ملحق (3)

برامج المحاكاة للتجارب الخمسة

في هذا الجزء من التحليل تم التركيز على الاوزان المصاحبة لكل تجربة من التجارب الخمسة اضافة الى البرنامج

أ. برنامج التجربة الاولى

```
function [Y,Xf,Af] = myNeuralNetworkFunction(X,~,~)
%MYNEURALNETWORKFUNCTION neural network simulation function.
% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:
%   X = 1xTS cell, 1 inputs over TS timesteps
%   Each X{1,ts} = 151xQ matrix, input #1 at timestep ts.
% and returns:
%   Y = 1xTS cell of 1 outputs over TS timesteps.
%   Each Y{1,ts} = 8xQ matrix, output #1 at timestep ts.
%
% where Q is number of samples (or series) and TS is the number of
timesteps.
%#ok<*RPMT0>
% ===== NEURAL NETWORK CONSTANTS =====
% Input 1
x1_step1.keep = [8 16 24 32 40 48 56 64 72 80 88 96 104 112 113 114 115
116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133
134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150
151];
x1_step2.xoffset =
[2323;627;258;3000;276;7263;137;2000;599;191;2898;271;4908;111;19.966;3
.928;3.988;3.928;3.928;3.928;3.604;185400;2.372;4.212;4.365;4.212;4.212
;4.212;3.89;10330;4.68;2.949;2.949;2.949;2.949;2.949;44.104;95888;0.174
;5.813;5.813;5.913;5.813;5.813;91.231;12000220;17250;20000;100000;30000
;20000;40000;-207900];
x1_step2.gain =
[0.000563063063063063;0.00119402985074627;0.0090702947845805;0.00131233
595800525;0.00333333333333333;0.000363570259952736;0.00888888888888889;
0.00632911392405063;0.00275103163686382;0.021505376344086;0.00399201596
806387;0.00641025641025641;0.000451977401129943;0.0540540540540541;0.28
3527076835838;3.4965034965035;2.19298245614035;3.4965034965035;3.496503
4965035;3.4965034965035;3.47222222222222;1.29366106080207e-
05;0.0948316737790422;1.07066381156317;1.16618075801749;1.0706638115631
7;1.07066381156317;1.07066381156317;0.91324200913242;8.63793947136674e-
06;1.13636363636364;1.14876507754164;1.14876507754164;1.14876507754164;
1.14876507754164;1.14876507754164;0.871080139372823;1.91916477948797e-
05;0.562429696287964;1.77462289263531;1.77462289263531;1.94741966893866
;1.77462289263531;1.77462289263531;0.565131393048884;5.5555740741358e-
08;1.6359918200409e-05;0.000307692307692308;3.14218381775334e-
05;0.000266666666666667;0.00036036036036036;0.000238095238095238;1.9627
0853778214e-05];
x1_step2.ymin = -1;
```

```
% Layer 1
b1 = [-1.4401041177616795697;-
1.4054852760196354655;1.4008691564980970679;-
1.3886480201200768203;1.3533239526344489345;1.1746348423648471648;1.235
6394785029840566;-1.1497297221221500774;-0.92383650434087793979;-
1.0369815468178547047;-1.0204313260437767319;-
1.0040678491143282258;0.91239530300008586661;0.87796140798956134876;0.7
2532604840851244 71;-0.61603918088760190042;0.71558091857681827008;-
0.70609688346976973161;0.56855596148344611507;0.31724168357655790906;0.
60987049794570735717;0.45844235352562812125;-0.62082171942810093768;-
0.23589127241180379846;-0.42960147645311719256;-
0.21554654877409124003;-0.36428904859213390166;-
0.03674619620816216292 2;0.088319374866639946409;-
0.18261764425240539778;0.083275777857965554984;-
0.25258594289013103218;-0.39407465071322844707;-
0.26090513876956983319;-0.58892636757685035143;0.42406337596567522663;-
0.39447855379631402428;-0.44433327256759741486;-
0.58103935998792555218;-0.55553252377347350777;0.67759647412704782088;-
0.85237636422365870015;-0.98890112230085991385;0.83040015401951228213;-
0.82908794784387573884;-0.94217644989709059544;-
0.95584093853963780774;1.039787996743813947;-1.1575466005286469962;-
1.258614672089181763;1.2092517183762525601;-
1.2307100893230371597;1.4372131123924780649;1.418247480389263071;-
1.4082018774384967852;1.6631526055494672001];
IW1_1 = [];
% Layer 2
b2 = [0.88584187757115584461;-0.40830596053365175724;-
0.66685225460493358085;0.36475450489800714804;-0.59510991793675249717;-
0.70610773435665052933;-0.28908710253069852536;-
0.21752946009812304529];
LW2_1 = [];
% Output 1
y1_step1.ymin = -1;
y1_step1.gain =
[0.000705965407695023;0.000868809730668983;0.0064059447166971;0.0004520
79566003617;0.00342465753424658;0.00016002560409 6655;0.0078431372549019
6;1.30712726171413e-09];
y1_step1.xoffset = [2264;0;75.79;0;0;0;0;66866100];
 % ===== SIMULATION ========
 % Format Input Arguments
isCellX = iscell(X);
if ~isCellX
    X = {X};
end
 % Dimensions
TS = size(X,2); % timesteps
if ~isempty(X)
    Q = size(X{1},2); % samples/series
else
    Q = 0;
end

% Allocate Outputs
Y = cell(1,TS);

% Time loop
```

```
for ts=1:TS

    % Input 1
    temp = removeconstantrows_apply(X{1,ts},x1_step1);
    Xp1 = mapminmax_apply(temp,x1_step2);

    % Layer 1
    a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*Xp1);

    % Layer 2
    a2 = repmat(b2,1,Q) + LW2_1*a1;

    % Output 1
    Y{1,ts} = mapminmax_reverse(a2,y1_step1);
end

% Final Delay States
Xf = cell(1,0);
Af = cell(2,0);

% Format Output Arguments
if ~isCellX
    Y = cell2mat(Y);
end
end

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);
y = bsxfun(@plus,y,settings.ymin);
end

% Remove Constants Input Processing Function
function y = removeconstantrows_apply(x,settings)
y = x(settings.keep,:);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings)
x = bsxfun(@minus,y,settings.ymin);
x = bsxfun(@rdivide,x,settings.gain);
x = bsxfun(@plus,x,settings.xoffset);
end
  ================================================================

% Solve an Input-Output Fitting problem with a Neural Network
```

```
% Script generated by Neural Fitting app
% Created 16-Apr-2020 19:34:25
%
% This script assumes these variables are defined:
%
%   input - input data.
%   output - target data.

x = input;
t = output;

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. Suitable in low memory situations.
trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 56;
net = fitnet(hiddenLayerSize,trainFcn);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.input.processFcns = {'removeconstantrows','mapminmax'};
net.output.processFcns = {'removeconstantrows','mapminmax'};

% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivision
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean Squared Error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
    'plotregression', 'plotfit'};

% Train the Network
[net,tr] = train(net,x,t);

% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)

% Recalculate Training, Validation and Test Performance
trainTargets = t .* tr.trainMask{1};
valTargets = t .* tr.valMask{1};
```

```
testTargets = t .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,y)
valPerformance = perform(net,valTargets,y)
testPerformance = perform(net,testTargets,y)

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, ploterrhist(e)
%figure, plotregression(t,y)
%figure, plotfit(net,x,t)

% Deployment
% Change the (false) values to (true) to enable the following code
blocks.
% See the help for each generation function for more information.
if (false)
    % Generate MATLAB function for neural network for application
    % deployment in MATLAB scripts or with MATLAB Compiler and Builder
    % tools, or simply to examine the calculations your trained neural
    % network performs.
    genFunction(net,'myNeuralNetworkFunction');
    y = myNeuralNetworkFunction(x);
end
if (false)
    % Generate a matrix-only MATLAB function for neural network code
    % generation with MATLAB Coder tools.
    genFunction(net,'myNeuralNetworkFunction','MatrixOnly','yes');
    y = myNeuralNetworkFunction(x);
end
if (false)
    % Generate a Simulink diagram for simulation or deployment with.
    % Simulink Coder tools.
    gensim(net);
end
```

==========

ب. برنامج التجربة الثانية

```
function [Y,Xf,Af] = myNeuralNetworkFunction(X,~,~)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Auto-generated by MATLAB, 16-Apr-2020 19:57:47.
%
% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:
%
%   X = 1xTS cell, 1 inputs over TS timesteps
%   Each X{1,ts} = 151xQ matrix, input #1 at timestep ts.
%
% and returns:
%   Y = 1xTS cell of 1 outputs over TS timesteps.
```

```
%   Each Y{1,ts} = 8xQ matrix, output #1 at timestep ts.
%
% where Q is number of samples (or series) and TS is the number of
timesteps.

%#ok<*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
x1_step1.keep = [8 16 24 32 40 48 56 64 72 80 88 96 104 112 113 114 115
116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133
134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150
151];
x1_step2.xoffset =
[2323;627;258;3000;276;7263;137;2000;599;191;2898;271;4908;111;19.966;3
.928;3.988;3.928;3.928;3.928;3.604;185400;2.372;4.212;4.365;4.212;4.212
;4.212;3.89;10330;4.68;2.949;2.949;2.949;2.949;2.949;44.104;95888;0.174
;5.813;5.813;5.913;5.813;5.813;91.231;12000220;17250;20000;100000;30000
;20000;40000;-207900];
x1_step2.gain =
[0.000563063063063063;0.00119402985074627;0.0090702947845805;0.00131233
595800525;0.00333333333333333;0.000363570259952736;0.00888888888888889;
0.00632911392405063;0.00275103163686382;0.021505376344086;0.00399201596
806387;0.00641025641025641;0.000451977401129943;0.0540540540540541;0.28
3527076835838;3.4965034965035;2.19298245614035;3.4965034965035;3.496503
4965035;3.4965034965035;3.47222222222222;1.29366106080207e-
05;0.0948316737790422;1.07066381156317;1.16618075801749;1.0706638115631
7;1.07066381156317;1.07066381156317;0.91324200913242;8.63793947136674e-
06;1.13636363636364;1.14876507754164;1.14876507754164;1.14876507754164;
1.14876507754164;1.14876507754164;0.871080139372823;1.91916477948797e-
05;0.562429696287964;1.77462289263531;1.77462289263531;1.94741966893866
;1.77462289263531;1.77462289263531;0.565131393048884;5.5555740741358e-
08;1.6359918200409e-05;0.000307692307692308;3.14218381775334e-
05;0.000266666666666667;0.00036036036036036;0.000238095238095238;1.9627
0853778214e-05];
x1_step2.ymin = -1;

% Layer 1
b1 = [-1.5104721265132210473;-
1.4555458673672856396;1.4006196082213504539;-1.3456933490754150462;-
1.2907670899294796385;1.2358408307835444528;1.1809145716376092672;1.125
9883124916738595;1.0710620533457386738;1.0161357941998032661;0.96120953
505386796945;0.90628327590793256174;0.85135701676199748711;-
0.79643075761606207941;-0.74150449847012678273;-
0.68657823932419148605;0.63165198017825607835;-0.57672572103232078167;-
0.52179946188638548499;-0.46687320274045013280;-0.41194694359451478061;-
0.35702068444857948393;0.30209442530264413174;-0.24716816615670897384;-
0.19224190701077364940;-0.13731564786483832497;0.08238938871890300053;-
0.02746312957296766568;0.02746312957296749568;-
0.08238938871890282011;0.13731564786483815843;0.19224190701077348287;-
0.24716816615670880730;-0.30209442530264413174;-
0.35702068444857948393;0.41194694359451478061;0.46687320274045013280;-
0.52179946188638548499;0.57672572103232078167;0.63165198017825607835;-
0.68657823932419148605;0.74150449847012678273;0.79643075761606207941;0.
85135701676199748711;-0.90628327590793278379;-
```

```
0.96120953505386808047;1.0161357941998030441;1.0710620533457384518;-
1.1259883124916736374;-1.1809145716376090451;-1.2358408307835444528;-
1.2907670899294796385;-1.3456933490754150462;1.4006196082213504539;-
1.4555458673672856396;1.5104721265132210473];
IW1_1 = [];

% Layer 2
b2 = [-
0.56974783896274772133;0.29034564599069612179;0.21227833246321248239;-
0.63530975017609359057;0.73538291248885068185;0.055436209165867333937;0
.94907525931186142998;-0.42903036553899909222];
LW2_1 = [];

% Output 1
y1_step1.ymin = -1;
y1_step1.gain =
[0.000705965407695023;0.000868809730668983;0.0064059447166971;0.0004520
79566003617;0.00342465753424658;0.000160025604096655;0.0078431372549019
6;1.30712726171413e-09];
y1_step1.xoffset = [2264;0;75.79;0;0;0;0;66866100];

% ===== SIMULATION ========

% Format Input Arguments
isCellX = iscell(X);
if ~isCellX
    X = {X};
end

% Dimensions
TS = size(X,2); % timesteps
if ~isempty(X)
    Q = size(X{1},2); % samples/series
else
    Q = 0;
end

% Allocate Outputs
Y = cell(1,TS);

% Time loop
for ts=1:TS

    % Input 1
    temp = removeconstantrows_apply(X{1,ts},x1_step1);
    Xp1 = mapminmax_apply(temp,x1_step2);

    % Layer 1
    a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*Xp1);

    % Layer 2
    a2 = repmat(b2,1,Q) + LW2_1*a1;

    % Output 1
    Y{1,ts} = mapminmax_reverse(a2,y1_step1);
```

```
end

% Final Delay States
Xf = cell(1,0);
Af = cell(2,0);

% Format Output Arguments
if ~isCellX
    Y = cell2mat(Y);
end
end

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);
y = bsxfun(@plus,y,settings.ymin);
end

% Remove Constants Input Processing Function
function y = removeconstantrows_apply(x,settings)
y = x(settings.keep,:);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings)
x = bsxfun(@minus,y,settings.ymin);
x = bsxfun(@rdivide,x,settings.gain);
x = bsxfun(@plus,x,settings.xoffset);
end
```

==========

```
% Solve an Input-Output Fitting problem with a Neural Network
% Script generated by Neural Fitting app
% Created 16-Apr-2020 19:58:37
%
% This script assumes these variables are defined:
%
%   input - input data.
%   output - target data.

x = input;
t = output;

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
```

```
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. Suitable in low memory situations.
trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 56;
net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
[net,tr] = train(net,x,t);

% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, ploterrhist(e)
%figure, plotregression(t,y)
%figure, plotfit(net,x,t)
```

==========

ج. برنامج التجربة الثالثة

```
function [Y,Xf,Af] = myNeuralNetworkFunction(X,~,~)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Auto-generated by MATLAB, 16-Apr-2020 20:23:07.
%
% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:
%
%   X = 1xTS cell, 1 inputs over TS timesteps
%   Each X{1,ts} = 151xQ matrix, input #1 at timestep ts.
%
% and returns:
%   Y = 1xTS cell of 1 outputs over TS timesteps.
%   Each Y{1,ts} = 8xQ matrix, output #1 at timestep ts.
%
% where Q is number of samples (or series) and TS is the number of
timesteps.

%#ok<*RPMT0>
```

```
% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
x1_step1.keep = [8 16 24 32 40 48 56 64 72 80 88 96 104 112 113 114 115
116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133
134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150
151];
x1_step2.xoffset =
[2323;627;258;3000;276;7263;137;2000;599;191;2898;271;4908;111;19.966;3
.928;3.988;3.928;3.928;3.928;3.604;185400;2.372;4.212;4.365;4.212;4.212
;4.212;3.89;10330;4.68;2.949;2.949;2.949;2.949;2.949;44.104;95888;0.174
;5.813;5.813;5.913;5.813;5.813;91.231;12000220;17250;20000;100000;30000
;20000;40000;-207900];
x1_step2.gain =
[0.000563063063063063;0.00119402985074627;0.0090702947845805;0.00131233
595800525;0.00333333333333333;0.000363570259952736;0.00888888888888889;
0.00632911392405063;0.00275103163686382;0.021505376344086;0.00399201596
806387;0.00641025641025641;0.000451977401129943;0.054054054054541;0.28
3527076835838;3.4965034965035;2.19298245614035;3.4965034965035;3.496503
4965035;3.4965034965035;3.47222222222222;1.29366106080207e-
05;0.0948316737790422;1.07066381156317;1.16618075801749;1.0706638115631
7;1.07066381156317;1.07066381156317;0.91324200913242;8.63793947136674e-
06;1.13636363636364;1.14876507754164;1.14876507754164;1.14876507754164;
1.14876507754164;1.14876507754164;0.871080139372823;1.91916477948797e-
05;0.562429696287964;1.77462289263531;1.77462289263531;1.94741966893866
;1.77462289263531;1.77462289263531;0.565131393048884;5.5555740741358e-
08;1.6359918200409e-05;0.000307692307692308;3.14218381775334e-
05;0.000266666666666667;0.00036036036036036;0.000238095238095238;1.9627
0853778214e-05];
x1_step2.ymin = -1;

% Layer 1
b1 = [-1.4799766172060757619;-
1.4821205597089706085;1.3906894647696006473;-
1.3467544700215015041;1.4249662236870357379;-
1.1653695625633899713;1.0981634212991964183;-1.1406184985725131842;-
1.085546037608197345;-1.0685391239026658106;-0.84025511354471882353;-
1.0179763705601478918;-0.94531135967230450756;0.82807791453669876702;-
0.73276169813296709865;0.79836941391203386509;0.67963035421615947129;-
0.46113041103271762422;-0.53166361288681907027;0.46287197781486488779;-
0.44204186078260249149;-
0.39911672569147432776;0.267595719131583476;0.25299388149309870455;0.12
106772744402957698;-0.14290821106353912562;-0.16956431283556633316;-
0.064064960988495442407;0.012546370347313266086;0.090167259491120907433
;0.12429297134754599408;0.36967218913914817735;0.54208612478627149489;0
.28119188199703848419;0.52880913111684979544;-0.51311719019698931454;-
0.55494581410990140302;-0.37864262365848494474;0.55903948104780043504;-
0.60698756057071978631;-0.62511515655922345047;-0.727867790068121856;-
0.76836109517011497161;-1.0119176697531075959;-
0.85029094071050392856;0.98965119031071524525;1.0233082611221799851;1.0
44789203145327 8605;1.0457102670717126625;-
1.1318367428412232645;1.1076426811952149176;1.2557011591652758042;-
1.4098760241257752845;-1.4667529165740831321;-1.4601097322430718339;-
1.4640642619001742464];
IW1_1 = [];
```

```
% Layer 2
b2 =
[0.023914228214870710681;0.097732599921039178814;0.52886849188400064659
;-
0.85070379132807660838;0.78647765077878384332;0.53661869998251321423;0.
28411322754421591830.47237612612956741964];
LW2_1 = [];

% Output 1
y1_step1.ymin = -1;
y1_step1.gain =
[0.00070596540769502300.00086880973066898300.006405944716697100.000452
079566003617;0.00342465753424658;0.000160025604096655;0.0078431372549019
6;1.30712726171413e-09];
y1_step1.xoffset = [2264;0;75.79;0;0;0;0;66866100];

% ===== SIMULATION ========

% Format Input Arguments
isCellX = iscell(X);
if ~isCellX
    X = {X};
end

% Dimensions
TS = size(X,2); % timesteps
if ~isempty(X)
    Q = size(X{1},2); % samples/series
else
    Q = 0;
end

% Allocate Outputs
Y = cell(1,TS);

% Time loop
for ts=1:TS

    % Input 1
    temp = removeconstantrows_apply(X{1,ts},x1_step1);
    Xp1 = mapminmax_apply(temp,x1_step2);

    % Layer 1
    a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*Xp1);

    % Layer 2
    a2 = repmat(b2,1,Q) + LW2_1*a1;

    % Output 1
    Y{1,ts} = mapminmax_reverse(a2,y1_step1);
end

% Final Delay States
```

```
Xf = cell(1,0);
Af = cell(2,0);

% Format Output Arguments
if ~isCellX
    Y = cell2mat(Y);
end
end

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);
y = bsxfun(@plus,y,settings.ymin);
end

% Remove Constants Input Processing Function
function y = removeconstantrows_apply(x,settings)
y = x(settings.keep,:);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings)
x = bsxfun(@minus,y,settings.ymin);
x = bsxfun(@rdivide,x,settings.gain);
x = bsxfun(@plus,x,settings.xoffset);
end
 ================================================================
                                    ==

% Solve an Input-Output Fitting problem with a Neural Network
% Script generated by Neural Fitting app
% Created 16-Apr-2020 20:23:51
%
% This script assumes these variables are defined:
%
%   input - input data.
%   output - target data.

x = input;
t = output;

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. Suitable in low memory situations.
trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.
```

```
% Create a Fitting Network
hiddenLayerSize = 56;
net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
[net,tr] = train(net,x,t);

% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, ploterrhist(e)
%figure, plotregression(t,y)
%figure, plotfit(net,x,t)
```

==========

==

د. برنامج التجربة الرابعة

```
function [Y,Xf,Af] = myNeuralNetworkFunction(X,~,~)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Auto-generated by MATLAB, 16-Apr-2020 23:40:03.
%
% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:
%
%   X = 1xTS cell, 1 inputs over TS timesteps
%   Each X{1,ts} = 151xQ matrix, input #1 at timestep ts.
%
% and returns:
%   Y = 1xTS cell of 1 outputs over TS timesteps.
%   Each Y{1,ts} = 8xQ matrix, output #1 at timestep ts.
%
% where Q is number of samples (or series) and TS is the number of
timesteps.

%#ok<*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====
```

```
% Input 1
x1_step1.keep = [8 16 24 32 40 48 56 64 72 80 88 96 104 112 113 114 115
116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133
134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150
151];
x1_step2.xoffset =
[2323;627;258;3000;276;7263;137;2000;599;191;2898;271;4908;111;19.966;3
.928;3.988;3.928;3.928;3.928;3.604;185400;2.372;4.212;4.365;4.212;4.212
;4.212;3.89;10330;4.68;2.949;2.949;2.949;2.949;2.949;44.104;95888;0.174
;5.813;5.813;5.913;5.813;5.813;91.231;12000220;17250;20000;100000;30000
;20000;40000;-207900];
x1_step2.gain =
[0.000563063063063063;0.00119402985074627;0.0090702947845805;0.00131233
595800525;0.00333333333333333;0.000363570259952736;0.00888888888888889;
0.00632911392405063;0.00275103163686382;0.021505376344086;0.00399201596
806387;0.00641025641025641;0.000451977401129943;0.0540540540540541;0.28
3527076835838;3.4965034965035;2.19298245614035;3.4965034965035;3.496503
4965035;3.4965034965035;3.47222222222222;1.29366106080207e-
05;0.0948316737790422;1.07066381156317;1.16618075801749;1.0706638115631
7;1.07066381156317;1.07066381156317;0.91324200913242;8.63793947136674e-
06;1.13636363636364;1.14876507754164;1.14876507754164;1.14876507754164;
1.14876507754164;1.14876507754164;0.871080139372823;1.91916477948797e-
05;0.562429696287964;1.77462289263531;1.77462289263531;1.94741966893866
;1.77462289263531;1.77462289263531;0.565131393048884;5.5555740741358e-
08;1.6359918200409e-05;0.000307692307692308;3.14218381775334e-
05;0.000266666666666667;0.00036036036036036;0.000238095238095238;1.9627
0853778214e-05];
x1_step2.ymin = -1;

% Layer 1
b1 = [1.5137514121180331017;-1.4041204207252473424;-
1.4840585159136079341;-
1.5091958921406982697;1.2789452358502806373;1.3004889424439136647;1.190
2548881815271375;1.1138921686060765381;1.2128792317630958664;0.98663044
009054134964;1.0248805230034001568;-
0.82471964163447297747;0.90203367710015802672;0.77628868985701304783;-
0.88820140627633414887;-0.71932492151756588505;-
0.60600874753036948483;0.56336122832065349542;0.44093230806215238937;0.
48178658161000342819;-
0.50134995265900039974;0.29355443399179298458;0.21446921212270680934;0.0
044470728102675577409;-0.21334913446926445646;-
0.10385603773632562374;0.055404996404899535534;-
0.072620938963958547863;0.11413594684019222336;-
0.086381949699519192111;0.039736802983782265608;-
0.32928892335896930987;0.32903598715582937917;0.47525662559680992292;0.
33257222957938553654;0.5859758419678423591;0.49216039051190357823;-
0.49684499365956186212;0.49756324270485219333;-0.68726027906110032095;-
0.94044997790732054455;-0.70888485573527526551;0.69567604601380805818;-
0.75749695381159887209;-1.0059222454849210404;0.87687130042050553946;-
1.1615413729262267228;0.89766357523141837582;-1.0919982408648245542;-
1.2543100752332752013;-
1.2495423112160819468;1.2996567708432922306;1.3505269911492518986;-
1.4167571658171103088;-1.5284731555307833339;1.7106163425245535237];
IW1_1 = [];

% Layer 2
```

```
b2 =
[0.36420882246827496198;0.28081920645042418583;0.95976799315146821279;0
.72854789208765213804;0.92187073232997529537;0.37973653978973448408;-
0.86836094248214923397;-0.43148344605842436916];
LW2_1 = [];

% Output 1
y1_step1.ymin = -1;
y1_step1.gain =
[0.000705965407695023;0.000868809730668983;0.006405944716697l;0.0004520
79566003617;0.00342465753424658;0.000160025604096655;0.0078431372549019
6;1.30712726171413e-09];
y1_step1.xoffset = [2264;0;75.79;0;0;0;0;66866100];

% ===== SIMULATION ========

% Format Input Arguments
isCellX = iscell(X);
if ~isCellX
    X = {X};
end

% Dimensions
TS = size(X,2); % timesteps
if ~isempty(X)
    Q = size(X{1},2); % samples/series
else
    Q = 0;
end

% Allocate Outputs
Y = cell(1,TS);

% Time loop
for ts=1:TS

    % Input 1
    temp = removeconstantrows_apply(X{1,ts},x1_step1);
    Xp1 = mapminmax_apply(temp,x1_step2);

    % Layer 1
    a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*Xp1);

    % Layer 2
    a2 = repmat(b2,1,Q) + LW2_1*a1;

    % Output 1
    Y{1,ts} = mapminmax_reverse(a2,y1_step1);
end

% Final Delay States
Xf = cell(1,0);
Af = cell(2,0);
```

```
% Format Output Arguments
if ~isCellX
    Y = cell2mat(Y);
end
end

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);
y = bsxfun(@plus,y,settings.ymin);
end

% Remove Constants Input Processing Function
function y = removeconstantrows_apply(x,settings)
y = x(settings.keep,:);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end
 % Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings)
x = bsxfun(@minus,y,settings.ymin);
x = bsxfun(@rdivide,x,settings.gain);
x = bsxfun(@plus,x,settings.xoffset);
end
 ================================================================================
                                        ==

% Solve an Input-Output Fitting problem with a Neural Network
% Script generated by Neural Fitting app
% Created 16-Apr-2020 23:40:35
%
% This script assumes these variables are defined:
%
%   input - input data.
%   output - target data.

x = input;
t = output;

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. Suitable in low memory situations.
trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 56;
net = fitnet(hiddenLayerSize,trainFcn);
```

```
% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
[net,tr] = train(net,x,t);

% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, ploterrhist(e)
%figure, plotregression(t,y)
%figure, plotfit(net,x,t)
```

==========

هـ. برنامج التجربة الخامسة

```
function [Y,Xf,Af] = myNeuralNetworkFunction(X,~,~)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Auto-generated by MATLAB, 17-Apr-2020 19:39:39.
%
% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:
%
%   X = 1xTS cell, 1 inputs over TS timesteps
%   Each X{1,ts} = 151xQ matrix, input #1 at timestep ts.
%
% and returns:
%   Y = 1xTS cell of 1 outputs over TS timesteps.
%   Each Y{1,ts} = 8xQ matrix, output #1 at timestep ts.
%
% where Q is number of samples (or series) and TS is the number of
timesteps.

%#ok<*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
```

```
x1_step1.keep = [8 16 24 32 40 48 56 64 72 80 88 96 104 112 113 114 115
116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133
134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150
151];
x1_step2.xoffset =
[2323;627;258;3000;276;7263;137;2000;599;191;2898;271;4908;111;19.966;3
.928;3.988;3.928;3.928;3.928;3.604;185400;2.372;4.212;4.365;4.212;4.212
;4.212;3.89;10330;4.68;2.949;2.949;2.949;2.949;2.949;44.104;95888;0.174
;5.813;5.813;5.913;5.813;5.813;91.231;12000220;17250;20000;100000;30000
;20000;40000;-207900];
x1_step2.gain =
[0.000563063063063063;0.00119402985074627;0.0090702947845805;0.00131233
595800525;0.00333333333333333;0.0003635702599527 36;0.00888888888888889;
0.00632911392405063;0.00275103163686382;0.021505376344086;0.00399201596
806387;0.00641025641025641;0.000451977401129943;0.0540540540540541;0.28
3527076835838;3.4965034965035;2.19298245614035;3.4965034965035;3.496503
4965035;3.4965034965035;3.47222222222222;1.29366106080207e-
05;0.0948316737790422;1.07066381156317;1.16618075801749;1.0706638115631
7;1.07066381156317;1.07066381156317;0.91324200913242;8.63793947136674e-
06;1.13636363636364;1.14876507754164;1.14876507754164;1.14876507754164;
1.14876507754164;1.14876507754164;0.871080139372823;1.91916477948797e-
05;0.562429696287964;1.77462289263531;1.77462289263531;1.94741966893866
;1.77462289263531;1.77462289263531;0.565131393048884;5.5555740741358e-
08;1.6359918200409e-05;0.000307692307692308;3.14218381775334e-
05;0.00026666666666667;0.00036036036036036;0.000238095238095238;1.9627
0853778214e-05];
x1_step2.ymin = -1;

% Layer 1
b1 = [1.5338477404035857354;-1.4875073550655968369;-
1.3446810144923115882;1.369134863989389439;1.3018603867692677412;-
1.18691429228720291;-1.1535025517124914973;-1.0333892801228539149;-
1.0594117352664540466;1.025813293600071 6739;-0.82863349103043604238;-
1.0295444762005998562;-0.83559066760231925919;0.83991492139829748709;-
0.71114750376656121755;0.59928749067403530582;0.70505143413877779945;-
0.60756598080713186683;0.69064359353127735552;0.54644108381245581896;-
0.44794079929974545129;0.32309656285769156048;0.12594805077253204417;0.
10003237608457894725;-
0.063867439994750044296;0.17399541941006893553;0.16371643575198344123;-
0.12638925083567029151;0.01244461497466275296;-
0.11564816588567589095;0.15813593375249232476;-
0.094369180360070969993;-0.045540228534143079031;-
0.31981284885810534746;-
0.38233066618335403986;0.36834087823884309465;0.57466674451236388066;0.
52661765690830331099;-0.50682642002809263904;-
0.51443399524710009 54;0.79201939516322961143;0.61520072338684217428;0.8
1605537399596239645;-0.90034026040722070494;-1.013752910331978 8959;-
0.86643301077613932293;1.0229054495516232137;-
1.1061975815337581341;1.0441037573432410124;1.2159847121932334524;-
1.1832437289471462449;-1.2417166103370769203;-1.4235701045068536175;-
1.4396915505906913157;1.470635532221902686;1.4760611616487380182];
IW1_1 = [];

% Layer 2
b2 =
[0.154325466223329405;0.74409382248532240922;0.706240668521329828 52;0.6
```

```
8358961515376659435;1.025596083298202954;0.66820879184283377139;0.36729
607052376889165;0.19862478339365055581;
LW2_1 = [];

% Output 1
y1_step1.ymin = -1;
y1_step1.gain =
[0.000705965407695023;0.000868809730668983;0.0064059447166971;0.0004520
79566003617;0.00342465753424658;0.000160025604096655;0.0078431372549019
6;1.30712726171413e-09];
y1_step1.xoffset = [2264;0;75.79;0;0;0;0;66866100];

% ===== SIMULATION ========

% Format Input Arguments
isCellX = iscell(X);
if ~isCellX
    X = {X};
end

% Dimensions
TS = size(X,2); % timesteps
if ~isempty(X)
    Q = size(X{1},2); % samples/series
else
    Q = 0;
end

% Allocate Outputs
Y = cell(1,TS);

% Time loop
for ts=1:TS

    % Input 1
    temp = removeconstantrows_apply(X{1,ts},x1_step1);
    Xp1 = mapminmax_apply(temp,x1_step2);

    % Layer 1
    a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*Xp1);

    % Layer 2
    a2 = repmat(b2,1,Q) + LW2_1*a1;

    % Output 1
    Y{1,ts} = mapminmax_reverse(a2,y1_step1);
end

% Final Delay States
Xf = cell(1,0);
Af = cell(2,0);

% Format Output Arguments
if ~isCellX
```

```
    Y = cell2mat(Y);
end
end

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);
y = bsxfun(@plus,y,settings.ymin);
end

% Remove Constants Input Processing Function
function y = removeconstantrows_apply(x,settings)
y = x(settings.keep,:);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings)
x = bsxfun(@minus,y,settings.ymin);
x = bsxfun(@rdivide,x,settings.gain);
x = bsxfun(@plus,x,settings.xoffset);
end
  ================================================================
% Solve an Input-Output Fitting problem with a Neural Network
% Script generated by Neural Fitting app
% Created 17-Apr-2020 19:40:26
%
% This script assumes these variables are defined:
%
%   input - input data.
%   output - target data.

x = input;
t = output;

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. Suitable in low memory situations.
trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 56;
net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
```

```
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
[net,tr] = train(net,x,t);

% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, ploterrhist(e)
%figure, plotregression(t,y)
%figure, plotfit(net,x,t)
```

==========