Nonlinear Parameter Estimation
Outline

A. Matrix formulation for nonlinear parameter estimation using ordinary least squares (OLS)

B. Steps before performing parameter estimation:
   1. Scaled sensitivity coefficients.
   2. Standard statistical assumptions;
   3. Choice of the model;
   4. Choice of the nonlinear parameter estimation method;
   5. Choice of the solution method (integral or differential);

C. After performing parameter estimation:
   6. Statistical results for both parameters and the dependent variable;
   7. Residual analysis;
1. Theoretical Background for Scaled Sensitivity Coefficients

Taylor Series on Parameter Values

\[ \eta(x,t,\beta) \]

\[ \eta(x,t,\beta_0) \]

\[ \beta_0 \quad \beta \]

Initial parameter guesses \quad true parameters

\[ \eta(\text{Estimated Parameters}) \]
Taylor Series on Parameter Values

Let \( \beta_0 = (\beta_{10}, \beta_{20}, \beta_{30}, \ldots, \beta_{p0}) \) initial guesses of parameters

\[
\eta(x, t, \beta) \approx \eta(x, t, \beta_0) + \frac{\partial \eta}{\partial \beta_1} \bigg|_{\beta_0} (\beta_1 - \beta_{10}) + \cdots + \frac{\partial \eta}{\partial \beta_p} \bigg|_{\beta_0} (\beta_p - \beta_{p0})
\]

\[
\eta(x, t, \beta_0) + \begin{bmatrix} \beta_1 & \frac{\partial \eta}{\partial \beta_1} \bigg|_{\beta_0} \end{bmatrix} \frac{\beta_1 - \beta_{10}}{\beta_1} + \cdots + \begin{bmatrix} \beta_p & \frac{\partial \eta}{\partial \beta_p} \bigg|_{\beta_0} \end{bmatrix} \frac{\beta_p - \beta_{p0}}{\beta_p}
\]

\( X'_1 = 1^{st} \text{ scaled sensitivity} \)

\( X'_p = p^{th} \text{ scaled sensitivity} \)

Want scaled sensitivity coefficients to be “large” compared to \( \eta \) (dependent variable) and uncorrelated with each other
What Insights can Scaled Sensitivity Coefficients (X’) Give?

• X’ can be plotted before the experiment is run as a map guiding the parameter estimation.
• The largest X’ will have the smallest relative error = standard deviation/estimate.
• Correlated X’ means potential estimation difficulties.
• Small X’(< ~5% of the span of $\eta$) means the parameter is insignificant and potentially can be eliminated from the model.
Step 2. Standard Statistical Assumptions and examples (Beck and Arnold, Revised Chapter 6, p. 5.28-5.34)

1. \( Y = \eta(X, \beta) + \varepsilon \); additive errors in measurements
2. \( E(\varepsilon) = 0 \); Zero mean measurements
3. Constant variance (\(\sigma^2\)) errors
4. Uncorrelated errors
5. \( \varepsilon \) has a normal distribution of errors
6. Known statistical parameters describing \( \varepsilon \)
7. Errorless independent variables
8. \( \beta \) is a constant parameter vector and statistics of \( \beta \) 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 
\( \geq (n+1)/2 \)
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 3. 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 4. 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 5. Choice of the Solution Method (Integral or Differential)

• If the model is in differential form, can use differential solution method (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:
  \[ y = y(0) \cdot \exp(-kt) \]

Can use tabulated temperature data, without fitting a function

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

Estimate \( k_r, E \) using nlinfit
Using MATLAB to perform nonlinear parameter estimation

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

- `[beta,r,J,COVB,mse] = nlinfit(x,yobs, 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 (anonymous function or 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^T X)^{-1}`
- `mse` is the mean square error = `sse/(n-p)`
MATLAB files for the inverse problems with differential equations

- `nlinfit` calls the forward problem multiple times, so we must nest the forward problem
- 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) %calling statement
function dy=fun(t,y)
dy(1)=beta(1)*y(1);
dy(2)=y(1)-beta(2)*y(2);
end
end

The forward problem
Inverse Problem Example 1

• First order equation \( \frac{dy}{dt} = -ky; \)

• For the inverse problem, we supply data = \( y_{obs} \)
yobs from exp_data.xls
%% example of nlinfit using file name = inv_ode.m

%This program can be used as a base for most nonlinear regression OLS

%% Housekeeping
%
clear all; % Clear the workspace.
close all; % Close all figures.

%% Read in data

data = xlsread('exp_data.xls');
yo=100;
k=0.6;
x=data(:,1);
yobs=data(:,2);
%% Initial parameter guesses

beta0(1)=yo; %initial guess
beta0(2)=k; %initial guess

%% nlinit returns parameters, residuals, Jacobian (sensitivity %coefficient matrix),
covariance matrix, and mean square error. ode45 is solved many times iteratively

[beta,resids,J,COVB,mse] = nlinit(x,yobs,'forderinv',beta0);
rmse=sqrt(mse);

%% confidence intervals for parameters

ci=nlparci(beta, resids,J);
R is the correlation matrix for the parameters, sigma is the standard deviation vector

\[
[R,\text{sigma}]=\text{corrcov}(\text{COVB});
\]
relstderr=\text{sigma}.\text{/beta}'; % relative standard error for each parameter

% computed \text{Cpredicted} by solving \text{ode45} once (forward problem) with the estimated parameters

ypred=\text{forderinv}(\text{beta},x);
%% Confidence and prediction intervals for the dependent variable

%nonlinear regression confidence intervals-- 'on' means simultaneous
%bounds; 'off' is for nonsimultaneous bounds; must use 'curve' for
%regression line, 'observation' for prediction interval
[ypred, delta] = nlpredci('forderinv',x,beta,resids,J,0.05,'on','curve');
  %confidence band for regression line
[ypred, deltaob] = nlpredci('forderinv',x,beta,resids,J,0.05,'on','observation');%prediction band for individual points

%simultaneous confidence bands for regression line
CBu=ypred+delta;
CBl=ypred-delta;

%simultaneous prediction bands for regression line
PBu=ypred+deltaob;
PBl=ypred-deltaob;
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')
Asymptotic confidence bands and prediction bands

%% Output --CIs and PIs

%plot Cobs, Cpred line, confidence band for regression line
h1(1) = plot(x,CBu,'--g','LineWidth',2);
plot(x,CBl,'--g','LineWidth',2);

%plot prediction band for regression line
h1(4) = plot(x,PBu,'-.','LineWidth',2);
plot(x,PBl,'-.','LineWidth',2);
• % 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--same as dfittool, but no curve fit here

[n1, xout] = hist(resids,10); %10 is the number of bins
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 using Jacobian

ysens1 = beta(1)*J(:,1);
ysens2 = beta(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')
title('scaled sensitivity coefficients from Jacobian')
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
%% scaled sensitivity coefficients using forward-difference

d=0.0001;
for i = 1:length(beta)  %scaled sens coeff for forward problem from IJFM paper
    betain = beta;
    betain(i) = beta(i)*(1+d);
    yhat{i} = forderinv(betain,x);
    Xp{i} = (yhat{i}-ypred)/d; %scaled sens coeff for ith parameter
end
ysensf1=Xp{1}; ysensf2=Xp{2}; %extract data from cell array into vectors

figure
hold on
YLine = [0 0];
XLine = [0 max(x)];
set(gca, 'fontsize',14,'fontweight','bold');
h2(1) = plot(x,ysensf1,'-b','LineWidth',2);
h2(2) = plot(x,ysensf2,'-r','LineWidth',2);
legend('X''_yo','X''_k')
title('scaled sensitivity coefficients from forward-difference')
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 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

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

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

\[ \text{rmse} = 4.84 \text{ (~5% of the total scale—this is good)} \]
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

The graph shows two curves labeled $X_{y_0}^*$ and $X_k^*$ as functions of time (in minutes). The $X_{y_0}^*$ curve starts high and decreases rapidly, approaching the horizontal axis. The $X_k^*$ curve starts at a lower value, decreases more gradually, and approaches a positive value as time increases.
$X'$ and ease of parameter estimation

- The accuracy of the parameter estimates is directly proportional to the absolute size of the $X'$s.
- That is, the parameter with the largest $X'$ will have the smallest relative standard error (most accurate).
• Relative standard errors:
  For $\beta_1$: 1.85%
  For $\beta_2$: 2.68%

How do these results compared to the size of the $X'$?
>> cond(J) = 241.1745  much smaller than 1 million, which is good

>> det(J'*J)

ans = 

   2.5055e+006

Large, are far away from zero, which is good.
Integral method?

• What would the function look like?

```matlab
function y = forderinv(beta,t)
%first-order reaction equation, integral method
y = beta(1) * exp(-beta(2) * t)
```
For HW#6

• If your data is in an excel file with multiple columns of different lengths, MATLAB reads the longest length as the length of every column. Where no data exist, MATLAB puts “NaN” in the cell.

• To remove these NaN cells in vector x, use

  \[ x = x(\text{isfinite}(x)) \]
Example of using isfinite to remove NaN cells

data =
    1   6
    2   7
    3   8
    4   9
    5  10
    NaN 11
    NaN 12

>> x = data(:,1)

x =
    1
    2
    3
    4
    5
    NaN
    NaN

>> x = x(isfinite(x))

x =
    1
    2
    3
    4
    5