

## Empirical Model Building Ib: Objectives:

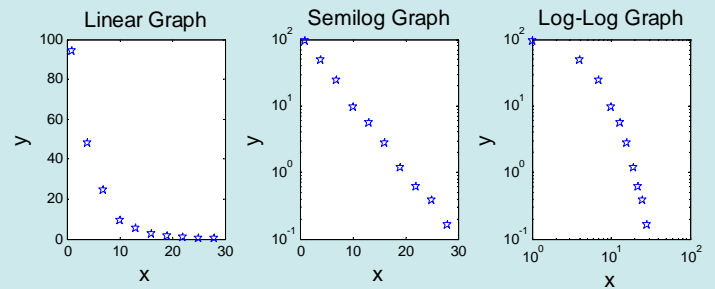
By the end of this class you should be able to:

- Determine the coefficients for any of the basic two parameter models
- Plot the data and resulting fits
- Calculate and describe residuals

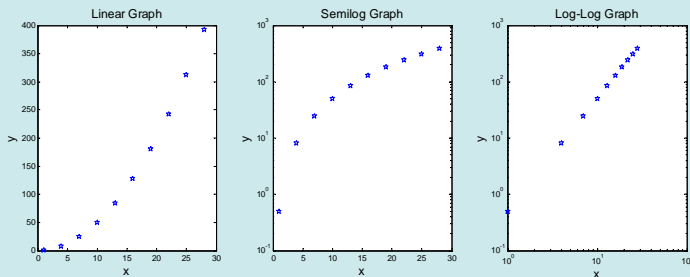
Palm, Section 5.5

Download file [FnDiscovery.mat](#) and load into MATLAB

1. Below are three graphs of the same dataset. What is the name and equation of the likely model that would match this data?



2. Here are the plots for another dataset. Name the model and write its equation for this case

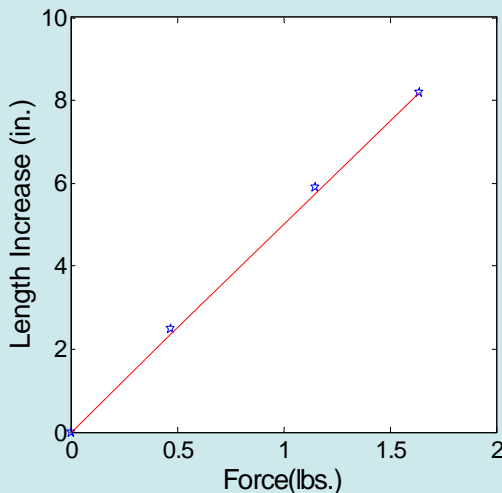


## Two Parameter Models

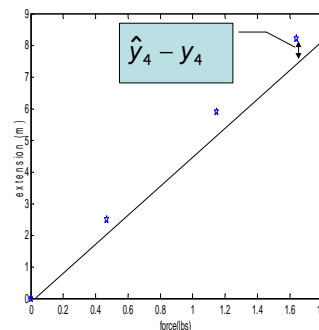
- Three simple models available
- Can "make" the latter two linear by taking logs of the equation
- The resulting equations suggest plots where the data should form a straight line if the model is true.
- What did you get for the three problems?
 

Problem 1: linear	$y = mx + b$
Problem 2: power	$y = bx^m$
Problem 3: exponential	$y = b10^{mx}$
- Note: on problem 3, power is also nearly straight. However, if similar the exponential model is slightly preferred over the power model.

How would you define the Best Fit line?



## Fitting data



- plot data (see points at left)
- looks linear
- equation:  $y = mx + b$
- need to find the "best"  $m$  &  $b$  (the parameters)
- what makes the best line?
- usual approach - minimize the squared distance between the points and the line in the  $y$  direction
- the predicted value is often displayed as  $\hat{y}$  and pronounced "y-hat"
- Calculate the predicted values
- then minimize the sum of squares

$$\sum_{i=1}^n (\hat{y}_i - y_i)^2$$

- next we look at how to do this

## Fitting a Linear equation via matrices

e.g., Fitting the Spring data

- **Model:**  $y = mx + b$
- **Setup:**
  1. Design Matrix: `>> X = [ones(length(Force),1), Force]`
  2. Response Vector `>> Y = Length`
- **Fit:** find the fitted parameters  
`>> B = X \ Y`      B will be  $\begin{bmatrix} b \\ m \end{bmatrix}$
- **Predict:** calculate predicted y for each x  
`>> Lhat = X*B`
- **Plot:** plot the result  
`>> plot(Force, Length, 'p', Force, Lhat)`      (plus labels ...)

## A Linear Model & it's Design Matrix

Linear Model:  $y = b(1) + mx$

Design Matrix:  $X = \begin{bmatrix} 1 & 0 \\ 1 & 0.47 \\ 1 & 1.15 \\ 1 & 1.64 \end{bmatrix}$

**Matlab Syntax:** `>> X = [ones(4, 1), L']`  
 (to convert a row vector of x values to a design matrix)

## Fitting a Linear Equation in Matrix Form

Matrix Equation:

$$Y = X \beta + \epsilon$$

$$\begin{bmatrix} 0 \\ 2.5 \\ 5.9 \\ 8.2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0.47 \\ 1 & 1.15 \\ 1 & 1.64 \end{bmatrix} \begin{bmatrix} b \\ m \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \end{bmatrix}$$

The Full Matrices

MATLAB Syntax for finding the parameter matrix

$$\gg B = X \backslash Y$$

## Linear Equation in Matrix Form

$$\hat{Y} = X \beta$$

$$\begin{bmatrix} 0.08 \\ 2.4 \\ 5.8 \\ 8.3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0.47 \\ 1 & 1.15 \\ 1 & 1.64 \end{bmatrix} \begin{bmatrix} 0.08 \\ 5.00 \end{bmatrix}$$

fits: `>> yhat = X*B`  
 residuals: `>> res = Y - X*B`

## Fitting Transformed models

- Same as linear model except set up design matrix (X) and response vector (Y) using the transformed variables
- e.g., the capacitor discharge from last time
- straight line on a semilog plot
  - what model is implied?
    - Exponential  $y = b10^{mx}$
    - or in this example  $V = b10^{mt}$
  - what is its linearized (transformed) form
    - $\log(V) = \log(b) + mt$

## E.G., Fitting the capacitor discharge data

**Model:** Last lecture we found the data was straight on a semilog plot implying an exponential model.

For the base-ten model the equations are:  
 $V = b10^{mt}$       or       $\log(V) = \log(b) + mt$

**Setup:** 1. Design Matrix: `>> X = [ones(length(t),1), t]`  
 2. Response Vector `>> Y = log10(V)`

**Fit:** determine parameters `>> B = X \ Y`

**Predict:**      Predict: `>> logVhat = X*B`  
                   Untransform: `>> Vhat = 10.^logVhat`

or

Untransform `>> b = 10^B(1), m = B(2)`  
 Predict `>> Vhat = b*10.^(m.*t)`

**Plot:** either on linear or semilog plot

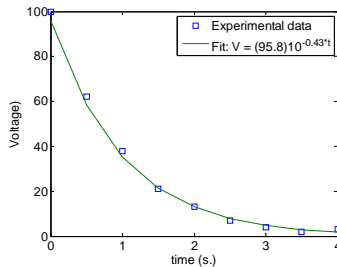
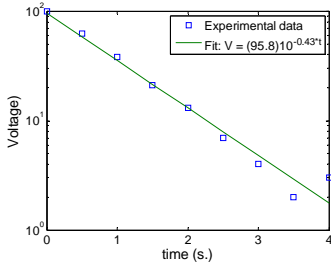
### Plotting the capacitor discharge fit:

#### semilog plot (left below)

```
>> semilogy(t, V, 's', t, Vhat)
>> xlabel('time (s.)', ylabel 'Voltage'
>> legend('Experimental data', ...
'Fit: V = (95.8)10-0.43*t')
```

#### linear plot (right below)

```
>> plot(t, V, 's', t, Vhat)
>> xlabel('time (s.)', ylabel 'Voltage'
>> legend('Experimental data', ...
'Fit: V = (95.8)10-0.43*t')
```



notes:

- linear plot preferred when y axis range is reasonable,
- semilog is used when detail at the low values is required
- I usually set variables m & b equal to the correct values to prevent mistakes
- full labeling required
- notice use of up carrot and curly brackets {} to superscript the powers in the legend
- as always: data is points, equation is a line

### Function Discovery (Review) 2. Fitting Parameters (m & b)

	Equation	Fit	Parameters
linear	$y = mx + b$	x vs. y	b = B(1) m = B(2)
power	$y = bx^m$	log(x) vs. log(y)	b = 10 <sup>B(1)</sup> m = B(2),
exponential	$y = be^{mx}$	x vs. ln(y)	b = e <sup>B(1)</sup> m = B(2),
	$y = b10^{mx}$	x vs. log(y)	b = 10 <sup>B(1)</sup> m = B(2)

### Fitting a 2-parameter models

#### "Normal Data"

##### Model: Identify Functional Form

- Plot data
  - is it linear?
  - is it monotonic?
- Log-Log (loglog(x,y))
- semilog (semilogy(x,y))
- look for straight graph

#### "Transformed Data"

##### Setup:

Transform Data to Linearize  
Create X & Y matrices

##### Fit linear model to transformed data

##### Predict and Untransform

Parameters to m & b

##### Plot:

Plot data and predicted equation.

### Steps in Function Discovery & Fitting

- Step 1: Model** - Use graphs to determine if function is linear, exponential or power. Write down the equation to be fitted
- Step 2: Setup** - create design matrix (X) & response vector (Y) for a model that will be linear (I.E., use transformed data for exponential and power models)
- Step 3: Fit** - Determine parameters for a linear model ( $B = X \setminus Y$ )
- Step 4: Predict** - Untransform: if exponential or power untransform the b coefficient  
Apply model to predict y values
- Step 5: Plot** - data and model - fix any problems encountered.

### Class Exercise:

For problems 3 (x2 vs. y2) from last class:

- What type of model will likely fit this data? (from last time)
- Determine the full model including parameter values.
- Plot the data and the fitted curve on one plot

For problem 2 (x1 vs. y1), repeat the above.

Handout Problem 3: (students to try) uses: x2, y2

Fitting an exponential model:

**Model:**  $y = b \cdot 10^{mx}$  or  $\log(y) = \log(b) + mx$

**Setup:** `>> X = [ones(length(x2),1), x2']; Y = log10(y2');`

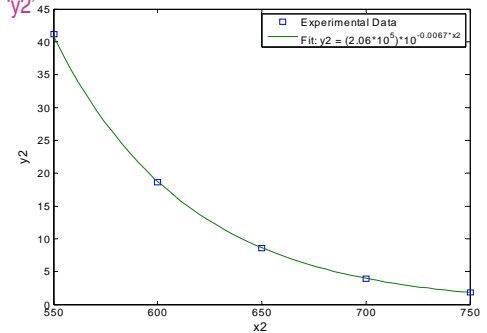
**Fit:** `>> B = X \ Y`  
`B =`  
 5.3143  
 -0.0067

**Predict:** `>> y2hat = 10.^(X*B)`  
 (& Untransform)  
 or  
`>> m = p(1), b = 10^p(2)`  
`m = -0.0067`  
`b = 2.0622e+005`  
`>> y2hat = b*10.^(m.*x2)`

**Plot:** `>> plot(x2, y2, 's', x2, y2hat)`  
 the resulting prediction curve is not smooth. It is a good idea to smooth this out by simply adding more x points

Commands used to create a graph with a smooth prediction curve

```
>> x2p = linspace(min(x2), max(x2),50);
>> y2hat = b*10.^(m.*x2p);
>> plot(x2, y2, 's', x2p, y2hat)
>> legend('Experimental Data', 'Fit: y2 = (2.06*10^5)*10^{-0.0067*x2}')
>> xlabel 'x2', ylabel 'y2'
```



Handout Problem 2: uses variables x1, y1, Fitting a power law model

**Model:** log-log plot is linear → power model  
 $y1 = b(x1)^m$  or  $\log(y1) = \log(b) + m \log(x1)$

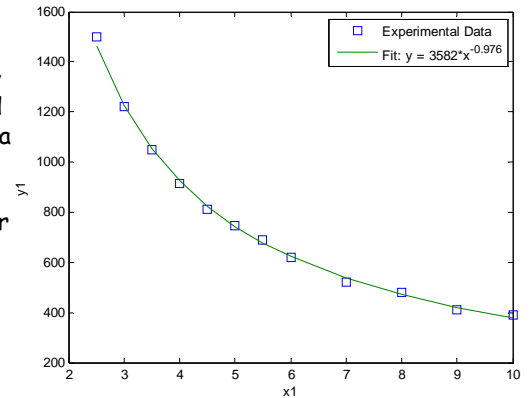
**Setup:** `>> X = [x1'./x1', log10(x1')]; Y = log10(y1');`

**Fit:** `>> B = X \ Y`  
`B =`  
 3.5541  
 -0.9764

**Predict:** `>> y1hat = 10.^(X*B);` or  
 (& Untransform)  
`>> m = B(2), b = 10^B(1)`  
`m = -0.9764`  
`b = 3.5821e+003`  
`>> y1hat = b*x1.^m;`

**Plot:** `>> plot(x1, y1, 's', x1, y1hat)`  
`>> legend('Experimental Data', 'Fit: y = 3582*x^{-0.976}')`  
`>> xlabel 'x1', ylabel 'y1'`

Notice that this function is very nearly  $1/x$ . It would be a good idea to fit this case and see if this simpler model works as well.



Extra, if time:  
 How could you fit  $1/x$ ? →

**Model:**  $y1 = m(1./x1)$

**Setup:** Design Matrix:  $X = (1./x1)'$   
 Response Vector  $Y = y1'$

**Fit:**  $m = X \ Y$

**Predict:**  $Yhat = X*m$

**Plot** the results