Unit 2 Error Analysis and Data Fitting

Numerical Analysis

Mar. 13, 2017

EE/NTHU

Computer Numbers

- Numbers are represented in computers using finite number of bits.
- In C and C++ we have the following `int` types with various numbers of bits:

Table 2.1.1. Integer Numbers

Type	Bits	Range
char	8	-128 127
short	16	-32,768 32,767
int	32	-2,147,483,648 2,147,483,647
long	64	-9,223,372,036,854,775,808 9,223,372,036,854,775,807
long long	128	-170,141,183,460,469,231,731,687,303,715,884,105,728 170,141,183,460,469,231,731,687,303,715,884,105,727

- Note implementation on each machine can be different
  - Use `#include <limits.h>` to find out the range of each type

Integer Overflow

- When using int types, care should be taken to avoid overflow problem
- For example,

```
short i,j;

i=32767;     // largest short number
i++;         // i is now -32768

j=-32768;    // smallest short number
j--;         // j is now 32767
```

- All types of int can have overflow problem
- C and C++ will not notify the user when overflow happens
  - It is the programmer's responsibility to guard against this problem

Floating Point Numbers

- Computers use floating point numbers to represent real numbers
- Floating point numbers have the following form

$$\underbrace{\pm 0.dddddd}_{\text{mantissa}}\, e\, \underbrace{\pm bbbb}_{\text{exponent}} \tag{2.1.1}$$

  where mantissa and exponent are usually in binary format with fixed numbers of bits.
- Typical real numbers representable in C and C++ are:

Table 2.1.2. Floating Point Numbers

Type	Bits mantissa	Bits exponent	$\lvert x\rvert_{\min}$	$\lvert x\rvert_{\max}$	ϵ
float	24	8	1.17549e-38	3.40282e+38	1.19209e-07
double	53	11	2.22507e-308	1.79769e+308	2.22045e-16
long double	65	15	3.3621e-4932	1.18973e+4932	1.0842e-19

- Note that numbers in between x and $x(1-\epsilon)$, or between x and $x(1+\epsilon)$ cannot be represented by C or C++.
- Special numbers such as $+\infty$, $-\infty$ and NaN are included in computer floating numbers today.
  - NaN: not a number.

Floating Point Numbers, II

- Any number in C or C++ has a finite number of bits.
  - The number of significant digits of any floating numbers are limited.
- Not all real numbers can be represented in a computer.
- Real numbers are approximated in computers.
  - Round-off errors exist in any computer arithmetic.

Example 2.1.3. 4-digit floating point numbers

Assuming a computer's floating number can have 4 significant digits, then

real number	4-digit representation	abs. error δ_{abs}	rel. error δ
1/7 (0.142857142857)	0.1429	4.28571e-05	3e-4
2/7 (0.285714285714)	0.2857	-1.42857e-05	-5e-5
3/7 (0.428571428571)	0.4286	2.85714e-05	6.66667e-5
4/7 (0.571428571429)	0.5714	-2.85714e-05	-5e-5
5/7 (0.714285714286)	0.7143	1.42857e-05	2e-5
6/7 (0.857142857143)	0.8571	-4.28571e-05	-5e-5

Note that the absolute error, $\delta_{abs}, |\delta_{abs}| \leq$ 5e-5.

Computer Number Approximation

- In C and C++, any real number, y, can be approximated only if $y \in [-|x|_{\max}, |x|_{\max}]$ where $|x|_{\max}$ are given in the table (2.1.2).
- Given any number, $-|x|_{\min} < y < |x|_{\min}$, then y is approximated by 0.
- If $y < -|x|_{\max}$ then y is represented by $-\infty$.
- If $y > |x|_{\max}$ then y is represented by ∞.
- When $-|x|_{\max} \leq y \leq |x|_{\max}$ then y is represented by $fl(y)$ such that

$$fl(y) = y(1+\delta) \quad \text{with } |\delta| \leq \epsilon. \tag{2.1.2}$$

  where ϵ is given in table (2.1.2).
- Given two real numbers x and y, their sum is approximated by

$$fl(x+y) = (x+y)(1+\delta_{x+y}), \quad \text{with } |\delta_{x+y}| \leq \epsilon. \tag{2.1.3}$$

  But, sum generated from the individual approximations is (assuming $x+y \neq 0$)

$$\begin{aligned} fl(x) + fl(y) &= x(1+\delta_x) + y(1+\delta_y) \\ &= (x+y)(1+\frac{x\delta_x + y\delta_y}{x+y}) \end{aligned}$$

Addition/Subtraction Errors

- If $xy > 0$ then it can be shown that

$$0 \leq \left| \frac{x\delta_x + y\delta_y}{x+y} \right| \leq |\delta_x| + |\delta_y| \leq 2\epsilon.$$

- Thus, the error in approximating the sum of two real numbers can accumulate after addition.
- It is also possible that the error decreases after addition (cancellation effect).
- If $xy < 0$ and $x + y \approx 0$, then $\dfrac{x\delta_x + y\delta_y}{x+y}$ can be a large number.
- Thus, the round-off error can be large after arithmetic operations.

Example 2.1.4. 4-digit Subtraction

Using 4-digit computer as the last example.

$$\begin{aligned} x &= \frac{100}{7} = 14.29, \quad \delta_x = 3e-4, \\ y &= \frac{99}{7} = 14.14, \quad \delta_y = -2.0202e-4, \\ x - y &= \frac{100}{7} - \frac{99}{7} = 0.15, \quad \delta_{x-y} = 5e-2. \end{aligned}$$

Thus, the error in subtraction can increase significantly.

Multiplication Errors

- Given two real numbers x and y, the ideal computer product is

$$fl(x \times y) = xy(1 + \delta_{xy}) \qquad \text{with } |\delta_{xy}| \leq \epsilon. \tag{2.1.4}$$

But, computer generates

$$\begin{aligned} fl(x \times y) &= x(1+\delta_x) \cdot y(1+\delta_y) \\ &= xy(1 + \delta_x + \delta_y + \delta_x\delta_y) \\ &\approx xy(1 + \delta_x + \delta_y) \end{aligned} \tag{2.1.5}$$

where $\delta_x\delta_y \ll 1$. Thus, the relative error can increase in multiplication.

$$\delta_{xy} \approx \delta_x + \delta_y. \tag{2.1.6}$$

Example 2.1.5. 4-digit Multiplication.

Using 4-digit computer,

$$\begin{aligned} x &= \frac{10}{7} = 1.429, \quad \delta_x = 3e-4, \\ y &= \frac{30}{7} = 4.286, \quad \delta_y = 6.66667e-5, \\ x \times y &= \frac{300}{49} = 6.125, \quad \delta_{xy} = 4.16667e-4, \\ \text{compared to } & \frac{300}{49} = 6.122, \quad \delta = -7.33333e-5. \end{aligned}$$

Division Errors

- For 2 real numbers x and y, the quotient is

$$fl(\frac{x}{y}) = \frac{x}{y}(1+\delta_{x/y}), \qquad \text{with } |\delta_{x/y}| \le \epsilon. \tag{2.1.7}$$

Computer arithmetic generates

$$\begin{aligned} fl(\frac{x}{y}) &= \frac{x(1+\delta_x)}{y(1+\delta_y)} \\ &= \frac{x}{y}\frac{1+\delta_x}{1+\delta_y} \approx \frac{x}{y}(1+\delta_x)(1-\delta_y) \\ &\approx \frac{x}{y}(1+\delta_x-\delta_y) \end{aligned} \tag{2.1.8}$$

- Since δ_x and δ_y can have different signs, the errors can also accumulate.
- The approximation formulas above are valid only if δ is very small.

Associative and Distributive Laws

Example 2.1.6. 4-digit Associative Law.

Using 4-digit computer as the last example.

$$\begin{aligned} x &= \frac{100}{7} = 14.29, \\ y &= -\frac{99}{7} = -14.14, \\ z &= \frac{1}{7} = 0.1429, \\ (x+y)+z &= 0.15 + 0.1429 = 0.2929, \\ x+(y+z) &= 14.29 - 14 = 0.29. \end{aligned}$$

Thus, $(x+y)+z \neq x+(y+z)$.

- It can also be shown that distribution law is not observed in general, that is

$$(x+y)z \neq xz + yz.$$

- Thus, the order of operations is important in numerical analysis.

More Significant numbers

Example 2.1.7. 6-digit Associative Law.

Using 6-digit computer as the last example.

$$
\begin{aligned}
x &= \frac{100}{7} = 14.2857, \\
y &= -\frac{99}{7} = -14.1429, \\
z &= \frac{1}{7} = 0.142857, \\
(x+y)+z &= 0.1428 + 0.142857 = 0.285657, \\
x+(y+z) &= 14.2857 - 14 = 0.2857.
\end{aligned}
$$

Again, $(x+y)+z \neq x+(y+z)$.

- With longer mantissa, the differences become smaller.

Round-off Errors

- Due to the finite length approximation of real numbers by the computer number system, errors exist.
    - With longer mantissa, the approximation error is smaller
    - With longer exponent, the range of the computer number is larger.
- Using longer number system can reduce approximation errors.
- Round-off error can increase significantly after arithmetic operations
    - The order of arithmetic operation is important to the resulting errors.
- Again, using longer number system can reduce arithmetic errors.
    - But need to trade-off the execution time.

Sources of Errors

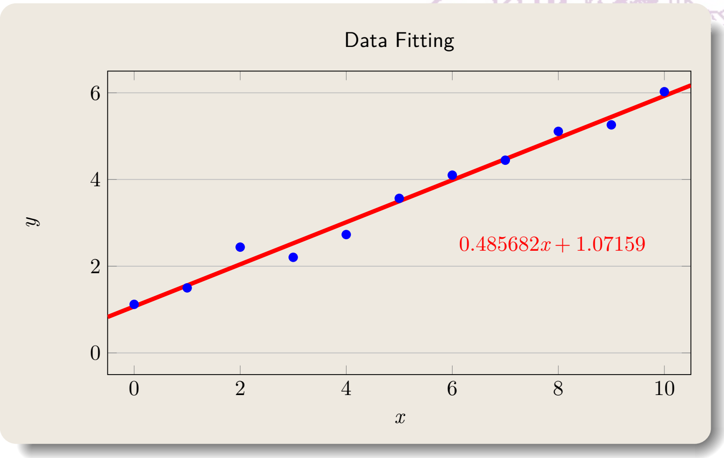
- Numerical analysis applies some mathematical models to approximate physical problems, then applies numerical methods to solve the numerical equations.
- Possible sources of errors:
  - Inaccurate model,
    - For example, in high doped source/drain region we need to apply concentration dependent mobility model for more accurate device simulations.
  - Inaccurate model parameters,
    - If the concentration dependent mobility model has wrong parameters, then the solution is not vary accurate either.
  - Data errors,
  - Truncation errors in approximating the model equations,
    - Example, $\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots$
    - Finite series sum results in errors.
  - Discretization errors in approximating the model is space/time dimensions,
    - Example the finite number of N_x and N_y when discretize the conductor for equivalent resistance calculation.
  - Round-off errors in solving the equations.
- To get accurate results, it is important to analyze all possible error sources.

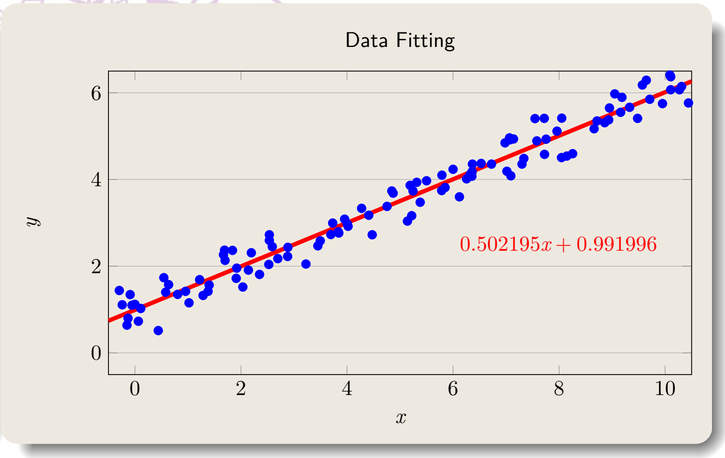
Data Fitting

- Experimental data contain errors, yet often one needs to extract a function from the measured data.
- Example
  - The data measured, y_i, are known to be a linear function of the independent variable, x_i, $i = 1, \cdots, n$

$$y_i = Ax_i + B \tag{2.1.9}$$

How does one find the coefficients, A and B.





Least Square Fit

- One of the popular methods to find the trend in a set of data is the Least Square Fitting method.
- Fitting to a straight line
  - Given a set of data $\{(x_i, y_i), i = 1, \cdots n\}$, least square fit finds $f(x) = Ax + B$ such that

$$S = \sum_{i=1}^{n} (y_i - f(x_i))^2 \qquad (2.1.10)$$

is minimum.

- In this formulation, two coefficients, A and B need to be determined.
- Step 1, taking derivative of S with respect to A and set it to 0

$$\begin{aligned}\frac{\partial S}{\partial A} &= 0 \\ &= \frac{\partial \sum_{i=1}^{n} (y_i - Ax_i - B)^2}{\partial A} \\ &= -2 \sum_{i=1}^{n} x_i (y_i - Ax_i - B) = 0\end{aligned}$$

and we have

$$A \sum_{i=1}^{n} x_i^2 + B \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i y_i \qquad (2.1.11)$$

Least Square Fit, II

- Taking derivative of S with respect to B and set it to 0

$$\begin{aligned}\frac{\partial S}{\partial B} &= 0 \\ &= \frac{\partial \sum_{i=1}^{n} (y_i - Ax_i - B)^2}{\partial B} \\ &= -2 \sum_{i=1}^{n} (y_i - Ax_i - B) = 0\end{aligned}$$

And

$$A \sum_{i=1}^{n} x_i + B \cdot n = \sum_{i=1}^{n} y_i \qquad (2.1.12)$$

- Combining the last two equations, we have the linear system to solve for A and B,

$$\begin{bmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & n \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} \sum x_i y_i \\ \sum y_i \end{bmatrix} \qquad (2.1.13)$$

where each summation is from $i = 1$ to n.

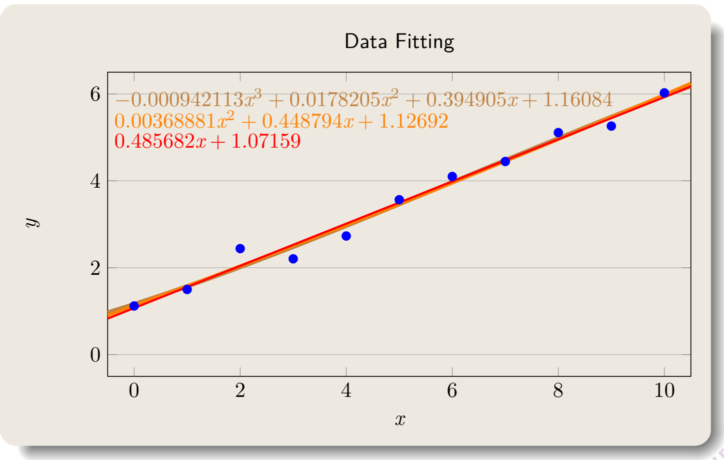
Least Square Fit, III

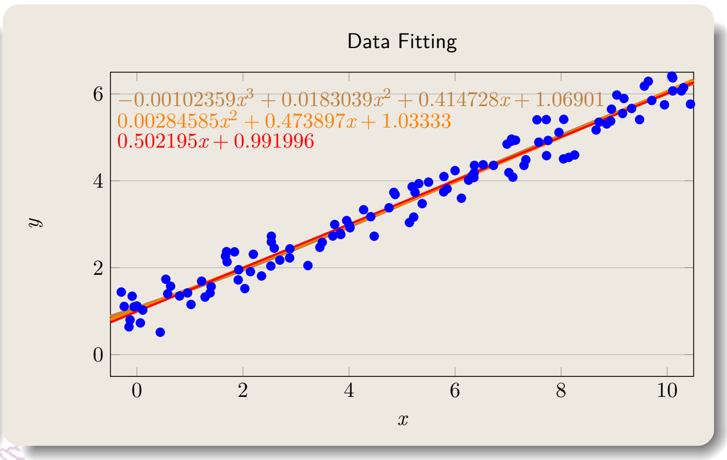
- The Least square fitting method can be generalized to finding a p-degree polynomial $f(x) = a_p x^p + a_{p-1}x^{p-1} + \cdots + a_1 x + a_0$ given the data set $\{(x_i, y_i), i = 1, \cdots, n\}$.
- The coefficients of the polynomial can be found by solving the following linear system of dimension $p+1$.

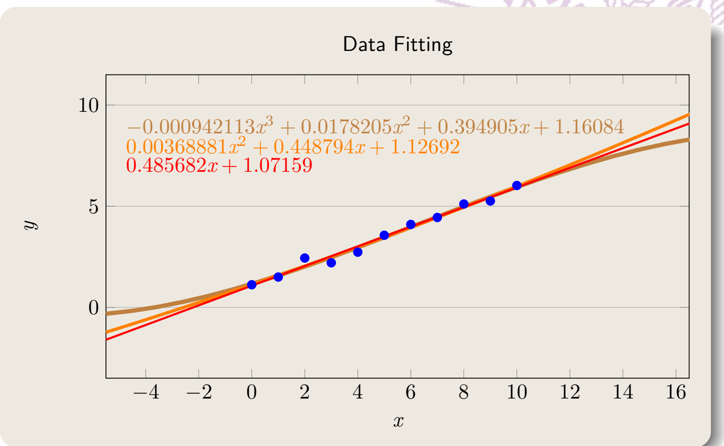
$$\begin{bmatrix} \sum x_i^{2p} & \sum x_i^{2p-1} & \sum x_i^{2p-2} & \cdots & \sum x_i^{p} \\ \sum x_i^{2p-1} & \sum x_i^{2p-2} & \sum x_i^{2p-3} & \cdots & \sum x_i^{p-1} \\ \sum x_i^{2p-2} & \sum x_i^{2p-3} & \sum x_i^{2p-4} & \cdots & \sum x_i^{p-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum x_i^{p} & \sum x_i^{p-1} & \sum x_i^{p-2} & \cdots & n \end{bmatrix} \begin{bmatrix} a_p \\ a_{p-1} \\ a_{p-2} \\ \vdots \\ a_0 \end{bmatrix} = \begin{bmatrix} \sum x_i^p y_i \\ \sum x_i^{p-1} y_i \\ \sum x_i^{p-2} y_i \\ \vdots \\ \sum y_i \end{bmatrix} \tag{2.1.14}$$

where each summation is from $i = 1$ to n.

Higher Degree Polynomial Fitting







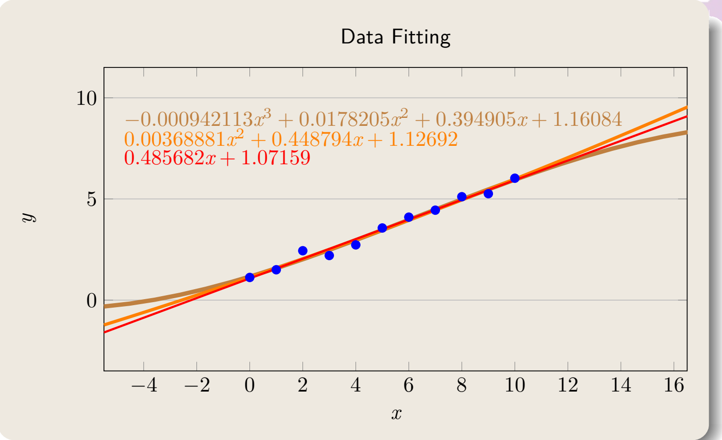
- Equation (2.1.14) can fit the data set to a polynomial of at most p-degree
  - Leading coefficients should be examined for statistical significance
- Limited extrapolation can be performed

Uncertainty in Coefficients

- The coefficients, a_i, of the fitted polynomial can be found by solving Eq. (2.1.14).
- Let $\mathbf{A}$ be the matrix in Eq. (2.1.14) and $\mathbf{A}^{-1}$ be its inverse, then it can be shown that the standard deviations of the coefficients $a_i, i = 0, \cdots, p$ be

$$\delta^2(a_{p-i}) = \delta^2(y) \cdot \mathbf{A}^{-1}_{\mathbf{i},\mathbf{i}} \tag{2.1.15}$$

where $\delta^2(y) = \dfrac{\sum(y_i - a_p x_i^p - a_{p-1}x^{p-1} - \cdots - a_1 x_i - a_0)^2}{N-p-1}$ is the standard deviation of y.



- For the 2nd-degree fitting

	coefficient	std. dev.
a_2	0.00368881	0.0341394
a_1	0.448794	0.354459
a_0	1.12692	0.761853

- $|a_2| < |\delta(a_2)|$, therefore 2nd-degree fitting does not yield more accurate results.

* J.R. Taylor, *An Introduction to Error Analysis*, University Science Books, 1982.

* P.R. Bevington, *Data Reduction and Error Analysis for the Physical Sciences*, McGraw-Hill, 1969.

Multiple Variable Fitting

- Least square methodology can be modified for multiple variable polynomials
- For example, given a data set $\{(x_i, y_i, z_i), i = 1, \cdots, n\}$, where the measured data z_i is a linear function of two variables, x_i and y_i, as

$$z_i = Ax_i + By_i + C \tag{2.1.16}$$

- The least square principle minimizes

$$S = \sum_{i=1}^{n} (z_i - Ax_i - By_i - C)^2 \tag{2.1.17}$$

Then

$$\frac{\partial S}{\partial A} = -2(A\sum x_i^2 + B\sum x_i y_i + C\sum x_i - \sum x_i z_i) = 0$$
$$\frac{\partial S}{\partial B} = -2(A\sum x_i y_i + B\sum y_i^2 + C\sum y_i - \sum y_i z_i) = 0$$
$$\frac{\partial S}{\partial C} = -2(A\sum x_i + B\sum y_i + C\cdot n - \sum z_i) = 0$$

where each summation is from $i = 1$ to n.

Multiple Variable Fitting, II

- Then the coefficients can be found by solving the following

$$\begin{bmatrix} \sum x_i^2 & \sum x_i y_i & \sum x_i \\ \sum x_i y_i & \sum y_i^2 & \sum y_i \\ \sum x_i & \sum y_i & n \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} \sum x_i z_i \\ \sum y_i z_i \\ \sum z_i \end{bmatrix} \tag{2.1.18}$$

  again each summation is from $i = 1$ to n.
- Fitting to functions with more variables can be carried out similarly.
- Also, fitting to higher degree polynomials with more than one variable can be derived.

Fitting Exponential Functions

- Given a data set $\{(x_i, y_i), i = 1, \cdots, n\}$, which has the following dependency

$$y_i = A x_i^B \tag{2.1.19}$$

- This function can be rewritten as

$$\log(y_i) = \log(A) + B \cdot \log(x_i) \tag{2.1.20}$$

  or

$$u_i = A' + B \cdot v_i \tag{2.1.21}$$

  with $u_i = \log(y_i)$ and $v_i = \log(x_i)$.
- Then the least square fitting can be applied to find $A' = \log(A)$ and B, and Eq. (2.1.19) can be found.
- Least square fitting to other functional forms can also be derived.

Summary

- Computer number systems
  - Integer types
  - Floating point types
- Round-off errors
  - Arithmetic error propagation
- Sources of errors
- Least square fitting
  - Polynomial fitting
  - Multi-variable fitting
  - Exponential fitting