cada pasada de entrenamiento con los obtenidos en la pasada anterior, y en caso de estos hayan empeorado, desactiva la función entrenamiento de manera que a partir de ese momento los pesos conservan el valor que tenían en la pasada anterior y ya no cambian nunca más de valor.

Para decidir si la red aprende o no se utiliza una variable , que sólo se actualiza en el caso que la red esté aprendiendo. Por tanto, una vez que la red ha dejado de aprender, la variable ya no cambiará de valor y por tanto no habrá más entrenamientos .


INICIO
INICIA VARIABLES
K<LONGITUD
NO
FIN
SI
CONTROLADOR PD
RN_APRENDE?
SI
ACTUALIZA
NO
ENTRENAMIENTO
SISTEMA
ACTUALIZACIÓN
VARIABLES

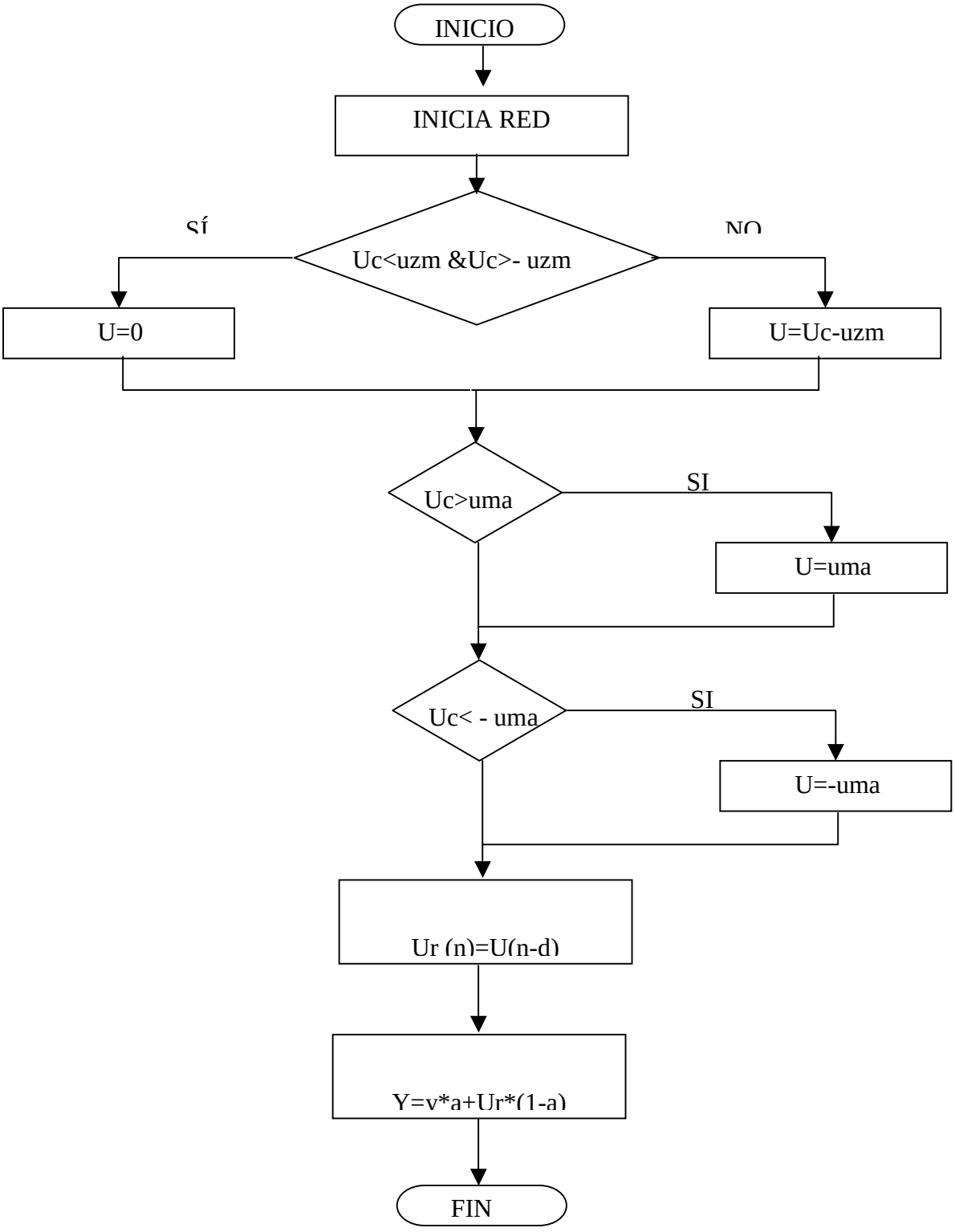
2.6.4 CONTROLADOR PD



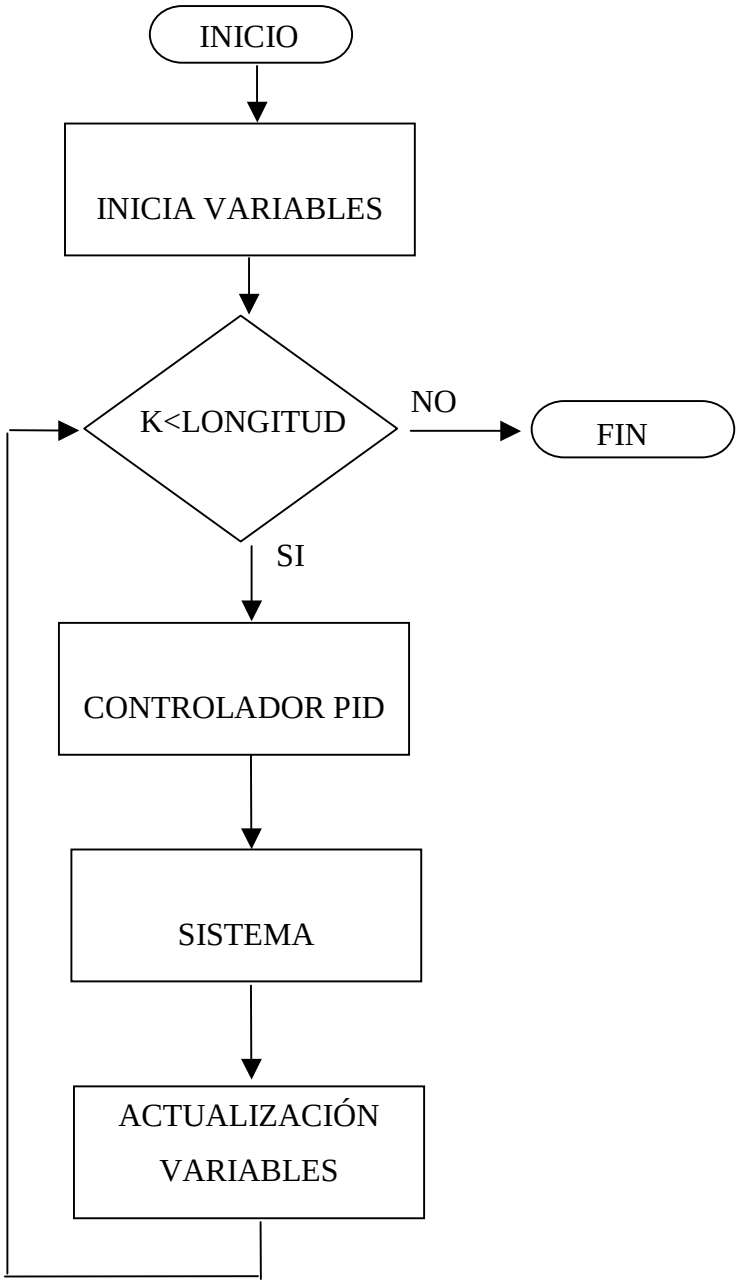
Esta rutina consiste en aplicar las ecuaciones de un controlador PID. Para ello necesita el error en la salida (efecto proporcional) , su variación (efecto derivada) y su acumulación(efecto integral).

2.6.5 SISTEMA

Esta rutina consiste en las ecuaciones que describen la salida de la planta. Primero se tiene en cuenta la zona muerta (valores inferiores a uzm y superiores a –umz), a continuación la saturación (valores superiores a uma o inferiores a –uma), y por último un retraso de intervalos de tiempo.

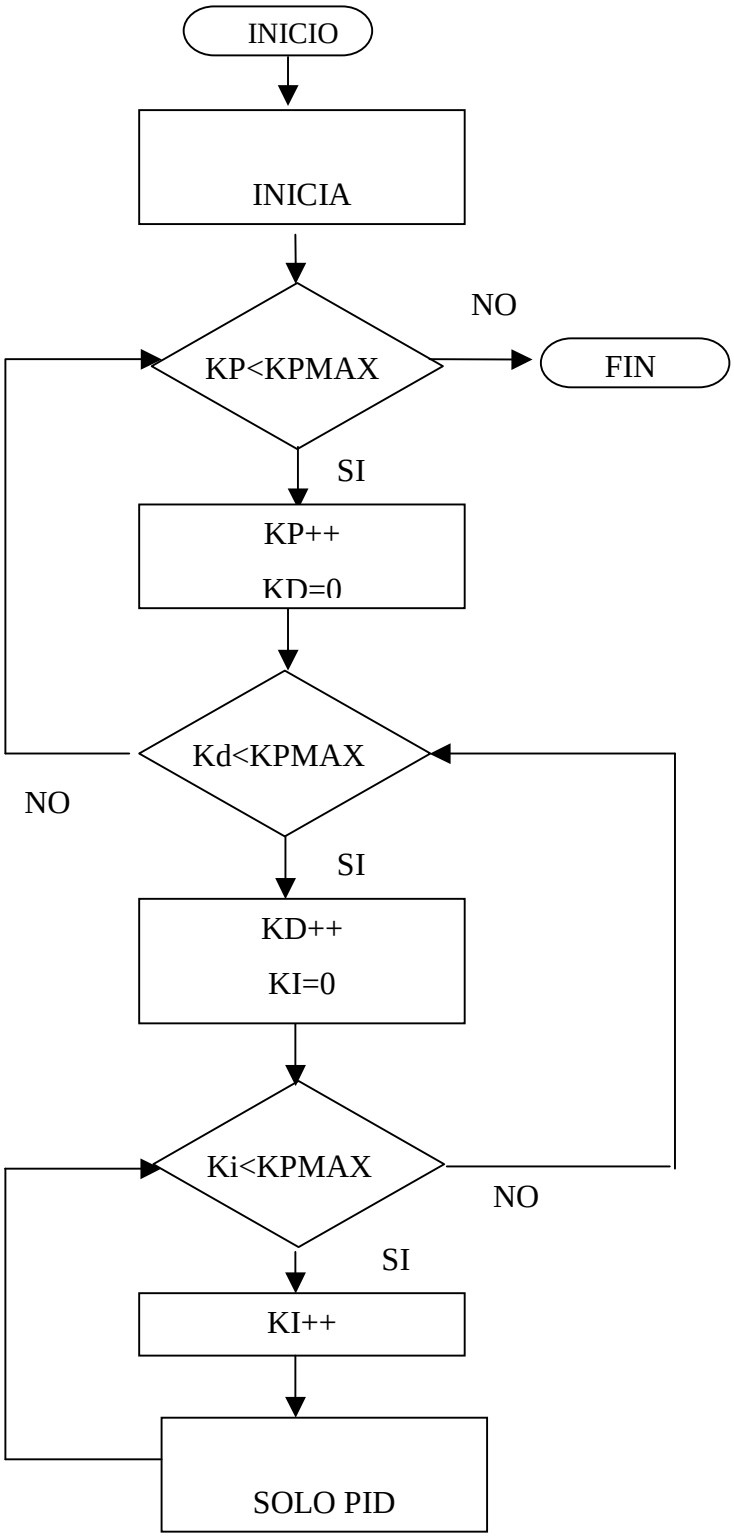

INICIO
INICIA RED
SÍ
Uc<uzm &Uc>- uzm
NO
U=0
U=Uc-uzm
Uc>uma
SI
U=uma
Uc< - uma
SI
U=-uma
Ur (n)=U(n-d)
Y=v*a+Ur*(1-a)
FIN

2.6.6 SOLOPI



Esta rutina calcula la salida del sistema y el error cometido cuando el controlador es sólo un PID

2.6.7 OPTIMIZASOLOPI



Esta rutina calcula el PID óptimo. Realiza bucles en los que va variando Kp, Ki , Kd dentro de un rango , ejecuta la rutina SOLOPI y devuelve aquellos valores del PID que hacen que el error total sea mínimo.

Esta rutina calcula el PID óptimo. Realiza bucles en los que va variando Kp, Ki , Kd dentro de un rango , ejecuta la rutina SOLOPI y devuelve aquellos valores del PID que hacen que el error total sea mínimo.

2.6.8 OPTIMIZACIÓN PID

La figura 10 muestra el esquema clásico de control utilizando un PID:

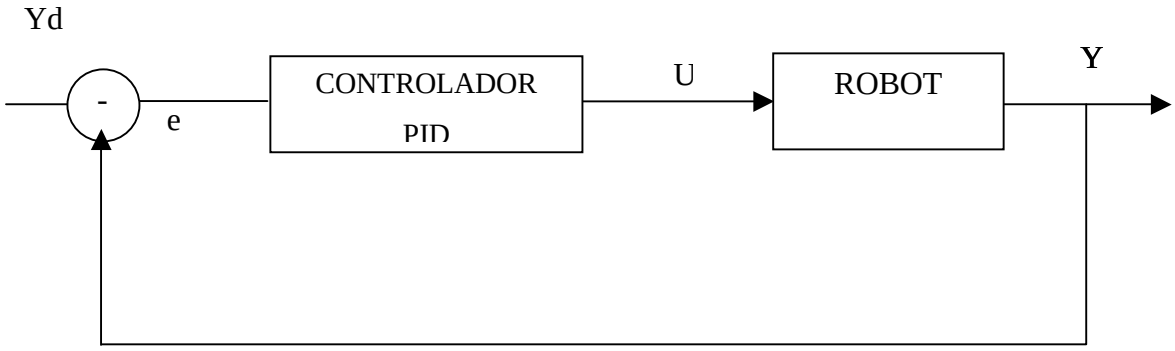


figura 10

En este apartado se pretende calcular un PID óptimo para nuestro sistema. En el punto 2.3 ya vimos cuáles son las ecuaciones y los parámetros que intervienen en controlador PID

El criterio de optimización es obtener la mínima suma de errores en cada instante al cuadrado a lo largo de la señal de referencia:

$$ErrorTotal = \sum_i \left(referencia(i) - salida(i)\right)^2$$

Cuando esta variable Error Total sea mínima estaremos ante el controlador PID óptimo para nuestro sistema.

Aplicamos un controlador PID al sistema en el que vamos incrementando los valores de Kp Kd y Ki desde 0 hasta 0,9 en incrementos de 0,1.Obtenemos así 100 resultados diferentes.

Si tomamos como objetivo el que Error Total sea el mínimo posible, el mejor resultado se obtiene para:

Kp=0.7;

Ki=0.3;

Kd=0.9;

Vamos a ver cuál es el comportamiento del sistema cuando usamos como controlador un PID con esos valores que hemos considerado óptimos.

En la figura 11 se representa la salida del sistema obtenida para esos valores de los parámetro del PID frente a la señal de referencia

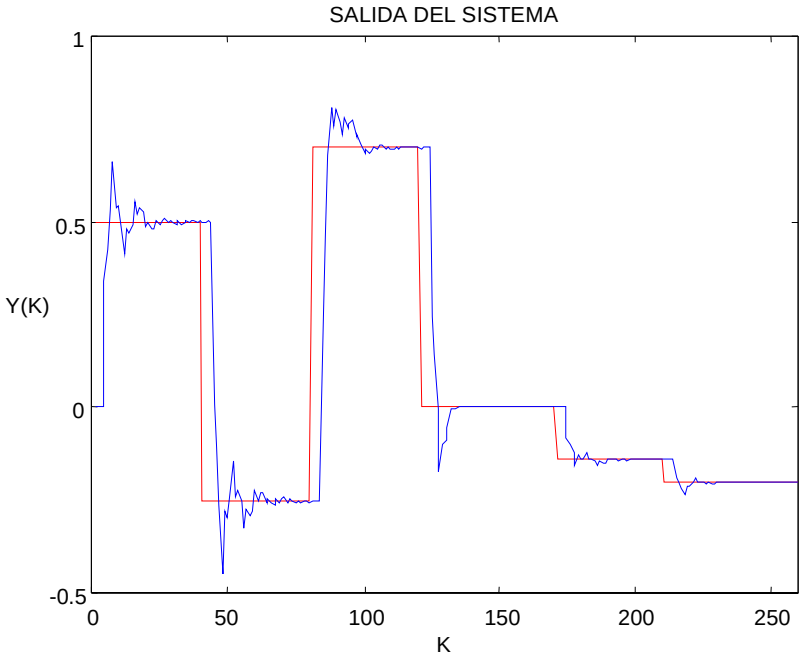


figura 11

En la figura 12 se representa el error obtenido en cada instante. Vemos que en los puntos en los que se produce la transición de un escalón a otro de amplitud diferente, el error experimenta un pico

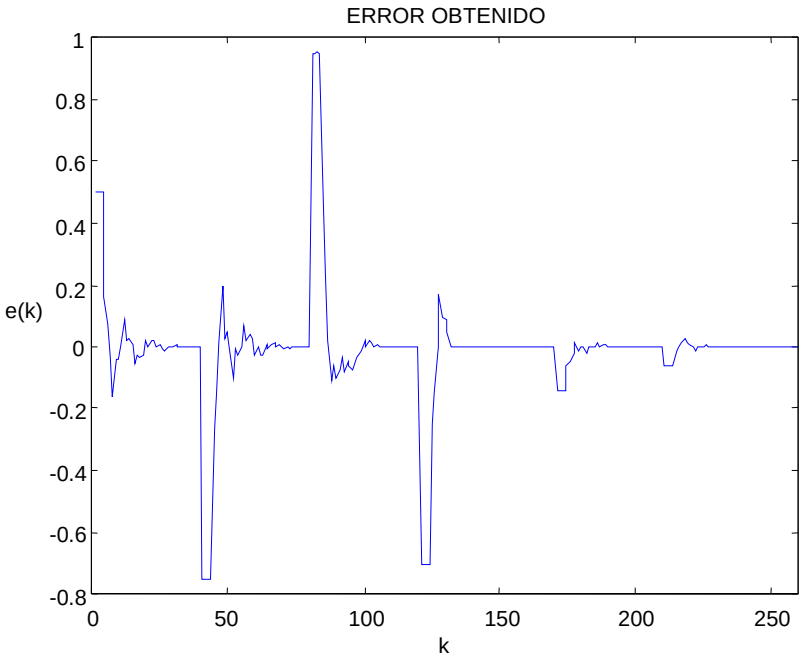


figura 12

Sería también interesante conocer cómo va evolucionando la salida del controlador con el transcurso de los instantes . Podemos representar también cómo es la señal de control

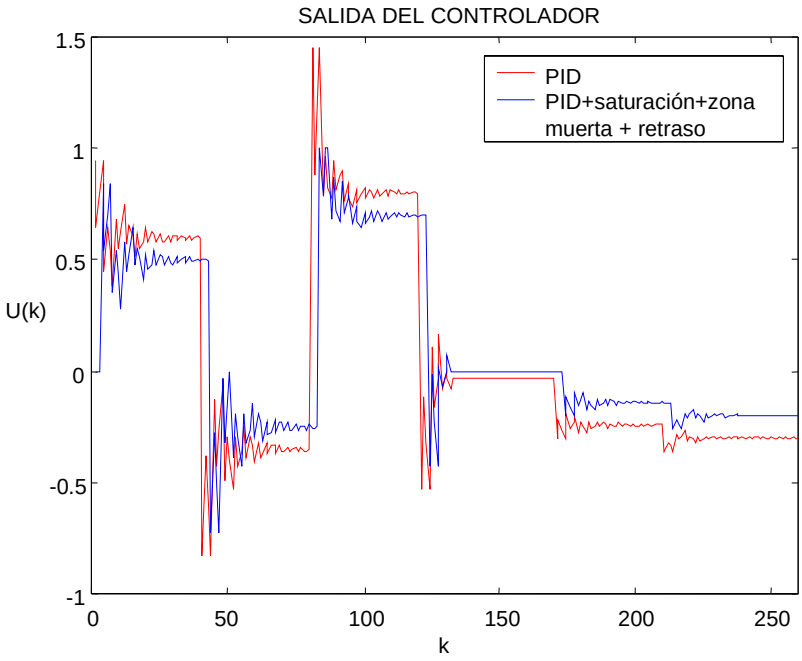


figura 13

En esta figura podemos observar cómo afectan a la señal de control la zona muerta , la saturación y el retraso del sistema.