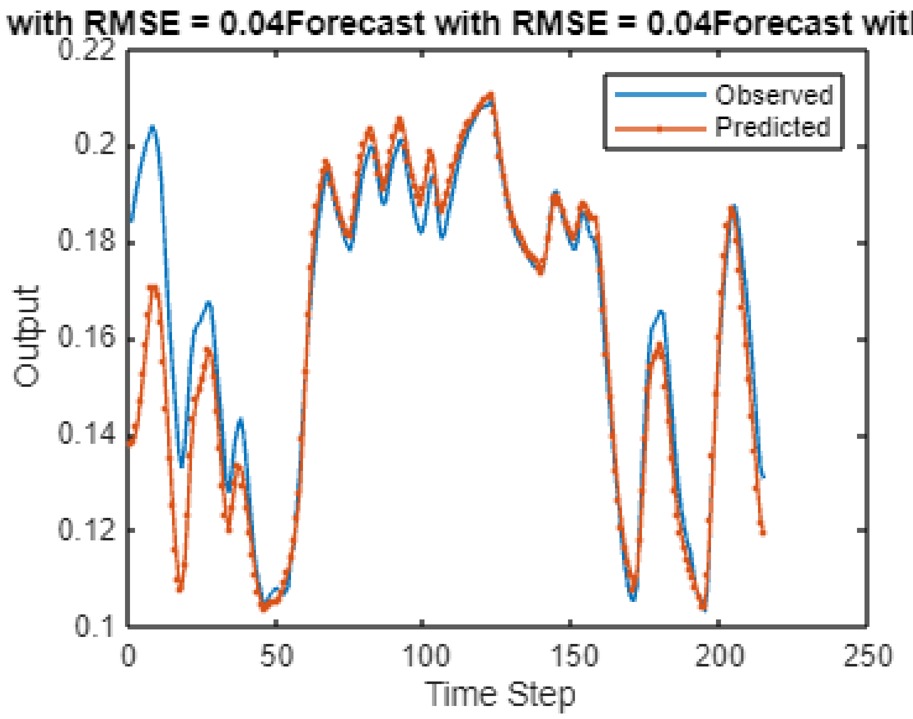
```
%clear

X = [u1, u2]; % Input variables (pump flow rates)
YY = y2;       % Output variable (bottom left tank level)

% Assuming U is the input and Y is the output
numTimeStepsTrain = 500;
U_Train = X(1:numTimeStepsTrain, :);
Y_Train = YY(1:numTimeStepsTrain);
U_est = X(numTimeStepsTrain+1:end, :);
Y_Test = YY(numTimeStepsTrain+1:end);

% Standardize the training data (both U and Y)
mu_UU = mean(U_Train);
sigma_UU = std(U_Train);
U_TrainStandardized = (U_Train - mu_UU) ./ sigma_UU;

mu_YY = mean(Y_Train);
sigma_YY = std(Y_Train);
Y_TrainStandardized = (Y_Train - mu_YY) / sigma_YY;

% Prepare sequences for LSTM
X_Train = U_TrainStandardized;

% ```matlab
numFeatures = 2; % as U has two columns
numResponses = 1;
numHiddenUnits = 200;

layers = [
    sequenceInputLayer(numFeatures)
    lstmLayer(numHiddenUnits)
    fullyConnectedLayer(numResponses)
    regressionLayer
         ];
```

```

options = trainingOptions('adam', ...
    MaxEpochs=200, ...
    SequencePaddingDirection='left', ...
    Shuffle='every-epoch', ...
    Plots='training-progress', ...
    Verbose=0);

net_1 = trainNetwork(X_Train', Y_TrainStandardized', layers, options);

U_TestStandardized = (U_est - mu_UU) ./ sigma_UU;
X_Test = U_TestStandardized;
```

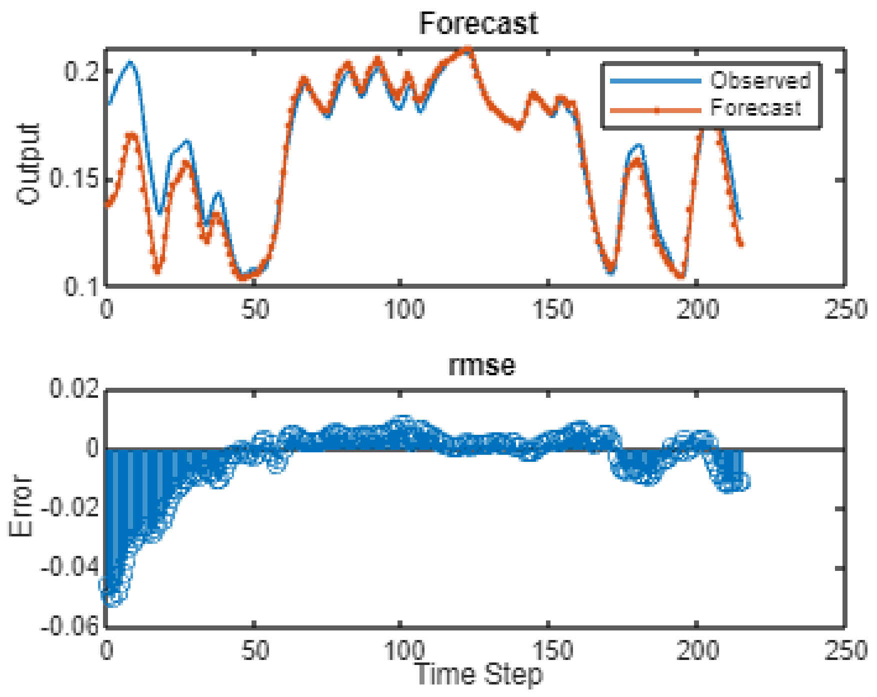
```
% Reset the state of the network for prediction
net_1 = resetState(net_1);
```

```


% Predict using the trained network
Y_PredStandardized = predict(net_1, X_Test');
Y_Pred = sigma_YY * Y_PredStandardized + mu_YY;

% Calculate RMSE
rmse_11 = sqrt(mean((Y_Pred - Y_Test).^2));


figure;
plot(Y_Test);
hold on;
plot(Y_Pred, '.-');
hold off;
legend(["Observed", "Predicted"]);
xlabel("Time Step");
ylabel("Output");
title(sprintf("Forecast with RMSE = %.2f", rmse_11));
```

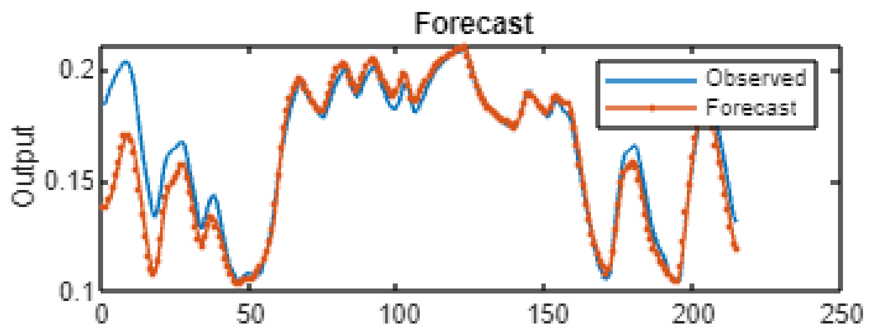


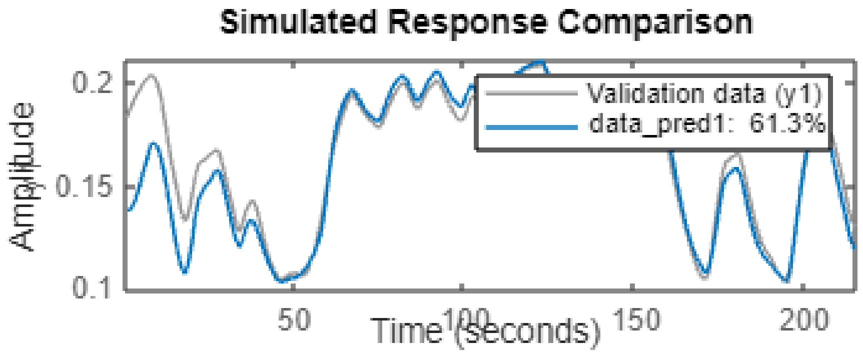
```
% figure;
% subplot(2,1,1); % Upper plot for observed vs. forecast
% plot(YTest);
% hold on;
% plot(YPred, '.-');
% hold off;
% legend(["Observed", "Forecast"]);
% ylabel("Output");
% title("Forecast");
%
% subplot(2,1,2); % Lower plot for errors
% stem(YPred - YTest);
% xlabel("Time Step");
% ylabel("Error");
% title(sprintf("RMSE = %.2f", rmse));
```

```
% Standardize the test data using the training mean and standard deviation
U_TestStandardized = (U_est - mu_UU) ./ sigma_UU;
X_Test = U_TestStandardized;

% Reset the state of the network for prediction
net_1 = resetState(net_1);

% Predict using the trained network
Y_PredStandardized = predict(net_1, X_Test');
Y_Pred = sigma_YY * Y_PredStandardized + mu_YY;

% Calculate RMSE
rmse_11 = sqrt(mean((Y_Pred - Y_Test).^2));

% Visualization of results and errors
figure;
subplot(2,1,1); % Upper plot for observed vs. forecast
plot(Y_Test);
hold on;
plot(Y_Pred, '.-');
hold off;
legend(["Observed", "Forecast"]);
ylabel("Output");
title("Forecast");

subplot(2,1,2); % Lower plot for errors
stem(Y_Pred' - Y_Test);
xlabel("Time Step");
ylabel("Error");
title(' rmse');
```



```
% Convert LSTM predictions from single to double precision
YPred_double1 = double(Y_Pred);

% Convert LSTM predictions and actual data to iddata objects
data_pred1 = iddata(YPred_double1', [], 1);  % Assuming 1 sample time
between each prediction
data_actual1 = iddata(Y_Test, [], 1); % Assuming 1 sample time between each
actual data point

% Compare LSTM predictions with actual data
compare(data_actual1, data_pred1);
```


Forecast
Output
Observed
Forecast
0.1
0.15
0.2
0
50
100
150
200
250


Simulated Response Comparison
Amplitude
Validation data (y1)
data_pred1: 61.3%
Time (seconds)
0.1
0.15
0.2
50
100
150
200