function [Y,Xf,Af] = myNeuralNetworkFunction(X,~,~)  
%MYNEURALNETWORKFUNCTION neural network simulation function.  
%  
% Auto-generated by MATLAB, 04-Dec-2022 14:52:18.  
%  
% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:  
%  
%   X = 1xTS cell, 1 inputs over TS timesteps  
%   Each X{1,ts} = Qx24 matrix, input #1 at timestep ts.  
%  
% and returns:  
%   Y = 1xTS cell of 1 outputs over TS timesteps.  
%   Each Y{1,ts} = Qx1 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.xoffset = data.xs;  
x1\_step1.gain = data.gi;  
x1\_step1.ymin = -1;  
  
% Layer 1  
b1 = data.b;  
IW1\_1 = data.iw;  
% Layer 2  
b2 = 0.74436015070712513442;  
LW2\_1 = data.lw;  
% Output 1  
y1\_step1.ymin = -1;  
y1\_step1.gain = 0.111111111111111;  
y1\_step1.xoffset = 1;  
  
% ===== 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},1); % samples/series  
else  
    Q = 0;  
end  
  
% Allocate Outputs  
Y = cell(1,TS);  
  
% Time loop  
for ts=1:TS  
      
    % Input 1  
    X{1,ts} = X{1,ts}';  
    Xp1 = mapminmax\_apply(X{1,ts},x1\_step1);  
      
    % 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);  
    Y{1,ts} = Y{1,ts}';  
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  
  
% 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

function [Y,Xf,Af] = myNeuralNetworkFunction(X,~,~)

%MYNEURALNETWORKFUNCTION neural network simulation function.

%% Auto-generated by MATLAB, 04-Dec-2022 14:52:18.

%% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:%%   X = 1xTS cell, 1 inputs over TS timesteps%   Each X{1,ts} = Qx24 matrix, input #1 at timestep ts.%% and returns:%   Y = 1xTS cell of 1 outputs over TS timesteps.%   Each Y{1,ts} = Qx1 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.xoffset = data.xs;

x1\_step1.gain = data.gi;

x1\_step1.ymin = -1;

% Layer 1

b1 = data.b;

IW1\_1 = data.iw;% Layer 2

b2 = 0.74436015070712513442;

LW2\_1 = data.lw;% Output 1

y1\_step1.ymin = -1;

y1\_step1.gain = 0.111111111111111;

y1\_step1.xoffset = 1;

% ===== 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},1); % samples/series**else**

    Q = 0;**end**

% Allocate Outputs

Y = cell(1,TS);

% Time loop

for ts=1:TS

    % Input 1

    X{1,ts} = X{1,ts}';

    Xp1 = mapminmax\_apply(X{1,ts},x1\_step1);

    % 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);

    Y{1,ts} = Y{1,ts}';**end**

% Final Delay States

Xf = cell(1,0);

Af = cell(2,0);

% Format Output Arguments

if ~isCellX

    Y = cell2mat(Y);**endend**

% ===== 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**

% 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**