ANN

```
clear all; close all; clc; warning off;

u=2*ones(151,1);
y=zeros(151,1);
% Assuming initial condition 'zero'
for k=2:length(u)
    if k<51
        u(k)=2*exp(-0.02*pi*(k-1));
    else
        u(k)=10*exp(-0.01*pi*(k-1))*sin(0.2*pi*(k-1));
    end
    y(k)= (y(k-1)/(1+(y(k-1))^2))+(u(k-1)^3);
end
```

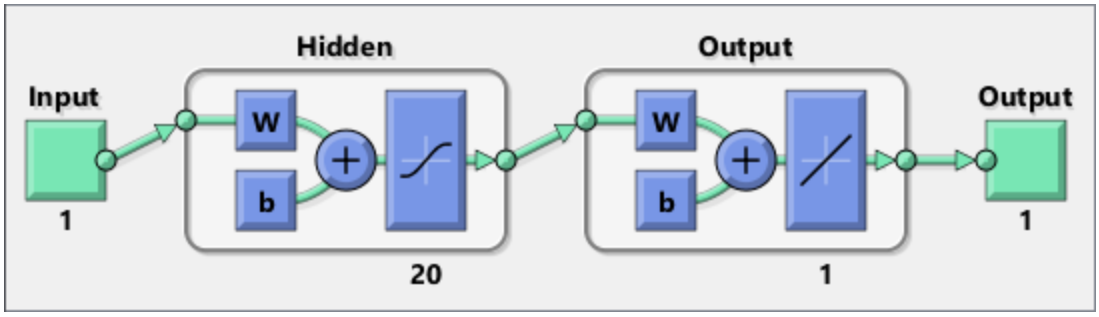
ARTIFICIAL NEURAL NETWORK

LM

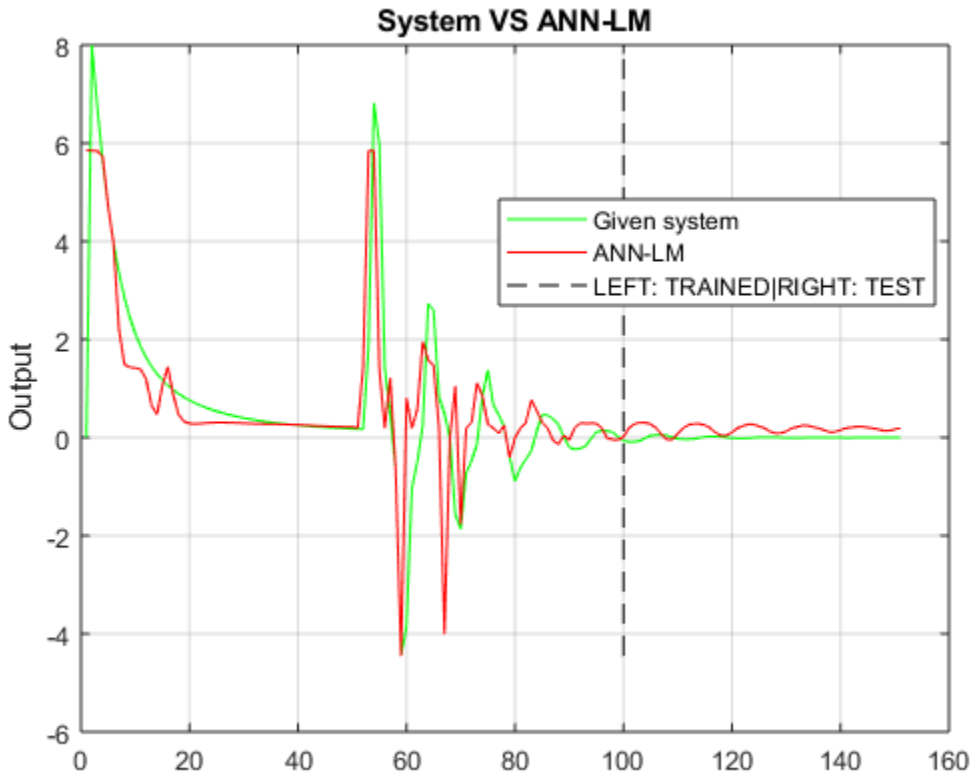
```
u1=u(1:100);
y1=y(1:100);

net = fitnet(20,'trainlm')
net = train(net,u1',y1');
view(net)
ylm = net(u');
```

```
net =

    Neural Network

              name: 'Function Fitting Neural Network'
          userdata: (your custom info)

    dimensions:

         numInputs: 1
         numLayers: 2
        numOutputs: 1
    numInputDelays: 0
    numLayerDelays: 0
 numFeedbackDelays: 0
 numWeightElements: 20
        sampleTime: 1

    connections:

       biasConnect: [1; 1]
      inputConnect: [1; 0]
      layerConnect: [0 0; 1 0]
     outputConnect: [0 1]
```

```
subobjects:

         input: Equivalent to inputs{1}
        output: Equivalent to outputs{2}

        inputs: {1x1 cell array of 1 input}
        layers: {2x1 cell array of 2 layers}
       outputs: {1x2 cell array of 1 output}
        biases: {2x1 cell array of 2 biases}
  inputWeights: {2x1 cell array of 1 weight}
  layerWeights: {2x2 cell array of 1 weight}

functions:

      adaptFcn: 'adaptwb'
    adaptParam: (none)
      derivFcn: 'defaultderiv'
     divideFcn: 'dividerand'
   divideParam: .trainRatio, .valRatio, .testRatio
    divideMode: 'sample'
       initFcn: 'initlay'
    performFcn: 'mse'
  performParam: .regularization, .normalization
      plotFcns: {'plotperform', plottrainstate, ploterrhist,
                plotregression, plotfit}
    plotParams: {1x5 cell array of 5 params}
      trainFcn: 'trainlm'
    trainParam: .showWindow, .showCommandLine, .show, .epochs,
                .time, .goal, .min_grad, .max_fail, .mu, .mu_dec,
                .mu_inc, .mu_max

weight and bias values:

            IW: {2x1 cell} containing 1 input weight matrix
            LW: {2x2 cell} containing 1 layer weight matrix
             b: {2x1 cell} containing 2 bias vectors

methods:

         adapt: Learn while in continuous use
     configure: Configure inputs & outputs
        gensim: Generate Simulink model
          init: Initialize weights & biases
       perform: Calculate performance
           sim: Evaluate network outputs given inputs
         train: Train network with examples
          view: View diagram
   unconfigure: Unconfigure inputs & outputs
```



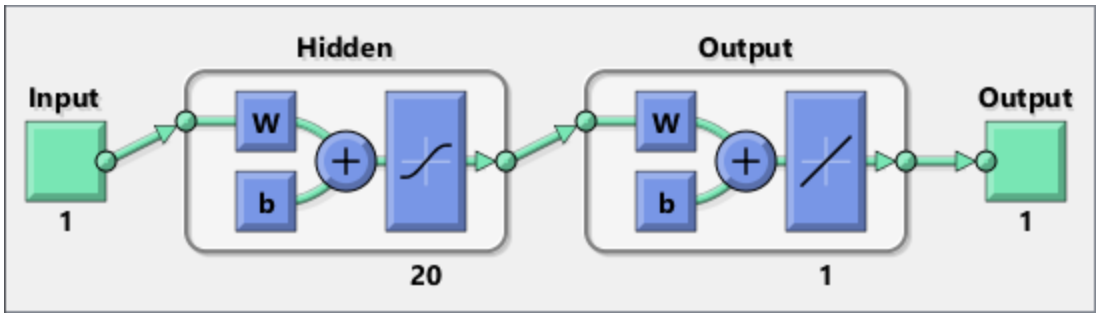

```
figure(1)
plot(y,'g-');
hold on;
plot(ylm,'r-');
hold on;
plot([100,100],[min(y),max(y)],'k--');
legend('Given system','ANN-LM','LEFT: TRAINED|RIGHT:
 TEST','location','best');
grid on;
ylabel('Output');
title('System VS ANN-LM');
```



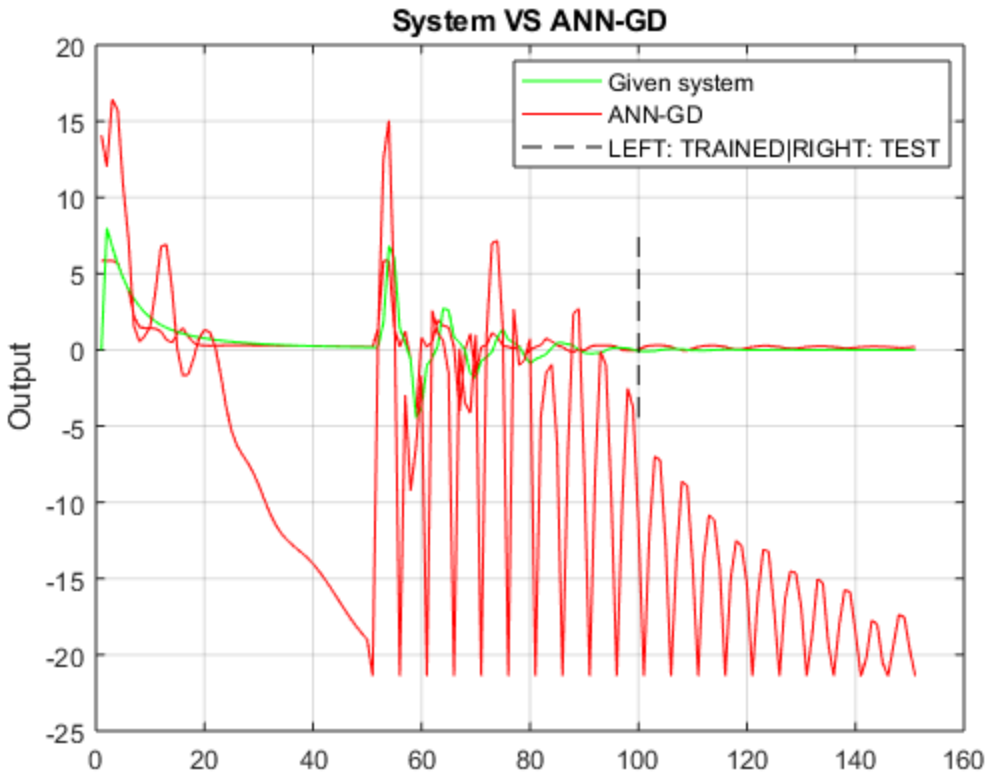

```
netG = fitnet(20,'traingd')
netG = train(netG,u1',y1');
view(netG)
ygd = netG(u');
```

```
netG =

    Neural Network

              name: 'Function Fitting Neural Network'
          userdata: (your custom info)

    dimensions:

         numInputs: 1
         numLayers: 2
        numOutputs: 1
    numInputDelays: 0
    numLayerDelays: 0
 numFeedbackDelays: 0
 numWeightElements: 20
        sampleTime: 1

    connections:

       biasConnect: [1; 1]
      inputConnect: [1; 0]
      layerConnect: [0 0; 1 0]
     outputConnect: [0 1]

    subobjects:

             input: Equivalent to inputs{1}
            output: Equivalent to outputs{2}

            inputs: {1x1 cell array of 1 input}
            layers: {2x1 cell array of 2 layers}
           outputs: {1x2 cell array of 1 output}
            biases: {2x1 cell array of 2 biases}
      inputWeights: {2x1 cell array of 1 weight}
      layerWeights: {2x2 cell array of 1 weight}

    functions:

          adaptFcn: 'adaptwb'
        adaptParam: (none)
          derivFcn: 'defaultderiv'
         divideFcn: 'dividerand'
       divideParam: .trainRatio, .valRatio, .testRatio
        divideMode: 'sample'
           initFcn: 'initlay'
        performFcn: 'mse'
      performParam: .regularization, .normalization
          plotFcns: {'plotperform', plottrainstate, ploterrhist,
                    plotregression, plotfit}
        plotParams: {1x5 cell array of 5 params}
          trainFcn: 'traingd'
        trainParam: .showWindow, .showCommandLine, .show, .epochs,
```

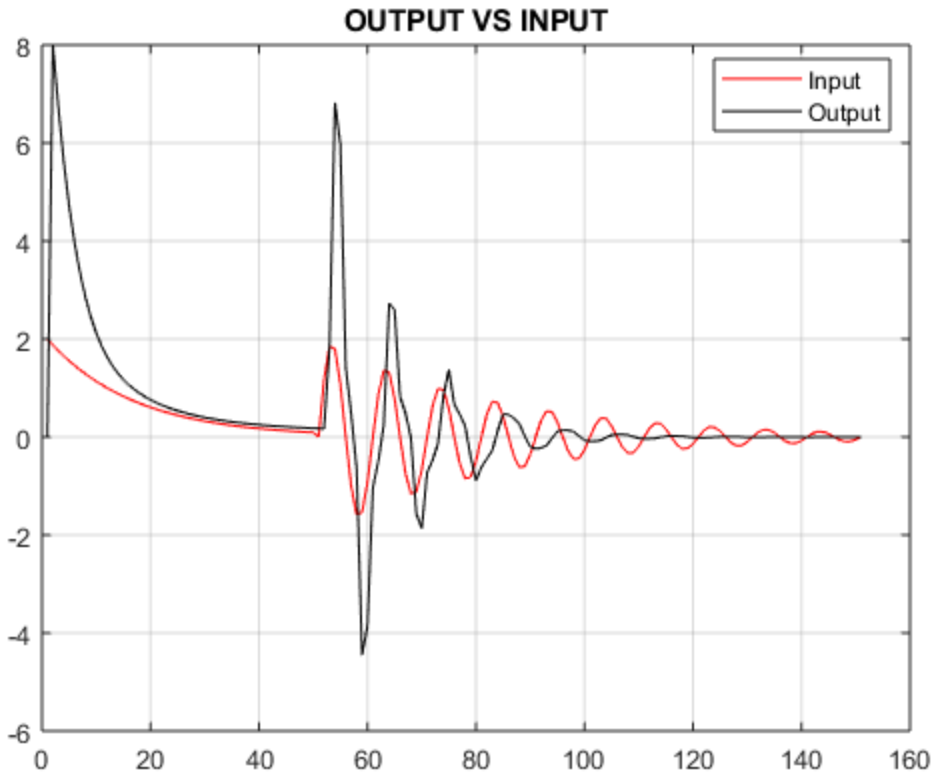
```
                .time, .goal, .min_grad, .max_fail, .lr

    weight and bias values:

                IW: {2x1 cell} containing 1 input weight matrix
                LW: {2x2 cell} containing 1 layer weight matrix
                 b: {2x1 cell} containing 2 bias vectors

    methods:

             adapt: Learn while in continuous use
         configure: Configure inputs & outputs
            gensim: Generate Simulink model
              init: Initialize weights & biases
           perform: Calculate performance
               sim: Evaluate network outputs given inputs
             train: Train network with examples
              view: View diagram
       unconfigure: Unconfigure inputs & outputs
```




```
figure(2)
plot(y,'g-');
hold on;
plot(ygd,'r-');
hold on;
plot([100,100],[min(y),max(y)],'k--');
legend('Given system','ANN-GD','LEFT: TRAINED|RIGHT:
 TEST','location','best');
grid on;
ylabel('Output');
title('System VS ANN-GD');
```




```
figure(3)
plot(u,'r');
hold on;
plot(y,'k');
title('OUTPUT VS INPUT');
legend('Input', 'Output','location','best');
grid on;
```



Published with MATLAB® R2018a