3.7 IDENTIFICACION DE UN SISTEMA DINÁMICO NO LINEAL

3.7.1 Descripción del Problema: La identificación de un sistema consiste en determinar una función que relacione las variables de entrada con las variables de salida, esta identificación tradicionalmente se hace a través de técnicas de regresión y con un conocimiento previo de la naturaleza del sistema.

Clásicamente el comportamiento de un sistema se describe a través de sus ecuaciones de estado, las cuales son un conjunto de ecuaciones diferenciales no lineales de primer orden, donde el objetivo es encontrar las funciones f y g mostradas en las siguientes ecuaciones:

$$\frac{\partial x(t)}{\partial t} = f[x(t), u(t)] \qquad (3.7.1)$$

$$y(t) = g[x(t)] \qquad (3.7.2)$$

En la identificación de un sistema pueden conocerse previamente el orden del sistema y desconocer de forma precisa sus parámetros o desconocer totalmente el orden del sistema y sus parámetros; la identificación de un sistema mediante redes neuronales no pretende encontrar las funciones f y g, sino un mapa de la relación entrada - salida con base a los patrones con que fue entrenada.

El sistema que se quiere identificar es el péndulo invertido propuesto por Kuschewski et al. (1993), mostrado en la figura 3.7.1

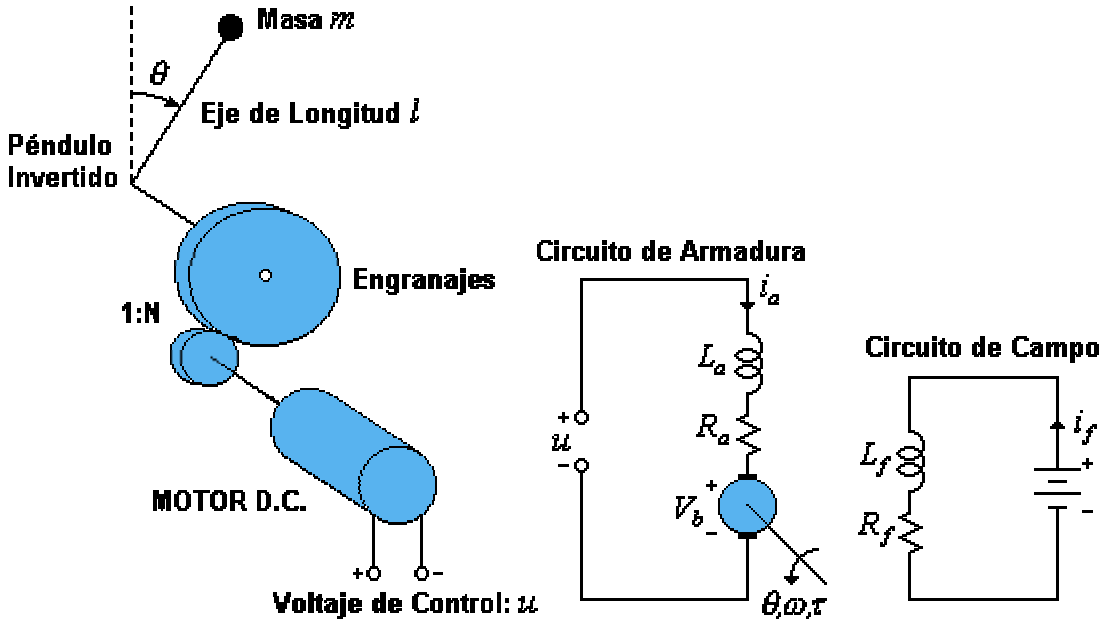

Figura 3.7.1 Péndulo Invertido

Las ecuaciones diferenciales no lineales que describen el comportamiento del sistema son:

$$ml^2\ddot{\theta} + mgl\ Sen\theta = \tau_m \qquad (3.7.3)$$

$$L_{fa}\frac{di_a}{dt} + R_{fa}i_{fa} + V_b = u \qquad (3.7.4)$$

$$\tau_m = N_t K_m i_a \qquad (3.7.5)$$

$$V_b = N_t K_b \theta_m \qquad (3.7.6)$$

K_m: Constante de Torque del motor = 0.1NM/A

K_b: Constante de Fuerza Contra electromotriz = 0.1Vs/rad

N_t: Relación de Transformación entre los piñones = 10

g = 9.8 m/s^2 l = 1m m = 1 kg R_a = 1 Ω L_a = 1 mH

Con estos parámetros, las ecuaciones de estado que representan el comportamiento del Péndulo Invertido están dadas por:

$$\begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \\ \dot{X}_3 \end{bmatrix} = \begin{bmatrix} X_2 \\ -9.8Sen(X_1) + X_3 \\ -10X_2 - 10X_3 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \quad (3.7.7)$$

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \quad (3.7.8)$$

Las variables de estado del sistema son:

$$X_1 = \theta,\ X_2 = \frac{d\theta}{dt},\ X_3 = i_a \quad (3.7.9)$$

3.7.2 Justificación del tipo de red.

Según lo demostrado en la sección 2.6.1 del capitulo 2, una red neuronal dinámica recurrente puede identificar el comportamiento de un sistema dinámico descrito

por ecuaciones diferenciales no lineales si se tiene abundante y representativa información de sus datos de entrada y salida.

Según lo demostrado en la sección 2.6.2 un sistema dinámico autónomo ($u=0$) puede ser identificado una red dinámica multicapa, utilizando en la capa estática un algoritmo tipo backpropagation

3.7.3 Entrenamiento de la red.

3.7.3.1 Identificación del Sistema Autónomo (u=0). Utilizando una red dinámica multicapa: Se identificará la dinámica del péndulo invertido autónomo considerando nulo el voltaje aplicado.

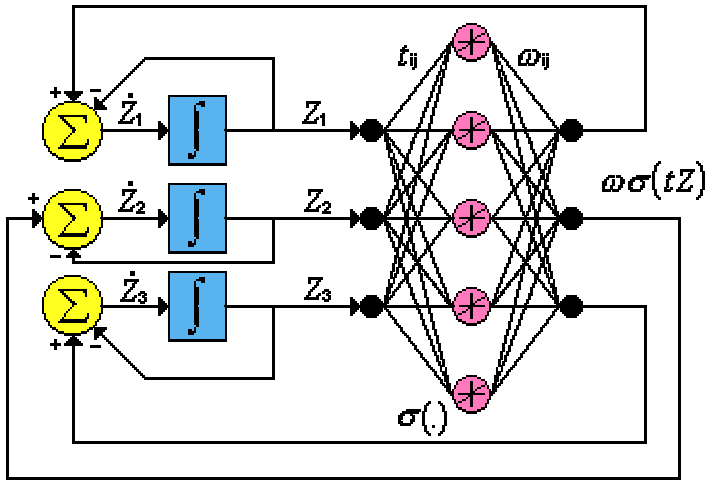

Figura 3.7.2 Red multicapa

Para encontrar los pesos tij y wij de la red recurrente multicapa, se entrenará la capa estática con un algoritmo de propagación inversa y luego se implementará la red de la figura 3.7.2 para identificar plenamente la dinámica del sistema autónomo.

El set de entrenamiento se generó con variaciones en el ángulo, la velocidad, la corriente y todas sus posibles combinaciones en los siguientes rangos:

	Valor Inicial	Valor Final	Incremento	Conversión
Ang	-30º	30º	12º	pi/180º
Vel	-115º	46º	115º	pi/180º
u	0	8	1.6	-

Tabla 3.7.1 Rangos del set de entrenamiento

Los patrones de entrenamiento esperados se calculan como el valor actual del patrón de entrada más el valor de cada una de sus derivadas como se explicó en la sección 2.6.2, el valor de las derivadas de las variables de estado se obtiene al evaluar su valor actual en el conjunto de ecuaciones que describe el comportamiento de la planta.

P	P1	P2		P42		P144	P145		P180		P216
Z1	-0.5236	-0.3142	....	0.5236	....	0.5236	-0.5236	....	0.5236	....	0.5236
Z2	-2.0071	-2.0071	....	-2.0071	....	2.0071	-2.0071	....	2.0071	....	2.0071
Z3	0.0000	0.0000	....	1.6000	....	4.8000	6.4000	....	6.4000	....	8.0000

t	t1	t2		t42		t144	t145		t180		t216
Z1+dZ1	-2.5307	-2.3213	...	-1.4835	...	2.5307	-2.5307	...	2.5307	...	2.5307
Z2+dZ2	2.8929	1.0212	...	-5.3071	...	1.9071	9.2929	...	3.5071	...	5.1071
Z3+dZ3	20.0713	20.0713	...	5.6713	...	-63.2713	-37.5287	...	-77.6713	...	-92.0713

Tabla 3.7.2 Patrones de entrenamiento

Para la capa estática tipo backpropagation los pesos iniciales son:

W1i=net.IW{1,1}

	Z1	Z2	Z3
N1	2.4024	-1.0466	-0.0833
N2	-2.3909	0.2432	-0.4966
N3	2.6014	0.2716	-0.4746
N4	2.2506	-0.6173	-0.4285
N5	-2.7268	0.1526	-0.4748

W2i=net.LW{2,1}

	N1	N2	N3	N4	N5
Z1	-0.2321	-0.9293	-0.9685	0.1738	0.2629
Z2	0.3662	0.2248	-0.9673	-0.8848	0.4353
Z3	-0.8143	0.2171	-0.6199	-0.2649	0.3853

Tabla 3.7.3 Valores iniciales de los Parámetros de la Capa Estática

El siguiente código crea una red de 3 entradas, 5 neuronas en la capa oculta y 3 salidas y la entrena con el algoritmo *trainlm* (Ver Anexo A):

```
net=newff(minmax(P),[5 3],{'tansig' 'purelin'},'trainlm');
net.trainParam.epochs=1000;
net.trainParam.goal=1e-06;
net.trainParam.lr=0.03
net.trainParam.mem_reduc=1
[net,tr]=train(net,P,t);
```

Luego de 500 iteraciones el error medio cuadrático cae por debajo de $1x10^{-6}$, los valores de los parámetros de la red luego del proceso de entrenamiento se muestran en la tabla 3.7.2

tij=W1f=net.IW{1,1}

	Z1	Z2	Z3
N1	0.7305	0.0000	0.0000
N2	0.0002	0.0211	0.0188
N3	-0.0003	-0.0212	-0.0189
N4	-0.0003	-0.0287	-0.0256
N5	-0.0078	-0.0074	0.0004

wij=W2i=net.LW{2,1}

	N1	N2	N3	N4	N5
Z1	-0.0004	10.1756	7.9010	-1.1967	-127.9416
Z2	-13.2698	16.1031	-28.6495	-24.1701	15.6964
Z3	0.0000	-217.4789	187.7513	218.6130	-13.3227

Tabla 3.7.4 Valores finales de los Parámetros de la Capa Estática

Según lo demostrado en la sección 2.6.2, el Jacobiano de la red multicapa en el origen se calcula como:

$$J_M = -I_N + WT \qquad (3.7.10)$$

Reemplazando en la ecuación 3.7.10, los valores de la tabla 3.7.4 el Jacobiano de la red multicapa es como se sigue:

JM

0.0002	10.260	0.0231
-9.7995	0.5254	1.4680
-0.0490	-14.7521	-14.2328

Tabla 3.7.5 Jacobiano de la Red Multicapa

Para una identificación exitosa los valores propios del Jacobiano de la red multicapa mostrado en la tabla 3.7.5 deben ser muy próximos a los valores propios del sistema descritos en la ecuación 3.7.10

$$\begin{aligned} \lambda_1 &= -0.5207 + j3.2816 \\ \lambda_2 &= -0.5207 - j3.2816 \\ \lambda_3 &= -12.6657 \end{aligned} \tag{3.7.11}$$

Los valores propios del sistema son calculados del sistema de ecuaciones 3.7.7

$$eigen\begin{bmatrix} 0 & 1 & 0 \\ -9.8 & 0 & 1 \\ 0 & -10 & -10 \end{bmatrix} \Rightarrow \begin{aligned} \lambda_1 &= -0.4952 + j3.2607 \\ \lambda_2 &= -0.4952 - j3.2607 \\ \lambda_3 &= -9.0096 \end{aligned} \tag{3.7.12}$$

Como se observa, los valores del Jacobiano de la red multicapa se encuentran próximos a los del sistema, por lo tanto la red a identificado el sistema correctamente, nótese además que todos los valores propios tanto de la red como del sistema se encuentran en el semiplano complejo izquierdo haciendo que el sistema y la red neuronal sean estables.

Según lo demostrado en la sección 2.6.2, una red dinámica multicapa puede ser transformada en una red dinámica tipo Hopfield por medio de la siguiente transformación:

$$\frac{d}{dt}\chi = -I_N\chi + TW\sigma(.) \tag{3.7.13}$$

Tomando los valores de los parámetros de la red multicapa y reemplazándolos en la ecuación 3.7.13 se obtiene la matriz de pesos de la red dinámica de Hopfield equivalente a la red dinámica recurrente, la matriz de pesos de la red de Hopfield se muestra en la tabla 3.7.6

Matriz de Pesos Red de Hopfield = TW

-0.0003	7.4337	5.7721	-0.8742	-93.4671
-0.2803	-3.7494	2.9285	3.6020	0.0568
0.2810	3.7589	-2.9370	-3.6117	-0.0488
0.3813	5.0998	-3.9839	-4.8997	-0.0721
0.0979	-0.2832	0.2229	0.2730	0.8793

Tabla 3.7.6 Matriz de pesos Red de Hopfield equivalente

El Jacobiano de la red de Hopfield según lo demostrado en la sección 2.6.2 se calculan como:

$$J_H = -I_N + TW \tag{3.7.14}$$

y tiene los siguientes valores propios:

$$\begin{aligned}
\lambda_1 &= -0.5207 + j3.2816 \\
\lambda_2 &= -0.5207 - j3.2816 \\
\lambda_3 &= -12.6657 \\
\lambda_4 &= -1 \\
\lambda_5 &= -1
\end{aligned} \tag{3.7.15}$$

Nótese que los tres primeros valores propios de la ecuación 3.7.15 coinciden con los valores propios de la ecuación 3.7.12, y éstos a su vez se encuentran muy próximos a los valores propios del sistema, los valores propios adicionales de la ecuación 3.7.15 corresponden a los estados adicionales que se agregaron para una adecuada identificación del sistema.

Para comprobar la efectividad de la aproximación, en la figura 3.7.3 se muestra la comparación de la respuesta de las variables de estado del sistema con la respuesta de la red neuronal multicapa; como la identificación que se realizó fue del sistema autónomo ($u=0$), para la simulación se requieren condiciones iniciales en las variables de estado del sistema y en las variables de estado estimadas de la res multicapa, los valores iniciales para la simulación fueron: $X1^0=0.2$, $X2^0=-0.3$, $X3^0=0$.

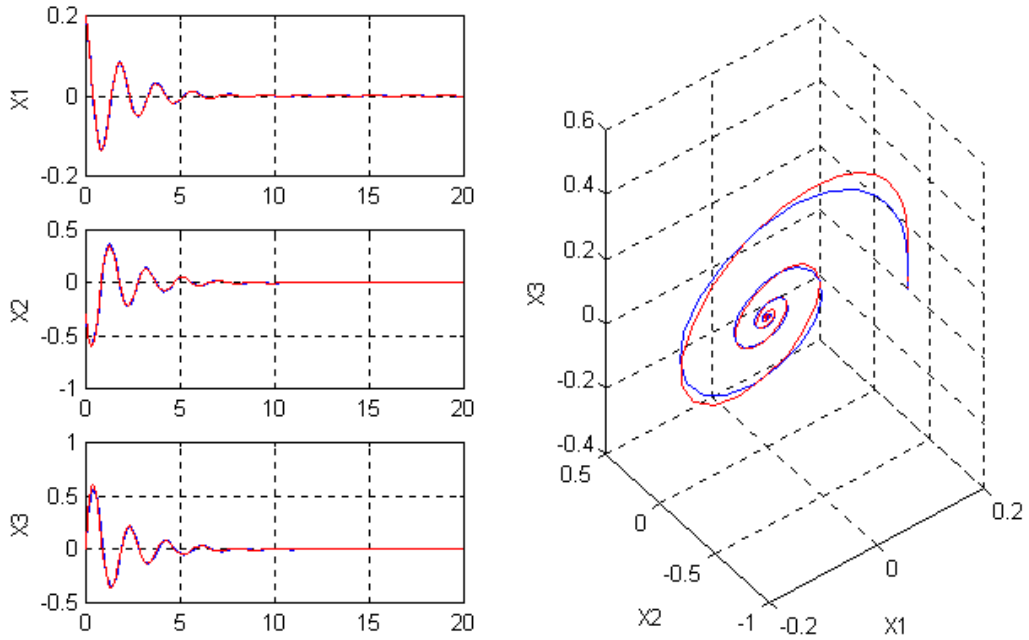

Figura 3.7.3 Respuesta del sistema Vs Red Multicapa

La línea azul representa la respuesta del sistema y la línea roja muestra la respuesta de la red dinámica multicapa, en la gráfica de la izquierda se grafica cada variable de estado en función del tiempo y en la gráfica de la derecha se tienen los planos de fase descritos por las tres variables de estado, de la gráfica 3.7.3 se observa, que se obtuvo una muy buena aproximación del sistema autónomo y de la ecuación 3.7.11 se tiene una buena aproximación de los valores propios del sistema

3.7.3.2 Identificación de la dinámica del Sistema. Se entrenará una red recurrente según el procedimiento descrito en el tutorial del Matlab versión 5.3 [34], esta red tiene como entradas el ángulo, la velocidad, la corriente de armadura y el voltaje aplicado, para identificar

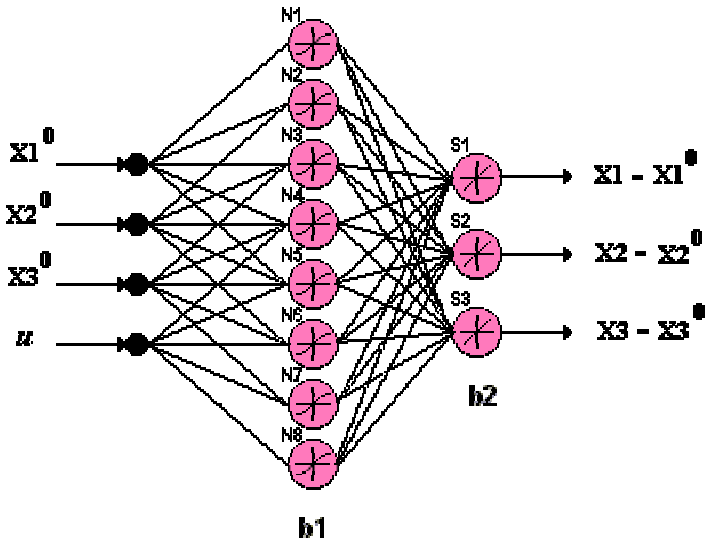

Figura 3.7.4 Capa Estática Red Recurrente

De la figura 3.7.4, se puede observar que la red recurrente posee una capa backpropagation, para la cual el set de entrenamiento fue generado con variaciones en el ángulo, la velocidad, la corriente y todas sus posibles combinaciones

	Valor Inicial	Valor Final	Incremento	Conversión
Ang	-30º	30º	15º	pi/180º
Vel	-115º	55º	115º	pi/180º
ia	0	8	2	-
u	-6	3	6	-

Tabla 3.7.7 Datos de entrenamiento

Adicionalmente se incluyen en el set de entrenamiento datos de posiciones iniciales con velocidad y corriente igual a cero

	Valor Inicial	Valor Final	Incremento	Conversión
Ang	-30º	30º	6º	pi/180º
Vel	0	0	0	-
ia	0	0	0	-
u	0	0	0	-

Tabla 3.7.8 Datos adicionales de entrenamiento

Todos los patrones de entrenamiento se evaluaron en el conjunto de ecuaciones que describen el comportamiento del sistema, las ecuaciones diferenciales no lineales fueron resueltas en cada caso para un tiempo tstep=0.05s, la salida esperada para cada patrón de entrada es el vector solución X menos el valor de Xo utilizado como condición inicial.

```
tstep=0.05;
t=zeros(3,length(P));
for i=1:length(P)
    [time,X]=ode45('modelo',[0 tstep],P(:,i));
   t(:,i)=X(length(X),1:3)'-P(1:3,i);
end
```

P	P1	P2		P72		P144	P145		P340		P636
$X1^0$	-0.5236	-0.2618	....	-0.2618	....	0.2618	0.5236	....	0.5236	....	0.5236
$X2^0$	-2.0071	-2.0071	....	1.8326	....	0.8727	0.8727	....	-0.0873	....	0.0000
$X3^0$	0.0000	0.0000	....	4.0000	....	0.0000	0.0000	....	6.0000	....	0.0000
u	-6.0000	-6.0000	....	-6.0000	....	-3.0000	-3.0000	....	0.0000	....	0.0000

t	t1	t2		t72		t144	t145		t340		t636
$X1-X1^0$	-0.0946	-0.0975	...	0.0972	...	0.0396	0.0367	...	-0.0041	...	-0.0061
$X2-X2^0$	0.2217	0.1069	...	0.1770	...	-0.1772	-0.2935	...	-0.0072	...	-0.2432
$X3-X3^0$	-1.6197	-1.5950	...	-4.7026	...	-1.4890	-1.4641	...	-2.3287	...	0.0520

Tabla 3.7.9 Patrones de entrenamiento P y patrones esperados t

La red fue entrenada con el algoritmo backpropagation *trainlm* con los siguientes pesos iniciales:

W1i=net.IW{1,1}

	I1	I2	I3	I4
N1	4.3432	0.1214	-0.1408	-0.0044
N2	1.8875	0.1985	-0.2226	0.3175
N3	1.7496	0.7324	-0.3268	0.1682
N4	3.6430	-0.1749	-0.3306	0.0347
N5	2.6789	0.6421	0.0647	0.2351
N6	-2.4939	-0.5881	0.3700	0.1018
N7	-1.4134	-0.1684	-0.4461	0.2178
N8	-1.7871	0.1220	0.0541	-0.3562

b1i=net.b{1}

N1	-1.7807
N2	-0.7742
N3	0.3622
N4	0.9709
N5	0.1337
N6	-2.5404
N7	0.0877
N8	-2.5602

W2i=net.LW{2,1}

	N1	N2	N3	N4	N5	N6	N7	N8
S1	-0.5745	-0.8180	-0.1714	0.8758	-0.3649	-0.6993	-0.2245	0.1744
S2	0.4294	-0.4508	-0.9462	-0.5202	0.7740	0.3627	-0.0005	0.6912
S3	-0.7391	-0.9940	0.4196	-0.6382	0.3041	-0.2284	-0.7049	0.1802

b2f=net.b{2}

S1	0.9108
S2	0.1123
S3	-0.7037

Tabla 3.7.10 Pesos iniciales red recurrente

El siguiente código entrena la red backpropagation de 4 entradas, 8 neuronas en la capa oculta y 3 salidas con el algoritmo de entrenamiento *trainlm.*

```
net=newff(minmax(P),[8 3],{'tansig' 'purelin'},'trainlm');
net.trainParam.epochs=700;
net.trainParam.goal=1e-8;
net.trainParam.lr=0.03
net.trainParam.mem_reduc=1
net.trainParam.min_grad=1e-10
[net,tr]=train(net,P,t);
```

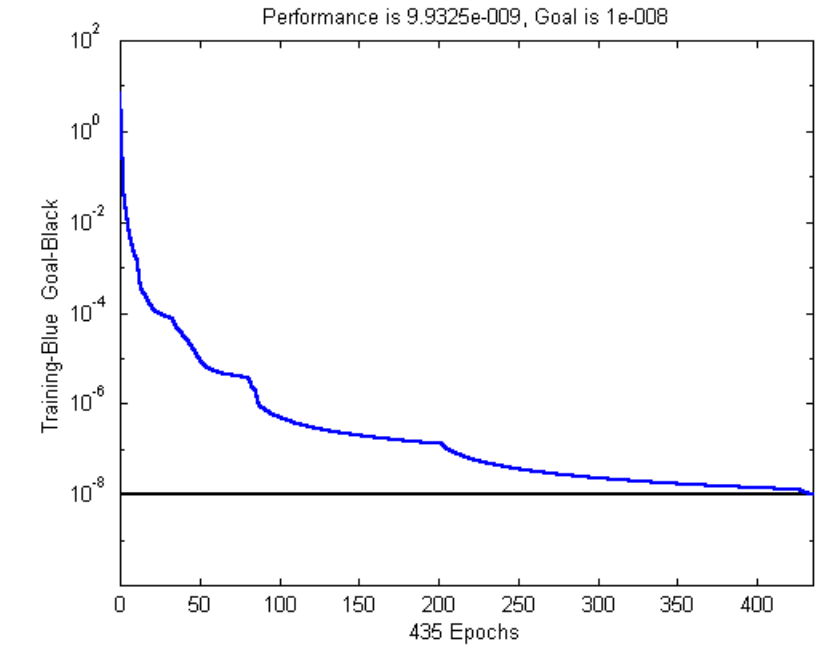

Figura 3.7.5 Error medio cuadrático utilizando *trainlm*

Después de 435 iteraciones de entrenamiento, el error medio cuadrático cayó por debajo de $1x10^{-8}$, los pesos y ganancias de la red entrenada son:

W1f=net.IW{1,1}

	I1	I2	I3	I4
N1	-0.0043	0.0241	0.0298	-0.0246
N2	0.7562	0.0193	0.0003	0.0001
N3	0.0139	0.0040	-0.0005	-0.0173
N4	5.3898	-0.9620	0.1449	0.0234
N5	0.0007	0.0333	-0.0006	-0.0002
N6	-0.2605	-0.0725	0.0168	0.2906
N7	-3.5076	-0.4256	-1.7086	0.0763
N8	-0.0047	0.0266	0.0329	-0.0272

b1f=net.b{1}

N1	0.5084
N2	0.0001
N3	-0.0546
N4	2.5040
N5	0.0052
N6	-1.8745
N7	0.2389
N8	-0.8470

W2f=net.LW{2,1}

	N1	N2	N3	N4	N5	N6	N7	N8
S1	0.0516	-0.0148	-0.1203	0.00001	1.4688	0.0002	0.00000	0.0425
S2	0.9975	-0.5914	-2.3910	0.00000	-0.9662	0.0033	-0.00001	0.8211
S3	-10.8348	0.1225	-3.2830	-0.00005	-1.5610	0.0045	0.00003	-8.8873

b2f=net.b{2}

S1	-0.0089
S2	-0.0235
S3	-1.2166

Tabla 3.7.11 Pesos finales de la red recurrente

Para comprobar si la red neuronal recurrente identifica correctamente la dinámica de la planta, esta fue simulada con los pesos finales anteriormente encontrados utilizando el siguiente algoritmo, con condiciones iniciales mostradas de las variables de estado X1 (ángulo) = -8º, X2 (velocidad) = 0 º/s, X3 (corriente de armadura) = 0 A, y u (voltaje)= 1 V, estos valores están contenidos en Xo

```
Xo=[-8*pi/180; 0; 0];
u=1;
tstep=0.05;
times=0:tstep:10;
estado=Xo;
estados=zeros(3,length(times));
estados(:,1)=estado;
for i=2:length(times)
  estado=estado+sim(net,[estado;u]);
  estados(:,i)=estado;
end
```

Para obtener la respuesta de la red en el tiempo deseado, esta debe ser simulada en intervalos de tiempo iguales al tiempo para el cual fueron resueltas las ecuaciones del set de entrenamiento, en este caso las ecuaciones fueron resueltas para un tiempo tstep=0.05s, por lo tanto para obtener la respuesta de la red en un tiempo total de 10s, debe simularse 201 veces en intervalos de 0.05s.

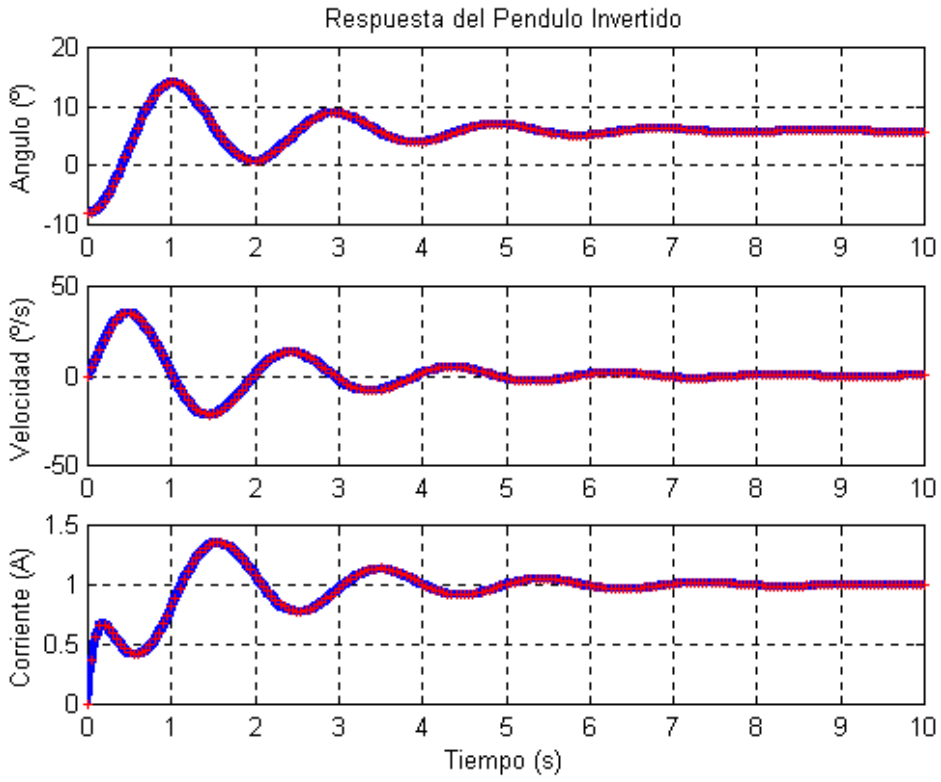

Figura 3.7.6 Comparación de la respuesta del sistema y la RN

La línea continua azul representa la respuesta del sistema simulado y las cruces rojas representan las simulaciones de la red neuronal entrenada, como se observa la red neuronal ha identificado la dinámica de la red perfectamente, en la siguiente gráfica puede observarse que los planos de fase descritos por la planta y por la red neuronal coinciden exactamente.

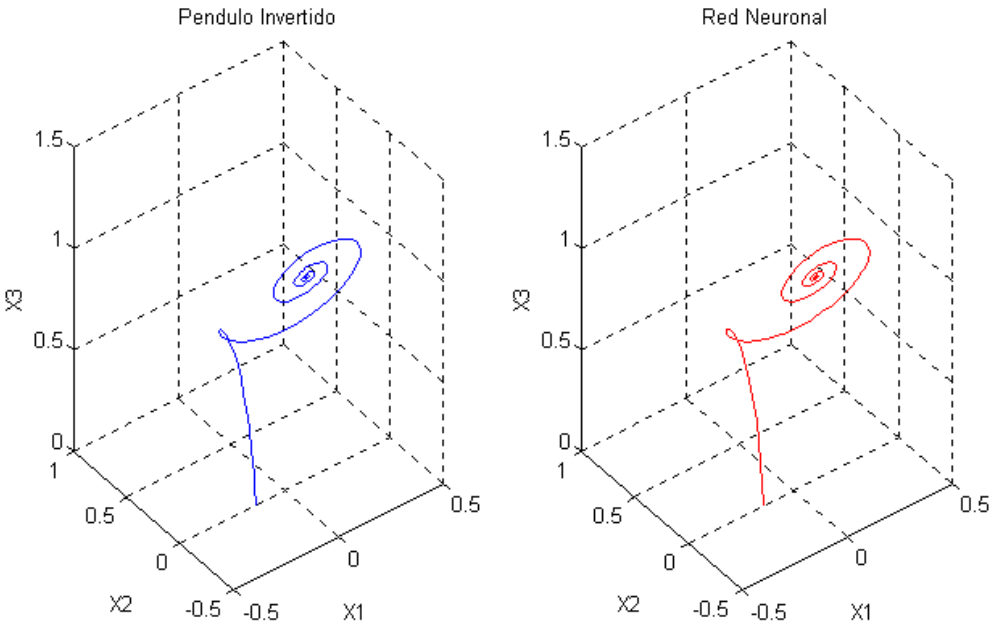

Figura 3.7.7 Comparación de los planos de fase

Como se observa en la figura 3.7.7, cuando la misma red neuronal es simulada en intervalos de tiempo distintos al tiempo para el cual fueron solucionadas las ecuaciones que generaron el set de entrenamiento, la red produce las mismas respuestas que la planta en estado estacionario, pero en el estado transitorio la

forma de la respuesta es similar pero en el caso de un intervalo de simulación menor tstep=0.3 la red neuronal responde mas rápido que la planta y en el caso de un intervalo de simulación mayor tstep=0.10 la red neuronal responde mas lento que la planta, para una identificación óptima de la planta tanto en estado estacionario como en estado el intervalo de simulación debe ser igual a tiempo de solución de las ecuaciones que generaron los patrones de entrenamiento.

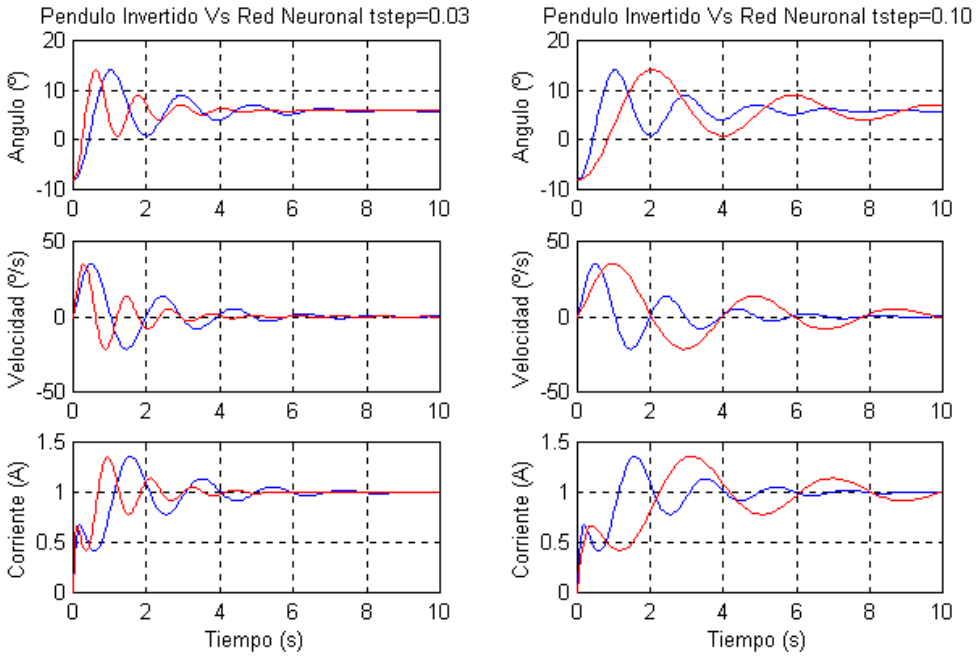

Figura 3.7.8 Red neuronal con diferentes tiempos de simulación