BIOE 198MI Biomedical Data Analysis. Spring Semester 2020.
Lab 5: Least Squares Curve Fitting and Data Modeling

A. Modeling Data

You have a pair of night vision goggles for seeing near infrared (NIR) fluorescent photons emerging from the skin of patients whose lymphatic vessels were injected with a fluorescent dye (see Fig 1). You can see the skin lesions that appear with this lymphatic disorder. Goggles are necessary to see light levels below the sensitivity of the human eye. *Fictitious Instruments Inc.* is a manufacturer who reports the output voltage signal is directly proportional to the input light levels with slope one. Great! We have a true "linear system." However, there is always the annoying additive noise.

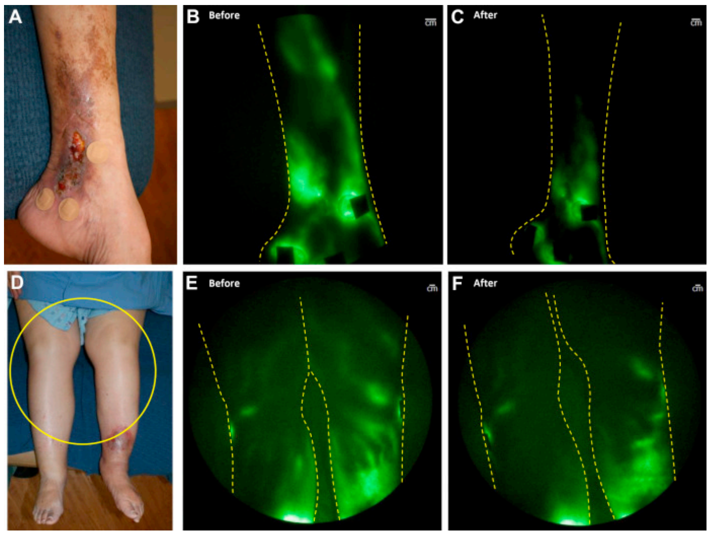


Fig. 1. Images from the lab of Dr. Eva Sevick-Muraca, University of Texas Health Science Center at Houston, TX. Publication: Journal of Vascular Surgery: Venous and Lymphatic Disorders, Volume 4, Issue 1, January 2016, Pages 9-17. Images show lymphatic emptying following sequential pneumatic compression.

Before noise suppression, the output signal from the goggles y relative to deterministic input x is modeled simply as: $y = x + n$, where at each input *light flux* value x the noise sample n is a time-independent sample, $\mathcal{N}(0,10)$. (Although x is a Poisson random variable, it is well approximated as a normal random variable in this example.) The units of flux for three terms is micro candela per square meter (μcd/m^2).

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 function provided by the manufacturer, $y = x$ (dotted blue line), the results are not impressive because of noise.

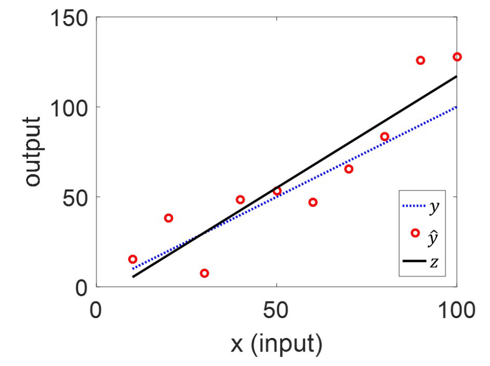
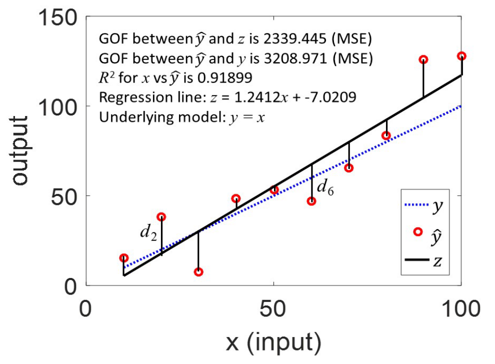


Fig. 2. (Above) Graphical representation of data in Example 1. Least-squares regression line is z while modeled output are labeled y and measured data $\hat{y}$. (Below) The distances d between $\hat{y}$ and z at each x are displayed along with z equation, GOF and R^2 values.



The device assumes a linear input-output relationship, so these data are fit to a linear regression curve of the form $z = P_1x + P_2$ using the method of least squares. It instructs us to search for parameters P_1 and P_2 that minimize the sum squared differences ($MSSD$) between each of the $N = 10$ measurements and a straight line.

$$MSSD = \underset{P_1,P_2}{\operatorname{argmin}} \sum_{i=1}^{N} (\hat{y}_i - z_i(P_1, P_2))^2 = \underset{P_1,P_2}{\operatorname{argmin}} \sum_{i=1}^{N} (\hat{y}_i - P_1x_i - P_2)^2$$

$$= \underset{P_1,P_2}{\operatorname{argmin}} \sum_{i=1}^{N} d_i^2\,(x),$$

where differences (Fig 2) are defined as

$$d_i \;=\; \hat{y}_i - z_i \;=\; \hat{y}_i - (P_1x_i + P_2)\,, \qquad \text{for } 1 \le i \le N.$$

For statistical optimization, we use d_i^2 instead of $|d_i|$. Notice that in this example, the least-squares regression curve, z is not the same as the true output function y. Consequently, regression curves are limited by bias and precision errors.

To apply the method of linear least squares, we must assume

- y is linearly related to x or a transformation of x
- deviations from regression line z (the residuals d_i) are normally distributed $\mathcal{N}(\bar{d}_i, \sigma_i)$
- variances $\sigma_i{}^2$ are assumed to be equal at each input x_i. (stationary random process)

My code for generating these results is

```
close all
x=10:10:100;rng('default');yh=x+10*randn(size(x)); % yh=x+n and n~N(0,10)
P=polyfit(x,yh,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,yh,'ro'),myplot1(x,z,'k')
xlabel('x (input)');ylabel('output');
legend('x','yh','z','Location','SouthEast')
q1=goodnessOfFit(yh,z,'MSE');  % Remember: MSE=sum_N(yh(i)-mu)^2
str1=['MSE between z and yh is ' num2str(q1)]; disp(str1);
q2=goodnessOfFit(yh,x,'MSE');
str2=['MSE between x and yh is ' num2str(q2)]; disp(str2);
% The following correlation coefficient relates x to y.
A2=corrcoef(x,yh);str2=['R^2 for x vs y is ' num2str(A2(1,2))]; disp(str2);
str4=['Regression line: z= ' num2str(P(1)) 'x + ' num2str(P(2))]; disp(str4);
%
% Note:  the degree of a polynomial is given by the highest power of x.
```

For example in Fig. 2, we have a first-order polynomial: `polyfit` gives `P = [1.24 -7.02]` and `polyval` generates $z(x) = 1.24\,x - 7.02$.

The equation $z(x) = 3x^4 - 2x^2 + 4$ is a fourth-order polynomial equation, where we find `P = [3 0 -2 0 4]`, with length `N+1`.

`polyfit` and `polyval` are used together to generate regression curve z. Mathematically, $z = P_1x^2 + P_2x + P_3$ for *N* = 2 (quadratic model) and $z = P_1x + P_2$ for *N* = 1 (linear model).

Examples:

```
m=-2;b=2;x=0:10;yh=m*x+b;

P=polyfit(x,yh,1);z=polyval(P,x);

myplot1(x,yh,'ko');hold on;myplot1(x,z,'r:');hold off
```

Now if you type `P` in the command window you see `P= [-2 2]` and that matches `m=-2, b=2`. Without noise, the match is perfect! Let's try a third-order polynomial, that fits four parameters in the model:

```
x=0:10;yh=(x.^3)/1000-4;

P=polyfit(x,yh,4);z=polyval(P,x);

myplot1(x,yh,'ko');hold on;myplot1(x,z,'r:');hold off
```

Again, the fit is perfect because I am living in a fantasy world without noise. Type `P` into the command window and be sure you can interpret the output

$$\begin{array}{cccccc} & x^4 & x^3 & x^2 & x^1 & x^0 \\ \texttt{P = [} & 0.0000 & 0.0010 & -0.0000 & 0.0000 & -4.0000\texttt{]} \end{array}$$

When you take BIOE 210, you will discover that Nth order polynomial equations form an orthogonal basis for representing many, many functions. Regression analysis is a great modeling tool.

Describing the Quality of a Fit

First, let's discuss two basic statistical analysis metrics: correlation coefficient and MSE.

The Pearson correlation coefficient R^2 between x and y is defined as

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

Note that $\bar{\hat{y}} = \frac{1}{N}\sum_{i=1}^{N}\hat{y}_i$, and R^2 has no units. Also, $\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 = \frac{1}{N}\left[\sum_{i=1}^{N} {x_i}^2\right] - \bar{x}^2$.

(1) Pearson's correlation coefficient R^2 describes the strength of the linear relationship between two functions or arrays of data. In Fig 3, the two **idealized** variables are SCI and FSI from flow cytometry. R^2 estimates the fraction of sample variance $s_{\hat{y}}^2$ explained by changes in x.

While $0 \leq R^2 \leq 1$, we also have $-1 \leq R \leq 1$. For example, $R \sim 1$ for S4 in Fig 3 where $R^2 \sim 0.96$ and $R \sim 0.98$ (and in Fig. 2 above). In both cases, variations in the x-axis measurement mostly explain variations seen in y-axis measurements. When increasing x by one unit increases y by approximately one unit; we say the two variables are strongly positively correlated. S1 is weakly negatively correlated. When $R = 0$ (see S2 and S3 in Fig. 3), there is no relationship between the two variables: they are uncorrelated.

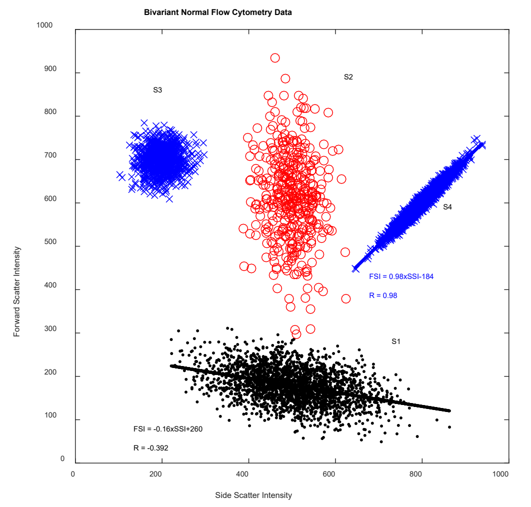


Fig. 3. 2-D scatter plots for two variables with different means, variances, and correlation coefficients. S1: $R = -0.40$, $s_y < s_x$, showing weak correlation between SSI and FSI. S2: $R = 0$, $s_y > s_x$ and S3: $R = 0$, $s_y = s_x$; both are uncorrelated. S4: $R = 0.98$, $s_y = s_x$, showing strong correlation between SSI and FSI. Code is given at the end. The results of linear regression for S1 and S4 are shown as lines.

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.

(2) Another test statistic of interest is the Goodness of Fit measure, GOF. 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. In this case, we are overfitting the data. ****The `goodnessOfFit` function was not included in recent Matlab releases, You will be asked to compute MSE using the equation above in a function you call from the main script. ****

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. Mathematically, is $\hat{y}(x)$ true or is $x(\hat{y})$? We prefer R^2 over R when correlation is used to explain variance in the data. Variance is a more fundamental quantity than standard deviation. However, we still 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 (like MSE) describe how well $\hat{y}$ is described by linear regression line z.

Run the code that generates Fig. 3 at the end of this lab write. You will see a few things. I estimated correlation from the data so I could compare with values input into the simulation. You can see estimates closely agree with input values!

I performed linear regression on S1 and S4 data and plotted the result in Fig. 3. If you go to the command line and type `P1` or `P4`, the first number that appears is the slope of the line. The slope for S1 data is -0.16 and the slope of the line fit to S4 is 0.98. Slopes are not the correlation coefficients!

To find R for these fits, use `A=corrcoeff(X,Yh)` that generates a 2x2 array with 1.0 on the diagonal elements and correlation coefficient R on both off-diagonal elements, i.e., `R = A(1,2) = A(2,1)`.

For X1 and Y1 in Fig. 3,

```
A=corrcoef(X1,Y1)

A =

    1.0000   -0.4258

   -0.4258    1.0000
```

What do you think will happen when S2 or S3 are fit to a line? (hint: R_2 = -0.095 and R_3 = 0.056).

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 for Lab 5:

(a) Obtain the file `lab5.mat` (**instead, see code on the next page that generates the data**) that contains three variables. `t` is a time axis while `yh1` and `yh2` are data arrays such that $\hat{y}_1(t)$ and $\hat{y}_2(t)$. Plot `yh1` and `yh2` versus `t`. Note that `t` is the independent variable for both `yh1` and `yh2`. Each of the three arrays has $N = 101$ points.

(b) Generate 9 polynomial fits for each of the two data sets, `yh1` and `yh2`. Fit a zeroth-order polynomial (m=0) through an eighth-order polynomial (m=8), generating 9 fit curves each for `z1` and `z2`.

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

$$MSE_1 \triangleq \frac{1}{N}\sum_{n=1}^{N}(\hat{y}_{1,n} - z_{1,n})^2, \quad MSE_2 \triangleq \frac{1}{N}\sum_{n=1}^{N}(\hat{y}_{2,n} - z_{2,n})^2 .$$

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

```
function [output] = meanse(input1,input2)
    output = {something goes here}   %MSE(input1,input2)
end
```

(d) Plot the two MSE versus `m` curves and use those two curves to select a value of `m` that you consider to provide the "best" fit. Try candidate values of `m` by having your main program ask the user to select one value of `m` from the keyboard. Use the `input` function to enter m values between 0 and 8 from the command window.

(e) Plot the "best" `z`-function regression curves on top of the `yh1` and `yh2` data.

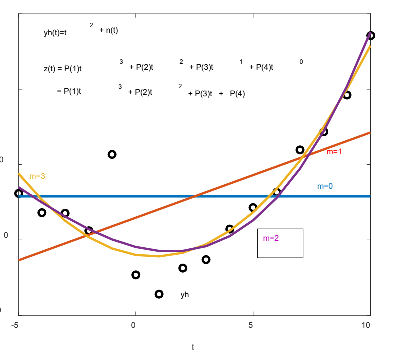


Fig. 4. Illustration of a solution.

```
t=-5:10;yh=t.^2+20*randn(size(t));
myplot1(t,yh,'ko');hold on;
m=0:3;
for j=1:length(m)
    P=polyfit(t,yh,m(j));
    z=polyval(P,t);
    myplot1(t,z,' ')
end
```

(f) <u>Another approach:</u> The true function is `yh1 = 1-exp(-t/2) + n1` where `n1` is normally-distributed noise. Try one extra fitting procedure as follows: Fit `yy1 = log(1-yh1)` to a linear function using `clear P;P=polyfit(t,yy1,1);zz1=real(polyval(P,t));` (Note that `log` in Matlab is $\ln$)
In a separate window, plot `1-exp(zz1)` versus `t` on top of the `yh1` data. Repeat the process for `yh2`.

*** Note that $\exp(x) = 1 + x + \frac{x^2}{2} + \frac{x^3}{6}$ ... This new method $e^x \to \ln(e^x) \to linear\ fit \to z \to e^z$. ***

<u>Report five plots</u>: (1) Plot in one window both MSE$_1$ and MSE$_2$ versus model order for `m=0:8`. (2) `yh1` versus `t` and `z1` versus `t` on the same plot for your selection of the best `m`. (3) On the third plot show `yh2` versus `t` and `z2` versus `t` together for the same value of `m` as `y1`. (4) and (5) Plot the data generated in part (f) above that are the counterparts to plots (2) and (3).

<u>Explain what you find.</u> You need to know that having a complex signal function without noise requires higher-order polynomial fits. However, when noise is present, selecting a large-order polynomial will lead to <u>over fitting</u>, where the model tries to fit noise. Balancing the two aspects needs to be addressed to select the "best" fit model.

<u>Rubric:</u>
1 point for describing the problem in the **Introduction** of the report.
1 point for programming MSE as a <u>separate function</u> that is called within a `for` loop from the main script.
2 points for explaining in the **Methods** section to two methods for fitting these data.
1 point for successfully coding keyboard entry of `m` values. Describe in Methods.
1 point for the appearance of the 5 plots. Make them look good and use captions.
2 points for the clarity of writing in the report. There is much to discuss in this lab but be concise.
1 point for **Conclusions** explaining your choice of `m` and the trade offs that must be considered.

```
% Code below generates data used in the assignment
%
%% Generate data here (lab 5 assignment)
%
t=0:0.1:10;a=0.5;x=1-exp(-a*t);rng('default');
y1=x+0.2*randn(size(t));plot(t,y1,'.')%plot(t,x,t,y1,'.')
y2=x+0.2/sqrt(10)*randn(size(t));figure;plot(t,y2,'.')
save('lab5','t','y1','y2');
%
```

```
%%%%%%%%%%%% 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); % X1, Y1 are statistically independent
P1=polyfit(X1,Y1,1);z=polyval(P1,X1);           % Fit Y1 to X1
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')
plot(X1,z,'k','linewidth',2)                   % Plot the fit data as z
%
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')
title('Bivariant Normal Flow Cytometry Data');
xlabel('Side Scatter Intensity');ylabel('Forward Scatter Intensity')
P4=polyfit(X4,Y4,1);z=polyval(P4,X4);plot(X4,z,'b','linewidth',2) % Fit & plot X4,Y4
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)
```