3.5 CONTROL DE VOLTAJE POR INYECCIÓN DE REACTIVOS EN UNA BARRA REMOTA

3.5.1 Descripción del problema. El objetivo de un sistema de potencia es suministrar a las barras de carga, potencia activa y reactiva para su consumo conservando los niveles de voltaje dentro de los limites establecidos. Sin embargo los niveles de tensión en las barras están determinados por las condiciones del sistema, no obstante es posible realizar un control en el nivel de tensión inyectando reactivos en una barra remota, para de este modo controlar los niveles de tensión. Cuando se desea especificar el nivel de voltaje en una barra de carga PQ, ésta barra se declara como tipo PQV, la barra donde se inyecta potencia reactiva que generalmente es de tipo PV se declara como tipo P.

Las ecuaciones básicas que describen el flujo de carga en un sistema de potencia son las siguientes:

$$P_k = V_k \sum_{m \in k} V_m \left(G_{km} Cos\theta_{km} + B_{km} Sen\theta_{km} \right) \qquad (3.5.1)$$

$$Q_k = V_k \sum_{m \in k} V_m \left(G_{km} Sen\theta_{km} - B_{km} Cos\theta_{km} \right) \qquad (3.5.2)$$

Para $k = 1, \ldots, Nb$ $\qquad$ $Nb \Longrightarrow$ Número de barras del sistema

Copyright©2000 Universidad Tecnológica de Pereira

La solución de las ecuaciones de flujo de potencia generalmente se realizan por el método de Newton-Raphson, donde se calculan las derivadas de P y Q de las ecuaciones (3.5.1) y (3.5.2) respecto a las variables V y θ. El sistema de ecuaciones no lineales en forma matricial se describe en la ecuación 3.5.3, donde el Jacobiano del sistema es la matriz después de la igualdad.

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = \begin{bmatrix} \frac{\partial P}{\partial \theta} & \frac{\partial P}{\partial V} \\ \frac{\partial Q}{\partial \theta} & \frac{\partial Q}{\partial V} \end{bmatrix} \begin{bmatrix} \Delta \theta \\ \Delta V \end{bmatrix} \qquad (3.5.3)$$

Para la solución de flujo de potencia cada barra aporta distintas ecuaciones e incógnitas según su tipo, como se puede ver en las tablas 3.5.1 y 3.5.2

• Barra tipo PV Ecuación: $\Delta P_k = P_k^{esp} - P_k^{cal}$ Incógnita: θ_k	• Barra tipo PQ Ecuaciones: $\Delta P_k = P_k^{esp} - P_k^{cal}$ $\Delta Q_k = Q_k^{esp} - Q_k^{cal}$ Incógnitas: V_k , θ_k

Tabla 3.5.1 Ecuaciones e incógnitas barras, tipo PV y PQ

Cuando se desea realizar control de voltaje por inyección de reactivos en una barra remota, las barras tipo P y PQV proporcionan las siguientes ecuaciones:

Copyright©2000 Universidad Tecnológica de Pereira

• Barra tipo P o de Control	• Barra tipo PQV o Controlada
Ecuación: $\Delta P_k = P_k^{esp} - P_k^{cal}$	Ecuaciones: $\Delta P_k = P_k^{esp} - P_k^{cal}$ $\Delta Q_k = Q_k^{esp} - Q_k^{cal}$
Incógnitas: V_k, θ_k	Incógnita: θ_k

Tabla 3.5.2 Ecuaciones e incógnitas barras tipo P y PQV

En resumen las ecuaciones e incógnitas que aporta cada tipo de barra se sintetiza en el grupo de ecuaciones (3.5.4).

$$\begin{matrix} NPQV \\ NPQ \\ NPV \\ NP \end{matrix}\left\{\begin{matrix} \\ \\ \\ \\ \end{matrix}\right.\left[\begin{matrix} \\ \Delta P \\ \\ \\ \end{matrix}\right] = \left[\begin{matrix} \dfrac{\partial P}{\partial \theta} & \dfrac{\partial P}{\partial V} \\ \\ \dfrac{\partial Q}{\partial \theta} & \dfrac{\partial Q}{\partial V} \end{matrix}\right]\left[\begin{matrix} \\ \Delta \theta \\ \\ \Delta V \end{matrix}\right]\left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\}\begin{matrix} NPQV \\ NPQ \\ NPV \\ NP \end{matrix}$$
$$\begin{matrix} NPQ \\ NPQV \end{matrix}\left\{\begin{matrix} \\ \\ \end{matrix}\right.\left[\begin{matrix} \\ \Delta Q \\ \\ \end{matrix}\right]\left.\begin{matrix} \\ \\ \\ \end{matrix}\right\}\begin{matrix} NPQ \\ NP \end{matrix} \qquad (3.5.4)$$

Donde: $NP = NPQV \Rightarrow$ Número de barras con control remoto de voltaje.

$NPQ, NPV \Rightarrow$ Número de barras tipo PQ y PV respectivamente

Esta aplicación se llevará a cabo en un sistema de tres barras donde la barra 1 es una barra slack o de compensación Vθ, la barra 2 es una barra tipo P, desde donde se pretende controlar por inyección de reactivos el voltaje de la barra 3, la cual es una barra tipo PQV.

Copyright©2000 Universidad Tecnológica de Pereira

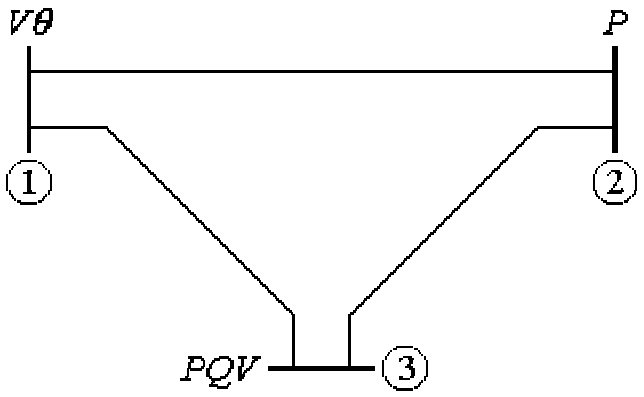

Figura 3.5.1 Sistema de 3 barras

El sistema tiene como potencia base Sb=100MVA y voltaje base Vb=230kV, el voltaje en el nodo slack se mantendrá en V1=1.015p.u.

Las ecuaciones no lineales que describen el comportamiento del flujo de potencia en el sistema de tres barras en forma matricial son las siguientes:

$$\begin{bmatrix} \Delta P_2 \\ \Delta P_3 \\ \Delta Q_3 \end{bmatrix} = \begin{bmatrix} \frac{\partial P_2}{\partial \theta_2} & \frac{\partial P_2}{\partial \theta_3} & \frac{\partial P_2}{\partial V_2} \\ \frac{\partial P_3}{\partial \theta_2} & \frac{\partial P_3}{\partial \theta_3} & \frac{\partial P_3}{\partial V_2} \\ \frac{\partial Q_3}{\partial \theta_2} & \frac{\partial Q_3}{\partial \theta_3} & \frac{\partial Q_3}{\partial V_2} \end{bmatrix} \begin{bmatrix} \Delta \theta_2 \\ \Delta \theta_3 \\ \Delta V_2 \end{bmatrix} \qquad (3.5.5)$$

$$\begin{bmatrix} \Delta P_2 \\ \Delta P_3 \\ \Delta Q_3 \end{bmatrix} = \begin{bmatrix} H_{22} & H_{32} & N_{22} \\ H_{32} & H_{33} & N_{32} \\ M_{32} & M_{33} & L_{22} \end{bmatrix} \begin{bmatrix} \Delta \theta_2 \\ \Delta \theta_3 \\ \Delta V_2 \end{bmatrix} \qquad (3.5.6)$$

Copyright©2000 Universidad Tecnológica de Pereira

$$H_{22} = -V_2 B_{22} - Q_2 \tag{3.5.7}$$

$$H_{23} = V_2 V_3 \left(G_{23} Sen\left(V_2 - V_3\right) - B_{23} Cos\left(V_2 - V_3\right)\right) \tag{3.5.8}$$

$$H_{32} = V_3 V_2 \left(G_{32} Sen\left(V_3 - V_2\right) - B_{32} Cos\left(V_3 - V_2\right)\right) \tag{3.5.9}$$

$$H_{33} = -V_3 B_{33} - Q_3 \tag{3.5.10}$$

$$N_{22} = \left(P_2 / V_2\right) + V_2 G_{22} \tag{3.5.11}$$

$$N_{32} = V_3 \left(G_{32} Cos\left(V_3 - V_2\right) + B_{32} Sen\left(V_3 - V_2\right)\right) \tag{3.5.12}$$

$$M_{32} = -V_3 V_2 \left(G_{32} Cos\left(V_3 - V_2\right) + B_{32} Sen\left(V_3 - V_2\right)\right) \tag{3.5.13}$$

$$M_{33} = -V_3^{\,2} G_{33} + P_3 \tag{3.5.14}$$

$$L_{32} = V_3 \left(G_{32} Sen\left(V_3 - V_2\right) - B_{32} Cos\left(V_3 - V_2\right)\right) \tag{3.5.15}$$

El proceso de cálculo se realiza de forma iterativa, cuando el sistema de ecuaciones ha convergido por debajo del error deseado se obtiene V_2, de las condiciones iniciales del flujo de carga se tienen V_1 y V_3 y con estos datos se calcula la potencia reactiva inyectada en la barra 2 (Q_2), que satisface las condiciones impuestas en la barra 3.

$$\begin{aligned} Q_2 = -V_2^2 B_{22} + V_2 V_1 \left(G_{21} Sen\left(V_2 - V_1\right) - B_{21} Cos\left(V_2 - V_1\right)\right) + \\ V_2 V_3 \left(G_{23} Sen\left(V_2 - V_3\right) - B_{23} Cos\left(V_2 - V_3\right)\right) \end{aligned} \tag{3.5.16}$$

Copyright©2000 Universidad Tecnológica de Pereira

Los datos de las líneas en p.u. son:

	Z serie	Y/2 paralelo
1-2	0.0229+j0.0400	j0.036
2-3	0.0118+j0.0333	j0.027
3-1	0.0046+j0.0267	j0.040

Tabla 3.5.3 Datos de las líneas del sistema de tres barras

Y bus en p.u.

	1	2	3
1	17.04606-j55.12630	-10.77946+j18.82875	-6.26660+j36.37354
2	-10.77946+j18.82875	22.37953-j41.94710	-12.60006+j23.18134
3	-6.26660+j36.37354	-12.60006+j23.18134	18.86667-j59.48788

Tabla 3.5.4 Ybus del sistema de tres barras

3.5.2 Justificación del tipo de red. El problema de flujos de potencia está regido por ecuaciones no lineales, las cuales según lo demostrado por Funahashi[16] pueden ser aprendidas por una red multicapa, en esta publicación el autor demuestra que siempre existe la red neuronal apropiada; este teorema es sólo de existencia pues prueba que la red existe pero no indica como construirla ni tampoco garantiza que la red aprenderá la función.

La red neuronal de propagación inversa o Backpropagation es una generalización para redes multicapa del algoritmo LMS. Ambas utilizan la técnica del error medio cuadrático, pero la red Backpropagation es más ampliamente utilizada para aproximación de funciones en especial no lineales.

Copyright©2000 Universidad Tecnológica de Pereira

3.5.3 Entrenamiento de la Red

3.5.3.1. Problema utilizando como entradas P3, Q3 y V3; para obtener Q2 utilizando 10 neuronas en la capa oculta

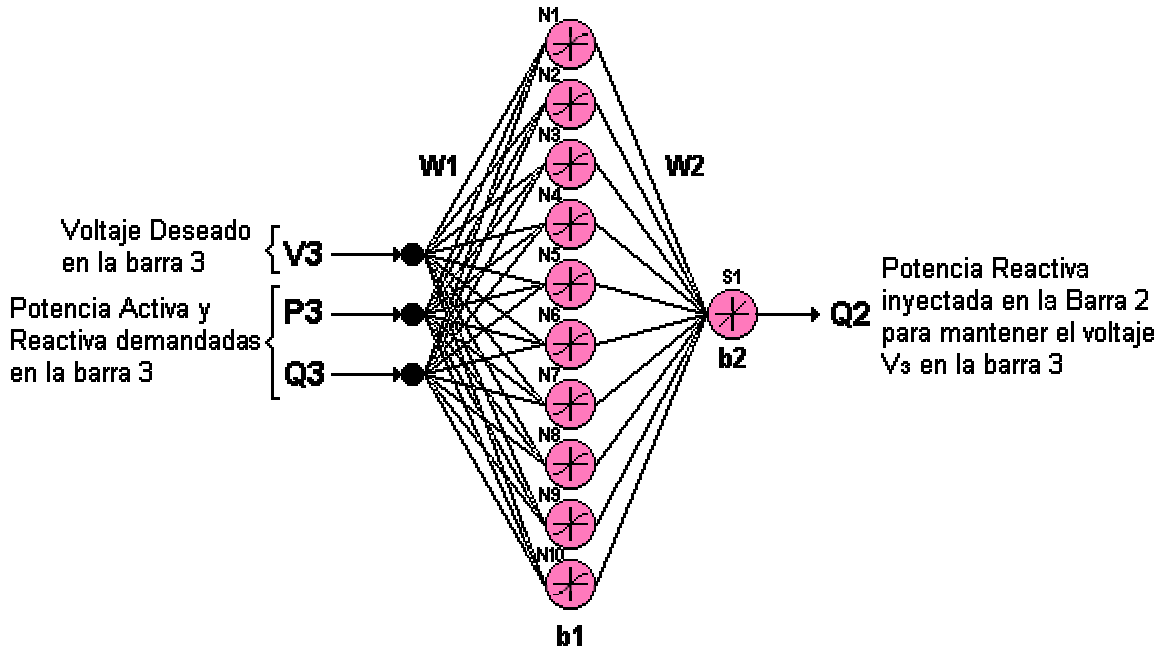

Figura 3.5.2 Red neuronal en configuración 3:10:1

La red será entrenada para determinar la cantidad de potencia reactiva en p.u. inyectada en la barra dos (Q_2) para satisfacer las condiciones impuestas por V_3, P_3 y Q_3, que son el voltaje, la potencia activa y reactiva demandada en la barra 3. Para generar el set de entrenamiento, se resolvieron las ecuaciones no lineales que describen el comportamiento del sistema por el método de Newton-Raphson considerando variaciones de V_3, P_3, Q_3; cada variable se incremento según lo mostrado en la tabla 3.5.3; se mantuvieron constantes el voltaje en el nodo slack V_1=1.015p.u. y la potencia activa demandada en la barra dos, P_2=0.7p.u.

Copyright©2000 Universidad Tecnológica de Pereira

	Valor Inicial	Valor Final	Incremento
V3	0.997	1.015	0.003
P3	0.700	0.850	0.030
Q3	0.250	0.400	0.030

Tabla 3.5.5 Patrones de entrenamiento

De estas variaciones se obtuvieron 252 combinaciones que conforman el vector de patrones de entrenamiento ***p***, que generan a su vez igual número de patrones objetivo o salidas deseadas ***t***.

P	P1	P2		P36		P78	P79		P140		P252
V3	0.9970	0.9970	....	0.9970	....	1.0030	1.0030	....	1.0060	....	1.0150
P3	0.8500	0.8500	....	0.7000	....	0.8500	0.8200	....	0.7000	....	0.7000
Q3	0.2500	0.2800	....	0.4000	....	0.4000	0.2500	....	0.2800	....	0.4000

t	t1	t2		t36		t78	t79		t140		t252
Q2	-0.7480	-0.6939	....	-0.5495	....	0.0685	-0.2244	....	0.0464	....	1.1281

Tabla 3.5.6 Patrones de entrenamiento de la red neuronal

Los pesos iniciales de la red fueron obtenidos de la función *initlay* que calcula los pesos iniciales con base al rango de los datos de entrada ***p***.

W1i=net.IW{1,1}

	I1	I2	I3
N1	-82.4449	-30.1178	24.7463
N2	251.4143	-20.6245	-16.7856
N3	-325.2046	-1.0347	9.6617
N4	325.4475	-9.5779	0.6391
N5	329.3532	-5.3503	-5.1664
N6	-142.9295	34.0779	12.7224
N7	64.3926	-27.4716	28.3361
N8	294.8647	-14.5486	-12.3960
N9	231.7537	21.5285	-19.5048
N10	-186.3720	-28.7647	-17.0220

b1i=net.b{1}

N1	101.2545
N2	-233.8294
N3	326.4934
N4	-321.1904
N5	-325.8389
N6	112.9068
N7	-51.6923
N8	-279.6544
N9	-241.1438
N10	212.2988

Copyright©2000 Universidad Tecnológica de Pereira

W2i=net.LW{2,1}

	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10
S1	-0.1769	-0.4281	-0.2117	0.0060	0.4440	-0.3876	-0.7757	-0.1134	-0.0665	-0.9707

b2i=net.b{2}

S1	0.3281

Tabla 3.5.7 Pesos iniciales para la red neuronal

Se entrenará la red de la figura 3.5.2 con 3 distintos algoritmos de aprendizaje y luego se simulará cada red con los siguientes patrones de prueba:

	1	2	3	4	5	6	7	8
V3	1.0120	**0.9950**	1.0100	**1.0200**	1.0070	**1.0160**	1.0050	1.0100
P3	0.7320	0.8300	0.8250	0.8370	0.7520	**0.9120**	**0.8720**	0.8500
Q3	0.3710	0.3600	**0.2300**	**0.4100**	0.3230	0.2500	**0.4500**	**0.2020**
Q2	0.7992	-0.7359	0.3895	1.7141	0.2457	1.0476	0.3588	0.3497

	9	10	11	12	13	14	15	16
V3	1.0100	**1.0170**	1.0050	1.0120	1.0030	**0.9950**	**1.0200**	1.0100
P3	0.8430	**0.9000**	0.7500	0.8500	0.8200	**0.6500**	**0.9000**	0.8000
Q3	0.3570	**0.1500**	0.3100	**0.5000**	0.2800	**0.1500**	**0.2000**	0.3000
Q2	0.6390	0.9463	0.0349	1.1089	-0.1691	-1.1977	1.3368	0.5088

Tabla 3.5.8 Patrones de prueba de las RN's

Los valores de prueba que se encuentran en negrilla se encuentran por fuera del rango de entrenamiento y buscan probar la capacidad de extrapolación de la red, la mayoría de los datos de prueba que se encuentran dentro del rango de entrenamiento no coinciden exactamente con el valor utilizado en el entrenamiento para probar la capacidad de interpolación de la red.

Copyright©2000 Universidad Tecnológica de Pereira

- Entrenamiento utilizando el algoritmo *trainrp*

El siguiente código entrena una red neuronal con 3 entradas, 10 neuronas en la capa oculta y 1 en la capa salida, utilizando la función de aprendizaje *trainrp* (Ver Anexo 1) y como objetivo un error medio cuadrático de $6x10^{-6}$.

```
net=newff([0.997 1.015;0.7 0.85;0.25 0.4],[10,1],
          {'tansig','purelin'},'trainrp');
net.trainParam.epochs=50000;
net.trainParam.goal=6e-6;
net.trainParam.min_grad=1e-10;
net.trainParam.lr=0.03;
net.trainParam.delt_inc=1.08;
net.trainParam.delt_dec=0.70;
net.trainParam.delta0=0.08;
net.trainParam.deltamax=50.0;
[net,tr]=train(net,P,t);
```

El error medio cuadrático objetivo fue fijado en $6x10^{-6}$, puesto que con la variación de los distintos parámetros no fue posible alcanzar un valor inferior; los mejores resultados se obtuvieron para una rata de aprendizaje de 0.03, para esta red un valor superior causa serias inestabilidades lo que a su vez ocasiona un proceso de aprendizaje bastante pobre (la red generaliza de manera incorrecta) y un valor inferior produce un aprendizaje bastante lento.

La figura 3.5.3 muestra la evolución del error medio cuadrático a través de 50000 iteraciones hasta alcanzar el error deseado.

Copyright©2000 Universidad Tecnológica de Pereira

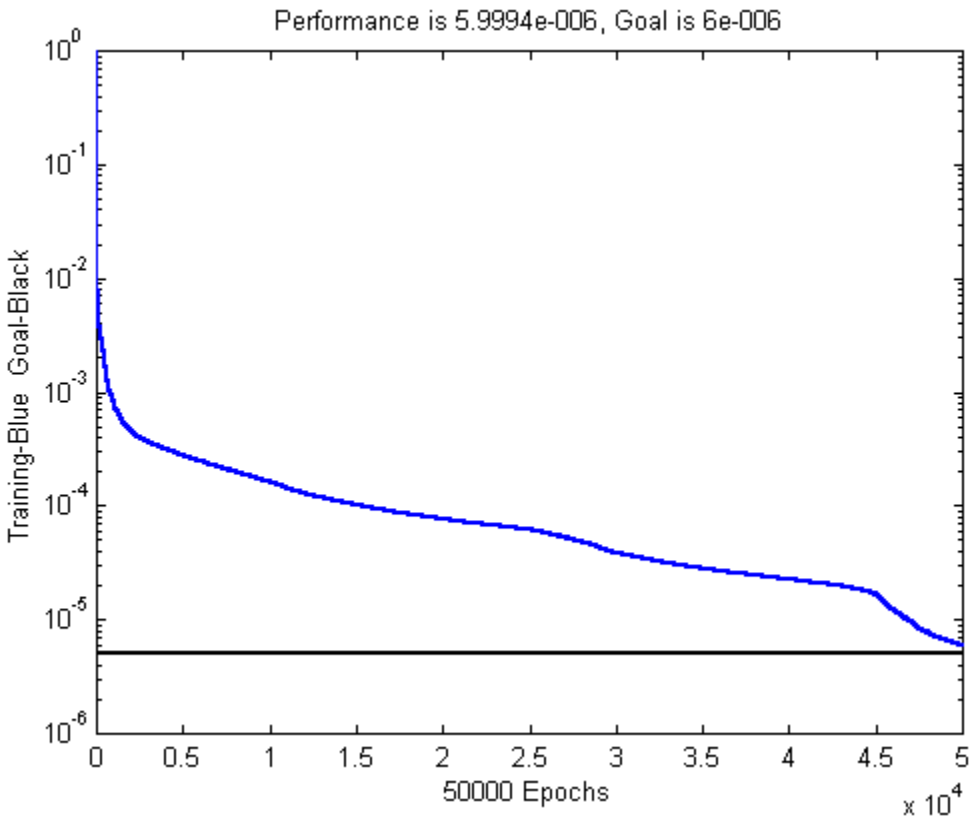

Figura 3.5.3 Error medio cuadrático utilizando *trainrp*

Luego de 50000 iteraciones el error medio cuadrático cayó por debajo de $6x10^{-6}$, los pesos y ganancias de la red entrenada son:

W1f=net.IW{1,1}

	I1	I2	I3
N1	-79.6128	-24.9124	34.8705
N2	252.3923	-15.6532	-27.2092
N3	-323.5910	-0.0197	0.3244
N4	322.9282	-0.2053	-0.1796
N5	325.8474	-0.0746	0.2680
N6	-133.9024	3.5832	15.9724
N7	59.2478	1.9485	7.0366
N8	275.5195	5.1249	21.3110
N9	234.3145	-18.7189	-245.3248
N10	-200.4902	-7.0055	-22.0806

b1f=net.b{1}

N1	112.1139
N2	-232.4849
N3	327.5410
N4	-324.6580
N5	-325.7568
N6	127.9533
N7	-63.8474
N8	-284.2129
N9	-129.0298
N10	212.8007

Copyright©2000 Universidad Tecnológica de Pereira

W2f=net.LW{2,1}

	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10
S1	-0.0688	-0.0069	-0.2988	0.2533	0.2656	0.0251	0.2475	0.0563	-0.0031	-0.0346

b2f=net.b{2}

S1	0.2726

Tabla 3.5.9 Pesos finales de la RN

Para hallar la respuesta de la red (Q2Rn) ante los patrones de prueba, se simuló la red y para hallar el error en las aproximaciones se comparó con la respuesta producida por las ecuaciones del flujo de carga (Q2Fc).

	1	2	3	4	5	6	7	8
Q2Fc	0.7992	-0.7359	0.3895	1.7141	0.2457	1.0476	0.3588	0.3497
Q2Rn	0.7980	-0.6457	0.3799	1.3230	0.2549	1.0028	0.3215	0.3437
Error	0.0011	-0.0902	0.0097	0.3911	-0.0092	0.0448	0.0374	0.0060

	9	10	11	12	13	14	15	16
Q2Fc	0.6390	0.9463	0.0349	1.1089	-0.1691	-1.1977	1.3368	0.5088
Q2Rn	0.6333	0.8745	0.0268	0.9868	-0.1672	-0.9485	1.0116	0.4964
Error	0.0057	0.0718	0.0081	0.1220	-0.0019	-0.2492	0.3252	0.0123

Tabla 3.5.10 Simulación de la RN con los patrones de prueba

Los errores más altos se presentaron en los patrones de prueba 4 (2), 12 (1), 14 (3) y 15 (3), donde los valores entre paréntesis son el número de datos extrapolados del set de entrenamiento; algunos resultados son muy próximos a los valores que producirían las ecuaciones de flujo de carga que describen el comportamiento del sistema, pero en general el aprendizaje fue muy lento (50000 iteraciones) y la aproximación es muy deficiente.

Copyright©2000 Universidad Tecnológica de Pereira

- Entrenamiento utilizando el algoritmo *trainbfg*

El siguiente código entrena una red neuronal con 3 entradas, 10 neuronas en la capa oculta y 1 en la salida, utilizando la función de aprendizaje *trainbfg* (Ver Anexo 1) y como objetivo un error medio cuadrático de $1.35x10^{-8}$.

```
net=newff([0.997 1.015;0.7 0.85;0.25 0.4],[10,1],
          {'tansig','purelin'},'trainbfg');
net.trainParam.epochs=1800;
net.trainParam.goal=1.35e-8;
net.trainParam.min_grad=1e-10;
net.trainParam.searchFcn='srchcha'
net.trainParam.scal_tol=20;
net.trainParam.alpha=0.001;
net.trainParam.beta=0.1;
net.trainParam.delta=0.01;
net.trainParam.gama=0.1;
net.trainParam.low_lim=0.1;
net.trainParam.up_lim=0.5
[net,tr]=train(net,P,t);
```

El error medio cuadrático fue fijado en $1.35x10^{-8}$, por debajo de este valor la matriz Jacobiana del algoritmo se vuelve singular o sin inversa y por este motivo se presenta un estancamiento en el proceso de aprendizaje, pero antes de que esto suceda los resultados obtenidos presentan un error medio cuadrático bastante pequeño, produciendo una muy buena aproximación y generalización de las funciones deseadas

La figura 3.5.4 muestra la evolución del error medio cuadrático a través de 1523 iteraciones hasta alcanzar el error deseado.

Copyright©2000 Universidad Tecnológica de Pereira

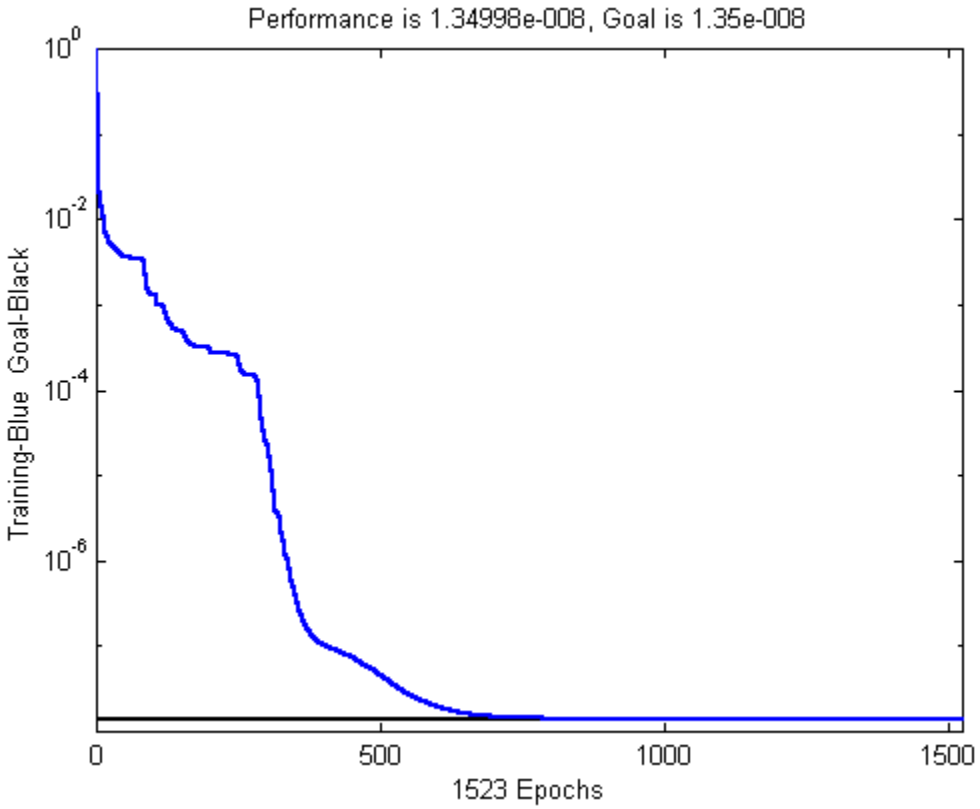

Figura 3.5.4 Error medio cuadrático utilizando *trainbfg*

Luego de 1523 iteraciones el error medio cuadrático cayó por debajo de 1.35×10^{-8}, alcanzado el error deseado, los pesos y ganancias de la red entrenada son:

W1f=net.IW{1,1}

	I1	I2	I3
N1	314.7873	292.0234	160.5730
N2	311.7056	-134.0357	138.9636
N3	-84.6597	-0.4099	-1.4174
N4	26.3003	0.1433	0.5297
N5	109.2728	0.6771	2.6136
N6	-63.4637	-296.9911	231.7073
N7	-75.4292	-98.1638	5.7991
N8	643.5915	286.1720	167.6307
N9	257.6940	33.3274	-21.1659
N10	-220.3097	125.5978	-512.3378

b1f=net.b{1}

N1	492.3094
N2	-159.4016
N3	87.8088
N4	-26.8251
N5	-109.7182
N6	208.1393
N7	-188.8066
N8	66.2581
N9	-215.0041
N10	171.3954

Copyright©2000 Universidad Tecnológica de Pereira

W2f=net.LW{2,1}

	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10
S1	-35.0242	-5.1142	-0.3124	3.4025	0.0932	0.0000	29.3423	-617.2979	-545.3703	0.2596

b2f=net.b{2}

S1	1233.1

Tabla 3.5.11 Pesos finales de la RN entrenada con *trainbfg*

Para hallar la respuesta de la red (Q2Rn) ante los patrones de prueba, se simuló la red y para hallar el error en las aproximaciones se comparó con la respuesta producida por las ecuaciones del flujo de carga (Q2Fc).

	1	2	3	4	5	6	7	8
Q2Fc	0.7992	-0.7359	0.3895	1.7141	0.2457	1.0476	0.3588	0.3497
Q2Rn	0.7992	-0.7356	0.3893	1.7148	0.2459	1.0474	0.3582	0.3492
Error	-0.0000	-0.0002	0.0002	-0.0007	-0.0001	0.0001	0.0007	0.0005

	9	10	11	12	13	14	15	16
Q2Fc	0.6390	0.9463	0.0349	1.1089	-0.1691	-1.1977	1.3368	0.5088
Q2Rn	0.6390	0.9454	0.0351	1.1073	-0.1690	-1.1897	1.3380	0.5089
Error	-0.0000	0.0008	-0.0001	0.0015	-0.0001	-0.0080	-0.0011	-0.0001

Tabla 3.5.12 Simulación de la RN con los patrones de prueba

Como se observa esta red tiene una amplia capacidad de generalización tanto para interpolar como para extrapolar, el error más grande se produjo en el patrón de prueba 15 el cual tiene 3 valores extrapolados con respecto al set de entrenamiento.

Copyright©2000 Universidad Tecnológica de Pereira

- Entrenamiento utilizando el algoritmo *trainlm*

El siguiente código entrena una red neuronal con 3 entradas, 10 neuronas en la capa oculta y 1 en la salida, utilizando la función de aprendizaje *trainlm* y como objetivo un error medio cuadrático de $1x10^{-8}$.

```
net=newff([0.997 1.015;0.7 0.85;0.25 0.4],[10,1],
          {'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);
```

Se utilizó una rata de aprendizaje de 0.03, para este valor se obtuvieron los mejores resultados de error medio cuadrático en el menor número de iteraciones, para valores diferentes a esta rata de aprendizaje se obtuvieron valores superiores en el error medio cuadrático para el mismo número de iteraciones.

No fue necesario fraccionar por submatrices el cálculo de la matriz Jacobiana del algoritmo de entrenamiento, lo cual en algunos casos se hace indispensable por razones computacionales cuando el conjunto de patrones de entrenamiento es muy extenso, pues el Jacobiano de este algoritmo tiene dimensiones QxN donde Q es el número de patrones de entrenamiento y N es el número de pesos y ganancias de la red.

La figura 3.5.5 muestra la evolución del error medio cuadrático a través de 601 iteraciones hasta alcanzar el error deseado.

Copyright©2000 Universidad Tecnológica de Pereira

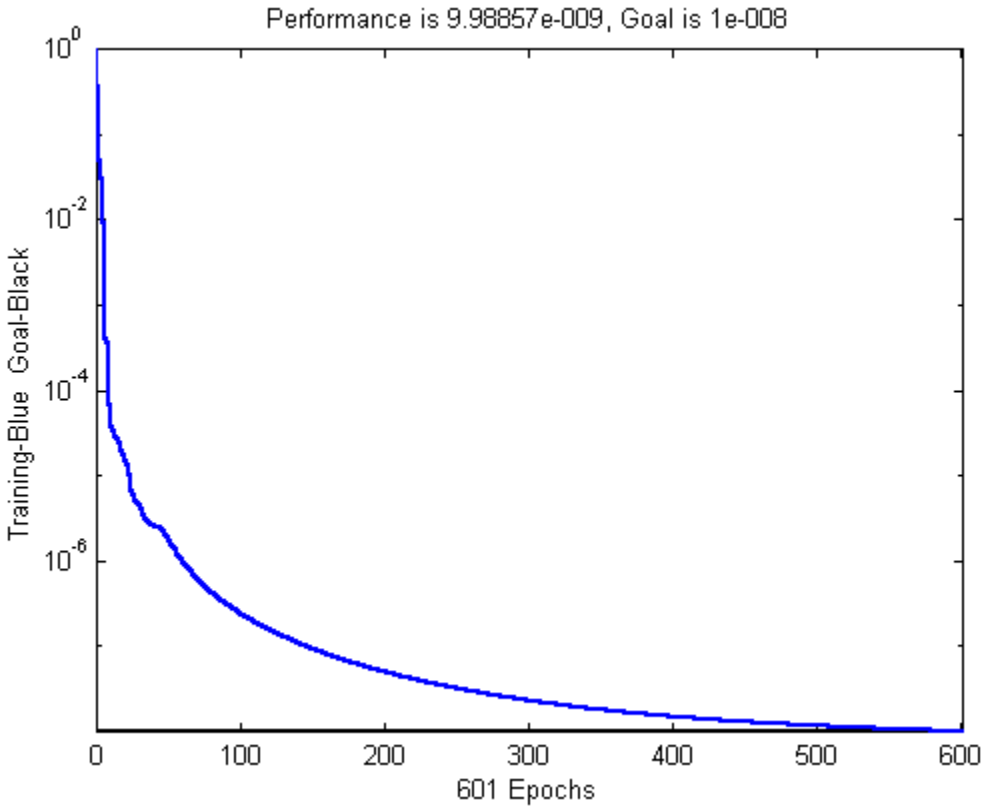

Figura 3.5.5 Error medio cuadrático utilizando *trainlm*

Luego de 601 iteraciones el error medio cuadrático cayó por debajo de $1x10^{-8}$, alcanzado el error deseado, los pesos y ganancias de la red entrenada son:

W1f=net.IW{1,1}

	I1	I2	I3
N1	-88.3684	-9.3759	-10.4875
N2	249.1384	-17.7323	-11.8856
N3	-289.7534	-1.2117	-8.7486
N4	333.5132	-34.3139	1.4535
N5	320.8474	1.9620	9.7026
N6	-143.5382	31.1396	23.1280
N7	13.7085	0.0753	0.2782
N8	298.9814	-14.0175	-39.8990
N9	241.2621	-9.1133	-10.0929
N10	-193.7428	0.4029	2.1040

b1f=net.b{1}

N1	105.0244
N2	-236.6178
N3	296.1395
N4	-312.9180
N5	-324.2801
N6	113.2598
N7	-14.1171
N8	-274.3762
N9	-232.4621
N10	195.5529

Copyright©2000 Universidad Tecnológica de Pereira

W2f=net.LW{2,1}

	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10
S1	-0.3137	-0.0002	0.0016	0.0002	0.0043	0.0001	6.9859	0.0001	0.0007	-0.0068

b2f=net.b{2}

S1	1.7103

Tabla 3.5.13 Pesos y ganancias RN entrenada con *trainlm*

Para hallar la respuesta de la red (Q2Rn) ante los patrones de prueba, se simuló la red y para hallar el error en las aproximaciones se comparó con la respuesta producida por las ecuaciones del flujo de carga (Q2Fc)

	1	2	3	4	5	6	7	8
Q2Fc	0.7992	-0.7359	0.3895	1.7141	0.2457	1.0476	0.3588	0.3497
Q2Rn	0.7992	-0.7361	0.3896	1.7055	0.2459	1.0469	0.3598	0.3497
Error	-0.0001	0.0002	-0.0001	0.0086	-0.0001	0.0007	-0.0010	0.0000

	9	10	11	12	13	14	15	16
Q2Fc	0.6390	0.9463	0.0349	1.1089	-0.1691	-1.1977	1.3368	0.5088
Q2Rn	0.6387	0.9443	0.0350	1.1132	-0.1692	-1.1856	1.3291	0.5087
Error	0.0003	0.0020	-0.0000	-0.0043	0.0001	-0.0120	0.0078	0.0000

Tabla 3.5.14 Simulación de la RN con los patrones de prueba

Como se observa, esta red tiene una amplia capacidad de generalización para interpolar como para extrapolar, los errores más grandes se produjeron en los patrones de prueba 4 y 15 los cuales tienen 2 y 3 valores extrapolados respectivamente con relación al set de entrenamiento; este algoritmo produjo resultados tan buenos como los obtenidos con el algoritmo *trainbfg* pero en menor número de iteraciones y sin problemas de singularidad de la matriz Jacobiana en el proceso de aprendizaje.

Copyright©2000 Universidad Tecnológica de Pereira

3.5.3.2 Problema utilizando como entradas P3, Q3, V3 y P2; para obtener Q2 utilizando 12 neuronas en la capa oculta

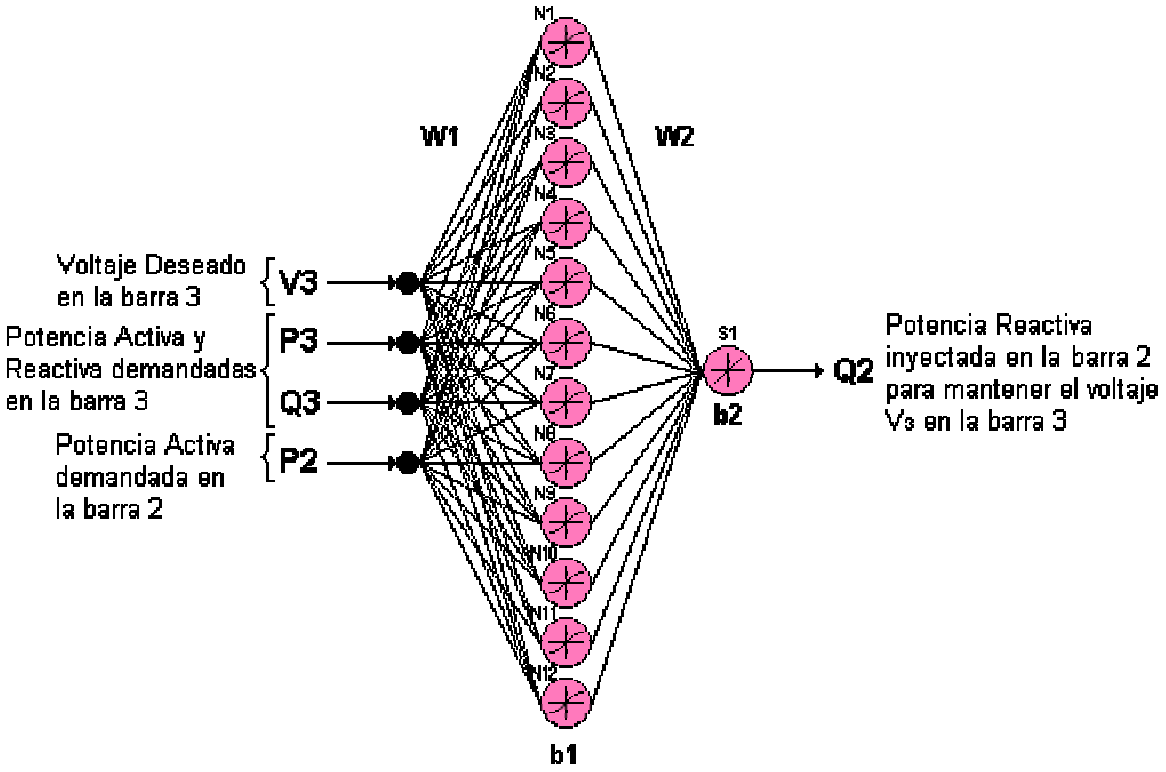

Figura 3.5.6 Red Neuronal en configuración 4:12:1

La red será entrenada para determinar la cantidad de potencia reactiva en p.u. inyectada en la barra 2 (Q_2) para satisfacer las condiciones impuestas por V_3, P_3 Q_3 y P_2, que son el voltaje, la potencia activa y reactiva deseados en la barra 3 y la potencia activa demandada en la barra 2. Para generar el set de entrenamiento se resolvieron las ecuaciones no lineales que describen el comportamiento del sistema por el método de Newton-Raphson considerando variaciones de V_3, P_3, Q_3 , P_2 y sus posibles combinaciones en los rangos mostrados en la tabla 3.5.13, el voltaje en el nodo slack V_1=1.015p.u se conservo constante.

Copyright©2000 Universidad Tecnológica de Pereira

	Valor Inicial	Valor Final	Incremento
V3	0.997	1.015	0.0045
P3	0.700	0.850	0.0500
Q3	0.250	0.400	0.0500
P2	0.600	0.750	0.0500

Tabla 3.5.15 Patrones de entrenamiento

De estas variaciones se obtuvieron 320 patrones de entrenamiento ***p***, que generaron a su vez igual número de respuestas esperadas t

P	p1	p2		p75		p148	p149		p263		p320
V3	0.9970	0.9970	....	1.0015	....	1.0060	1.0060	....	1.0150	....	1.0150
P3	0.8500	0.8500	....	0.8500	....	0.8000	0.8000	....	0.8500	....	0.7000
Q3	0.2500	0.2500		0.3500		0.2500	0.3000		0.3000		0.4000
P2	0.6000	0.6500	....	0.7000	....	0.7500	0.6000	....	0.7000	....	0.7500

t	t1	t2		t75		t148	t149		t263		t320
Q2	-0.7800	-0.7641	....	-0.1617	....	0.0577	0.1008	....	1.0139	....	1.1456

Tabla 3.5.16 Patrones de entrenamiento de la red neuronal

Los pesos iniciales de la red fueron obtenidos de la función *initlay* que calcula los pesos iniciales con base en el rango de los datos de entrada

W1i=net.IW{1,1}

	I1	I2	I3	I4
N1	214.9557	18.4104	8.3064	11.5648
N2	85.1487	-25.6540	-20.9690	2.1943
N3	-84.5134	12.2602	12.9213	28.0521
N4	-203.3228	18.7339	16.1259	-0.8630
N5	77.5282	-22.7461	-7.6400	-23.3397
N6	130.9888	27.2670	-5.8177	13.5142
N7	-68.2392	4.5660	10.2405	31.8477
N8	-207.9645	6.7104	17.1889	15.6131
N9	-63.1775	-25.1240	-12.1456	19.2571
N10	272.6824	6.2926	-9.6419	-1.9396
N11	129.7327	22.8632	5.0148	-20.4160
N12	250.8019	-11.8106	4.7112	11.8144

b1f=net.b{1}

N1	-243.6249
N2	-62.5758
N3	54.0424
N4	186.5500
N5	-42.8385
N6	-160.3749
N7	40.0478
N8	187.1758
N9	72.7921
N10	-273.0942
N11	-133.9472
N12	-250.0536

Copyright©2000 Universidad Tecnológica de Pereira

W2i=net.LW{2,1}

	N1	N2	N3	N4	N5	N6
S1	-0.2693	-0.7199	0.1335	0.6460	0.3479	0.9989

	N7	N8	N9	N10	N11	N12
S1	0.9233	-0.8823	-0.2794	0.0970	-0.4765	0.1947

b2f=net.b{2}

S1	-0.9014

Tabla 3.5.17 Pesos iniciales para la red neuronal

Se entrenará la red de la figura 3.5.6 con dos distintos algoritmos de aprendizaje y luego se simulará cada red con los siguientes patrones de prueba:

	1	2	3	4	5	6	7	8
V3	1.0120	**0.9950**	1.0100	**1.0200**	1.0070	**1.0160**	1.0050	1.0100
P3	0.7320	0.8300	0.8250	0.8370	0.7520	**0.9120**	**0.8720**	0.8500
Q3	0.3710	0.3600	**0.2300**	**0.4100**	0.3230	0.2500	**0.4500**	**0.2020**
P2	**0.5300**	**0.5500**	0.6200	0.6100	**0.7580**	0.7420	0.6320	**0.5840**
Q2	0.7416	-0.7836	0.3626	1.6821	0.2653	1.0623	0.3359	0.3108

	9	10	11	12	13	14	15	16
V3	1.0100	**1.0170**	1.0050	1.0120	1.0030	**0.9950**	**1.0200**	1.0100
P3	0.8430	**0.9000**	0.7500	0.8500	0.8200	**0.6500**	**0.9000**	0.8000
Q3	0.3570	**0.1500**	0.3100	**0.5000**	0.2800	**0.1500**	**0.2000**	0.3000
P2	0.6500	**0.7800**	0.6200	**0.5800**	**0.5500**	0.6000	0.6800	0.7000
Q2	0.6220	0.9743	0.0085	1.0673	-0.2180	-1.2288	1.3298	0.5088

Tabla 3.5.18 Patrones de prueba de las RN's

Los valores de la tabla 3.5.18 que se encuentran en negrilla se encuentran por fuera del rango de entrenamiento y con ellos se busca probar la capacidad de extrapolación de la red, la mayoría de los datos de prueba que se encuentran dentro del rango de entrenamiento no coinciden exactamente con el valor utilizado en el entrenamiento para probar la capacidad de interpolación de la red.

Copyright©2000 Universidad Tecnológica de Pereira

- Entrenamiento utilizando el algoritmo *trainbfg*

El siguiente código entrena una red neuronal con 4 entradas, 12 neuronas en la capa oculta y 1 en la salida, utilizando la función de aprendizaje *trainbfg* (Ver Anexo 1) y como objetivo un error medio cuadrático de $2.15x10^{-8}$.

```
net=newff([0.997 1.015;0.7 0.85;0.25 0.4;0.6 0.75],[12,1],
          {'tansig','purelin'},'trainbfg');
net.trainParam.epochs=1500;
net.trainParam.goal=5e-8;
net.trainParam.min_grad=1e-10;
net.trainParam.searchFcn='srchcha'
net.trainParam.scal_tol=20;
net.trainParam.alpha=0.001;
net.trainParam.beta=0.1;
net.trainParam.delta=0.01;
net.trainParam.gama=0.1;
net.trainParam.low_lim=0.1;
net.trainParam.up_lim=0.5;
[net,tr]=train(net,P,t);
```

En el entrenamiento de la red con este algoritmo se presentaron de nuevo inconvenientes en la invertibilidad de la matriz Jacobiana del algoritmo en el proceso iterativo, por lo cual el error medio cuadrático objetivo fue fijado en $5x10^{-8}$, por debajo de este valor la matriz Jacobiana del algoritmo se vuelve singular y presenta los inconvenientes antes descritos, pero antes de que esto suceda los resultados obtenidos proveen una muy buena aproximación.

La figura 3.5.7 muestra la evolución del error medio cuadrático a través de 1066 iteraciones hasta alcanzar el error deseado.

Copyright©2000 Universidad Tecnológica de Pereira

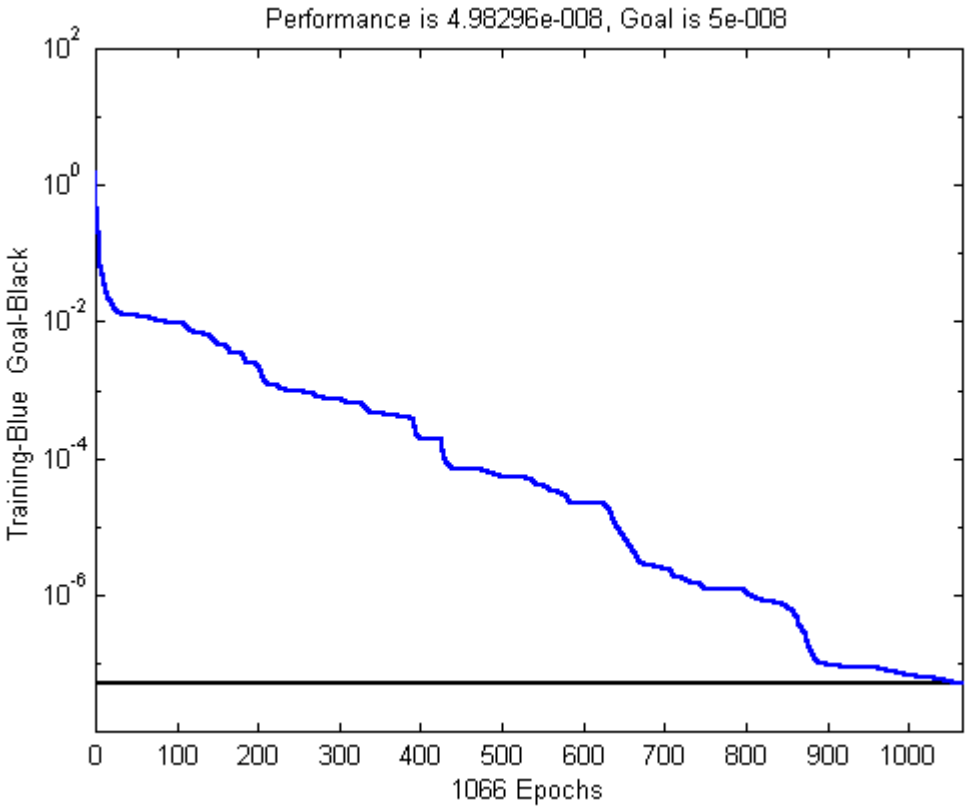

Figura 3.5.7 Error medio cuadrático utilizando *trainbfg*

Luego de 1066 iteraciones el error medio cuadrático cayó por debajo de $5x10^{-8}$, alcanzado el error deseado, los pesos y ganancias de la red entrenada son:

W1f=net.IW{1,1}

	I1	I2	I3	I4
N1	224.4737	1.1518	4.1945	0.7659
N2	102.8281	19.5357	-69.0280	-18.4298
N3	132.9784	173.1409	67.2342	171.2176
N4	-783.1072	-316.8951	-154.8182	-441.7653
N5	-151.4010	-190.8937	-109.9698	-159.8299
N6	425.0669	259.6459	85.3641	183.3998
N7	361.8879	328.3253	140.2923	276.1773
N8	47.0275	204.3169	40.6780	175.2289
N9	-45.6292	-0.2511	-0.9316	-0.1655
N10	268.0715	1.2620	4.5158	0.8389
N11	82.6771	4.2783	13.3543	-42.3772
N12	243.0169	1.3006	4.7068	0.8643

b1f=net.b{1}

N1	-231.1824
N2	-48.5460
N3	272.5634
N4	-385.5031
N5	-271.9678
N6	132.7761
N7	464.9202
N8	445.8823
N9	46.1722
N10	-272.7277
N11	-177.9131
N12	-248.8931

Copyright©2000 Universidad Tecnológica de Pereira

W2f=net.LW{2,1}

	N1	N2	N3	N4	N5	N6
S1	0.2130	10.4271	10.5727	-2.6097	27.1693	-906.5415

	N7	N8	N9	N10	N11	N12
S1	794.0887	-14.8412	-1.9511	0.0293	-0.4926	0.0775

b2f=net.b{2}

S1	130.1609

Tabla 3.5.19 Pesos y ganancias RN entrenada con *trainbfg*

Para hallar la respuesta de la red (Q2Rn) ante los patrones de prueba, se simuló la red y para hallar el error en las aproximaciones se comparó con la respuesta producida por las ecuaciones del flujo de carga (Q2Fc)

	1	2	3	4	5	6	7	8
Q2Fc	0.7416	-0.7836	0.3626	1.6821	0.2653	1.0623	0.3359	0.3108
Q2Rn	0.7412	-0.7841	0.3629	1.5963	0.2658	1.0627	0.3350	0.3112
Error	0.0004	0.0005	-0.0003	0.0858	-0.0005	-0.0004	0.0010	-0.0003

	9	10	11	12	13	14	15	16
Q2Fc	0.6220	0.9743	0.0085	1.0673	-0.2180	-1.2288	1.3298	0.5088
Q2Rn	0.6221	0.9742	0.0083	1.0634	-0.2184	-1.2087	1.3310	0.5088
Error	-0.0002	0.0001	0.0003	0.0040	0.0004	-0.0201	-0.0012	-0.0001

Tabla 3.5.20 Simulación de la RN con los patrones de prueba.

Como se observa, esta red tiene una amplia capacidad de generalización para interpolar como para extrapolar, los errores más grandes se produjeron en los patrones de prueba 4 y 14 los cuales tienen 2 y 3 valores extrapolados respectivamente en relación al set de entrenamiento.

Copyright©2000 Universidad Tecnológica de Pereira

- Entrenamiento utilizando el algoritmo *trainlm*

El siguiente código entrena una red neuronal con 4 entradas, 12 neuronas en la capa oculta y 1 en la salida, utilizando la función de aprendizaje *trainbfg* y como objetivo un error medio cuadrático de $1x10^{-8}$.

```
net=newff([0.997 1.015;0.7 0.85;0.25 0.4;0.6 0.75],[12,1],
          {'tansig','purelin'},'trainlm');
net.trainParam.epochs=700;
net.trainParam.goal=1e-8;
net.trainParam.lr=0.03;
net.trainParam.max_fail=5;
net.trainParam.mem_reduc=1;
net.trainParam.min_grad=1e-10;
[net,tr]=train(net,P,t);
```

Después de realizar varias pruebas, se decidió utilizar una rata de aprendizaje de 0.03, para este valor se obtuvieron los mejores resultados en el entrenamiento de la red, llegando a un valor muy pequeño en el error medio cuadrático a través de pocas iteraciones.

Como en el caso anterior de entrenamiento con este algoritmo, no fue necesario fraccionar por submatrices el cálculo de la matriz Jacobiana, pues los patrones de entrenamiento y el número de parámetros de la red son estrictamente los necesarios

La figura 3.5.8 muestra la evolución del error medio cuadrático a través de 362 iteraciones hasta alcanzar el error deseado.

Copyright©2000 Universidad Tecnológica de Pereira

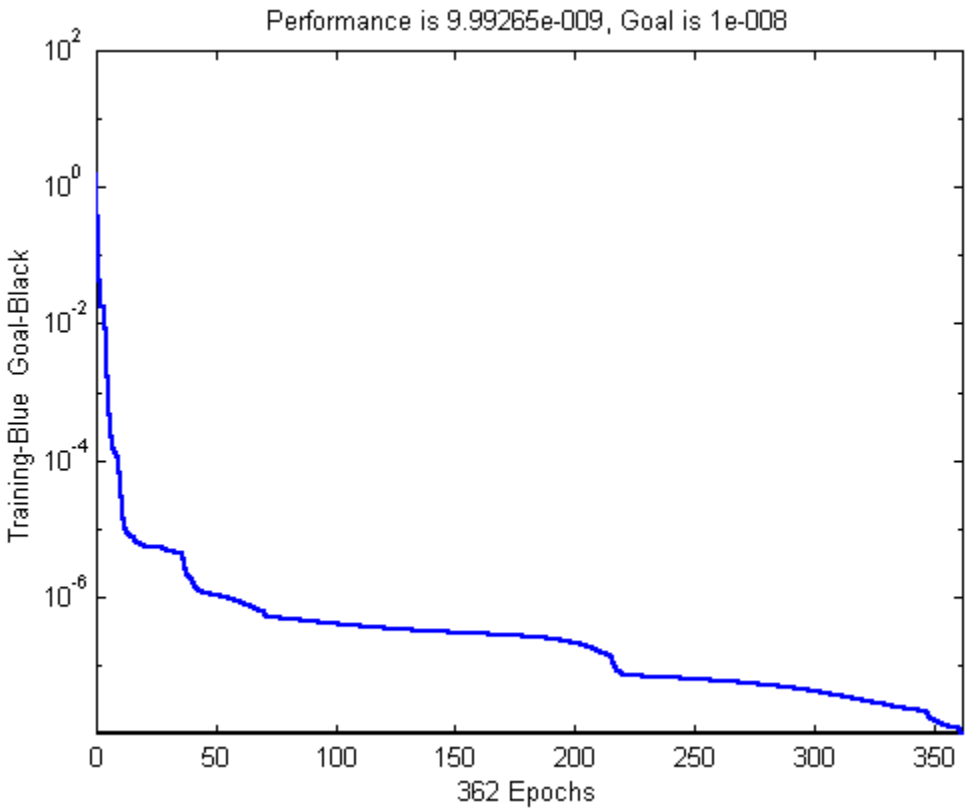

Figura 3.5.8 Error medio cuadrático utilizando *trainlm*

Luego de 362 iteraciones el error medio cuadrático cayó por debajo de $1x10^{-8}$, alcanzado el error deseado, los pesos y ganancias de la red entrenada son:

W1f=net.IW{1,1}

	I1	I2	I3	I4
N1	217.3050	1.1784	4.3958	0.7495
N2	86.3682	-28.0087	-14.8760	2.4348
N3	-82.5289	9.5925	36.1293	22.6072
N4	-196.4316	15.3102	-1.1168	-1.8863
N5	73.2020	-19.7579	-4.5863	-25.2722
N6	144.0374	0.7851	2.9423	0.5168
N7	-66.5832	7.3605	5.8988	24.5765
N8	-191.3809	-1.0280	-3.8772	-0.6760
N9	-70.7928	-0.3863	-1.4178	-0.2546
N10	263.8718	9.4256	8.6166	1.3268
N11	126.0300	11.5735	16.9349	2.5421
N12	246.1641	-2.9151	-7.5082	-1.6426

b1f=net.b{1}

N1	-221.8890
N2	-60.9291
N3	50.6335
N4	187.3245
N5	-49.0199
N6	-145.1983
N7	41.1724
N8	194.3911
N9	72.9238
N10	-275.1674
N11	-139.0503
N12	-244.2148

Copyright©2000 Universidad Tecnológica de Pereira

W2f=net.LW{2,1}

	N1	N2	N3	N4	N5	N6
S1	0.0595	-0.0001	0.0001	-0.0001	-0.4521	0.4525

	N7	N8	N9	N10	N11	N12
S1	0.0003	-0.1357	-1.3382	0.0004	0.0024	0.0023

b2f=net.b{2}

S1	0.1047

Tabla 3.5.21 Pesos y ganancias RN entrenada con *trainlm*

Para hallar la respuesta de la red (Q2Rn) ante los patrones de prueba, se simuló la red y para hallar el error en las aproximaciones se comparó con la respuesta producida por las ecuaciones del flujo de carga (Q2Fc).

	1	2	3	4	5	6	7	8
Q2Fc	0.7416	-0.7836	0.3626	1.6821	0.2653	1.0623	0.3359	0.3108
Q2Rn	0.7406	-0.7842	0.3624	1.6512	0.2652	1.0621	0.3357	0.3105
Error	0.0010	0.0007	0.0002	0.0308	0.0001	0.0002	0.0002	0.0004

	9	10	11	12	13	14	15	16
Q2Fc	0.6220	0.9743	0.0085	1.0673	-0.2180	-1.2288	1.3298	0.5088
Q2Rn	0.6222	0.9730	0.0089	1.0680	-0.2182	-1.1664	1.3250	0.5089
Error	-0.0002	0.0013	-0.0004	-0.0006	0.0002	-0.0624	0.0048	-0.0002

Tabla 3.5.22 Simulación de la RN con los patrones de prueba

De la tabla 3.5.22 se observa que esta red tiene una amplia capacidad de generalización para interpolar como para extrapolar, los errores más grandes se produjeron en los patrones de prueba 4 (2), 14 (3) y 15 (3) donde los valores entre paréntesis son el número de datos extrapolados en relación con el set de entrenamiento; este algoritmo produjo resultados tan buenos como los obtenidos

Copyright©2000 Universidad Tecnológica de Pereira

con el algoritmo *trainbfg* pero en menor número de iteraciones y sin problemas de singularidad de la matriz Jacobiana en el proceso de aprendizaje.

Para entrenar una red neuronal se debe invertir cierta cantidad de tiempo en la elección del tipo de red, conjunto de patrones de entrenamiento, topología de la red, proceso de entrenamiento; la inversión de esta cantidad de tiempo se ve recompensada cuando la red ha sido entrenada, pues la velocidad de la respuesta de la red es muy grande y la confiabilidad en las respuestas es muy alta.

Los sistemas de potencia poseen diferentes tópicos en los cuales se dificulta la solución de las complejas ecuaciones que los describen, una red neuronal entrenada correctamente con abundante y representativa información puede controlar las condiciones de operación de un sistema de potencia, con la precisión y acierto de un operador experto.

Utilizando redes neuronales se podrían entrenar dispositivos inteligentes para lograr despacho económico en mercado abierto y obtener flujos óptimos en el sistema de potencia sin necesidad de recurrir a resolver el sistema de ecuaciones no lineales que describen este de difícil solución.

Copyright©2000 Universidad Tecnológica de Pereira