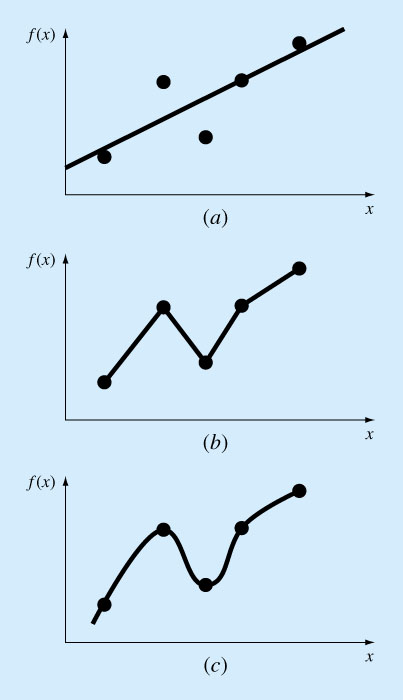
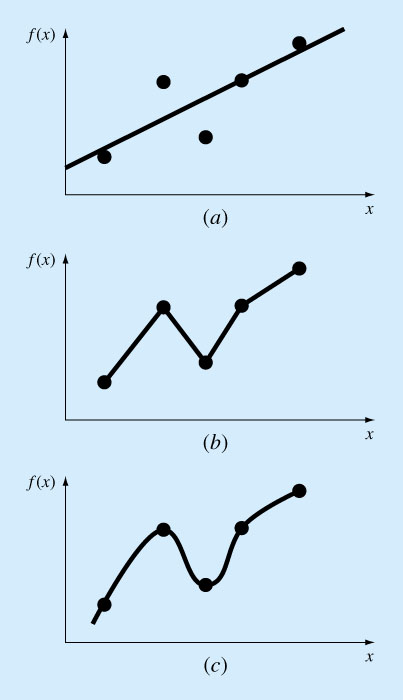
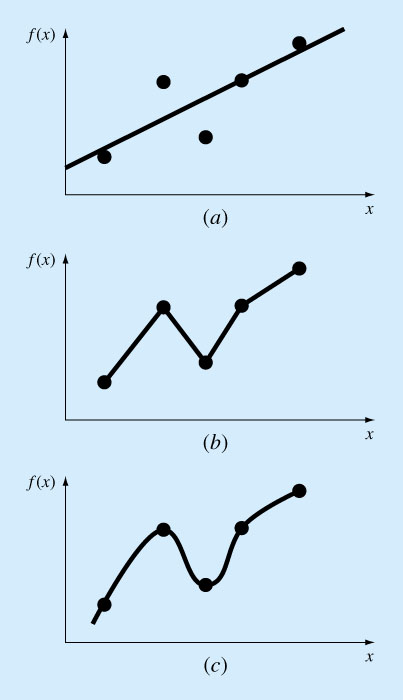
**Lecture Note for Numerical Analysis (10): Least-Squares Regression**

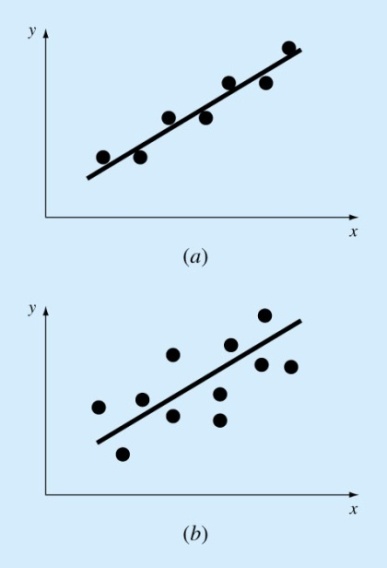
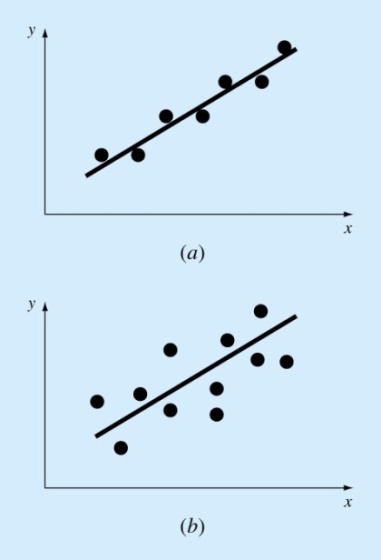
1. **Regression and Interpolation (Curve Fitting)**

* Given n data points : 
* Regression: find a curve fitting best to the points 
* Interpolation: find a curve fitting best to and passing the points 



1. linear regression (b) linear interpolation (c) nonlinear interpolation

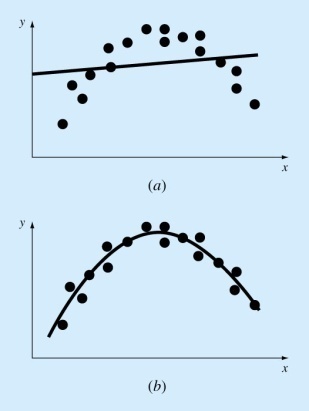
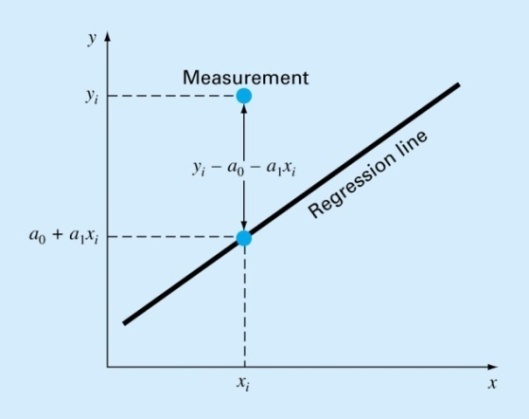
* Regression Error

** **

1. small error (b) large error
2. **Mathematical Expression of One Dimensional Least Squares Repression Problem**

* Approximation function:where is a parameter vector to be determined and  is an independent variable.
* Approximation error: 
* Problem statement for least squares regression as an unconstrained NLP

find f(x;**a**) such that 



1. Example#1: linear regression

(1st order)

1. Example #2: polynomial regression

 (2nd order)

1. Example #3: regression with power equation



Multi-dimensional nonlinear regression

1. **Solution of Linear Least Squares Repression Problem**

* Given n data points : 
* approximation function: 
* approximation error: 
* find the regression parameters to minimize the regression error such that



🡪 2-dimensional minimization problem to find  with the following cost function 



- The first order optimality condition



🡪 

Define the mean values as

🡪 🡪 

1. **Solution of Least Squares Polynomial Repression for One Dimensional Problem**

* Given n data points : 
* approximation function:  where 
* approximation error: 
* find such that 

🡪 (m+1)-dimensional minimization problem to find with the following cost function J()

* 

- The first order optimality condition



or 



**Example: 2nd order Polynomial Repression Problem (m=2)**





1. **General Nonlinear Least Squares Regression for One Dimensional Problem**

* Given n data points : 
* approximation function: 

in general  and  is a linear function of the parameter vector 

* approximation error: 
* find f(x) such that 
* find such that 

🡪 (m)-dimensional minimization problem to find with the following cost function J()

* 

- The first order optimality condition (KKT condition: Karush-Kuhn-Tucker condition)



**🡪 m nonlinear algebraic equations for** 

****

****

****

**[Example 1]**

* Problem statement

- given data: 

- regression equation: 

* Solution

****



Therefore, with the given n-th interate of , we can get the (n+1)-th iterate as



**[Example 2]**

* Problem statement

- given data: 

- regression equation: 

* Solution

****



Therefore, with the given nth iterate of , we can get the (n+1)-th iterate as



**[Example 3] Regression with the power equations in the general form**

* Problem statement

- given data: 

- regression equation: 



* Solution using linear least squares regression

By taking the natural logarithmic operation for each side



which is the same form as the multi-dimensional linear regression

**[Example 4] Regression for the Lagrange interpolation polynomial**

* Problem statement

- given data: 





* Given Data Points with the different m

****

* regression equation in the general polynomial form



* Solution





 where 

- least squares solution





* Homework for m=10

(1) n=1: 

(2) n=2: 

(3) n=3: 

(4) n=4: 

1. **Matrix expression for multi-dimensional linear regression problem**

* Regression formula to find the regression coefficient 



 : independent variable vector

 : regression parameter vector

* Given data (measured data) at each position of 



Where

 : independent variable vector

 : dependent variable

* Approximation error



* Least square solution



Optimality condition using 

🡪



where 

The matrix is called the **pseudo inverse** of .

If is full row rank (all row is linearly independent, which means the given data point are not

overlapped), is non-singular and the pseudo inverse exists.

**[Example 5] Local quadratic approximation of a multivariable function**

Let’s consider the data generated from the following functions



If we have function values at each of the following independent variable vector such that





Find the coefficients of the following approximation function using the least square method.



Or

 where 

[Solution]

Using the given data points, we can define the regression error as



The least square method can be applied to get the coefficient  as



Then, the first order optimality condition can be derived using the result of Appendix C and D such as



Therefore,



The first order optimality condition can be expressed as



Using



The final form of the optimality condition can be expressed by



Results of the quadratic approximation



****

**Exact function distribution Approximated function distribution**

** **

**Distribution of approximation error Pointwise errors at the given points**

**[Program List] Main program plus User defined function**

1. **Main program**

%----------------------------------------------------------------

% (1) Data generation using the precribed user function

%----------------------------------------------------------------

xd = 0:1:2 ; xd=xd' ;

yd = 0:1:2 ; yd=yd' ;

%

No\_data = 0;

for j=1:3

for k=1:3

%--data --------------------------------------------------------------

x1 = xd(j,1) ;

y1 = yd(k,1) ;

%--data vector for (1 x y x^2 xy y^2) ------------------------------

No\_data = No\_data + 1 ;

Nd = No\_data ;

xvec(1,1) = 1.0 ;

xvec(2,1) = x1 ;

xvec(3,1) = y1 ;

xvec(4,1) = x1^2 ;

xvec(5,1) = x1\*y1 ;

xvec(6,1) = y1^2 ;

%

xd\_vec(1:6,Nd) = xvec(1:6,1) ;

%--function data vector -----------------------------------------------

gfun = User\_fun(x1,y1) ;

gvec(Nd,1) = gfun ;

end

end

%

%----------------------------------------------------------------

% (2) Regression Analysis

%----------------------------------------------------------------

% (2-1) Build Matrix and RHS vector

%----------------------------------------------------------------

AA(1:6,1:6) = 0.0 ;

bb(1:6,1) = 0.0 ;

for j=1:No\_data

xvec(1:6,1) = xd\_vec(1:6,j) ; % data vector

%

AA(1:6,1:6) = AA(1:6,1:6) + xvec(1:6,1)\*xvec(1:6,1)' ; % A=A + (x^t)\*x

bb(1:6,1) = bb(1:6,1) + gvec(j,1)\*xvec(1:6,1) ; % b=b + gj\*x

end

%----------------------------------------------------------------

% (2-2) Solution of the coefficients

%----------------------------------------------------------------

avec(1:6,1) = AA(1:6,1:6) \ bb(1:6,1) ; % a = inv(AA)\*bb

%

%----------------------------------------------------------------

% (3) Error Analysis for the given data

%----------------------------------------------------------------

for j=1:No\_data

xvec(1:6,1) = xd\_vec(1:6,j) ; % data vector

%----------------------------------------------------------------

% approximation

%----------------------------------------------------------------

fun(j,1) = xvec(1:6,1)'\*avec(1:6,1) ;

eer(j,1) = abs(fun(j,1) - gvec(j,1) ) ;

end

%----------------------------------------------------------------

% plot the error

%----------------------------------------------------------------

figure(1),

Nd = 1:No\_data; Nd = Nd' ;

semilogy(Nd(1:No\_data,1),eer(1:No\_data,1),'-or') ; hold on ;

xlabel('data point'); ylabel('error') ;

%

%

%----------------------------------------------------------------

% (4) Surface plot of functions and error

%----------------------------------------------------------------

Nx = 31 ;

xmin = 0.0 ;

xmax = 2.0 ;

Dx = (xmax - xmin)/(Nx-1) ;

%

Ny = 31 ;

ymin = 0.0 ;

ymax = 2.0 ;

Dy = (ymax - ymin)/(Ny-1) ;

%

No\_xy = 0 ;

for j=1:Nx

for k=1:Ny

x1 = xmin + Dx\*(j-1) ;

y1 = ymin + Dy\*(k-1) ;

% exact dunction values

xg(j,k) = x1 ;

yg(j,k) = y1 ;

gf(j,k) = User\_fun(x1,y1) ;

% approximation

xvec(1,1) = 1.0 ;

xvec(2,1) = x1 ;

xvec(3,1) = y1 ;

xvec(4,1) = x1^2 ;

xvec(5,1) = x1\*y1 ;

xvec(6,1) = y1^2 ;

ff(j,k) = xvec(1:6,1)'\*avec(1:6,1) ;

% error

err(j,k) = abs(ff(j,k) - gf(j,k) ) ;

end

end

%

% surface plot of functions and error

%

figure(2)

surfc(xg,yg,gf)

colormap hsv

xlabel('x'); ylabel('y'); zlabel('exact function'); hold on ;

%

figure(3)

surfc(xg,yg,ff)

colormap hsv

xlabel('x'); ylabel('y'); zlabel('approximated function'); hold on ;

%

figure(4)

surfc(xg,yg,err)

colormap hsv

xlabel('x'); ylabel('y'); zlabel('approximation error'); hold on ;

%----------------------------------------------------------------

1. **User defined function program**

function fun = User\_fun(x,y)

%---------------------------------------

% function #1

%---------------------------------------

% fun = (x-1)^2 + 4.0\*(y-1)^2 + (x-1)\*(y-1) + 3 ;

%

%---------------------------------------

% function #2

%---------------------------------------

% xm = x - 1 ;

% ym = y - 1 ;

% fun = xm^4 + 4\*ym^4 + (xm^2)\*(ym^2) + 5\*xm\*ym + 3 ;

%---------------------------------------

% function #3

%---------------------------------------

% xm = x - 1 ;

% ym = y - 1 ;

% fun = xm^4 + ym^4 + 5\*xm\*ym + 3 ;

%---------------------------------------

% function #4

%---------------------------------------

xm = x - 1 ;

ym = y - 1 ;

fun = xm^4 + ym^4 +0.3\*xm\*ym + 3 ;

%---------------------------------------

**Appendix: Important Relations**

**[Appendix A] Definition of a Vector, Gradient of a function (in row vector form), Hessian of a function**

, , 

 and the Hessian matrix is symmetric

**[Appendix B] Gradient of a function vector and time derivative of a vector function of **



where 





**[Appendix C] Gradient of the product of vectors and the multiplication between a matrix and a vector**

,

,

**[Appendix D] Gradient of a quadratic function of**

, 

Let , then  and 



**If A is a symmetric matrix**, 