# Nonlinear Parameter Estimation

# Outline

- A. Matrix formulation for nonlinear parameter estimation using ordinary least squares (OLS)
- B. Steps before performing parameter estimation:
  1. Standard statistical assumptions;
  2. Choice of the model;
  3. Choice of the nonlinear parameter estimation method;
  4. Choice of the solution method (integral or differential);
  5. Scaled sensitivity coefficients.
- C. After performing parameter estimation:
  6. Statistical results for both parameters and the dependent variable;
  7. Residual analysis;

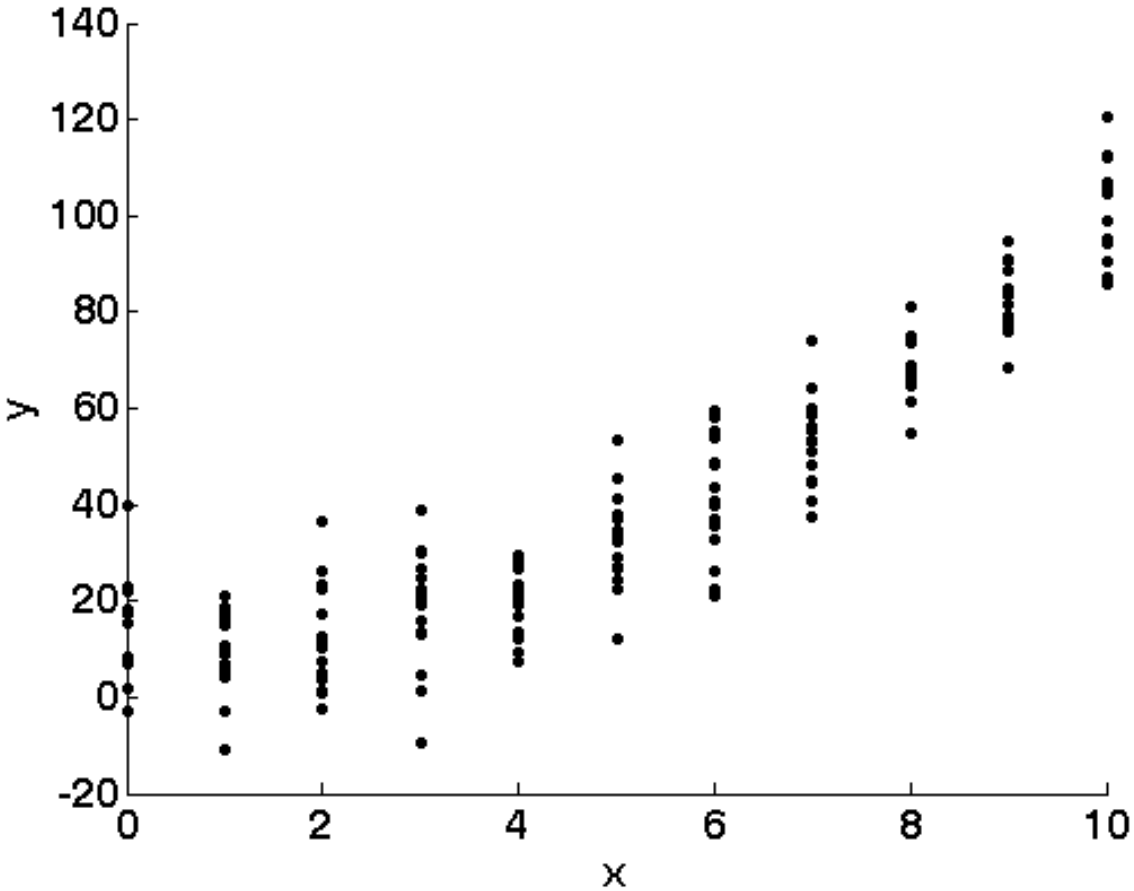
# Step 1. Standard Statistical Assumptions and examples (Beck and Arnold, Revised Chapter 6, p. 5.28-5.34)

1. **Y = η(X,β)** + $\varepsilon$ ; additive errors in measurements
2. E(ε)= 0; Zero mean measurements
3. Constant variance ($\sigma^2$ ) errors
4. Uncorrelated errors
5. ε has a normal distribution of errors
6. Known statistical parameters describing $\varepsilon$
7. Errorless independent variables
8. β is a constant parameter vector and statistics of β are unknown (such as covariance).

# Why are the assumptions important for parameter estimation?

- The estimation problems are generally less difficult when the standard assumptions are valid.
- When nothing is known regarding the measurement errors, OLS is recommended.
- The assumptions should be examined after obtaining results.

# Assumption 1: Example of additive errors



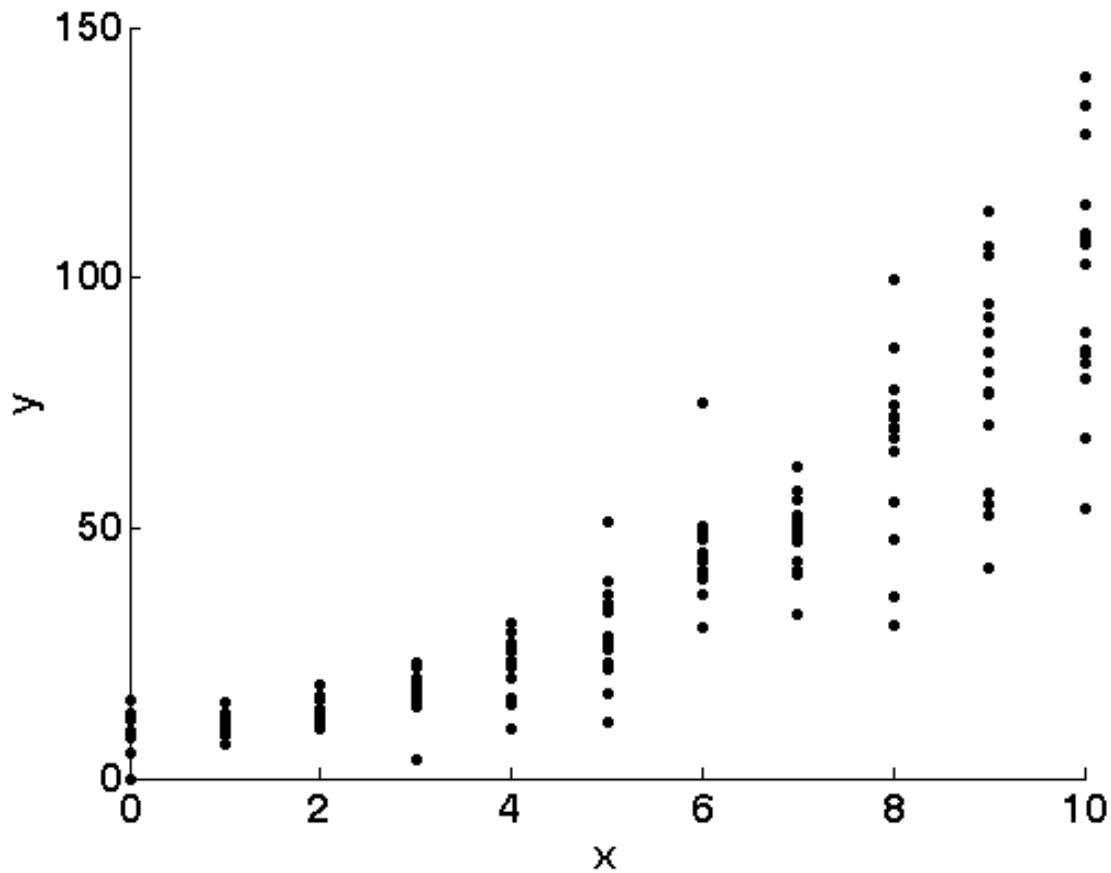
# Violation of Assumption 1: Example of multiplicative errors

Multiplicative Errors:

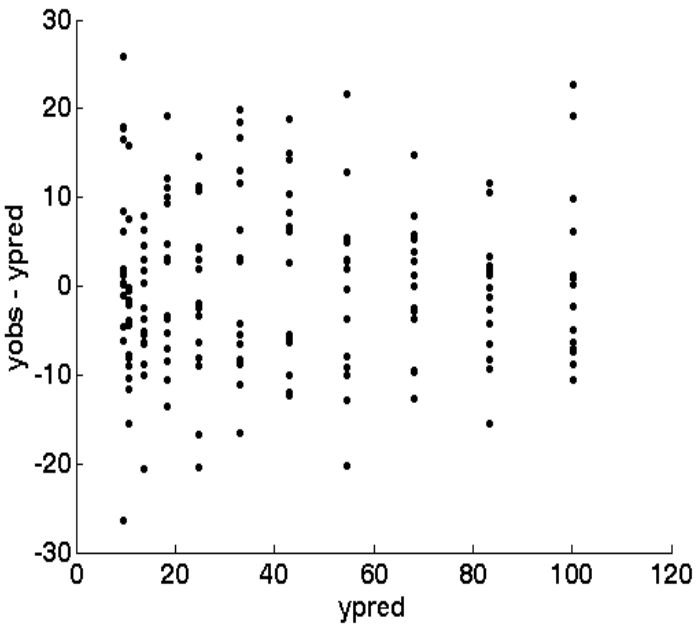
$Y_i = \eta_i(1+\varepsilon_i)$

p. 5.78

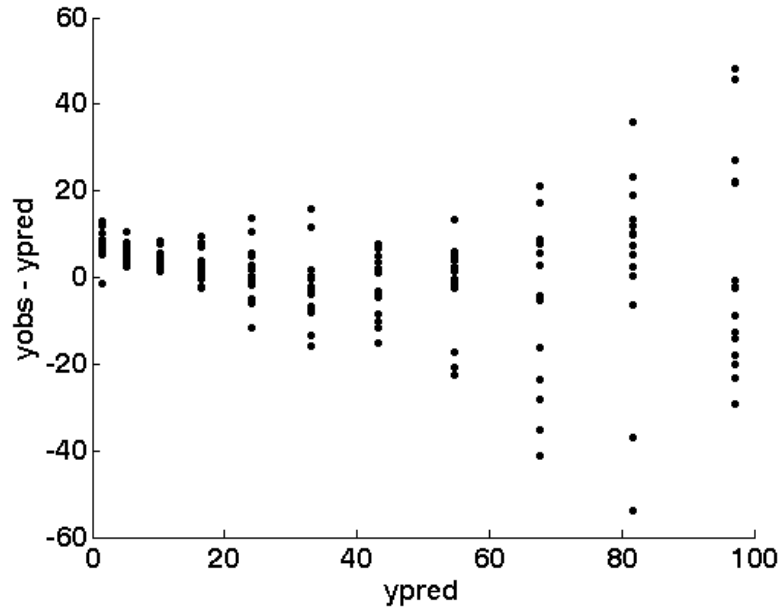
Revised Chap



# Plot residuals vs. $Y_{predicted}$ to check Assumption 1




Plot 1, additive errors




Plot 2, multiplicative errors

# Assumption 2: Mean of errors = 0

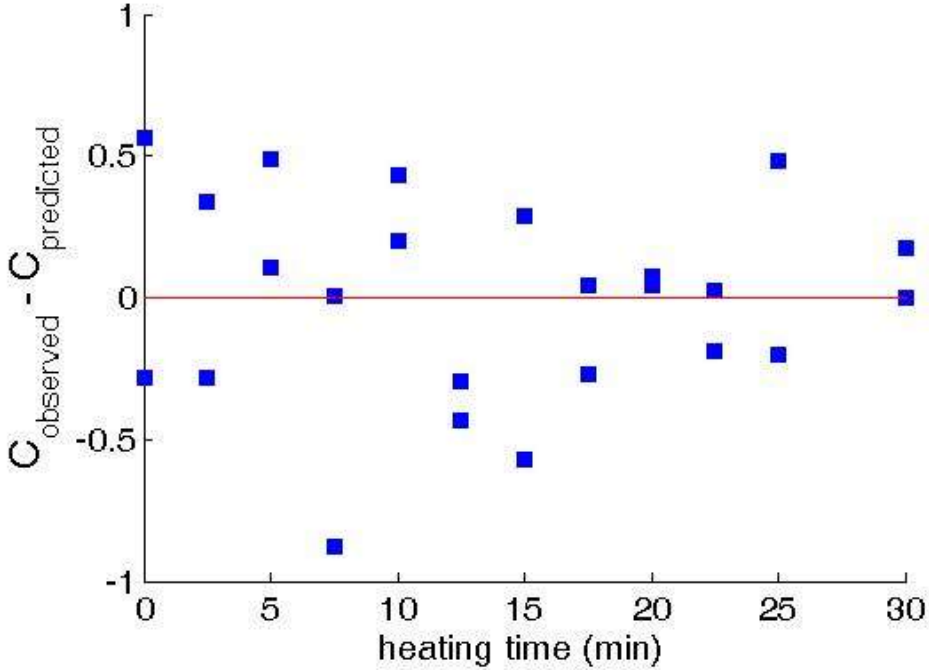
- Can usually approach this by calibration for additive errors.

- Assumption 3: Constant standard deviation of the errors = $\sigma^2$
  - Check by plotting residuals versus x.

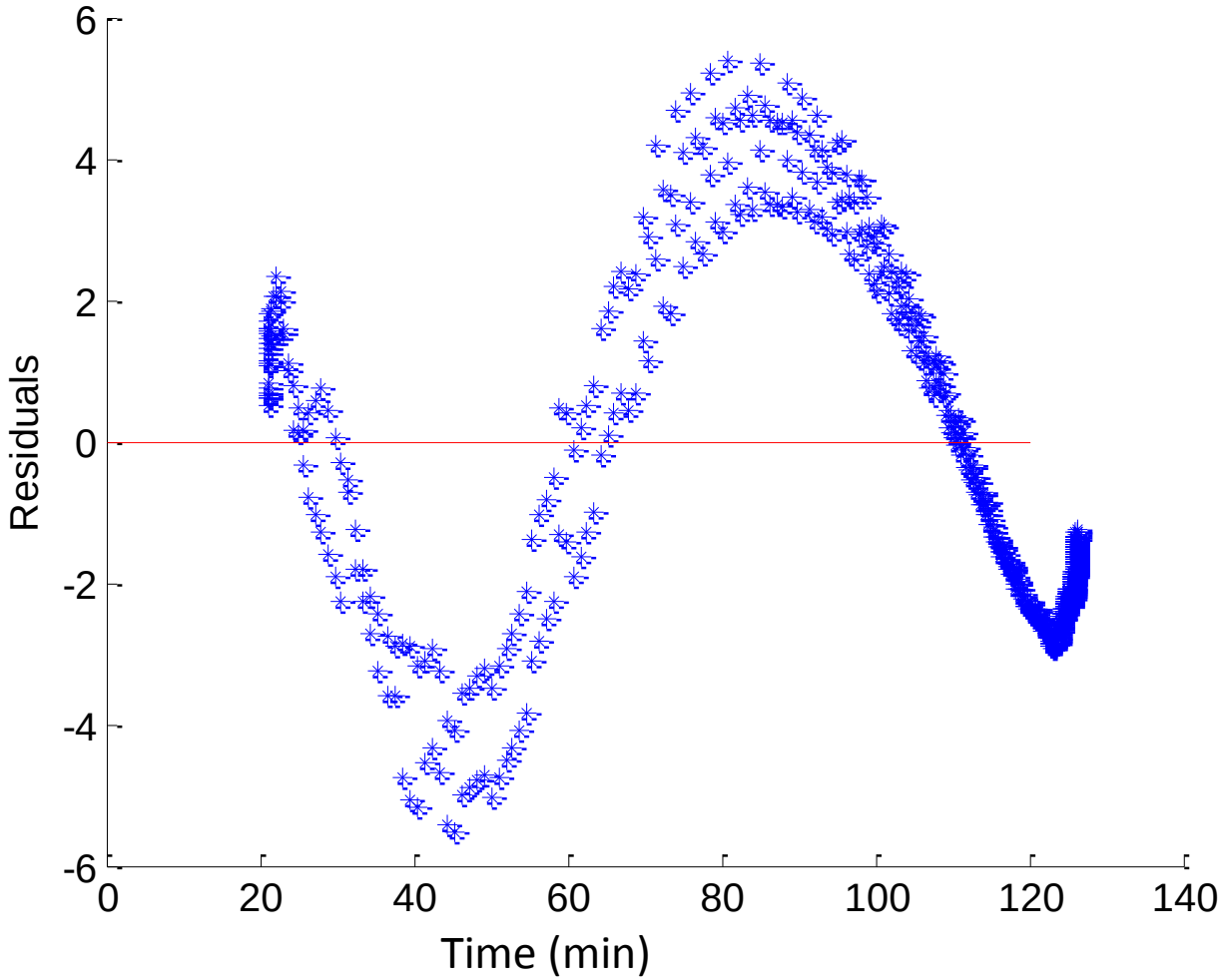
# Assumption 4: Uncorrelated errors

Simple test for uncorrelated errors is the number of runs >= (n+1)/2

A run is a change in sign from one residual to the next



# Violation of Assumption 4: these are correlated errors



# Assumption 5: Normal Distribution of errors

- Simplest test is to plot a frequency histogram
- Use dfittool

- Assumption 6: Covariance matrix of the errors is completely known
- Prior experiments could give this information.

# Assumption 7: Errorless independent variables

- Usually this assumption is true when the independent variable is time. If the independent variable error is much smaller than the dependent variable error, this assumption is nearly true.

# Assumption 8: $\beta$ is a constant parameter vector and there is no prior information

- Usually this is true, unless you are comparing batch sizes where a property changes slightly with each batch. Then $\beta$ would be random.
- We will deal with prior information when we learn MAP and sequential estimation

- Notation: Use "1" to mean that the assumption is valid, "0" if not valid, and a dash "-" if the assumption is not known. So if all 8 assumptions are valid, use "11111111"

# Step 2. Choice of the model

- Choose a model that allows all parameters to be estimated. Plot scaled sensitivity coefficients (Step 5) will assist in this. If you find that some parameters cannot be estimated, either drop them or adjust the model.
- Determine how many dependent variables, independent variables, and parameters there are. You need at least one equation for each dependent variable.

# Step 3. Choice of the nonlinear parameter estimation method

- If nothing is known about the errors (none of the 8 assumptions are known), use ordinary least squares (OLS).
- If  covariance of errors is known, use Maximum Likelihood (ML)
- If covariance of errors AND covariance of parameter are known, use Maximum a posteriori (MAP).  Using MAP, we can also do sequential estimation
- We will spend most of our time on OLS

# Step 4. Choice of the Solution Method (Integral or Differential)

- If the model is in differential form, can use <u>differential solution method</u> (ode45 or something similar)
- Differential method: Advantage—can use the differential equations directly. Disadvantage—If there is tabulated input data, such as temperature vs. time, must fit to a curve. Why?
- Example:

# Example of differential solution

- Primary model :

$$\frac{dy}{dt} = -ky$$

(differential equation)

Rate equation:

$$k = k_r \exp\left(-\frac{E}{R}\left[\frac{1}{T(t)} - \frac{1}{T_r}\right]\right)$$

Must first fit a T-t function: $T(t) = mt + b$

Estimate kr, E, y(0) using ode45 and nlinfit (or other nonlinear regression routine

# Example of integral solution

- Integrate the differential equation first:

$$y = y(0) * \exp(-kt)$$

$$k = k_r \exp\left(-\frac{E}{R}\left[\frac{1}{T(t)} - \frac{1}{T_r}\right]\right)$$

Can use tabulated temperature data, without fitting a function

Estimate kr, E using nlinfit

# Using Matlab to perform nonlinear parameter estimation

- The two main functions for parameter estimation are nlinfit, lsqnonlin, and cftool (Graphic User Interface).
- lsqnonlin allows limits on the parameters, while nlinfit does not.
- I prefer nlinfit because the statistics on the parameter and the predicted value are obtained more directly.
- However, you can generate the Jacobian from lsqnonlin and feed it into nlinparci and nlinpredci to get the same statistic.
- cftool will give standard errors and confidence intervals on the parameters, but no Jacobian. Cannot do differential equation models.
- We must plot scaled sensitivity coefficents, requiring a Jacobian.
- We will focus on nlinfit for this class.

# Syntax for nlinfit

- [beta,r,J,COVB,mse] = nlinfit(x,y,obs, fun, beta0);
- 
- Where beta is a vector are the  p parameters
- *n* is the number of data collected;
- x is a matrix of *n* rows of the independent variable; there can be more than one column to pass in other information needed inside fun;
- *yobs* is n-by-1 vector of the observed data
- fun is a function handle to a separate m-file to a function of this form:
- yhat = fun(b,X)
- where yhat is an n-by-1 vector of the predicted responses, and b is a vector of length p of the parameter values.
- beta0 is length p, the initial guesses of the parameters.
- r is a vector of residuals, J is the Jacobian (sensitivity) matrix X=dy/dparameter
- COVB is the covariance matrix =mse* $(X^TX)^{-1}$
- mse is the mean sqare error = sse/(n-p)

# Matlab files for the inverse problems with differential equations

- Place the calling statement and the function fun together inside another function funinv. Then use nlinfit to call funinv and pass the parameters to funinv.

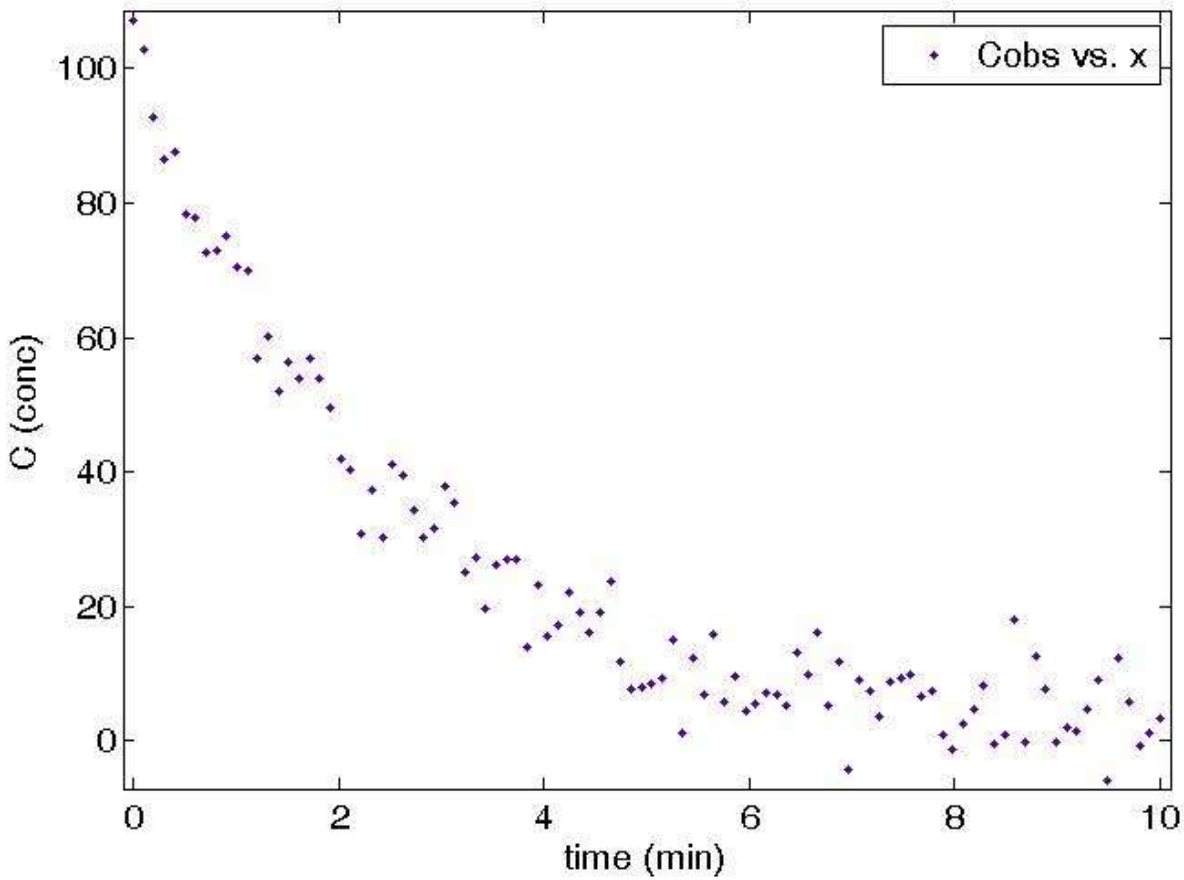
```
set initial guesses beta0(1), beta0(2)
[beta,r,J,COVB,mse] = nlinfit(t,y, funinv, beta0);

function y = funinv(beta,t)
        (t,y)=ode45(@fun,tspan,y0)
            function dy=fun(t,y)
                dy(1)=beta(1)*y(1);
                dy(2)=y(1)-beta(2)*y(2) ;
            end
end
```

# Inverse Problem Example 1

- First order equation dy/dt = -ky;
-
- For the inverse problem, we supply data = yobs

# yobs from exp_data.xls



# inv_ode.m—this file has the nlinfit statement

```
%example of nlinfit file name = inv_ode.m
clear all %clear all variables
data =xlsread('exp_data.xls');
yo=100;
k=0.6;
beta0(1)=yo; %initial guess
beta0(2)=k; %initial guess
x=data(:,1);
yobs=data(:,2);
```

# inv_ode.m page 2

```
%nlinfit returns parameters, residuals, Jacobian (sensitivity
    %coefficient matrix),
%covariance matrix, and mean square error.  ode45 is solved many
    times iteratively
[param,resids,J,COVB,mse] = nlinfit(x,yobs,'forderinv',beta0);
rmse=sqrt(mse);

%confidence intervals for parameters
ci=nlparci(param,resids,J);

%computed Cpredicted by solving ode45 once (forward problem) with
    the estimated parameters
ypred=forderinv(param,x);
```

# inv_ode.m page 3 (print results)

```
figure
hold on
h1(1)=plot(x,ypred,'-','linewidth',3); %predicted y
    values
h1(2)=plot(x,yobs,'square', 'Markerfacecolor', 'r');
legend(h1,'ypred','yobs')
xlabel('time (min)','fontsize',16,'fontweight','bold')
ylabel('y','fontsize',16,'fontweight','bold')

%residual scatter plot
figure
hold on
plot(x, resids, 'square','Markerfacecolor', 'b');
YLine = [0 0];
XLine = [0 max(x)];
plot (XLine, YLine,'R'); %plot a straight red line at
    zero
ylabel('Observed y - Predicted
    y','fontsize',16,'fontweight','bold')
xlabel('time (min)','fontsize',16,'fontweight','bold')
```

```
%residuals histogram
[n1, xout] = hist(resids,10);
 figure
 hold on
 set(gca, 'fontsize',14,'fontweight','bold');
 bar(xout, n1) % plots the histogram
 xlabel('M_{observed} - M_{predicted}','fontsize',16,'fontweight','bold')
 ylabel('Frequency','fontsize',16,'fontweight','bold')

%scaled sensitivity coefficients
ysens1=param(1)*J(:,1);
ysens2=param(2)*J(:,2);
figure
hold on
YLine = [0 0];
XLine = [0 max(x)];
set(gca, 'fontsize',14,'fontweight','bold');
h2(1) = plot(x,ysens1,'-b','LineWidth',2);
h2(2) = plot(x,ysens2,'-r','LineWidth',2);
legend('X''_{yo}','X''_k')
xlabel('time (min)','fontsize',16,'fontweight','bold')
ylabel('scaled sensitivity coefficient','fontsize',16,'fontweight','bold' )
plot (XLine, YLine,'k'); %plot a straight black line at zero
```

# Function with ode45

```
function y = forderinv(beta,t)
%first-order reaction equation, differential method

tspan=t; %we want y at every t
[t,y]=ode45(@ff,tspan,beta(1));%beta(1) is y(0)

   function dy = ff(t,y) %function that computes the dydt
            dy(1)= -beta(2)*y(1);
   end

end
```

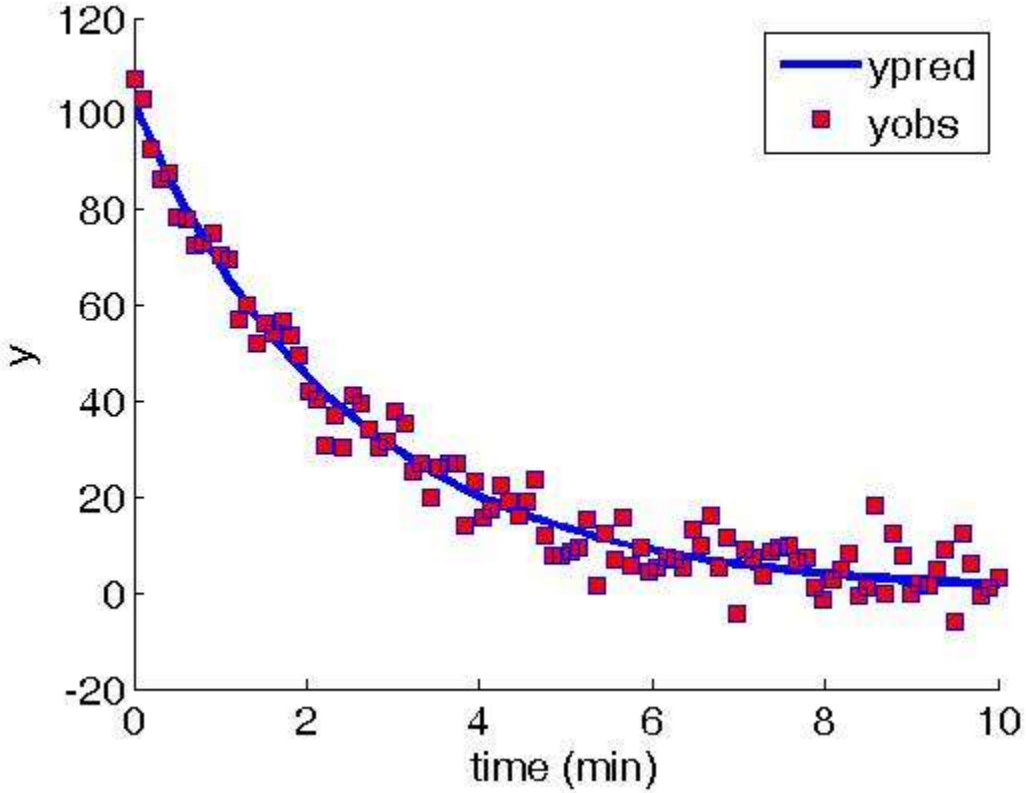
# Results

- beta(1) = 102.225 $\pm$ 1.89 (mean $\pm$ std error)
- beta(2)= 0.406 $\pm$0.011 $min^{-1}$

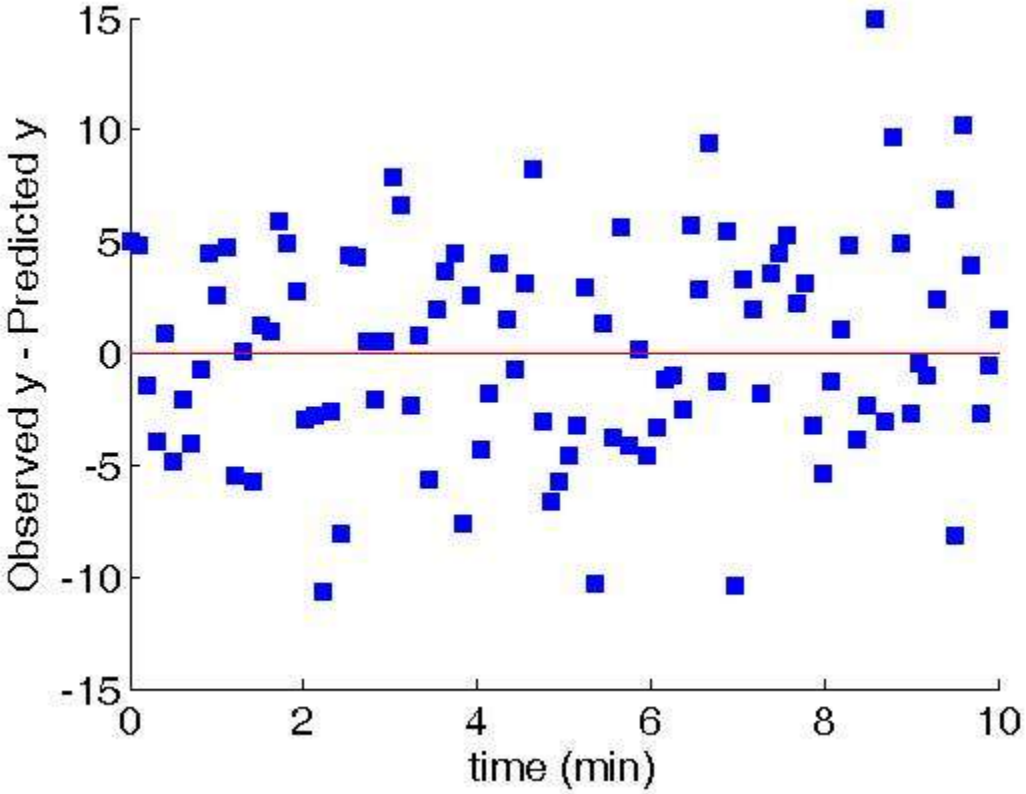
- 95% Confidence intervals (ci)
- Parameter 1: (98.48, 105.97)
- Parameter 2: (0.384, 0.4276)

- rmse = 4.84 (~5% of the total scale—this is good)

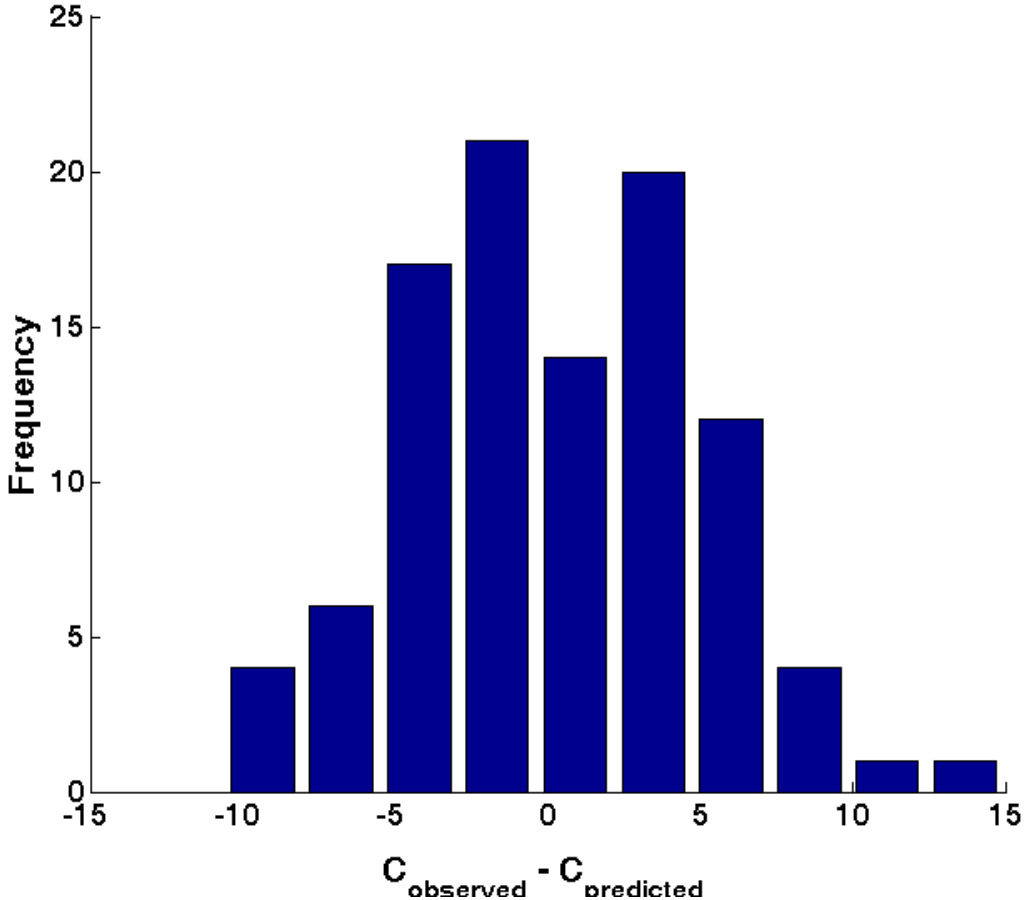
# Results



# Residual scatter plot



# From matlab plot resid hist

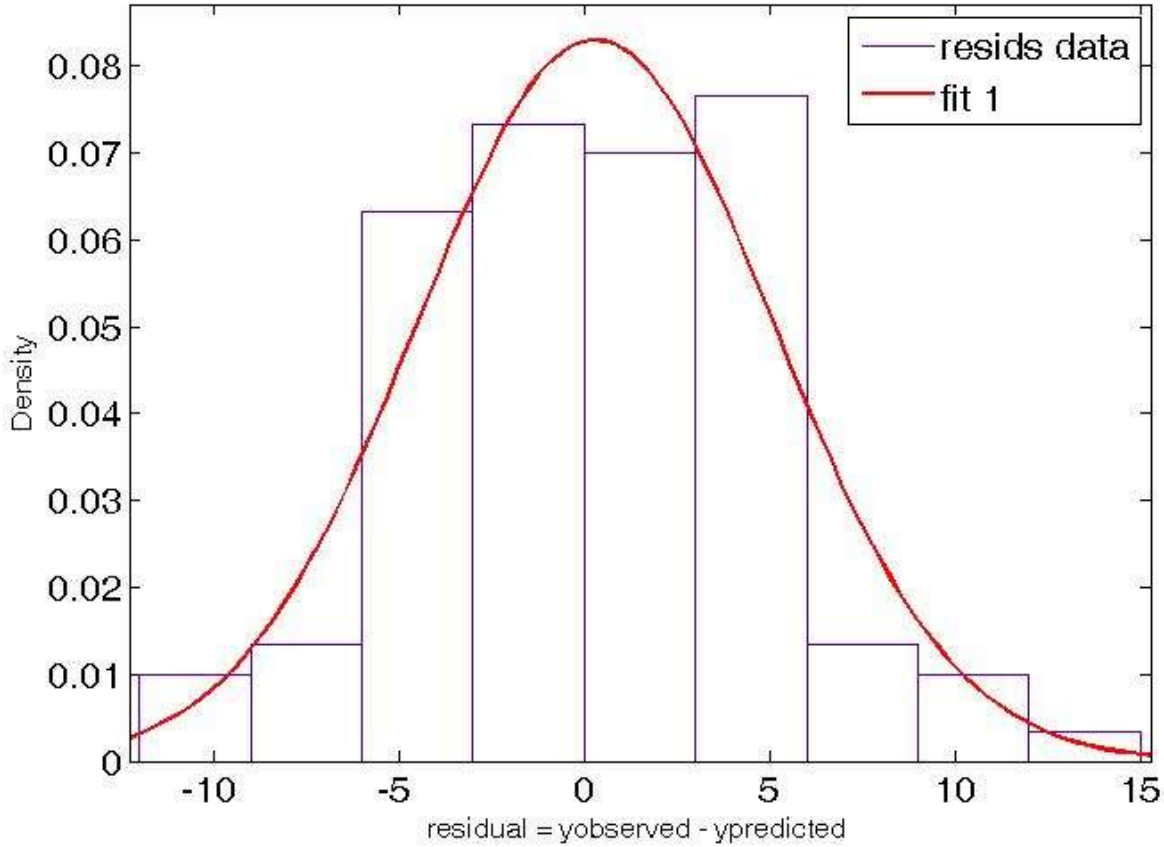


# Residual analysis

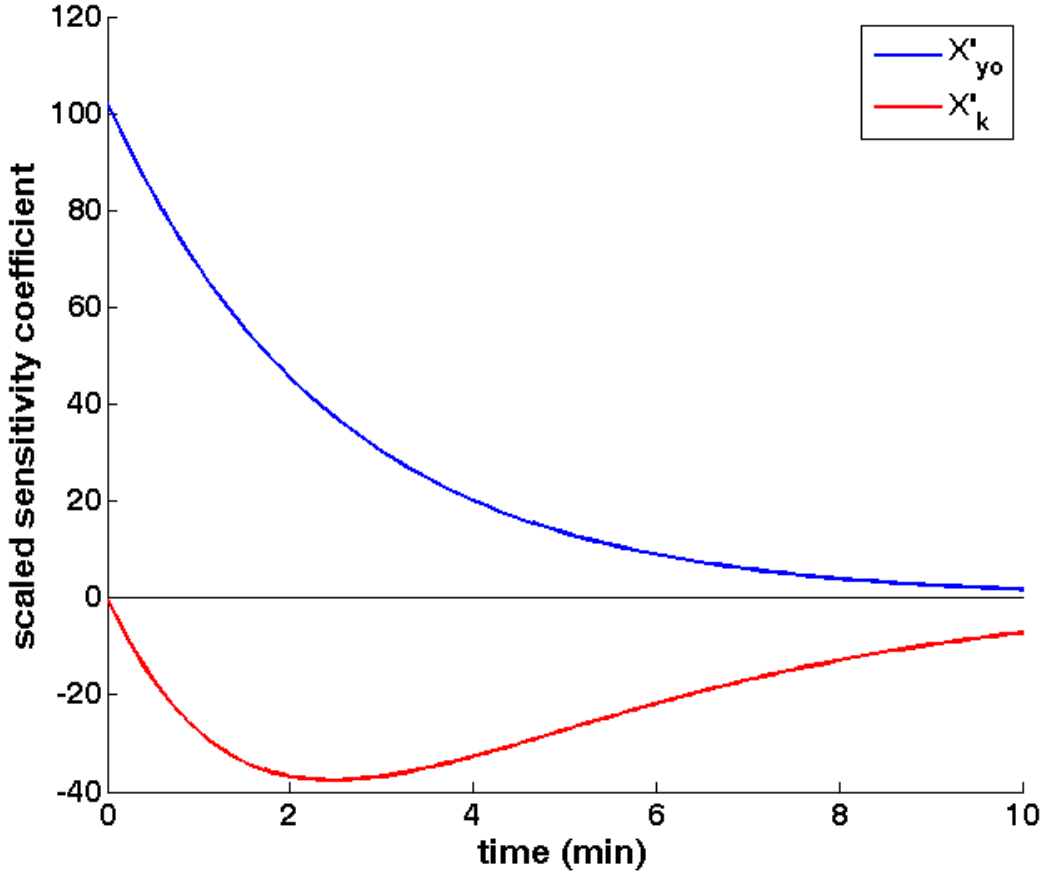
- For histogram,
- Open dfittool and fit normal distribution

# Results

- Residuals fit a reasonably normal distribution



# Scaled Sensitivity Coefficients



```
>> cond(J) = 241.1745  much smaller than 1
   million, which is good
>> det(J'*J)
ans =
  2.5055e+006
```

Large, which is good.

# Integral method?

- What would the function look like?

```
function y = forderinv(beta,t)
%first-order reaction equation, integral method
 ??
```