Table of Contents

ELEC 4700 PA-8 Diode Parameter Extraction 1
Tasks 1 - Generate Some Data 1
Task 2 - Polynomial fitting 3
Task 2 Conclusion 5
Task 3 - Nonlinear curve fitting 5
Task 3 a) 5
Task 3 b) 7
Task 3 c) 9
Task 3 Conclusion 11
Task 4 - Fitting using the Neural Net model 11
Task 4 Conclusion 13

ELEC 4700 PA-8 Diode Parameter Extraction

This PA will investigate the use of non-linear curve fitting for extracing parameters

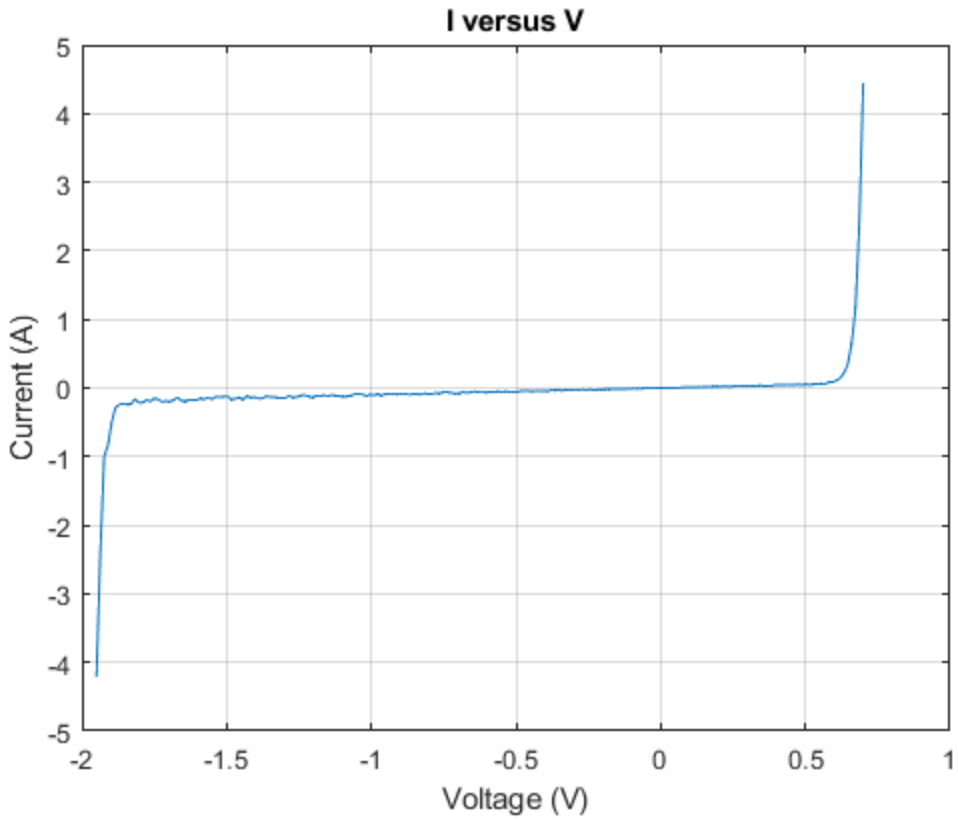
```
% % Docking figures
% set(0,'DefaultFigureWindowStyle','docked')
```

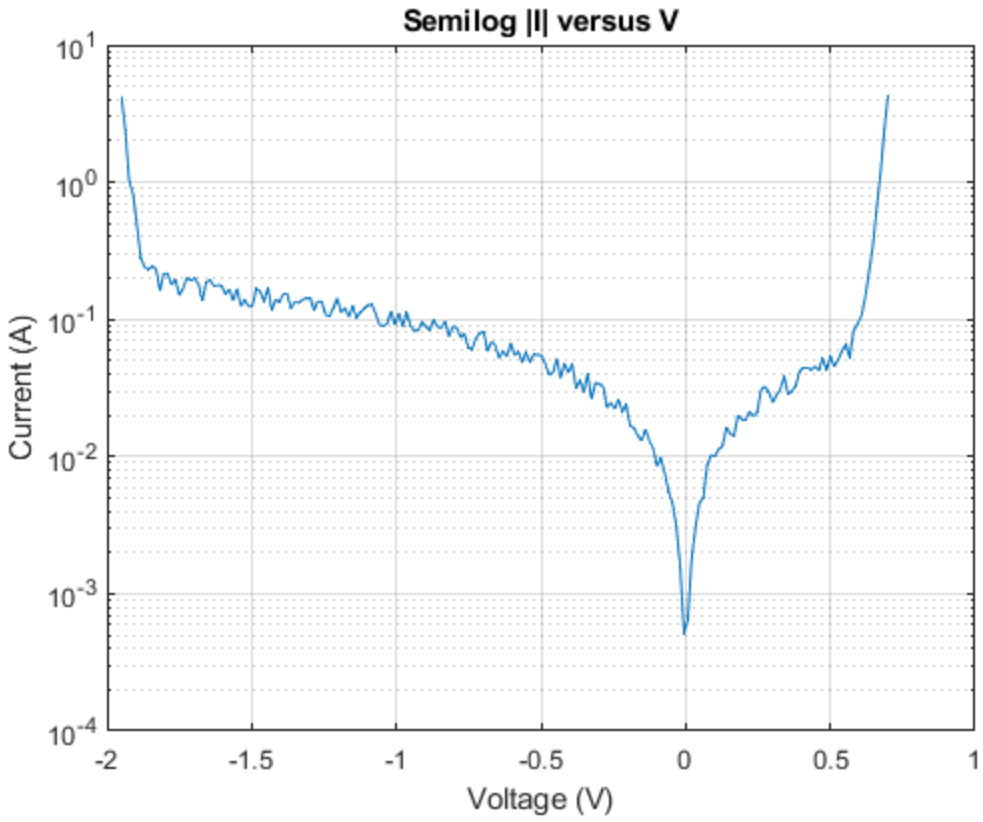
Tasks 1 - Generate Some Data

Generate some data for Is = 0.01pA, Ib=0.1pA, Vb=1.3V and Gp=0.1 Ohm^-1

```
Is = 0.01e-12;  % A
Ib = 0.1e-12;  % A
Vb = 1.3;  % V
Gp = 0.1;  % Ohm^-1

% Create a V vector from -1.95 to 0.7 volts with 200 steps
V = linspace(-1.95, 0.7, 200);

% Create the I vector
I = Is*(exp(1.2*V/0.025)-1) + Gp*V - Ib*(exp(-1.2/0.025*(V+Vb)) - 1);

% Create a second I vector with 20% random variation
randNum = 0.8+rand(1,200)*(1.2-0.8);  % list of 20% variation
I = I .* randNum;  % current vector with 20% random variation

% Plot the data
figure(1)
plot(V, I)
title("I versus V")
xlabel("Voltage (V)")
ylabel("Current (A)")
grid on

figure(2)
```

```
semilogy(V, abs(I))
title("Semilog |I| versus V")
xlabel("Voltage (V)")
ylabel("Current (A)")
grid on
```



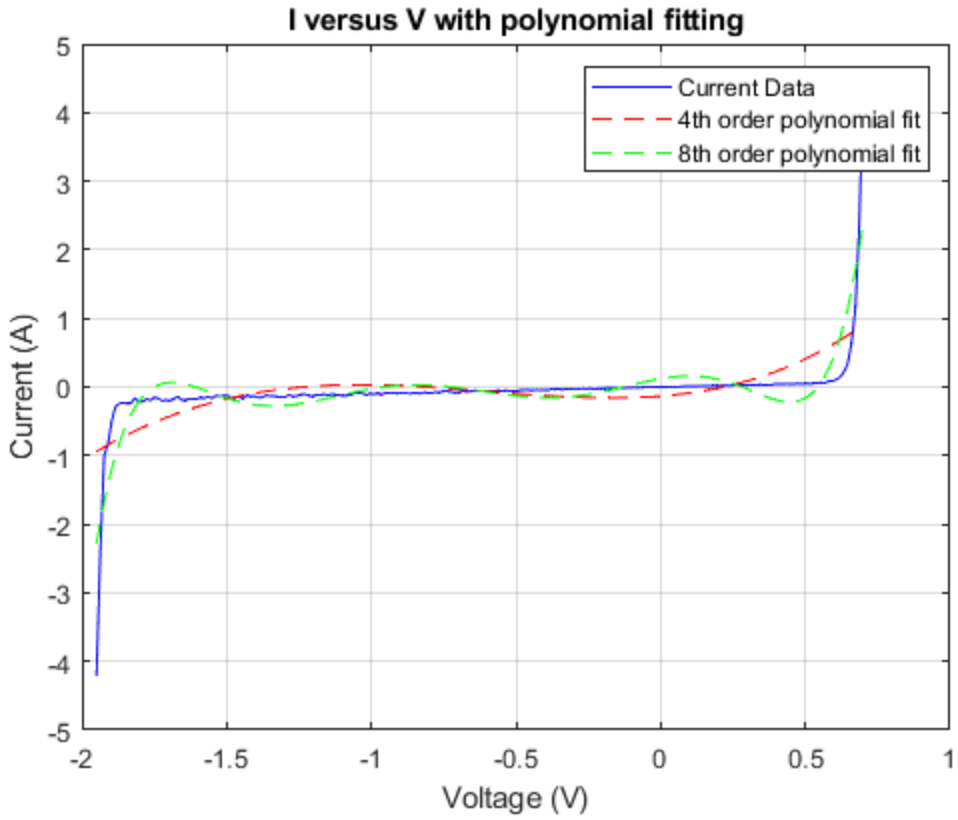


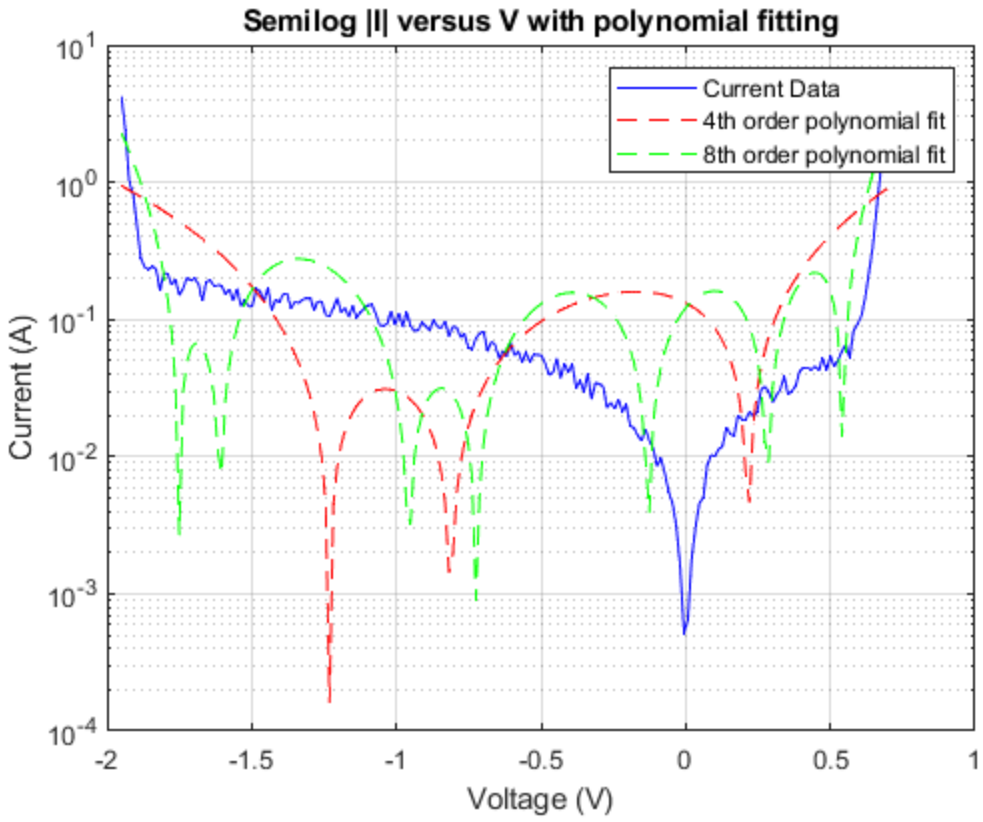
Task 2 - Polynomial fitting

Create a 4th order and 8th order polynomial fit for the two data vectors

```
p4 = polyfit(V, I, 4);  % 4th order
I4 = polyval(p4, V);  % Current with 4th order polynomial fit
p8 = polyfit(V, I, 8);  % 8th order
I8 = polyval(p8, V);  % Current with 8th order polynomial fit

% Plot the data with polynomial fitting
figure(3)
plot(V, I, "-b")
hold on
plot(V, I4, "--r")
plot(V, I8, "--g")
hold off
title("I versus V with polynomial fitting")
xlabel("Voltage (V)")
ylabel("Current (A)")
legend("Current Data", "4th order polynomial fit", "8th order polynomial fit")
grid on

figure(4)
semilogy(V, abs(I), "-b")
hold on
semilogy(V, abs(I4), "--r")
```

```
semilogy(V, abs(I8), "--g")
hold off
title("Semilog |I| versus V with polynomial fitting")
xlabel("Voltage (V)")
ylabel("Current (A)")
legend("Current Data", "4th order polynomial fit", "8th order polynomial fit")
grid on
```





Task 2 Conclusion

The 8th order polynomial fit does fit better than the 4th order polynomial especially at the two sides of the curve. The 8th order polynomial fit have more ripples than the 4th order polynomial fit. As shown in the semilogy plot, the polynomial fits in this case did not fit very well to the data. As the order of polynomial keep increasing, the polynomial fit curve will fit better and better to the actual curve. In theory (Taylor series), if the polynomial term reached infinity, the fitting will be perfect :)

Task 3 - Nonlinear curve fitting

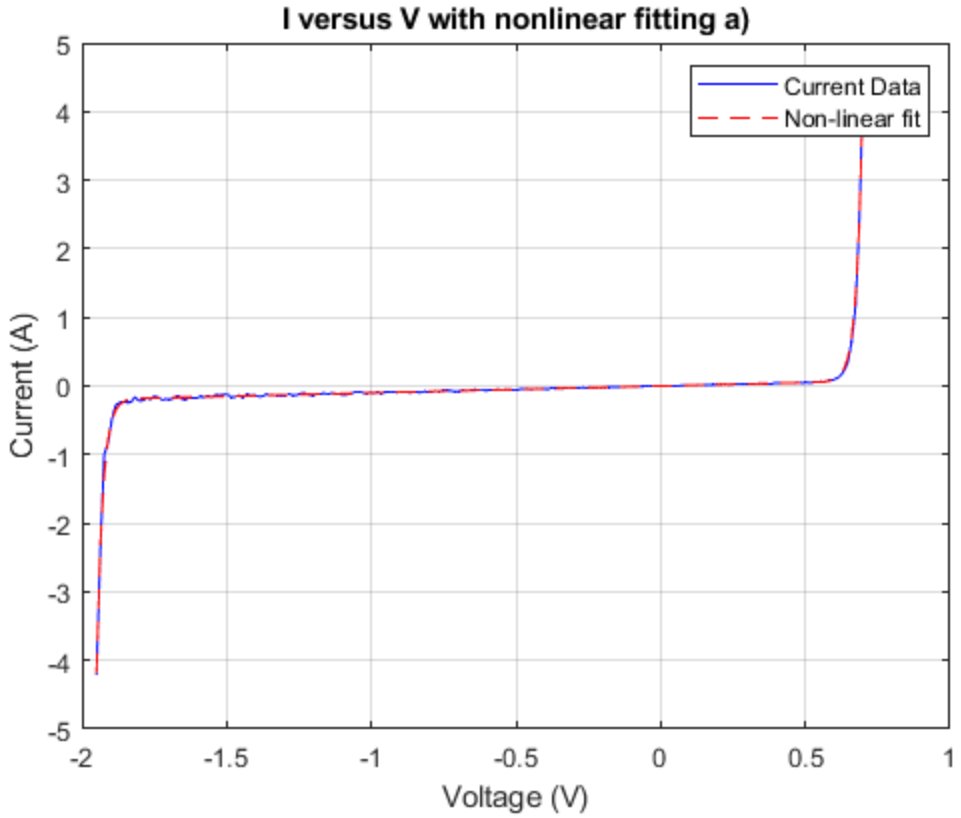
Nonlinear curve fitting to a physical model

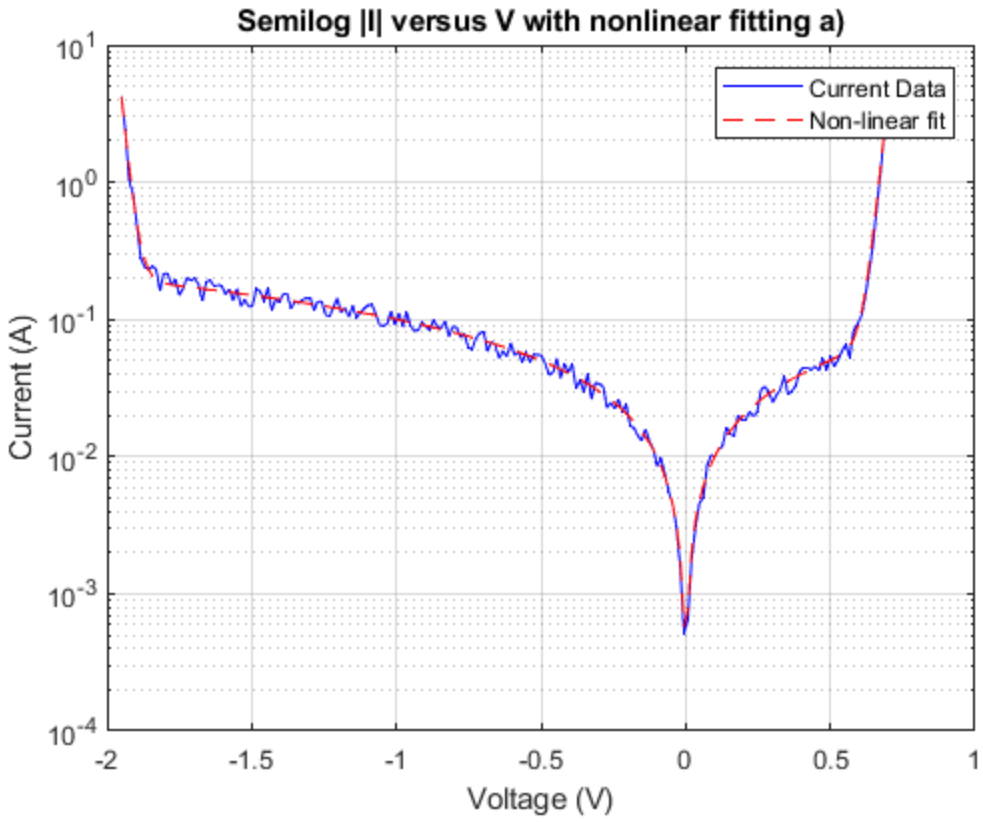
Task 3 a)

Only two fitted parameters A and C by explicitly setting B and D to the values that are already known

```
fo = fitoptions('Method','NonlinearLeastSquares', ...
                'StartPoint',[1,1]);
% Pass a string containing the non-linear function to fit
fo_a = fittype('A.*(exp(1.2*x/25e-3)-1) + 0.1.*x - C*(exp(1.2*(-(x
+1.3))/25e-3)-1)', 'options',fo);

% Extract the parameters
ff_a = fit(V',I',fo_a);
```

```
% Generate the curve
If_a = ff_a(V);

% Plot the data with nonlinear fitting case a)
figure(5)
plot(V, I, "-b")
hold on
plot(V, If_a, "--r")
hold off
title("I versus V with nonlinear fitting a)")
xlabel("Voltage (V)")
ylabel("Current (A)")
legend("Current Data", "Non-linear fit")
grid on

figure(6)
semilogy(V, abs(I), "-b")
hold on
semilogy(V, abs(If_a), "--r")
hold off
title("Semilog |I| versus V with nonlinear fitting a)")
xlabel("Voltage (V)")
ylabel("Current (A)")
legend("Current Data", "Non-linear fit")
grid on
```



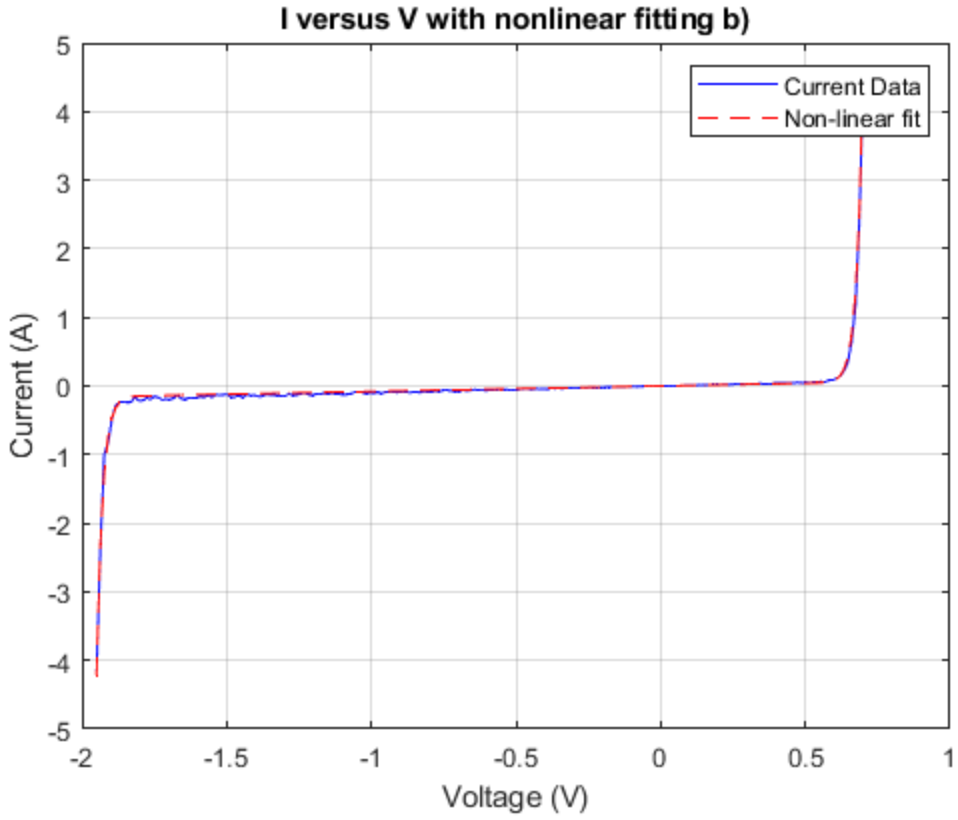


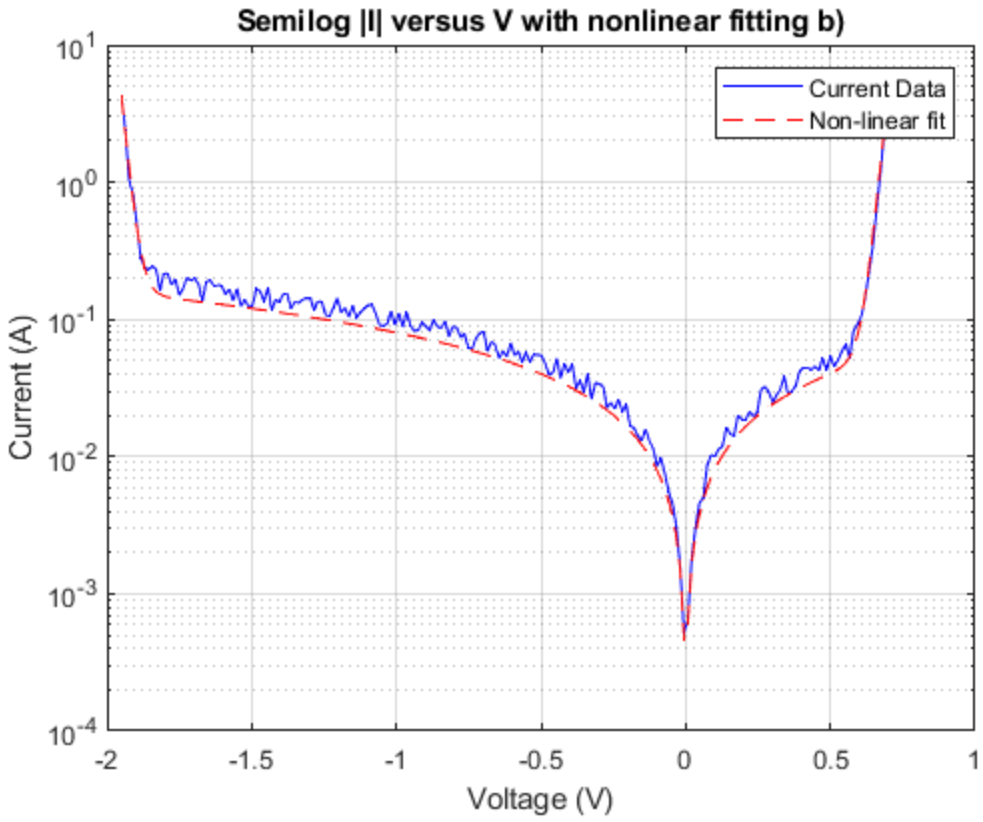
Task 3 b)

Using three fitted parameters A, B and C by explicitly setting D to the value that are already known

```
fo = fitoptions('Method','NonlinearLeastSquares', ...
                'StartPoint',[1e-12,0.08,1e-12]);
% Pass a string containing the non-linear function to fit
fo_b = fittype('A.*(exp(1.2*x/25e-3)-1) + B.*x - C*(exp(1.2*(-(x
+1.3))/25e-3)-1)', ...
        'options',fo);

% Extract the parameters
ff_b = fit(V',I',fo_b);
% Generate the curve
If_b = ff_b(V);

% Plot the data with nonlinear fitting case b)
figure(7)
plot(V, I, "-b")
hold on
plot(V, If_b, "--r")
hold off
title("I versus V with nonlinear fitting b)")
xlabel("Voltage (V)")
ylabel("Current (A)")
legend("Current Data", "Non-linear fit")
```

```
grid on

figure(8)
semilogy(V, abs(I), "-b")
hold on
semilogy(V, abs(If_b), "--r")
hold off
title("Semilog |I| versus V with nonlinear fitting b)")
xlabel("Voltage (V)")
ylabel("Current (A)")
legend("Current Data", "Non-linear fit")
grid on
```



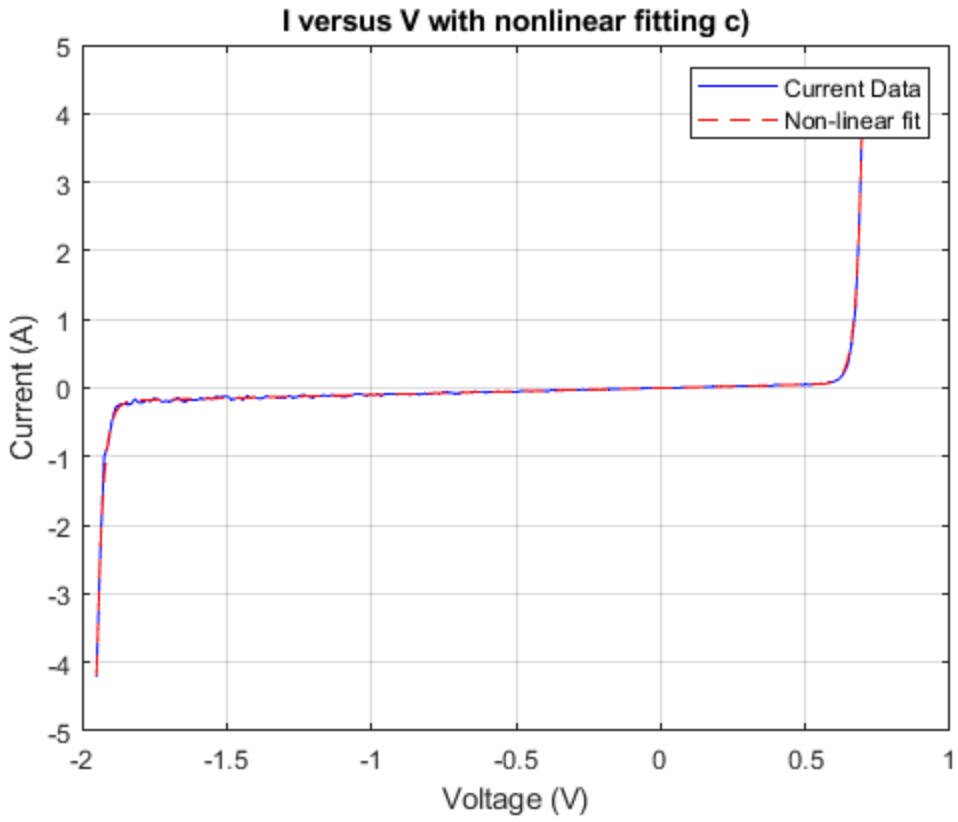


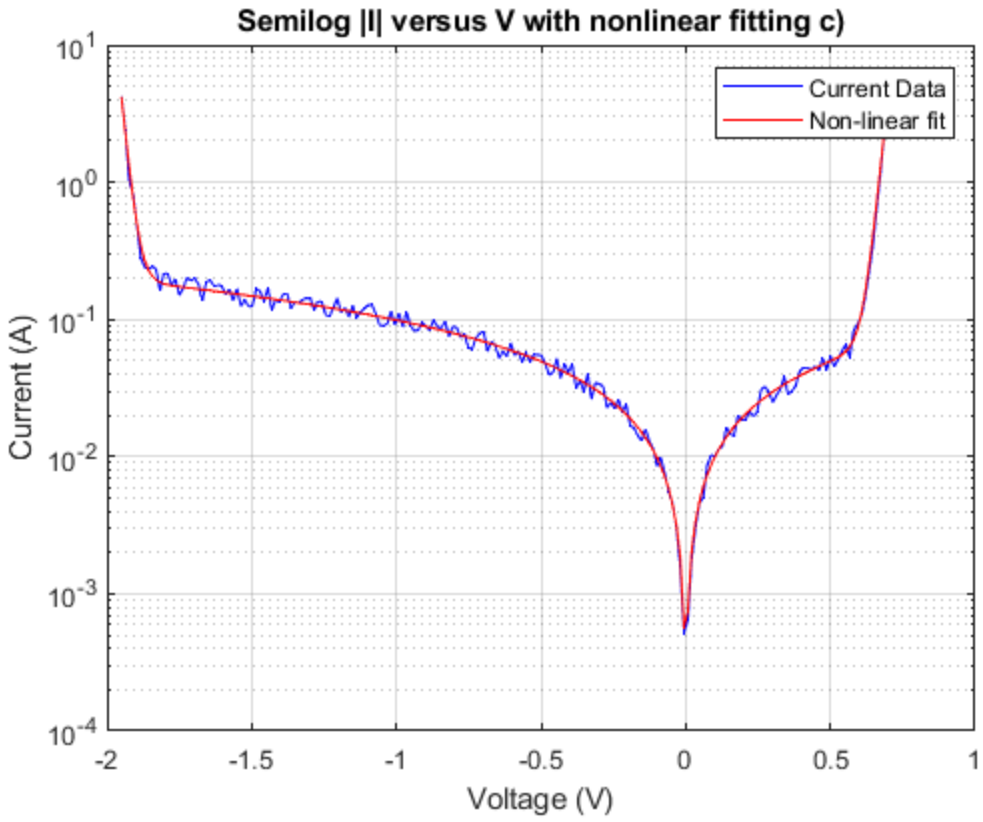
Task 3 c)

Fitting all four parameters A, B, C and D

```
fo = fitoptions('Method','NonlinearLeastSquares', ...
                'StartPoint',[1e-12,1,1e-12,1]);
% Pass a string containing the non-linear function to fit
fo_c = fittype('A.*(exp(1.2*x/25e-3)-1) + B.*x - C*(exp(1.2*(-(x
+D))/25e-3)-1)','options',fo);

% Extract the parameters
ff_c = fit(V',I',fo_c);
% Generate the curve
If_c = ff_c(V);

% Plot the data with nonlinear fitting case c)
figure(9)
plot(V, I, "-b");
hold on
plot(V, If_c, "--r");
hold off
title("I versus V with nonlinear fitting c)")
xlabel("Voltage (V)")
ylabel("Current (A)")
legend("Current Data", "Non-linear fit")
grid on
```

```
figure(10)
semilogy(V, abs(I), "-b")
hold on
semilogy(V, abs(If_c), "-r")
hold off
title("Semilog |I| versus V with nonlinear fitting c)")
xlabel("Voltage (V)")
ylabel("Current (A)")
legend("Current Data", "Non-linear fit")
grid on
```





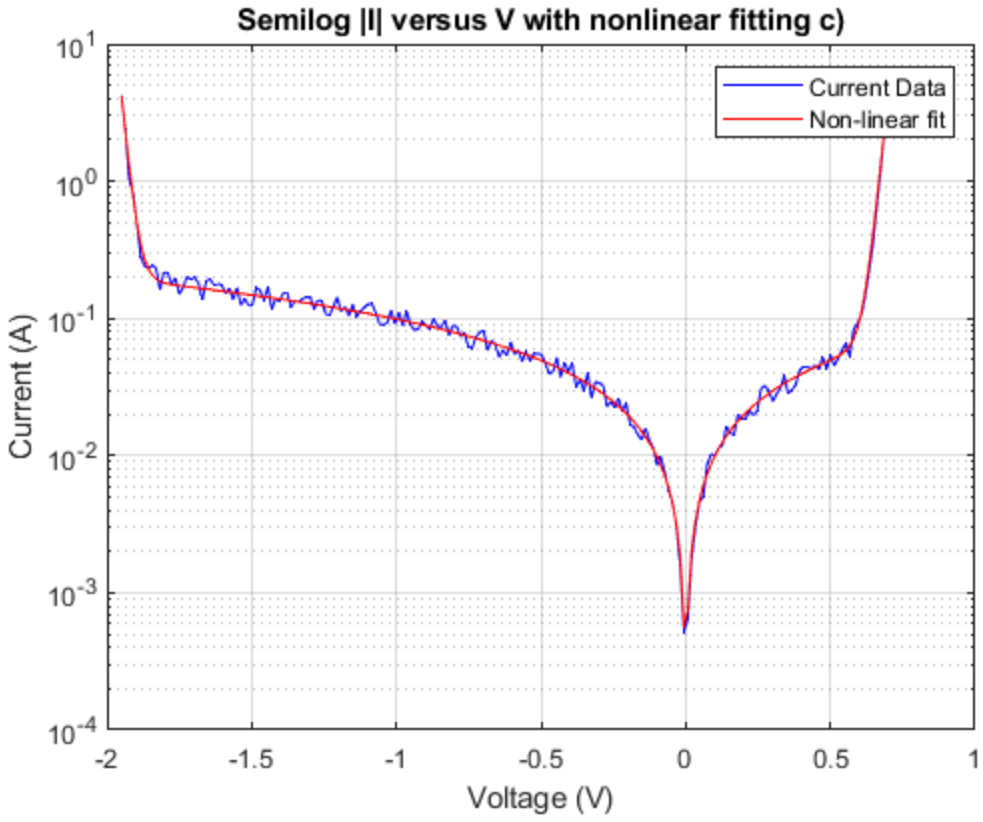
Task 3 Conclusion

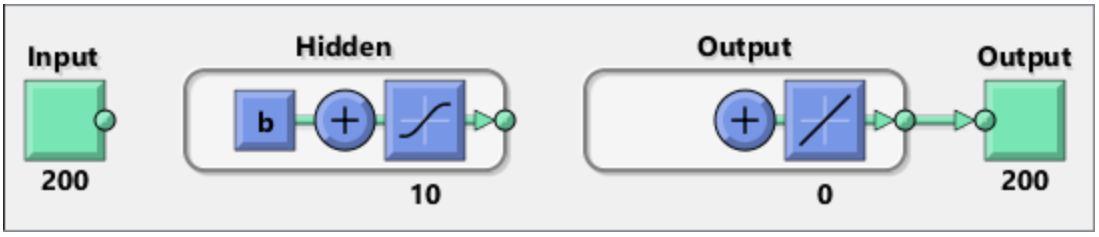
The nonlinear fit in general can fit better than the polynomial fit. The non-linear fit requires the form of the equation to fit. The nonlinear fit can fit the curve very well if the initial guests for the fitting parameters are appropriate. If the initial guests are not appropriate, the nonlinear fit sometimes cannot converge to the curve. If the fitting terms become more, such as the case c) which have 4 terms, the fitting will be very inaccurate if the initial guests are not appropriate. The non-linear fitting are successful in our cases because we already know the actual parameter values, so we can passed in some appropriate initial guests that are relatively close to the "true" value.

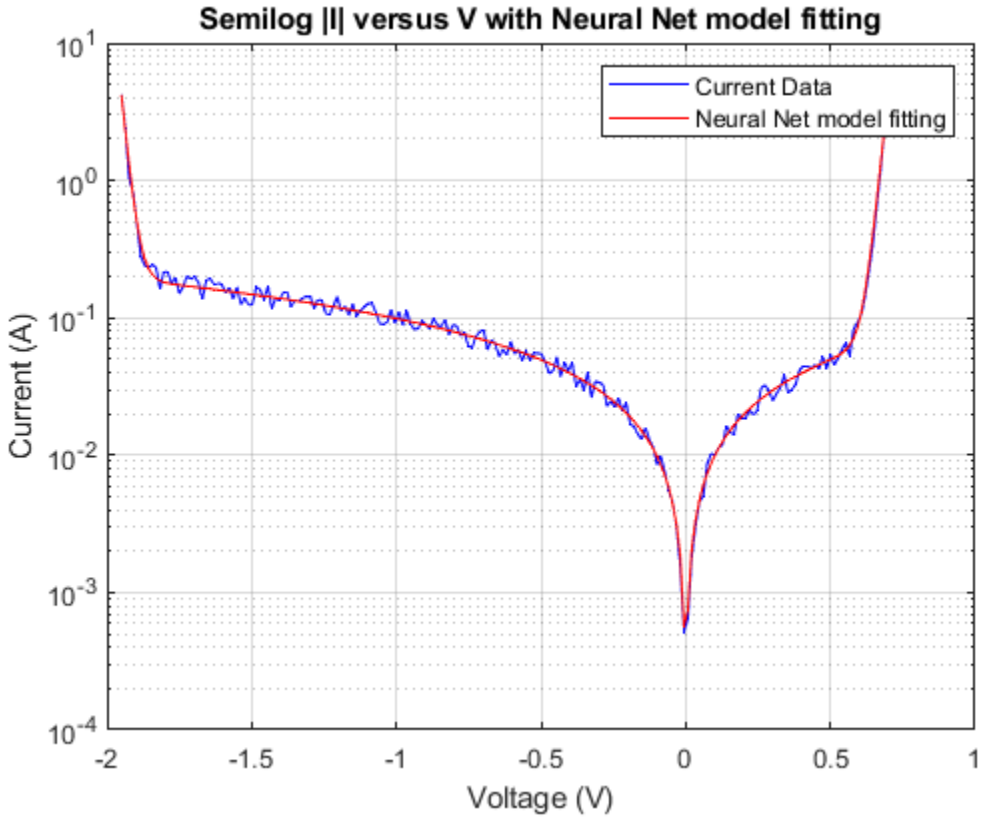
Task 4 - Fitting using the Neural Net model

```
% Code provided from the manual:
inputs = V.';
targets = I.';
hiddenLayerSize = 10;
net = fitnet(hiddenLayerSize);
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;
[net,tr] = train(net,inputs,targets);
outputs = net(inputs);
errors = gsubtract(outputs,targets);
performance = perform(net,targets,outputs);
view(net)
Inn = outputs;
```

```
% Plot the data with Neural Net model fitting
figure(11)
plot(V, I, "-b");
hold on
plot(V, Inn, "--r");
hold off
title("I versus V with Neural Net model fitting")
xlabel("Voltage (V)")
ylabel("Current (A)")
legend("Current Data", "Neural Net model fitting")
grid on

figure(12)
semilogy(V, abs(I), "-b")
hold on
semilogy(V, abs(If_c), "-r")
hold off
title("Semilog |I| versus V with Neural Net model fitting")
xlabel("Voltage (V)")
ylabel("Current (A)")
legend("Current Data", "Neural Net model fitting")
grid on
```







Task 4 Conclusion

The fitting with the Neural Net model fit very well to the curve, and it do not require initial guests such as the non-linear fitting. The Neural Net model fitting fit very well to the data range that it is trained from. If the data range goes beyond the training data range, the output behavior of the neural net model can be very wrong.

Published with MATLAB® R2021b