

---

## Table of Contents

closed_loop.m .....	1
Load Data .....	1
Plot Raw Data .....	1
Remove Errors .....	2
Retime, Smooth, and Remove Outliers .....	2
Generate Calibrations .....	4

## closed\_loop.m

This script is for generating calibration models for the closed-loop test between the ELT S300 sensor and LICOR LI-7810 reference instrument. This script does not support multiple calibration files currently.

Lincoln Scheer 7/8/2024

This script has not been optimized and can take a while to run, sit back.

```
clc, clear, close all
```

## Load Data

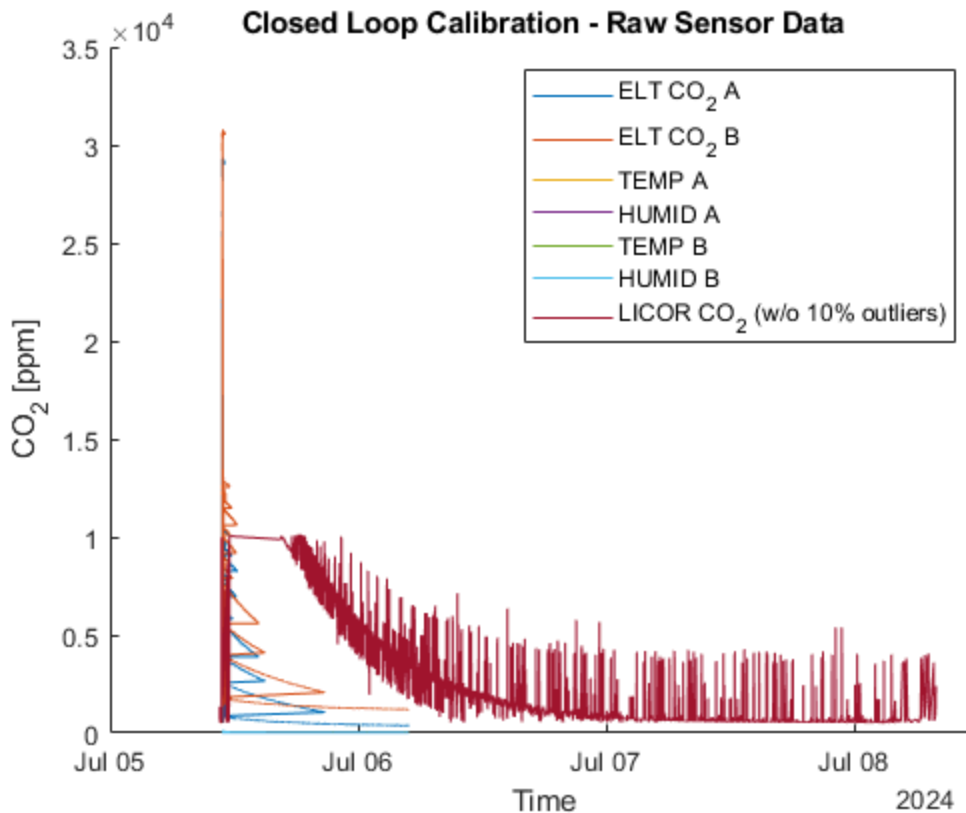
```
% import elt sensor dataset
daq = IMPORTDAQFILE("data/7.5.2024/daq.csv");
daq = rmmissing(daq);

% from elt sensor dataset, grab per-sensor dataset
sensors = {[daq(:,[2,3,4])], [daq(:,[5,6,7])]};

% import licor reference instrument dataset
licor = IMPORTLICORFILE("data/7.5.2024/licor.txt");
licor = rmmissing(licor);
```

## Plot Raw Data

```
figure();
hold on;
plot(daq.T, daq.CA, 'DisplayName', 'ELT CO_2 A');
plot(daq.T, daq.CB, 'DisplayName', 'ELT CO_2 B');
plot(daq.T, daq.TA, 'DisplayName', 'TEMP A');
plot(daq.T, daq.HA, 'DisplayName', 'HUMID A');
plot(daq.T, daq.TB, 'DisplayName', 'TEMP B');
plot(daq.T, daq.HB, 'DisplayName', 'HUMID B');
licor_tmp = rmoutliers(licor, 'percentile', [10, 90]);
plot(licor_tmp.T, licor_tmp.C, 'DisplayName', 'LICOR CO_2 (w/o 10% outliers)');
xlabel("Time")
ylabel("CO_2 [ppm]")
legend();
title("Closed Loop Calibration - Raw Sensor Data");
```



## Remove Errors

```
errorVals = [-999, 500, 2815, 64537, 231753, 65535, 2500, 2559];
```

```
for index = 1:2
    sensor = sensors{1,index};
    sensor = timetable2table(sensor);
    errorMask = (ismember(table2array(sensor(:,2)), errorVals));
    sensor = sensor(~errorMask, :);
    sensor = table2timetable(sensor);
    sensors{1,index} = sensor;
end
```

## Retime, Smooth, and Remove Outliers

```
% section settings
smooth_dt = minutes(1);
retime_dt = seconds(5);
outlier_bounds = [10, 90];
outlier_remove = true;

% smooth and retime sensor datasets
for index = 1:2
    sensor = sensors{1,index};
    sensor = retime(sensor,"regular", 'mean', 'TimeStep', retime_dt);
```

---

```

    sensor = smoothdata(sensor, 'movmean', smooth_dt);
    if (outlier_remove)
        sensor = rmoutliers(sensor, 'percentile', outlier_bounds);
    end
    sensors{1,index} = sensor;
end

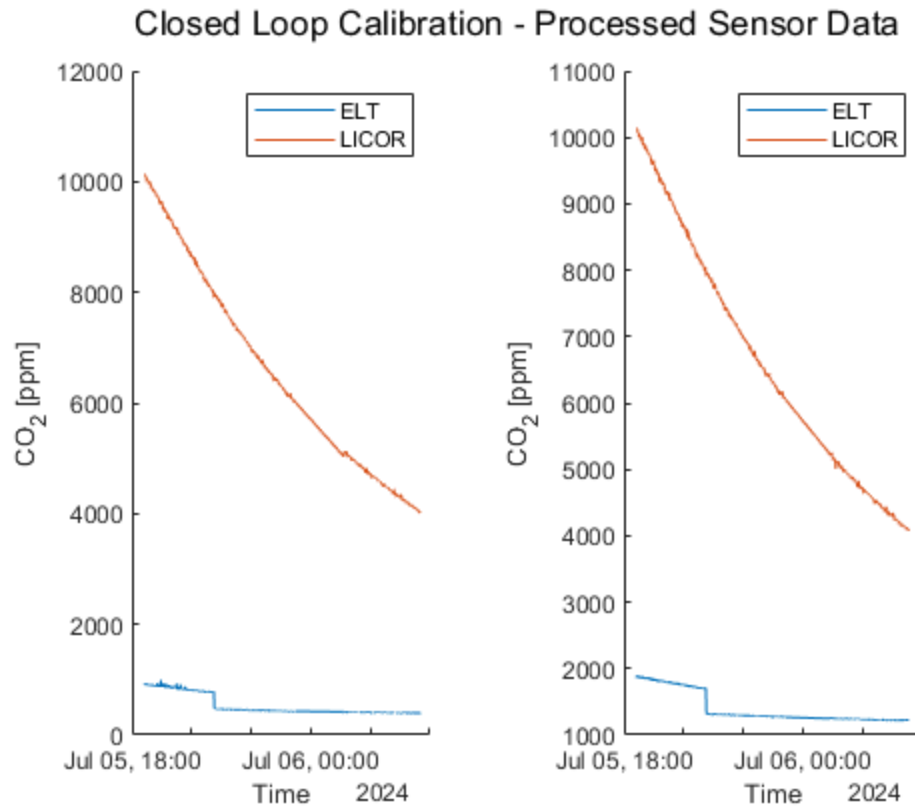
% smooth and retime reference dataset
licor = retime(licor,"regular", 'mean', 'TimeStep', retime_dt);
licor = smoothdata(licor, 'movmean', smooth_dt);
if (outlier_remove)
    licor = rmoutliers(licor, 'percentile', outlier_bounds);
end

% sync sensors with reference
for index = 1:2
    sensor = sensors{1,index};
    sensor = synchronize(sensor, licor);
    sensor = rmmissing(sensor);
    sensors{1,index} = sensor;
end

% plot datasets
figure();
sgtitle("Closed Loop Calibration - Processed Sensor Data");
for index = 1:2
    subplot(1,2,index);
    sensor = sensors{1,index};
    hold on;
    plot(sensor, 1, 'DisplayName', 'ELT');
    plot(sensor, 4, 'DisplayName', 'LICOR');
    xlabel("Time");
    ylabel("CO_2 [ppm]");
    legend();
    sensors{1,index} = sensor;
end

```

---



## Generate Calibrations

```
fprintf("Closed Loop Calibration - Calibration Results\n")
models = cell(2,2);
figure();
for index = 1:2
    sensor = sensors{1,index};
    sensor = timetable2table(sensor);

    % partition data
    cv = cvpartition(size(sensor,1), 'HoldOut', 0.3);
    trainX = table2array(sensor(~cv.test, 2:4));
    trainY = table2array(sensor(~cv.test, 5));
    testX = table2array(sensor(cv.test, 2:4));
    testY = table2array(sensor(cv.test, 5));

    % linear regression
    models{1,index} = fitlm(trainX, trainY);

    % network regression
    models{2,index} = feedforwardnet([16, 16]);
    models{2,index} = train(models{2,index}, trainX', trainY');

    % calculate metrics
    lin_pred = predict(models{1,index}, testX);
```

---

```

    net_pred = models{2,index}(testX)';
    lin_r2 = 1 - ((sum((lin_pred - testY).^2))/(sum(((testY -
mean(testY)).^2)))));
    net_r2 = 1 - ((sum((net_pred - testY).^2))/(sum(((testY -
mean(testY)).^2)))));
    lin_rmse = models{1,index}.RMSE;
    net_rmse = sqrt(mean((net_pred - testY).^2));

    % plot metrics
    figure()
    title("Closed Loop Calibration - Calibration Model Response - Sensor " +
index)
    hold on;
    plot(testX(:,1), testY, 'r-.', 'DisplayName', "Ground Truth (LICOR)");
    plot(testX(:,1), lin_pred, 'gs', 'DisplayName', "Linear Model Response");
    plot(testX(:,1), net_pred, 'mo', 'DisplayName', "Neural Model Response");
    legend();
    xlabel("X CO2 [ppm]")
    ylabel("Y CO2 [ppm]")

    fprintf("Sensor %d\n", index);
    fprintf("\tLinear\n")
    fprintf("\t\tRMSE:\t%0.1f\n", lin_rmse);
    fprintf("\t\tR^2:\t%0.3f\n", lin_r2);
    fprintf("\tNetwork\n")
    fprintf("\t\tRMSE:\t%0.1f\n", net_rmse);
    fprintf("\t\tR^2:\t%0.3f\n", net_r2);

    sensor = table2timetable(sensor);
    sensors{1,index} = sensor;
end

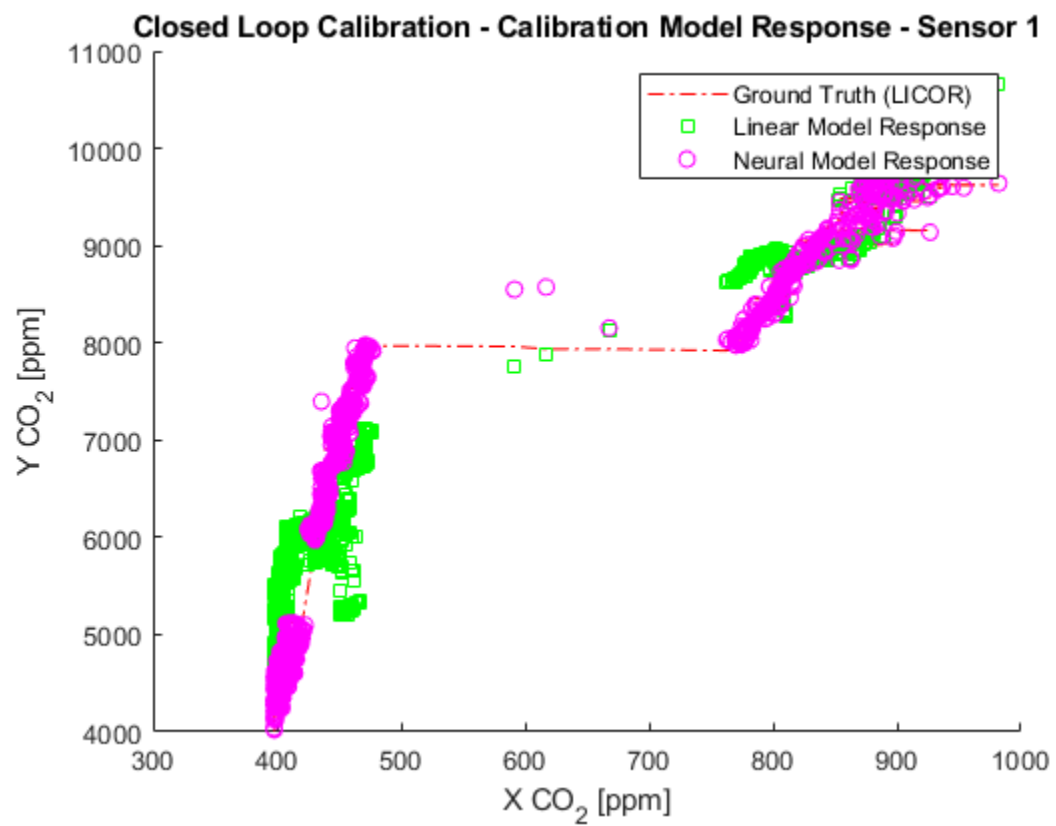
```

#### *Closed Loop Calibration - Calibration Results*

```


Sensor 1
Linear
  RMSE: 782.6
  R^2: 0.828
Network
  RMSE: 64.4
  R^2: 0.999
Sensor 2
Linear
  RMSE: 759.9
  R^2: 0.838
Network
  RMSE: 69.9
  R^2: 0.999

```



Network Diagram

### Training Results

Training finished: Met validation criterion 

### Training Progress

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	226	1000
Elapsed Time	-	00:00:52	-
Performance	5.95e+07	4.16e+03	0
Gradient	2.01e+08	5.82e+04	1e-07
Mu	0.001	10	1e+10
Validation Checks	0	6	6

### Training Algorithms

Data Division: Random dividerand

Training: Levenberg-Marquardt trainlm

Performance: Mean Squared Error mse

Calculations: MEX

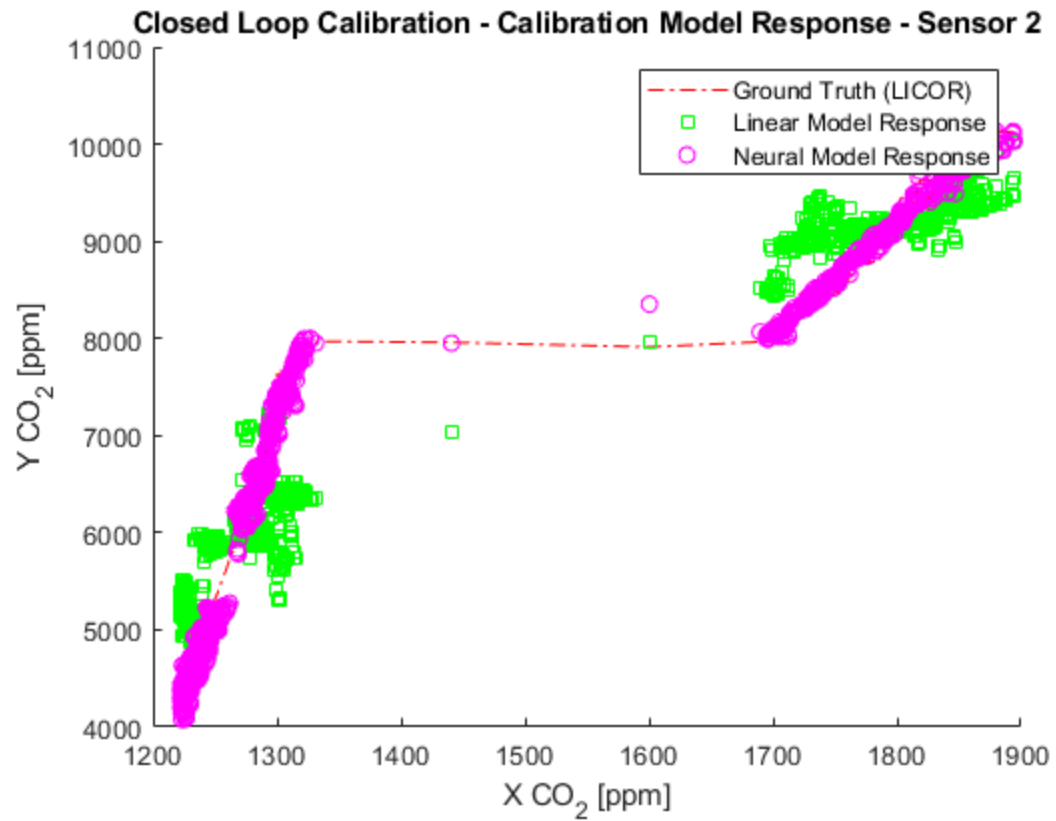
### Training Plots

Performance

Training State

Error Histogram

Regression



*Published with MATLAB® R2022a*