BIOE 198MI Biomedical Data Analysis. Spring Semester 2019.
Lab 6: Least Squares Curve Fitting Methods

A. Modeling Data

You have a pair of night vision goggles for seeing NIR fluorescent photons emerging from the skin of patients whose lymphatic vessels were injected with a fluorescent dye (see Fig 1). These goggles are necessary because the detector can see light levels below the sensitivity of the human eye. The manufacturer says the output voltage signal is directly proportional to the input light levels with slope one. Great! However, there is additive noise. Before noise suppression, the output signal from the goggles y relative to deterministic input x is modeled simply as: $y = x + n$, where for each x the noise sample n is a time-independent sample drawn from a normal pdf, $\mathcal{N}(0,10)$. (In reality, x is a Poisson random variable, which we ignore in this example.) Assume the units of all three terms is micro candela per square meter (μcd/m^2).

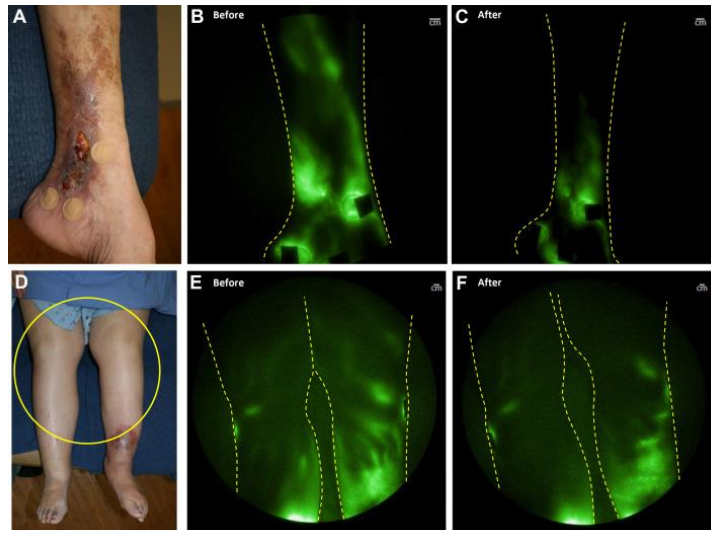


Fig. 1. Images from the lab of Dr. Eva Sevick-Muraca, University of Texas Health Science Center at Houston, TX. Publication: Journal of Vascular Surgery: Venous and Lymphatic Disorders, Volume 4, Issue 1, January 2016, Pages 9-17. Images show lymphatic emptying following sequential pneumatic compression.

Example 1. You obtain one of these devices and decide to experiment in the lab before seeing patients. The experiment is to input a known quantity of light x into the device and measure its response y. The experiment begins with input flux $x(1) = 10\ \mu\text{cd/m}^2$ and is indexed in steps of 10 μcd/m^2, i.e., 10, 20, … , 100 μcd/m^2 while output $\hat{y}(t)$ is recorded. The results are:

Experimental Data:

x =	10.000	20.000	30.000	40.000	50.000	60.000	70.000	80.000	90.000	100.000
$\hat{y}$ =	15.377	38.339	7.412	48.622	53.188	46.923	65.661	83.426	125.784	127.694

Plotting the x,y pairs of recorded measurements, you find the red circles in Fig 2. Compared to the noise free model provided by the manufacturer, $y = x$ (solid black line), the results are not very impressive because of the noise.

The device assumes a linear input-output relationship, so these data can be fit to a linear regression line of the form $z = P_1 x - P_2$ using the method of least squares:

$$MSSD = \underset{P_1,P_2}{\operatorname{argmin}} \sum_{i=1}^{N} d_i^2\,(x) = \underset{P_1,P_2}{\operatorname{argmin}} \sum_{i=1}^{N} (\hat{y}_i - P_1 x_i - P_2)^2\,.$$

This equation tells us to find parameters P_1 and P_2 that yield the minimum sum squared differences ($MSSD$) between each of the $N = 10$ measurements and a straight line. The differences (see Fig 2 below) are defined as

$$d_i \;=\; \hat{y}_i - z_i \;=\; \hat{y}_i - (P_1 x_i - P_2)\,, \qquad \text{for } 1 \le i \le N.$$

For statistical optimization reasons, we use d_i^2 instead of $|d_i|$. Notice that in this example, the least-squares regression line z is not the same as the model function y.

To apply the method of linear least squares, we assume

- y is linearly related to x or a transformation of x
- deviations from the regression line (residuals d_i) are normally distributed random variables $\mathcal{N}(\overline{d_i}, \sigma_i)$
- all variances $\sigma_i{}^2$ are equal.

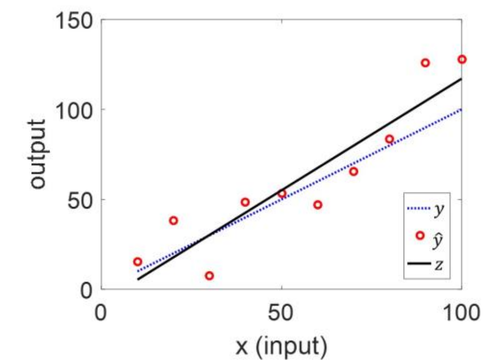
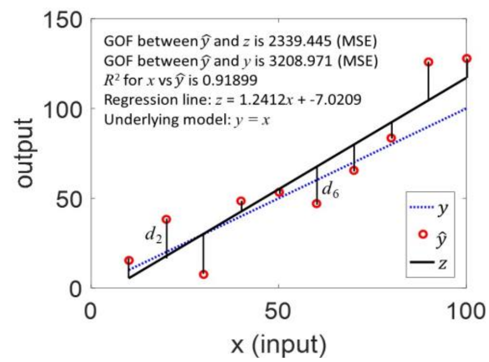


Fig. 2. (Above) Graphical representation of data in Example 1. Least-squares regression line is z while modeled output are labeled y and measured data $\hat{y}$. (Below) The distances d between $\hat{y}$ and z at each x are displayed along with z equation, GOF and R^2 values.



My code for generating these results is (note the new plotting function at the end!)

```
close all
x=10:10:100;rng('default');y=x+10*randn(size(x)); %y=x+n where n~N(0,10)
P=polyfit(x,y,1);        %Apply the method of least squares: P(1),P(2)
z=polyval(P,x);          %Compute the regression line z=P1*x + P2
myplot1(x,x,':b');hold on;myplot1(x,y,'or'),myplot1(x,z,'-k')
xlabel('x (input)');ylabel('output');
legend('x','y','z','Location','SouthEast')
q1=goodnessOfFit(y,z,'MSE');
str1=['GOF between z and y is ' num2str(q1)]; disp(str1);
q2=goodnessOfFit(y,x,'MSE');
str2=['GOF between x and y is ' num2str(q2)]; disp(str2);
%The following correlation coefficient relates x to y.
A2=corrcoef(x,y);str2=['R^2 for x vs y is ' num2str(A2(1,2))]; disp(str2);
str4=['Regression line: y= ' num2str(P(1)) 'x + ' num2str(P(2))]; disp(str4);
%
%
function myplot1(x,y,z)   %I adapted myplot to modify line/point types
%
plot(x,y,num2str(z),'linewidth',2)
ax = gca; % current axes
ax.FontSize = 20;
end
```

Note how I adapted `myplot` from Lab 1b, I called it `myplot1`, to add the ability to control line and point properties.

The Pearson correlation coefficient R^2 is found from either of the two off-diagonal terms obtained from the correlation matrix generated between x and y by the Matlab function `corrcoef(x,y);`

$$R^2 = \frac{square\ of\ sample\ covariance}{product\ of\ two\ sample\ variances} = \frac{s_{x\hat{y}}^2}{s_x^2 s_{\hat{y}}^2} = \frac{\left(\sum_{i=1}^N (x_i-\bar{x})\,(\hat{y}_i-\bar{\hat{y}})\right)^2}{\sum_{i=1}^N (x_i-\bar{x})^2 \sum_{i=1}^N (\hat{y}_i-\bar{\hat{y}})^2}$$

$$= \frac{\left[\sum_{i=1}^N \left(x_i\hat{y}_i - x_i\frac{1}{N}\sum_{j=1}^N \hat{y}_j - \hat{y}_i\frac{1}{N}\sum_{j=1}^N x_j + \frac{1}{N^2}\sum_{j=1}^N x_j \sum_{j=1}^N \hat{y}_j\right)\right]^2}{\left[\sum_{i=1}^N \left(x_i^2 - 2x_i\frac{1}{N}\sum_{j=1}^N x_j + \left(\frac{1}{N}\sum_{j=1}^N x_j\right)^2\right)\right]\left[\sum_{i=1}^N \left(\hat{y}_i^2 - 2\hat{y}_i\frac{1}{N}\sum_{j=1}^N \hat{y}_j + \left(\frac{1}{N}\sum_{j=1}^N \hat{y}_j\right)^2\right)\right]}$$

$$= \frac{\left[\sum_{i=1}^N x_i\hat{y}_i - \sum_{i=1}^N x_i\frac{1}{N}\sum_{j=1}^N \hat{y}_j - \sum_{i=1}^N \hat{y}_i\frac{1}{N}\sum_{j=1}^N x_j + \frac{1}{N^2}\sum_{i=1}^N\left(\sum_{j=1}^N x_j \sum_{j=1}^N \hat{y}_j\right)\right]^2}{\left[\sum_{i=1}^N x_i^2 - \frac{2}{N}\left(\sum_{i=1}^N x_i\right)\left(\sum_{j=1}^N x_j\right) + N\left(\frac{1}{N}\sum_{j=1}^N x_j\right)^2\right]\left[\sum_{i=1}^N \hat{y}_i^2 - \frac{2}{N}\left(\sum_{i=1}^N \hat{y}_i\right)\left(\sum_{j=1}^N \hat{y}_j\right) + N\left(\frac{1}{N}\sum_{j=1}^N \hat{y}_j\right)^2\right]}$$

$$= \frac{\left[\sum_{i=1}^N x_i\hat{y}_i - \frac{1}{N}\sum_{i=1}^N x_i\sum_{i=1}^N \hat{y}_i - \frac{1}{N}\sum_{i=1}^N \hat{y}_i\sum_{i=1}^N x_i + \frac{N}{N^2}\sum_{i=1}^N x_i \sum_{i=1}^N \hat{y}_i\right]^2}{\left[\sum_{i=1}^N x_i^2 - \frac{2}{N}\left(\sum_{i=1}^N x_i\right)\left(\sum_{i=1}^N x_i\right) + N\left(\frac{1}{N}\sum_{i=1}^N x_i\right)^2\right]\left[\sum_{i=1}^N \hat{y}_i^2 - \frac{2}{N}\left(\sum_{i=1}^N \hat{y}_i\right)\left(\sum_{i=1}^N \hat{y}_i\right) + N\left(\frac{1}{N}\sum_{i=1}^N \hat{y}_i\right)^2\right]}$$

$$= \frac{\left[N\sum_{i=1}^N x_i\hat{y}_i - \left(\sum_{i=1}^N x_i\right)\left(\sum_{i=1}^N \hat{y}_i\right)\right]^2}{\left[N\sum_{i=1}^N x_i^2 - \left(\sum_{j=1}^N x_j\right)^2\right]\left[N\sum_{i=1}^N \hat{y}_i^2 - \left(\sum_{j=1}^N \hat{y}_j\right)^2\right]} = \frac{\left[\frac{1}{N}\sum_{i=1}^N x_i\hat{y}_i - \bar{x}\bar{\hat{y}}\right]^2}{\left[\frac{1}{N}\sum_{i=1}^N x_i^2 - \bar{x}^2\right]\left[\frac{1}{N}\sum_{i=1}^N \hat{y}_i^2 - \bar{\hat{y}}^2\right]}$$

Whew! To follow the math, note that $\bar{\hat{y}} = \frac{1}{N}\sum_{i=1}^N \hat{y}_i = \frac{1}{N}\sum_{j=1}^N \hat{y}_j$. Also, $\frac{1}{N}\sum_{i=1}^N\left(\sum_{j=1}^N \hat{y}_j\right) = \sum_{i=1}^N \bar{\hat{y}} = N\bar{\hat{y}}$ and similarly for x.

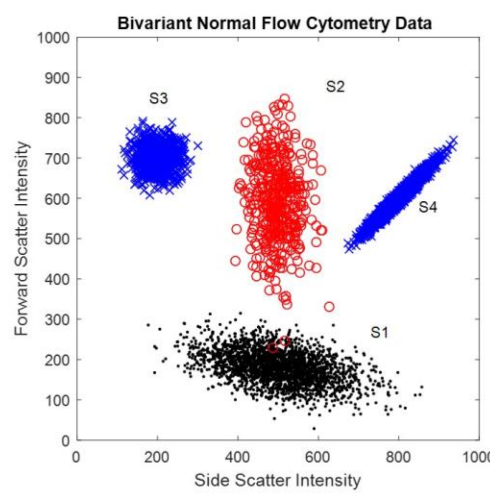


Fig. 3. 2-D scatter plots for two variables with different means, variances, and correlation coefficients. S1: R = -0.40, $s_y < s_x$. S2: R = 0, $s_y > s_x$. S3: R = 0, $s_y = s_x$. S4: R = 0.98, $s_y = s_x$. The code is given at the end of this lab write up.

<u>Pearson's correlation coefficient R^2</u> describes the strength of the linear relationship between two functions or arrays of data (Fig 3). It estimates the fraction of sample variance $s_{\hat{y}}^2$ explained by changes in x. While $0 \le R^2 \le 1$, we also have $-1 \le R \le 1$. For example, when $R \sim 1$, (see S4 in Fig 3 and in Example 1 above where $R \sim 0.96$), we see that variations in x mostly explain variations in $\hat{y}$; increasing one variable by one unit increases the other by approximately one unit and we say the two variables are strongly positively correlated. However, when $R = 0$ (see S2 and S3 in Fig 3), there is no relationship between the two variables.

The downside of correlation is that it tells us nothing about which variable is dependent. We will come back to R and R^2 in a minute.

First, another test statistic of interest is the <u>Goodness of Fit measure, GOF</u>. Using the Matlab function `goodnessOfFit`, I separately tested how well x and z in Fig 2 each represent $\hat{y}$ by selecting mean-square error as the GOF metric: $MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - z_i)^2$. Since regression line z is the $MSSD$ result, it make sense that z will fit data $\hat{y}$ even better than the true underlying model y. If you do not have the `goodnessOfFit` function, you can always compute MSE using the equation above in a function you call.

Summary: Correlation R is useful for quantifying the existence of relationships between variables, but it cannot establish causal relationships. That is, there is no way to tell if one variable is "causing" the response observed by the other; i.e., is $\hat{y}(x)$ true or is $x(\hat{y})$ true? We prefer R^2 over R when correlation is used to explain variance in the data. However, we use R to describe scatter plots (Fig 3) because it tells us if the correlation between variables is negative or positive (compare S1 and S4 in Fig 3). While R describes correlation between x and $\hat{y}$, GOF describe how well $\hat{y}$ is described by linear regression line z.

Exercise. Look at the line of code below. It generates random data in three columns. The first two columns are "uncorrelated" random data, $R \sim 0$, while the last two are "perfectly correlated" random data, $R = 1$. If the correlation coefficient between columns i and j is R_{ij}, the resulting correlation matrix from Matlab has elements given by $\begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix}$. What fraction of each random variable can be explained by the others? Why is the matrix symmetric about the diagonal?

```
clear all;Y=zeros(1000,3);Y=randn(1000,2);Y(:,3)=3*Y(:,2);R=corrcoef(Y)

R =

    1.0000    0.0006    0.0006
    0.0006    1.0000    1.0000
    0.0006    1.0000    1.0000
```

Assignment 1:

(a) Obtain the file `lab6.mat` that contains three variables. `t` is a time axis while `y1` and `y2` are data arrays such that $y_1(t)$ and $y_2(t)$. Plot `y1` and `y2` versus `t`.

(b) Use `polyfit` and `polyval` to generate a set of polynomial fits for data, `y1` and `y2`. That is, compute nine fitted functions for a zeroth-order polynomial through an eighth-order polynomial. Specifically, use a `for` loop for `m=0:8`, where `m` is the polynomial order, to find nine pairs `z1` and `z2` from `polyval` after running `polyfit`.

(c) Within the `for` loop, compute a goodness-of-fit measure MSE for each value of `m` by programming

$$MSE_1 \triangleq \frac{1}{N}\sum_{n=1}^{N}(y_{1,n} - z_{1,n})^2 \text{ and } MSE_2 \triangleq \frac{1}{N}\sum_{n=1}^{N}(y_{2,n} - z_{2,n})^2 .$$

Do this by writing a function called `meanse` that you call from the `for` loop. Here's is a start…

```
function [output] = meanse(input1,input2)     % Use this for the goodness-of-fit metric
%
output = {something goes here}        %compute the mean-squared error between input1 and input2
end
%
```

(d) Plot z-function outputs on top of the data and decide which order of polynomial fit m best represents both sets of data. To do this, generate a section than uses the input function to request keyboard command to input a value between 0 and 8 before plots are generated.
For example: `x=input('Order of polynomial to be plotted: ');`

Report three plots: (a) Plot both MSE_1 and MSE_2 versus model order for m=0:8. (b) y1 versus t and z1 versus t on the same plot for your selection of the best m. (b) On the third plot show y2 versus t and z2 versus t together for the same value of m as y1.

Extra Credit: The function being fit is `y1 = 1-exp(-t/2) + n1` where n is normally-distributed noise. Try one extra fitting procedure as follows: Fit `yy1 = log(1-y1)` to a linear function using

```
clear P;P=polyfit(t,yy1,1);zz1=real(polyval(P,t));
```

Also plot t versus `1-exp(zz1)` on top of the y1 data. Repeat the whole process for y2. Explain what you find.

Rubric:

1 point for describing the problem in the **Introduction** of the report.

1 point for programming the MSE function that is called in the for loop.

2 points for explaining in the **Methods** section how you solved this fitting problem.

1 point for coding keyboard entry of m values.

1 point for using a myplot1 type function.

2 points for overall appearance and clarity of the report including the plots in the **Results** section.

1 point for **Discussion** explaining your choice of m.

1 point for trying the extra credit and 2 more points for achieving and explaining the extra credit part.

```
%%%%%%%%%%%% Code that generated Figure 3 scatter plots. Not an assignment! %%%%%%%%%
% The following script simulates four flow cytometry data sets that are
% each bivariate normal.  Parameter vectors vary between data sets.
%
clear all; close all;
rho=-0.4;s1=[100 40];m1=[500 180];N1=50; %First data set
z=randn(N1); %N1^2 is the number of data points simulated
X1=s1(1)*z+m1(1);       %X1 and Y1 convert from standard normal pdf
Y1=s1(2)*(rho*z+sqrt(1-rho^2)*randn(N1))+m1(2);
plot(X1,Y1,'k.');axis([0 1000 0 1000]);axis square; %plot on fixed axis
text(730,270,'S1');hold on  %label the first group S1
xlabel('Side Scatter Intensity');ylabel('Forward Scatter Intensity')
%
rho=0;s2=[40 100];m2=[500 600];N2=20;   %Second data set
z=randn(N2);
X2=s2(1)*z+m2(1);
Y2=s2(2)*(rho*z+sqrt(1-rho^2)*randn(N2))+m2(2);
plot(X2,Y2,'ro');text(620,880,'S2')
%
rho=0;s3=[30 30];m3=[200 700];N3=30;    %Third data set
z=randn(N3);
X3=s3(1)*z+m3(1);
Y3=s3(2)*(rho*z+sqrt(1-rho^2)*randn(N3))+m3(2);
plot(X3,Y3,'bx')
text(180,850,'S3')
%
rho=0.98;s4=[40 40];m4=[800 600];N4=50; %Fourth data set
z=randn(N4);
X4=s4(1)*z+m4(1);
Y4=s4(2)*(rho*z+sqrt(1-rho^2)*randn(N4))+m4(2);
plot(X4,Y4,'bx');hold off
title('Bivariant Normal Flow Cytometry Data');
xlabel('Side Scatter Intensity');ylabel('Forward Scatter Intensity')
text(850,580,'S4');hold off
% save('FN1','X1','Y1','X2','Y2','X3','Y3') %use these for the assignment
%load('FN1.mat'); %then use whos command to see what is loaded
%
[rho1]=corr(X1,Y1);R1=trace(rho1)/N1;       %Here we estimate Pearson's
str = ['rho_1 = ' num2str(R1)];disp(str)     %correlation and average for
[rho2]=corr(X2,Y2);R2=trace(rho2)/N2;       %all points along diagonal
str = ['rho_2 = ' num2str(R2)];disp(str)
[rho3]=corr(X3,Y3);R3=trace(rho3)/N3;
str = ['rho_3 = ' num2str(R3)];disp(str)
[rho4]=corr(X4,Y4);R4=trace(rho4)/N4;
str = ['rho_4 = ' num2str(R4)];disp(str)
%
```