Salinity Calibration fit with MATLAB

EAS 199B Notes

Gerald Recktenwald
Portland State University
Department of Mechanical Engineering
gerry@me.pdx.edu

Overview

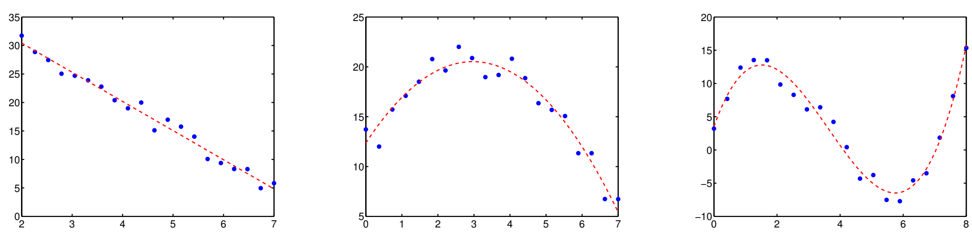
These slides are divided into three main parts

1. A review of least squares curve fitting
2. An introduction to least squares curve fitting with MATLAB
3. Application of least squares fitting to calibration of the salinity sensor

1. Review of Least Squares Curve Fitting

Introduction

Recall curve fitting notes from EAS 199A



Basic Idea

- Given data set (x_i, y_i), $i = 1, \ldots, n$
- Find a function $y = f(x)$ that is *close* to the data

The least squares process avoids guesswork.

Some sample data

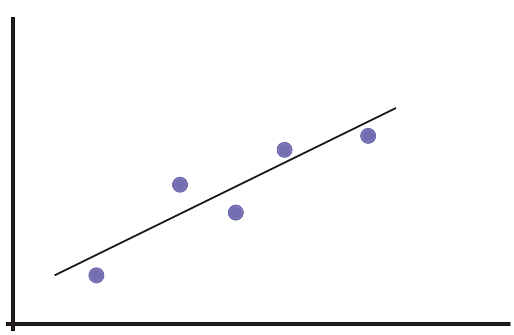
x (time)	y (velocity)
1	9
2	21
3	28
4	41
5	47

It is aways important to visualize your data.
You should be able to plot this data by hand.

- Compute slope and intercept in a way that minimizes an error (to be defined).
- Use calculus or linear algebra to derive equations for m and b.
- There is only one slope and intercept for a given set of data that satisfies the least squares criteria.

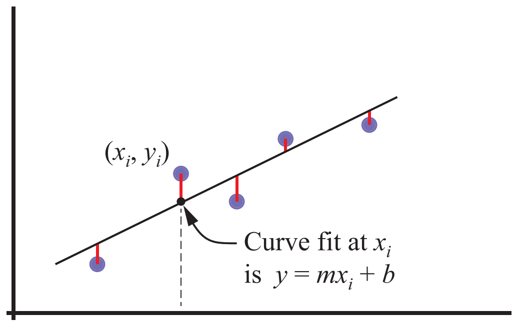
Do not guess m and b! Use least squares!

Least Squares: The Basic Idea



The best fit line goes near the data, but not through them.

Least Squares: The Basic Idea



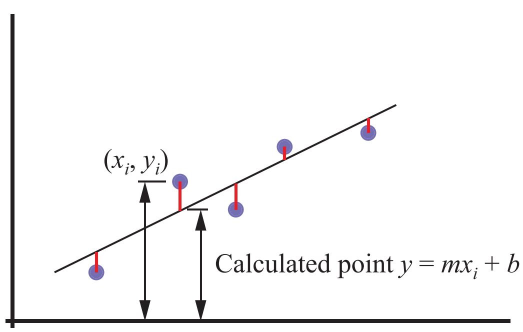
The best fit line goes near the data, but not through them.

The equation of the line is

$$y = mx + b$$

The data (x_i, y_i) are known.
m and b are unknown.

Least Squares: The Basic Idea



The discrepancy between the known data and the unknown fit function is taken as the *vertical distance*

$$y_i - (mx_i + b)$$

The error can be positive or negative, so we use the *square of the error*

$$[y_i - (mx_i + b)]^2$$

Least Squares Computational Formula

Use calculus to *minimize the sum of squares* of the errors

$$\text{Total error in the fit} = \sum_{i=1}^{n} \left[y_i - (mx_i + b)\right]^2$$

Minimizing the total error with respect to the two parameters m and b gives

$$m = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum x_i^2 - \left(\sum x_i\right)^2} \qquad b = \frac{\sum y_i - m \sum x_i}{n}$$

Notice that b depends on m, so solve for m first.

The R^2 Statistic

R^2 is a measure of how well the fit function follows the trend in the data. $0 \leq R^2 \leq 1$.

Define:

$\hat{y}$ is the value of the fit function at the known data points.

For a line fit $\quad \hat{y}_i = c_1 x_i + c_2$

Then:

$$R^2 = \frac{\sum (\hat{y}_i - \bar{y})^2}{\sum (y_i - \bar{y})^2} = 1 - \frac{\sum (y_i - \hat{y})^2}{\sum (y_i - \bar{y})^2}$$

When $R^2 \approx 1$ the fit function follows the trend of the data.
When $R^2 \approx 0$ the fit is not significantly better than approximating the data by its mean.

2. Introduction to least squares curve fitting with MATLAB

Least Squares Fitting with MATLAB

Built-in functions

`polyfit` performs a polynomial curve fit and returns coefficients in a vector

```
c = polyfit(xdata,ydata,n)
```

`polyval` evaluates a polynomial curve fit and returns coefficients in a vector

```
xfit = linspace(min(xdata),max(xdata);
yfit = polyval(c,xfit);
```

GWR function `expfit` performs a linearized curve fit to $y = c_1 e^{c_2 x}$

```
c = expfit(xdata,ydata)
```

`powfit` performs a linearized curve fit to $y = c_1 x^{c_2}$

```
c = powfit(xdata,ydata)
```

Polynomial Curve Fits with `polyfit` (1)

Syntax:

```
c = polyfit(x,y,n)
[c,S] = polyfit(x,y,n)
```

x and y define the data
n is the desired degree of the polynomial.

c is a vector of polynomial coefficients stored in order of *descending powers* of x

$$p(x) = c_1 x^n + c_2 x^{n-1} + \cdots + c_n x + c_{n+1}$$

S is an optional return argument for `polyfit`. S is used as input to `polyval`

Polynomial Curve Fits with `polyfit` (2)

Evaluate the polynomial with `polyval`

Syntax:

```
yf = polyval(c,xf)
[yf,dy] = polyval(c,xf,S)
```

c contains the coefficients of the polynomial (returned by `polyfit`)

xf is a scalar or vector of x values at which the polynomial is to be evaluated

yf is a scalar or vector of values of the polynomials: $yf = p(xf)$.

If S is given as an optional input to `polyval`, then dy is a vector of estimates of the uncertainty in yf

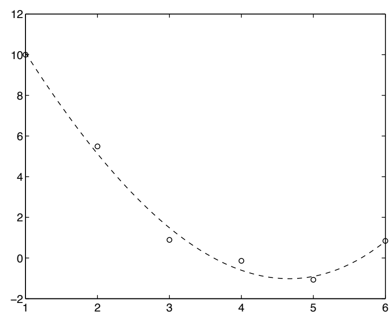
Example: Polynomial Curve Fit (1)

Fit a polynomial to Consider fitting a curve to the following data.

x	1	2	3	4	5	6
y	10	5.49	0.89	-0.14	-1.07	0.84

In MATLAB:

```
>> x = 1:6;
>> y = [10  5.49  0.89  -0.14  -1.07  0.84];
>> c = polyfit(x,y,3);
>> xfit = linspace(min(x),max(x));
>> yfit = polyval(c,xfit);
>> plot(x,y,'o',xfit,yfit,'--')
```



Fitting Transformed Non-linear Functions (1)

- Some nonlinear fit functions $y = F(x)$ can be transformed to an equation of the form $v = \alpha u + \beta$
- perform a linear least squares fit on the transformed variables.
- Parameters of the nonlinear fit function are obtained by transforming back to the original variables.
- The linear least squares fit to the transformed equations does not yield the same fit coefficients as a direct solution to the *nonlinear* least squares problem involving the original fit function.

Examples:

$$y = c_1 e^{c_2 x} \longrightarrow \ln y = \alpha x + \beta$$

$$y = c_1 x^{c_2} \longrightarrow \ln y = \alpha \ln x + \beta$$

$$y = c_1 x e^{c_2 x} \longrightarrow \ln(y/x) = \alpha x + \beta$$

Fitting Transformed Non-linear Functions (2)

Consider

$$y = c_1 e^{c_2 x} \tag{1}$$

Taking the logarithm of both sides yields

$$\ln y = \ln c_1 + c_2 x$$

Introducing the variables

$$v = \ln y \qquad b = \ln c_1 \qquad a = c_2$$

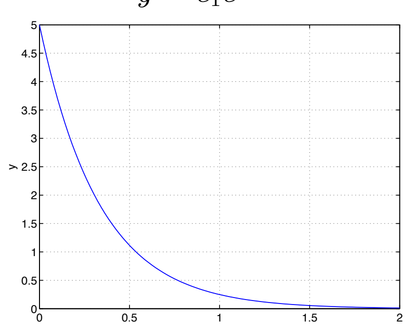
transforms equation (1) to

$$v = ax + b$$

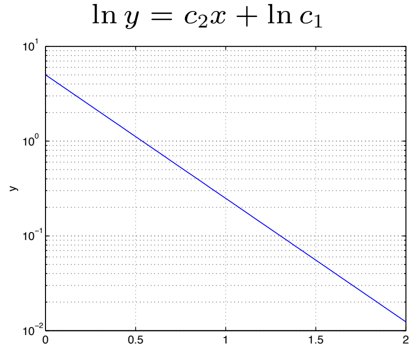
Fitting Transformed Non-linear Functions (3)

The preceding steps are equivalent to graphically obtaining c_1 and c_2 by plotting the data on semilog paper.

$$y = c_1 e^{c_2 x}$$



$$\ln y = c_2 x + \ln c_1$$



Fitting Transformed Non-linear Functions (4)

Consider $y = c_1 x^{c_2}$. Taking the logarithm of both sides yields

$$\ln y = \ln c_1 + c_2 \ln x \tag{2}$$

Introduce the transformed variables

$$v = \ln y \qquad u = \ln x \qquad b = \ln c_1 \qquad a = c_2$$

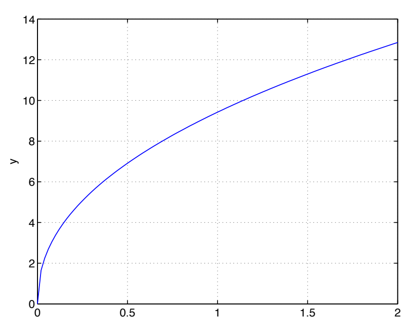
and equation (2) can be written

$$v = au + b$$

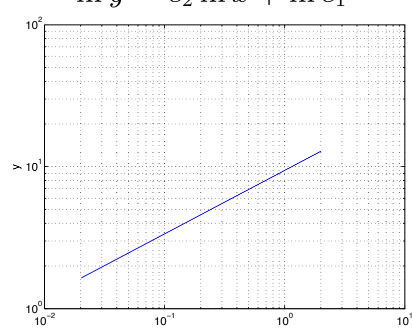
Fitting Transformed Non-linear Functions (5)

The preceding steps are equivalent to graphically obtaining c_1 and c_2 by plotting the data on log-log paper.

$y = c_1 x^{c_2}$



$\ln y = c_2 \ln x + \ln c_1$



3. Application to calibration of the salinity sensor

Matlab code for curve fitting Salinity Sensor Data (1)

The data set is small, so you can enter it manually

```
Sref = [0, 0.05, 0.10, 0.15];          % Calibration reference values
Rout = [ ... ]                         % your raw output

c = polyfit(Sref,Rout,1);              % perform the fit
Sfit = linspace(min(Sref),max(Sref));  % Evaluate the fit
rfit = polyval(c,Rout) - Sref;         % Evaluate the residuals
plot(Rout,rfit,'o')
```

Notes:

- The curve fit may work better if you leave off the `Sref` $= 0$ point
- How do you evaluate R^2

Forward and Reverse Calibration