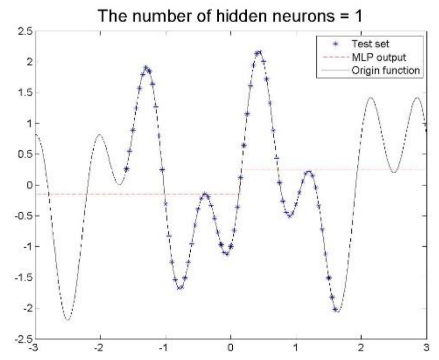
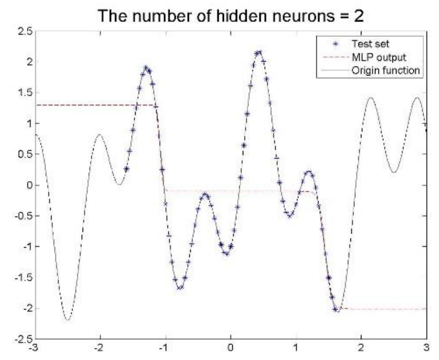
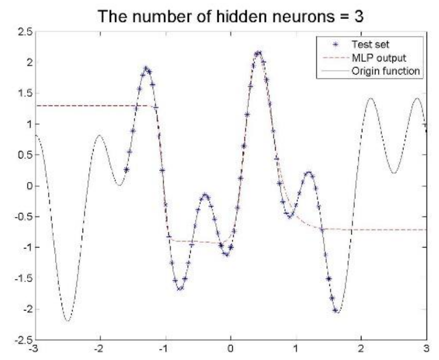
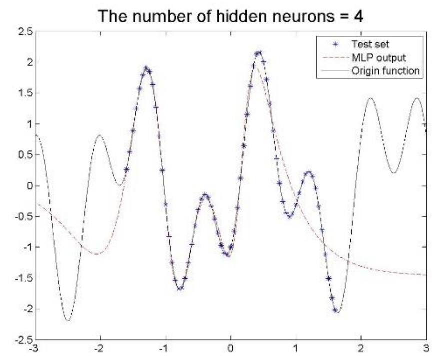
Q2.

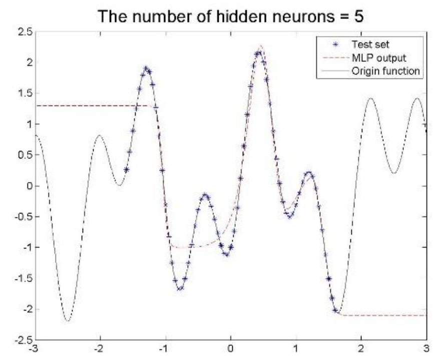
a) Use the sequential mode with BP algorithm and experiment with the following different structures of the MLP: 1-n-1 (where n = 1, 2… 10, 20, 50, 100).

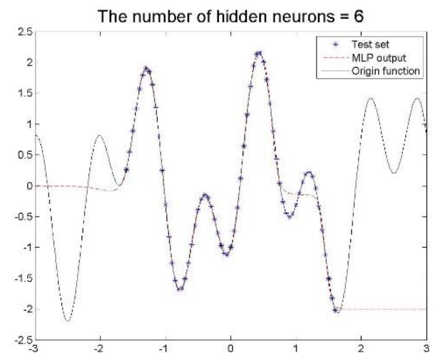


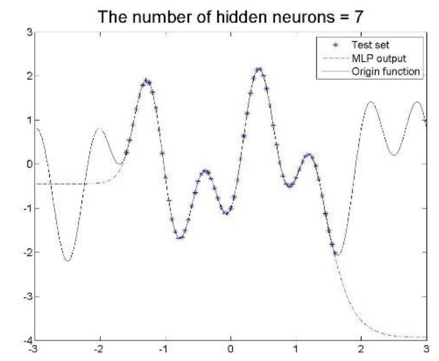


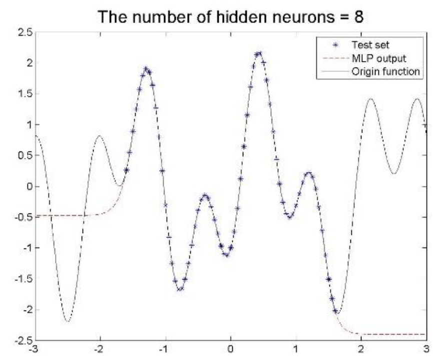


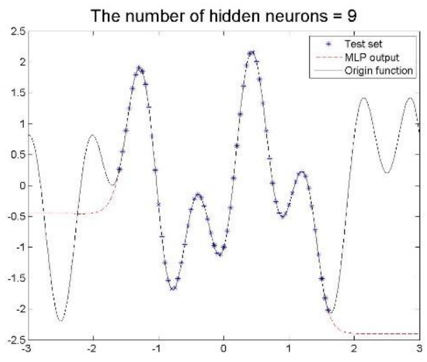


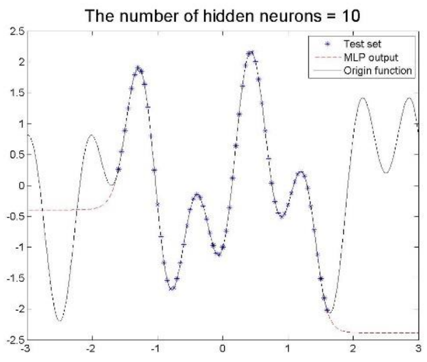


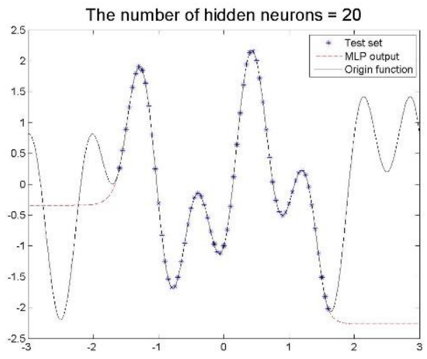


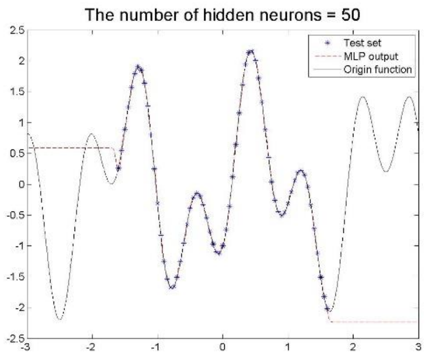












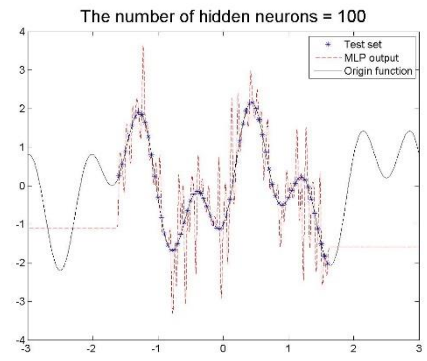


Table 1: the fitting state of different perceptions of BP algorithm

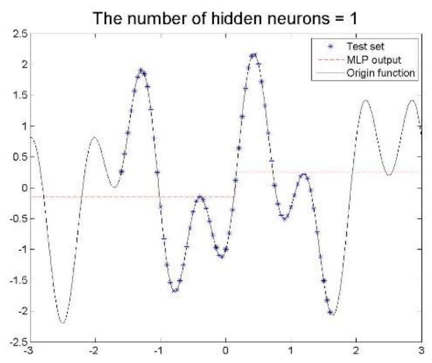
N	1	2	3	4	5	6	7	8	9	10	20	50	100
Fitting state	UF	UF	UF	UF	UF	UF	UF	PF	UF	UF	PF	PF	OF

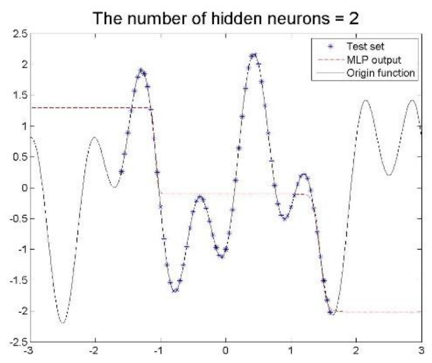
Description of table: Under-fitting: UF Proper fitting: PF Over-fitting: OF

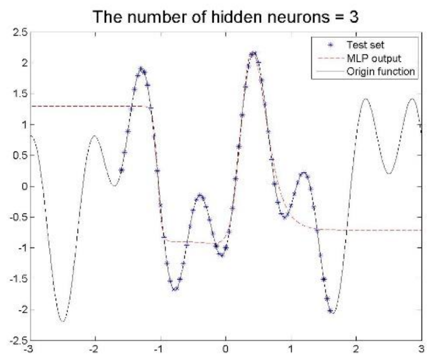
Depending on this table, we could find that the minimal number of hidden neurons for BP algorithm is 8. From the guideline given in the lecture slides, the minimal number of hidden neurons for BP is 8. The result is consistent with the guideline given in the lecture slides.

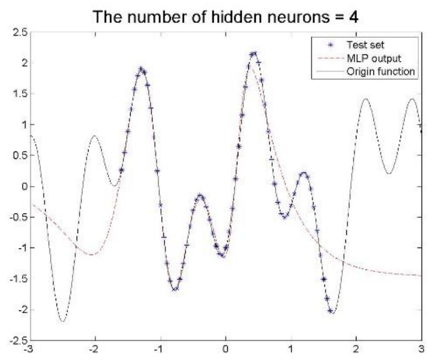
For all MLP of BP algorithm, they couldn't make reasonable predictions outside of the domain of the input limited by the training set.

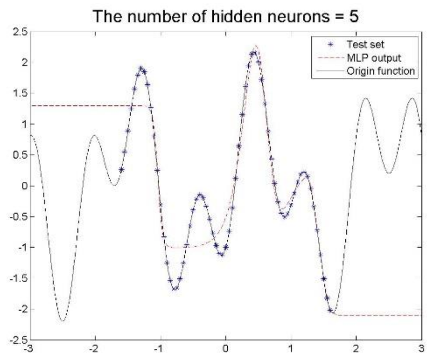
b) Use the batch mode with trainlm algorithm and experiment with the following different structures of the MLP: 1-n-1 (where n = 1, 2… 10, 20, 50, 100).

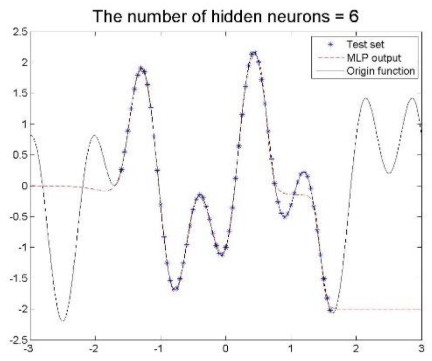


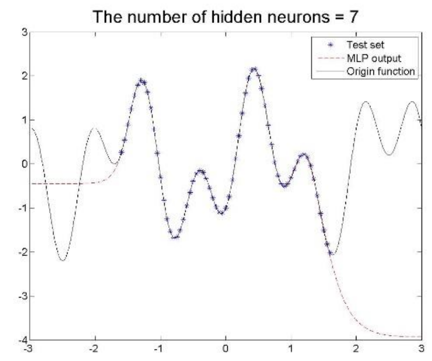


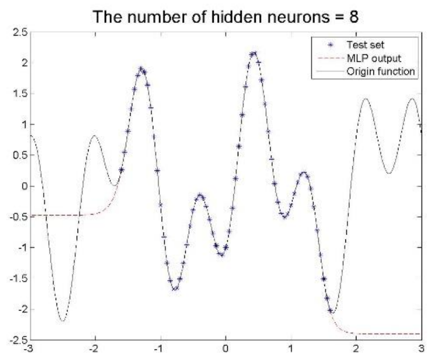


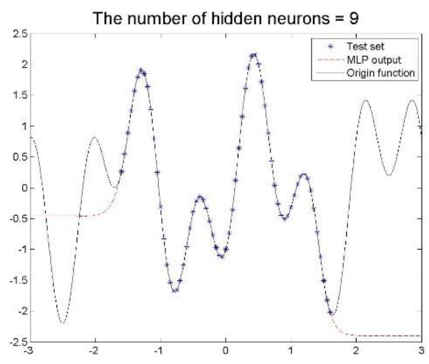


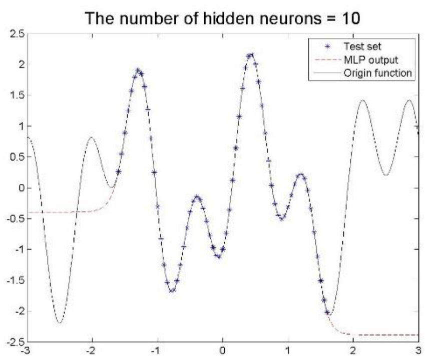


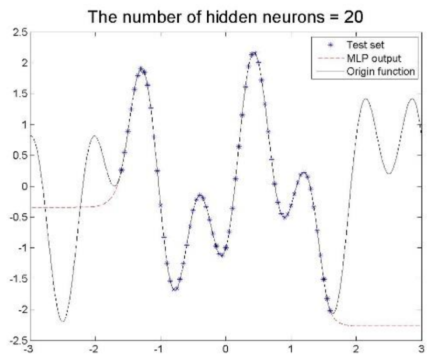


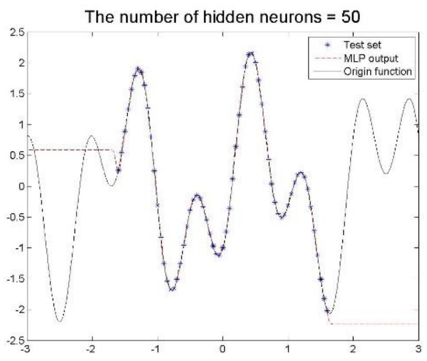












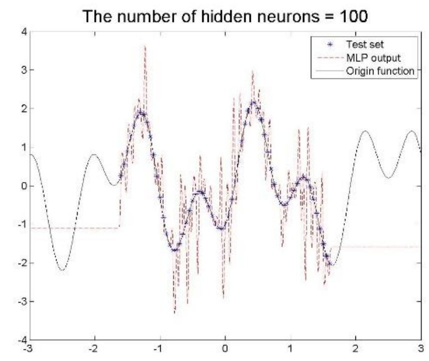


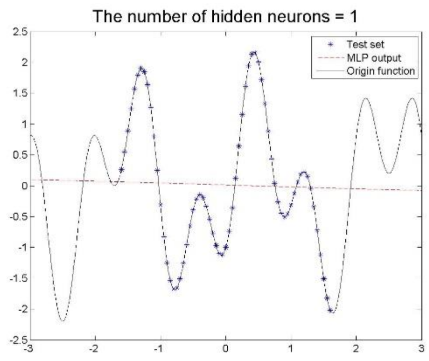
Table 2: the fitting state of different perceptions of trainlm algorithm

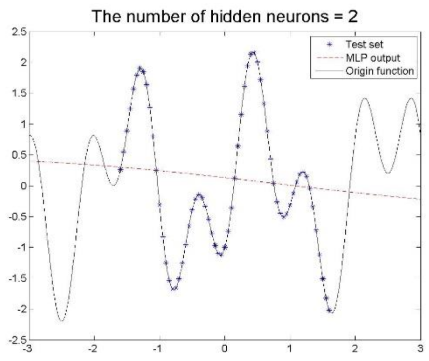
N	1	2	3	4	5	6	7	8	9	10	20	50	100
Fitting state	UF	UF	UF	UF	UF	UF	PF	PF	PF	PF	PF	PF	OF

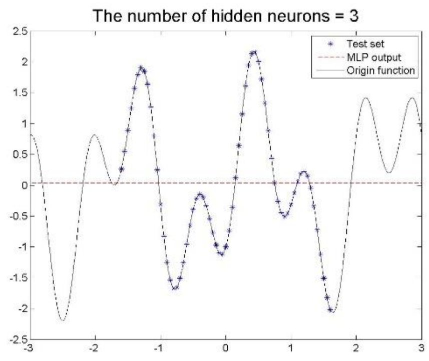
Description of table: Under-fitting: UF Proper fitting: PF Over-fitting: OF

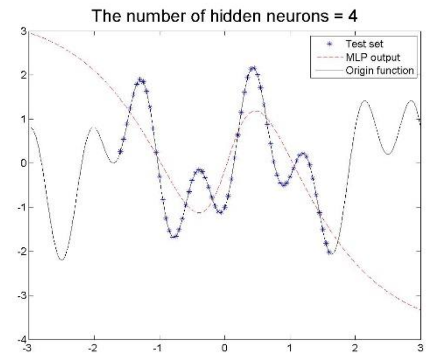
Depending on this table, we could find that the minimal number of hidden neurons for BP algorithm is 7. The result is consistent with the guideline given in the lecture slides. For all MLP of BP algorithm, they couldn’t make reasonable predictions outside of the domain of the input limited by the training set.

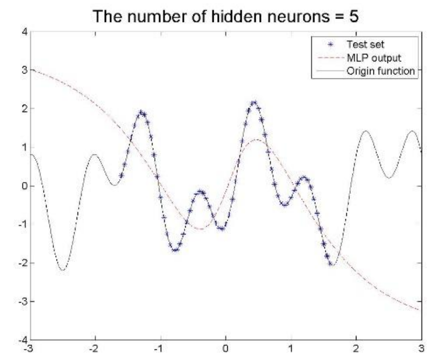
c) Use the batch mode with trainbr algorithm and experiment with the following different structures of the MLP: 1-n-1 (where n = 1, 2… 10, 20, 50, 100).

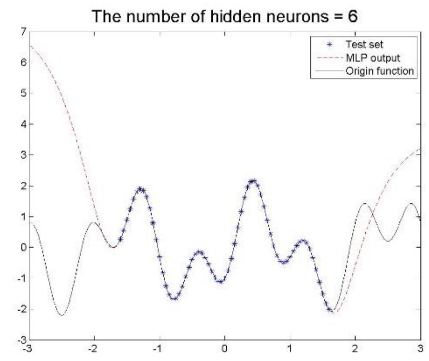


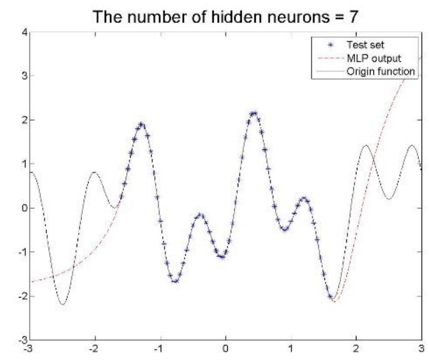


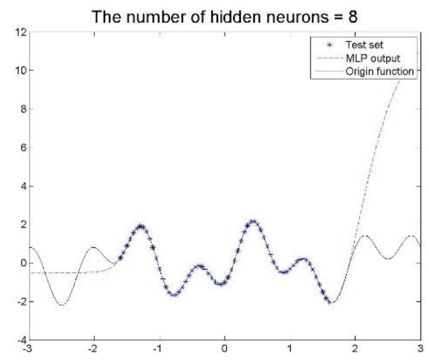


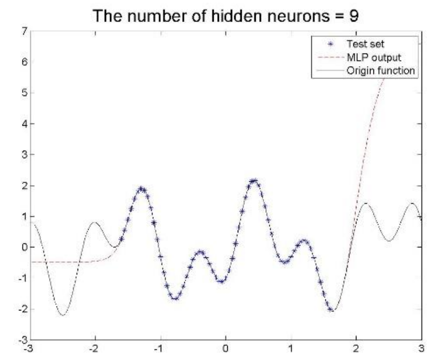


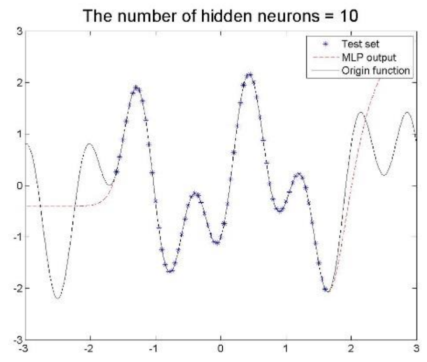


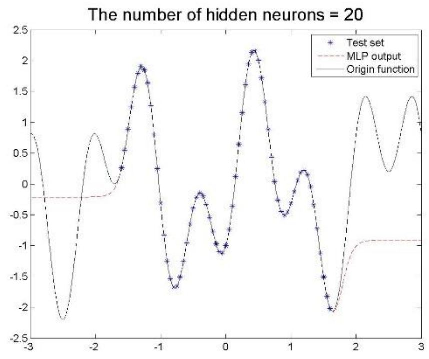


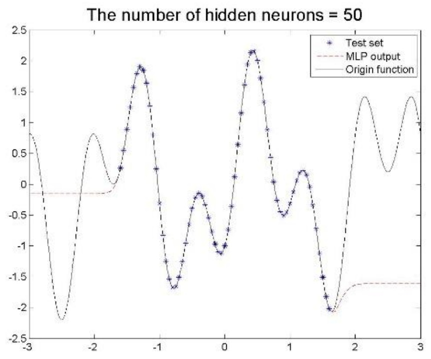












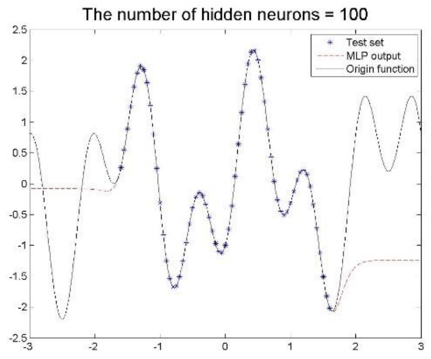


Table 3: the fitting state of different perceptions of trainbr algorithm

N	1	2	3	4	5	6	7	8	9	10	20	50	100
Fitting state	UF	UF	UF	UF	UF	PF	PF	PF	PF	PF	PF	PF	PF

Description of table: Under-fitting: UF Proper fitting: PF Over-fitting: OF

Depending on this table, we could find that the minimal number of hidden neurons for BP algorithm is 6. The result is not consistent with the guideline given in the lecture slides.

For all MLP of BP algorithm, they couldn't make reasonable predictions outside of the domain of the input limited by the training set.

The following is the MATLAB code of BP algorithm:

```
%%CACULATION
close all;
clear all;
k=100;
x = -1.6:0.05:1.6;
y = 1.2*sin(pi*x) - cos(2.4*pi*x);
net = feedforwardnet(k,'traingd');
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'purelin';
net = configure(net,x,y);
net.trainparam.lr=0.01;
net.trainparam.epochs=10000;
net.trainparam.goal=1e-8;
net.divideParam.trainRatio=1.0;
net.divideParam.valRatio=0.0;
net.divideParam.testRatio=0.0;
[net,tr]=adapt(net,x,y);
for i = 1 : 10000
    index = randperm(65);
    net = adapt(net,x(:,index),y(index));
end

xtest = -3:0.01:3;
ytest = 1.2*sin(pi*xtest) - cos(2.4*pi*xtest);
net_output = sim(net,xtest);

%%OUTPUT
x_design_output=-3:0.01:3;
y_design_output=1.2*sin(pi*x_design_output) -
cos(2.4*pi*x_design_output);

plot(x,y,'b*');
hold on;
plot(xtest,net_output,'r--');
plot(x_design_output,y_design_output,'k-');
legend('Test set','MLP output','Origin function')
hold off
discription1 = sprintf('The number of hidden neurons = %d',k);
title(discription1,'FontSize',20);
set(gca,'FontSize',12);
discription2 = sprintf('Q2_1_%d.jpg',k);
saveas(gcf,discription2)
```

The following is the MATLAB code of trainlm algorithm:

```
%%CACULATION
close all;
clear all;
i=100;
x = -1.6:0.05:1.6;
y = 1.2*sin(pi*x) - cos(2.4*pi*x);
net = feedforwardnet(i,'trainlm');
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'purelin';
net = configure(net,x,y);
net.trainparam.lr=0.01;
net.trainparam.epochs=1000;
net.trainparam.goal=1e-8;
net.divideParam.trainRatio=1.0;
net.divideParam.valRatio=0.0;
net.divideParam.testRatio=0.0;
[net,tr] = train(net,x,y);

xtest = -3:0.01:3;
ytest = 1.2*sin(pi*xtest) - cos(2.4*pi*xtest);
net_output = sim(net,xtest);

%OUTPUT
x_design_output=-3:0.01:3;
y_design_output=1.2*sin(pi*x_design_output) -
cos(2.4*pi*x_design_output);

plot(x,y,'b*');
hold on;
plot(xtest,net_output,'r--');
plot(x_design_output,y_design_output,'k-');
legend('Test set','MLP output','Origin function')
hold off
discription1 = sprintf('The number of hidden neurons = %d',i);
title(discription1,'FontSize',20);
set(gca,'FontSize',12);
discription2 = sprintf('Q2_2_%d.jpg',i);
saveas(gcf,discription2);
```

The following is the MATLAB code of trainbr algorithm:

```
%%CACULATION
close all;
clear all;
i=100;
x = -1.6:0.05:1.6;
y = 1.2*sin(pi*x) - cos(2.4*pi*x);
net = feedforwardnet(i,'trainbr');
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'purelin';
net = configure(net,x,y);
net.trainparam.lr=0.01;
net.trainparam.epochs=1000;
net.trainparam.goal=1e-8;
net.divideParam.trainRatio=1.0;
net.divideParam.valRatio=0.0;
net.divideParam.testRatio=0.0;
[net,tr] = train(net,x,y);

xtest = -3:0.01:3;
ytest = 1.2*sin(pi*xtest) - cos(2.4*pi*xtest);
net_output = sim(net,xtest);

%OUTPUT
x_design_output=-3:0.01:3;
y_design_output=1.2*sin(pi*x_design_output) -
cos(2.4*pi*x_design_output);

plot(x,y,'b*');
hold on;
plot(xtest,net_output,'r--');
plot(x_design_output,y_design_output,'k-');
legend('Test set','MLP output','Origin function')
hold off
discription1 = sprintf('The number of hidden neurons = %d',i);
title(discription1,'FontSize',20);
set(gca,'FontSize',12);
discription2 = sprintf('Q2_3_%d.jpg',i);
saveas(gcf,discription2);
```