KU KONKUK UNIVERSITY

Numerical Analysis
Introduction

목 차

Lecture Note-Numerical Analysis (1): Introduction

1. Sequence of numerical approach to engineering and scientific problems

- System mathematical modeling
  - Mathematical formulation (expression) of physical laws
  - Initial and/or boundary condition
  - System constraints
  - Etc.

- Numerically approximate the system mathematical model using numerical algorithm such as
  - Descretization (ex: finite difference method)
  - Linearization for nonlinear system
  - Interpolation (linear, quadratic, polynomial, etc)
  - Integration (line, surface, volume integrals)
  - Etc.

- Convert the numerical approximation into a computer program
  - Using computer languages such as Fortran, C, Basic, Matlab, etc
  - Using software package such as IMSL libraries, BLAS, Lapack, etc
  - Naive generation of computer codes (time consuming, involve high probability of mistakes)
  - etc

- Run the program in computers to get solutions
  - Compliers/linkers
  - Consider the limitation of computer hardware/software resources
  - etc

- Analyze the computed results
  - Accuracy/convergence
  - interprete physical meanings
  - apply or use the computed results to resolve the engineering and scientific problems
  - etc

2. Objective and Major concerns of numerical analysis

- Objective

  “Application of the computed results to resolve the engineering and scientific problems”

- Major concerns for real applications

(1) Required accuracy

- Accuracy in mathematical modeling of the system
- Accuracy in numerical analysis

(2) Means to overcome physical limits of computing machines

- Floating point handling capability (8-bit, 16-bit, 32-bit, 64 bit machines)
- Number of processors — How to use multi-processor machines
- Memory: — How to reduce the required memory
- CPU time: — How to reduce the computing time
- Software such as compilers: — What software to choose
- etc

(3) Computational efficiency of numerical methods

- Number of floating point operations
- Numerical stability
- Fast convergence
- etc

(4) Others

- Reusability
- Range of applications

목 차

3. Type of Mathematical Problems

(3-1) Roots of equations : $\mathbf{f}(\mathbf{x}) = 0$

(example 1) $f(x) = ax^2 + bx + c = 0 \quad (a \neq 0) \rightarrow x = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

(example 2) $f(x) = ax^5 + bx^4 + cx^3 + dx^2 + ex^1 + g = 0 \rightarrow$ (root formula ?)

(example 3) Roots of system of equation with more than two unknowns such as

$$\begin{aligned} f_1(x,y) &= a_1x^2 + b_1x + c_1 + y = 0 \\ f_2(x,y) &= b_2x + c_2 + y = 0 \end{aligned} \rightarrow \mathbf{f}(\mathbf{x}) = \mathbf{f}(x,y) = \begin{pmatrix} f_1(x,y) \\ f_2(x,y) \end{pmatrix} = \begin{pmatrix} a_1x^2 + b_1x + c_1 + y \\ b_2x + c_2 + y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

with $\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix} \in R^2, \mathbf{f} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} \in R^2$

(example 4) General polynomial equation with one unknown variable

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x^1 + a_0 = 0 \quad , \quad f(x) \in R, \quad x \in R$$

It generally has n solutions

(example 4) General nonlinear algebraic equation with multiple unknown variables

(n unknowns)

$$\mathbf{f}(\mathbf{x}) = 0, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} \in R^n, \quad \mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ f_3(\mathbf{x}) \\ \vdots \\ f_n(\mathbf{x}) \end{pmatrix} \in R^n \quad , \quad f(x) \in R, \quad x \in R$$

Remarks: Notational convention in this lecture

a. A component of the real valued set is represented by R.

- **Real number: *R***
- **Complex number: *C***
- **positive integer (Natural number): *N*, etc.**

b. A Lower Case Plane Letter represents a scalar or scalar function such as:

$$x \in R,\ y \in R,\ f(x) \in R,\ f(\mathbf{x}) \in R,\ a_5 \in R, \quad \textbf{etc.}$$

c. A Lower Case Bold Letter represents a vector or vector function such as:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in R^2,\ \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{pmatrix} \in R^2,\ \mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ f_3(\mathbf{x}) \end{pmatrix} \in R^3 \quad \text{where} \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \in R^3 \quad \textbf{etc.}$$

d. An Upper Case Letter represents a matrix such as: A, B

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{1n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{pmatrix} = (a_{ij}) \in R^{m \times n}$$

(3-2) Solution of simultaneous linear algebraic equations (or matrix equation) $\mathbf{Ax} = \mathbf{b}$

(example1) Linear algebraic equation with three unknowns (x_1, x_2, x_3)

$$
\begin{aligned}
2x_2 + 3x_3 &= 8 \\
4x_1 + 6x_2 + 7x_3 &= -3 \\
2x_1 - 3x_2 + 6x_3 &= 5
\end{aligned}
$$

$$
\rightarrow \begin{pmatrix} 0 & 2 & 3 \\ 4 & 6 & 7 \\ 2 & -3 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 8 \\ -3 \\ 5 \end{pmatrix}
$$

$$
\rightarrow \mathbf{Ax} = \mathbf{b} \quad \text{where} \quad \mathbf{A} = \begin{pmatrix} 0 & 2 & 3 \\ 4 & 6 & 7 \\ 2 & -3 & 6 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 8 \\ -3 \\ 5 \end{pmatrix}
$$

(example 2) Find x satisfying $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A} \in R^{3\times 3}$, $\mathbf{x} \in R^3$, $\mathbf{b} \in R^3$

$$
\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \quad \mathbf{A} = \begin{pmatrix} 1 & 0 & 7 \\ 2 & 5 & 0 \\ 3 & 6 & 9 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \rightarrow \textbf{solution:} \quad \mathbf{x} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}
$$

(example3) General linear algebraic equation with n unknowns

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \\
a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots + a_{3n}x_n &= b_3 \\
&\vdots \\
a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n &= b_n
\end{aligned}
$$

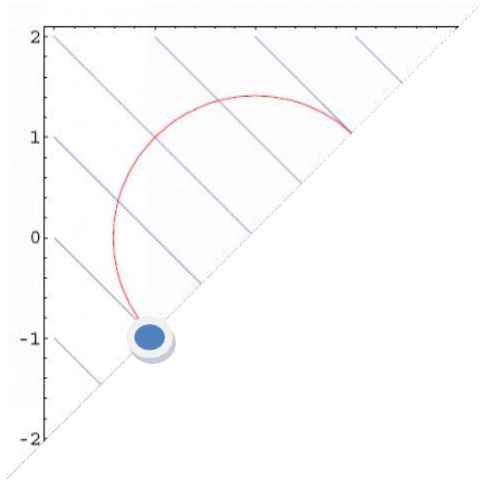
$$
\rightarrow \mathbf{Ax} = \mathbf{b}, \quad \mathbf{A} \in R^{n\times n}, \mathbf{x} \in R^n, \mathbf{b} \in R^n
$$

$$
\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & u_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix} \in R^{n\times n}, \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix} \in R^n, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} \in R^n
$$

(3-3) Optimization

(example 1) solve the following

$$\min f(\mathbf{x}) = x_1 + x_2 \quad s.t. \quad x_1^2 + x_2^2 - 2 = 0 \rightarrow \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \end{pmatrix} \text{ and } f(\mathbf{x}) = -2$$



- **Unconstrained Optimization Problems**

$$\min_{\mathbf{x} \in R^n} f(\mathbf{x})$$

1st order Optimization condition

$$\mathbf{g}(\mathbf{x}) = \frac{df(\mathbf{x})}{d\mathbf{x}} = 0 \rightarrow \textbf{becomes a root-finding problem}$$

(example 1) Nonlinear programming problem with an equality constraint

$$\min \; f(x,y) = x^2 + y^2$$

s. t. $x + y - 1 = 0$ → **solution** $x = y = \frac{1}{2}, \quad f = \frac{1}{2}$

(example 2) Nonlinear Programming problem with one equality constraint and one inequality constraint)

$$\min \; f(x,y) = x^2 + y^2$$

s. t. $x + y - 1 = 0$

$x \geq 1$ → **solution** $x = 1, \quad y = 0, \quad f = 1$

- **General form of constrained nonlinear programming problem (Nonlinear Programming):**

find unknown $\mathbf{x}$ **which minimizes the function** $f(\mathbf{x})$ **subject to** $\begin{matrix}\mathbf{g}(\mathbf{x}) = 0 \\ \mathbf{h}(\mathbf{x}) \leq 0\end{matrix}$

$$\min_{\mathbf{x} \in R^n} f(\mathbf{x})$$

Subject to

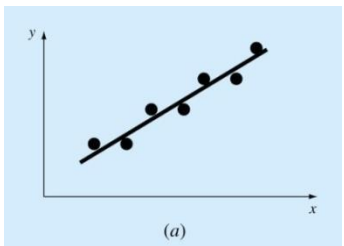
$$\mathbf{g}(\mathbf{x}) = 0$$
$$\mathbf{h}(\mathbf{x}) \leq 0$$

Where $\mathbf{g}(\mathbf{x}) = 0$ **is an equality constraint and** $\mathbf{h}(\mathbf{x}) \leq 0$ **is an inequality constraint**

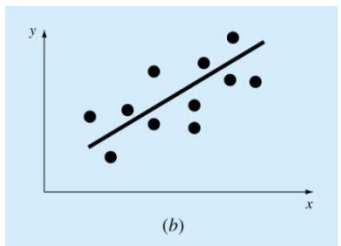
If f(x) is a linear function of x and g(x) and h(x) are also linear function of x, the resultant problem is called linear programming problem

(3-4) Curve fitting (interpolation, least squares approximation)

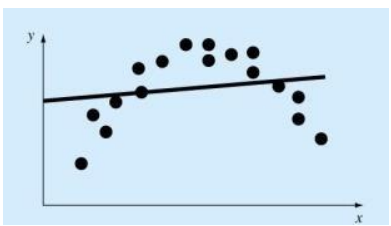
- **Regression: approximation curve doesn't have to pass all given data point but is determined to minimize the approximation error in the sense of Gaussian norm**
- **Interpolation : approximation curve should pass the given data points**



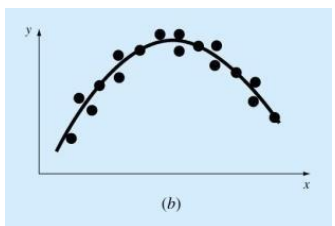
(a) linear regression with small error



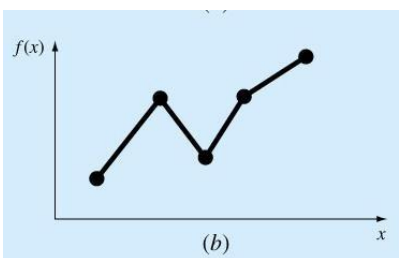
(b) linear regression with large error



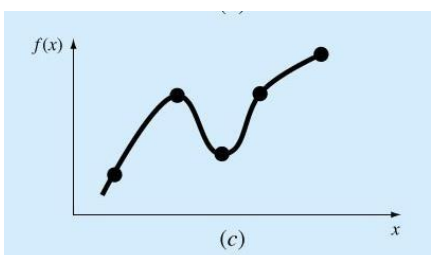
(c) linear regression with large error



(d) nonlinear regression



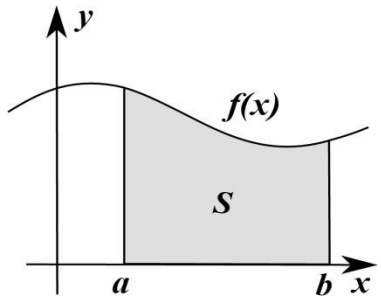
(e) linear interpolation

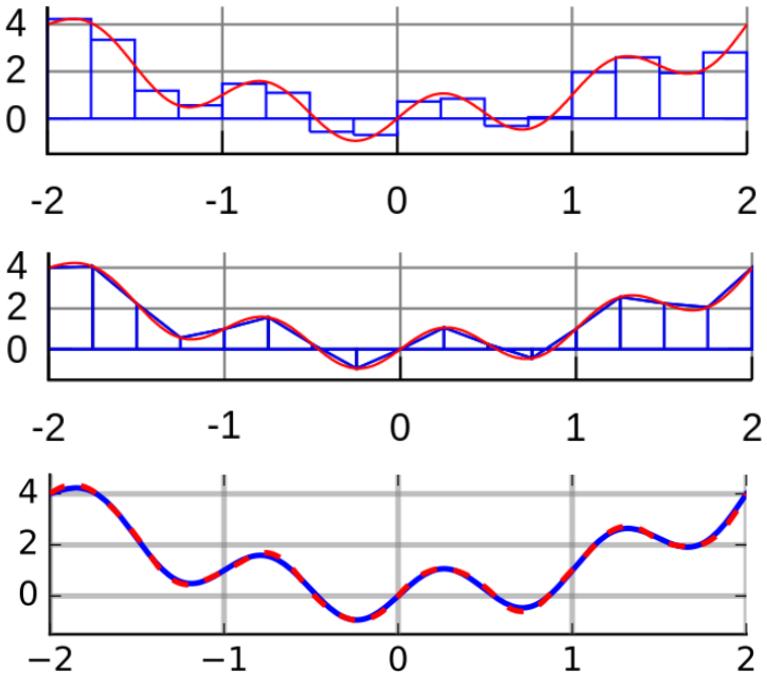


(f) nonlinear interpolation

(3-5) Numerical Integration: Numerical approximation of the following Integral

Line integral: $\int_a^b f(x)dx$





Surface integral: $\iint_S f(x, y)dx, dy$

(example 1): Application to the solution of a first order ordinary differential equation

$$\frac{dy}{dx} = f(x), \quad y(0) = y_0$$

$$y(x) = y(0) + \int_0^x f(t)dt$$

(3-6) Solution of ordinary differential equations such as

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(x,t), \quad \mathbf{x}(x_0) = \mathbf{x}_0$$

(example 1) Problem definition and analytic solution

$$\ddot{x} + 2\dot{x} + 16x = 0.1\cos t, \quad t \in [0,2]$$

$$\text{with initial conditions of } x(0) = 0, \dot{x}(0) = 1$$

Homogeneous solution x_h:

Characteristic equation $\lambda^2 + 2\lambda + 16 = 0$

Characteristic roots

$$\lambda_1 = -1 + \sqrt{1-16} = -1 + j\sqrt{15}$$

$$\lambda_1 = -1 - \sqrt{1-16} = -1 - j\sqrt{15}$$

Solution x_h

$$\begin{aligned} x_h &= c_1 e^{(-1+j\sqrt{15})t} + c_2 e^{(-1-j\sqrt{15})t} \\ &= c_1 \left\{ e^{-t}\left(\cos\sqrt{15}t + j\sin\sqrt{15}t\right)\right\} + c_2 \left\{ e^{-t}\left(\cos\sqrt{15}t - j\sin\sqrt{15}t\right)\right\} \\ &= (c_1 + c_2)e^{-t}\cos\sqrt{15}t + j(c_1 - c_2)e^{-t}\sin\sqrt{15}t \\ &= e_1 e^{-t}\cos\sqrt{15}t + e_2 e^{-t}\sin\sqrt{15}t \end{aligned}$$

Particular solution x_p

$$x_p = K_1\cos t + K_2\sin t \quad x$$

$$\dot{x}_p = -K_1\sin t + K_2\cos t$$

$$\ddot{x}_p = -K_1\cos t - K_2\sin t$$

$$\ddot{x}_p + 2\dot{x}_p + 16x_p = 0.1\cos t$$

$$(-K_1\cos t - K_2\sin t) + 2(-K_1\sin t + K_2\cos t) + 16(K_1\cos t + K_2\sin t) = 0.1\cos t$$

$$(-K_1 - 2K_2 + 16)\cos t + (-K_2 - 2K_1 + 16K_2)\sin t = 0.1\cos t$$

$$\rightarrow \begin{cases} 15K_1 - 2K_2 = 0.1 \\ -2K_1 + 15K_2 = 0 \end{cases} \rightarrow \begin{cases} 225K_2 - 4K_2 = 0.2 \\ K_1 = \dfrac{15}{2}K_2 \end{cases} \rightarrow \begin{cases} K_2 = \dfrac{0.2}{221} \\ K_1 = \dfrac{15}{2}\times\dfrac{0.2}{221} = \dfrac{1.5}{221} \end{cases}$$

Solution $x = x_h + x_p$

$$\begin{aligned} x &= x_h + x_p \\ &= \bar{c}_1 e^{-t}\cos\sqrt{15}t + \bar{c}_2 e^{-t}\sin\sqrt{15}t + \frac{1.5}{221}\cos t + \frac{0.2}{221}\sin t \end{aligned}$$

Applying initial conditions to determine coefficient c_1, c_2

$$\text{initial conditions of } x(0) = 0, \dot{x}(0) = 1$$

$$x(0) = \bar{c}_1 + \frac{1.5}{221} = 0 \rightarrow \bar{c}_1 = -\frac{1.5}{221}$$

$$\dot{x}(0) = -c_1 + \sqrt{15}c_2 + \frac{0.2}{221} = 1 \rightarrow c_2 = \frac{\sqrt{15}}{15} \times \frac{(221-1.7)}{221} = \frac{\sqrt{15}}{15} \times \frac{219.3}{221}$$

Therefore, the solution becomes

$$x(t) = -\frac{1.5}{221} e^{-t} \cos\sqrt{15}t + \frac{\sqrt{15}}{15} \times \frac{219.3}{221} e^{-t} \sin\sqrt{15}t + \frac{1.5}{221}\cos t + \frac{0.2}{221}\sin t$$

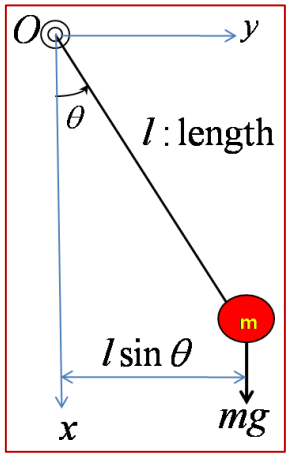
(example 2) Motion of a pendulum hanging on a hinge via a massless string

Motion equation : moment equation around O

$$I\ddot{\theta} = -mgl\sin\theta \quad \text{where} \quad I = ml^2$$

$$ml^2\ddot{\theta} = -mgl\sin\theta$$

$$\boxed{\ddot{\theta} = -\frac{g}{l}\sin\theta = -\alpha^2\sin\theta} \quad \text{where} \quad \alpha = \sqrt{\frac{g}{l}}$$



Initial conditions

$$\boxed{\begin{aligned}\theta(0) &= \theta_0 \\ \dot{\theta}(0) &= \dot{\theta}_0\end{aligned}}$$

How to compute the pendulum motion?

(3-7) Partial differential equations (not covered)

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0 \quad \text{with suitable boundary conditions}$$

목 차

4. Pseudocode

- Formal language : C, C++, Fortran (77,90,99), Matlab, Basic, etc
- Pseudo code is used to describe the general coding procedure using an informal language
- Pseudo code should be translated using one of the formal language for its running in a computer

Pseudo code example 1 to calculate $y = a + b$ with $a = 1, \quad b = 2$

Pseudo code	Description
$a = 1$	save 1 into memory a
$b = 2$	save 2 into memory b
$y = a + b$	add the values in memories a and b, save the result into memory y

Pseudo code example 2 to calculate $y = ax + b$ with $a = 1, \quad x = 5, \quad b = 2$

Pseudo code	Description
$a = 1$	save 1 into memory a
$x = 5$	save 5 into memory x
$b = 2$	save 2 into memory b
$y \leftarrow ax$	replace the value in memory y with the multiplication ax
$y \leftarrow y + b$	add b to the values in memories y and replace it with the result

Pseudo code example 3 to calculate $\mathbf{z} = \mathbf{x} + \mathbf{y}$, where x,y,z, are vectors such as

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} z_1 \\ \vdots \\ z_N \end{bmatrix}$$

Pseudo code (1)

$z_j = x_j + y_j, \quad j = 1, \cdots, N$

Description

For each value of $j = 1, \cdots, N$, add x_j and y_j .

Then save the result

Pseudo code (2)

```
do j=1, N
  z_j = x_j + y_j
```

Description

Repeat for each value of $j = 1, \cdots, N$

Pseudo code (3)

```
for j=1:1:N
  z_j = x_j + y_j
```

Description

Repeat for each value of $j = 1, \cdots, N$ with an increment 1

Pseudo code (4) Description

Repeat for j=1 until j=N Repeat for each value of $j = 1, \cdots, N$

$z_j = x_j + y_j$

Pseudo code (5) Description

Repeat for j=1 until j=N Repeat for each value of $j = 1, \cdots, N$

$z_j \leftarrow x_j + y_j$

Pseudo code example 4 for logical statement

(1)

If a>0, b=2 if a>1, then set b with 2

(2) if a>0, then

b=2

else

b=3

end if

(3) if a>0, b=2

Otherwise, b=3

(4) if a>0, b=2

Else if a=0

b = 0

Otherwise

b=3

end if

목 차

5. Tips for building an efficient numerical analysis program

(1) Memory saving in calculating $y = x^5$

High memory

$y1 = x$

$y2 = y1 \times x$

$y3 = y2 \times x$ → Memory required: $x, y1, y2, y3, y4, y$

$y4 = y3 \times x$

$y = y4 \times x$

Minimum memory

$y \leftarrow x$

$y \leftarrow y \times x$

$y \leftarrow y \times x$ → Memory required: x, y

$y \leftarrow y \times x$

$y \leftarrow y \times x$

Computer Program MATL

AB: $y = x\text{^}5$

FORTRAN: $y = x**5$

(2) Number of floating point operations in calculating $y = 5a^5 + 4a^4 + 3a^3 + 2a^2 + a + 1$

- $y = 5 \times a \times a \times a \times a \times a + 4 \times a \times a \times a \times a + 3 \times a \times a \times a + 2 \times a \times a + a + 1$

  Number of additions: 5 Number of multiplications: 5+4+3+2= 14

- $y = a \times \{a \times [a \times (a \times (5 \times a + 4) + 3) + 2] + 1\} + 1$

  Number of additions: 5 Number of multiplications: 5

(3) Miscellaneous for basic numerical operations in computers

- Calculation of $y_j = \frac{x_j}{a}, \quad j = 1,2,\cdots,N$ with given values of a constant $a(\neq 0)$ and an array x_j

  Naive calculation: $y_j = \frac{x_j}{a}, \quad j = 1,2,\cdots,N$

  Other method set $b = \frac{1}{a}$

  $y_j = b \times x_j, \quad j = 1,2,\cdots,N$

- Calculation of $y_j = -\frac{x_j}{a}, \quad j = 1,2,\cdots,N$ with given values of a constant $a(\neq 0)$ and an array x_j

  Naive calculation: $y_j = -\frac{x_j}{a}, \quad j = 1,2,\cdots,N$

Other method (1) set $b = \frac{1}{a}$

$$y_j = -b \times x_j, \quad j = 1,2,\cdots,N$$

Other method (2) set $b = -\frac{1}{a}$

$$y_j = b \times x_j, \quad j = 1,2,\cdots,N$$

- Calculation of $z_j = x_j - ay_j, \quad j = 1,2,\cdots,N$

Naive calculation: $z_j = x_j - ay_j, \quad j = 1,2,\cdots,N$

Other method (1) when (y_j)s are not used at the later part of program

$$\begin{aligned} y_j &\leftarrow ay_j \\ z_j &\leftarrow x_j - y_j \end{aligned}, \quad j = 1,2,\cdots,N$$

Other method (2) Set $b = -a$

$$\begin{aligned} y_j &\leftarrow by_j \\ z_j &\leftarrow x_j + y_j \end{aligned}, \quad j = 1,2,\cdots,N$$

- Matrix Calculation of $\mathbf{y} = \alpha\mathbf{A}\mathbf{x} + \mathbf{b}$ with a given constant α and

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{N1} \\ a_{21} & a_{22} & a_{2N} \\ & & \\ a_{N1} & a_{N2} & a_{NN} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix}$$

Inefficient calculation procedure (number of multiplications: NxN+NxN)

$$\mathbf{M} \leftarrow \alpha\mathbf{A}$$

$$\mathbf{v} \leftarrow \mathbf{M}\mathbf{x}$$

$$\mathbf{v} \leftarrow \mathbf{v} + \mathbf{b}$$

Normal calculation procedure (number of multiplications: NxN+N)

$$\mathbf{v} \leftarrow \mathbf{A}\mathbf{x}$$

$$\mathbf{v} \leftarrow \alpha\mathbf{v}$$

$$\mathbf{v} \leftarrow \mathbf{v} + \mathbf{b}$$

- Matrix Calculation of $\mathbf{y} = \mathbf{ABx}$ with a given constant α and

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{1n} \\ a_{21} & a_{22} & a_{2n} \\ & & \\ a_{n1} & a_{n2} & a_{nn} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & b_{1n} \\ b_{21} & b_{22} & b_{2n} \\ & & \\ b_{n1} & b_{n2} & b_{nn} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

(1) Sequence of operation (1) ; multiplication

$$\mathbf{M} \leftarrow \mathbf{AB} \quad \text{number of operation (multiplication)} \approx n^3$$

$$\mathbf{y} \leftarrow \mathbf{Mx} \quad \text{number of operation (multiplication)} \approx n^2$$

$$n^3 + n^2 = 1010000 \quad (n = 100)$$

(2) Sequence of operation (2) ; multiplication $2n^2 \rightarrow 20000 \quad (n = 100)$

$$\mathbf{m} \leftarrow \mathbf{Bx} \quad \text{number of operation (multiplication)} \approx n^2$$

$$\mathbf{y} \leftarrow \mathbf{Am} \quad \text{number of operation (multiplication)} \approx n^2$$

$$2n^2 = 20000 \quad (n = 100)$$

- Calculation of Exponential $y = x^5$ with integer index

Using syntax such as $y = x**5$ Fortran

$y = x\wedge 5$ Matlab

Normal calculation (1)

$$\begin{array}{l} y \leftarrow x \times x \\ y \leftarrow x \times y \\ y \leftarrow x \times y \\ y \leftarrow x \times y \end{array} \leftarrow \begin{array}{l} y \leftarrow x^2 \\ y \leftarrow x^3 \\ y \leftarrow x^4 \\ y \leftarrow x^5 \end{array}$$

Normal calculation (2)

$$\begin{array}{l} y \leftarrow x \times x \\ y \leftarrow y \times y \\ y \leftarrow x \times y \end{array} \leftarrow \begin{array}{l} y \leftarrow x^2 \\ y \leftarrow x^4 \\ y \leftarrow x^5 \end{array}$$

"Compare

number of multiplications/divisions

number of additions/subtractions a

nd run time for each algorithm"

목 차

6. Examples of math intrinsic function in Fortran

Name	Description
ACOS	Returns the arc cosine of the argument, expressed in radians between 0 and pi.
ACOSD	Returns the arc cosine of the argument, expressed in degrees between 0 and 180.
ALOG	Returns natural log of the argument.
ALOG10	Returns common log (base 10) of the argument.
ASIN	Returns the arc sine of the argument, expressed in radians between ±pi/2.
ASIND	Returns the arc sine of the argument, expressed in degrees between ±90°.
ATAN	Returns the arc tangent of the argument, expressed in radians between ±pi/2.
ATAND	Returns the arc tangent of the argument, expressed in degrees between ±90°.
ATAN2	Returns the arc tangent of the second argument divided by the first argument, expressed in radians between ±pi.
ATAN2D	Returns the arc tangent of the second argument divided by the first argument, expressed in degrees between ±180°.
CCOS	Returns complex cosine of the argument.
CDCOS	Returns the double-precision complex cosine of the argument.
CDEXP	Returns double-precision complex exponential value of the argument.
CDLOG	Returns the double-precision complex natural log of the argument.
CDSIN	Returns the double-precision complex sine of the argument.
CDSQRT	Returns the double-precision complex square root of the argument.
CEXP	Returns the complex exponential value of the argument.
CLOG	Returns the complex natural log of the argument.

COS	Returns the cosine of the argument, which is in radians.
COSD	**COSD**(*x*). Returns the cosine of the argument, which is in degrees.
COSH	Returns the hyperbolic cosine of the argument.
COTAN	Returns the cotangent of the argument, which is in radians.
COTAND	Returns the cotangent of the argument, which is in degrees.
CSIN	Returns the complex sine of the argument.
CSQRT	Returns the complex square root of the argument.
DACOS	Returns the double-precision arc cosine of the argument radians between 0 and pi.
DACOSD	Returns the arc cosine of the argument in degrees between 0 and 180.
DASIN	Returns the double-precision arc sine of the argument in radians between ±pi/2.
DASIND	Returns the double-precision arc sine of the argument degrees between ±90°.
DATAN	Returns the double-precision arc tangent of the argument radians between ±pi/2.
DATAND	Returns the double-precision arc tangent of the argument degrees between ±90°.
DATAN2	Returns the double-precision arc tangent of the second argument divided by the first argument radians between ±pi.
DATAN2D	Returns the double-precision arc tangent of the second argument divided by the first argument degrees between ±180°.
DCOS	Returns the double-precision cosine of the argument radians.
DCOSD	Returns the double-precision cosine of the argument degrees.
DCOSH	Returns the double-precision hyperbolic cosine of the argument.
DCOTAN	Returns the double-precision cotangent of the argument.
DEXP	Returns the double-precision exponential value of the argument.
DLOG	Returns the double-precision natural log of the argument.

DLOG10	Returns the double-precision common log (base 10) of the argument.
DSIN	Returns the double-precision sin of the argument radians.
DSIND	Returns the double-precision sin of the argument degrees.
DSINH	Returns the double-precision hyperbolic sine of the argument.
DSQRT	Returns the double-precision square root of the argument.
DTAN	Returns the double-precision tangent of the argument radians.
DTAND	Returns the double-precision tangent of the argument degrees.
DTANH	Returns the double-precision hyperbolic tangent of the argument.
EXP	Returns the exponential value of the argument.
LOG	Returns the natural log of the argument.
LOG10	Returns the common log (base 10) of the argument.
SIN	Returns the sine of the argument, which is in radians.
SIND	Returns the sine of the argument, which is in degrees.
SINH	Returns the hyperbolic sine of the argument.
SQRT	Returns the square root of the argument.
TAN	Returns the tangent of the argument, which is in radians.
TAND	Returns the tangent of the argument, which is in degrees.
TANH	Returns the hyperbolic tangent of the argument.