공학 수치해석

Numerical Analysis for Engineers

구조해석 및 동역학 연구실

(Structural Analysis & Dynamics Laboratory)

단국대학교 건축대학

LECTURE NOTE
NOTE-2012-ver-20120918

18 September 2012

EUNCHURN PARK (eunchurn@kmctech.co.kr)
Structrual Analysis & Dynamics Laboratory
Deppartment of Architectural Engineering
Dankook University School of Architecture

Department of R&D Development
Korea Maintenance Co., LTD.
KM Tower 12F
130-5, Guro-5-dong, Gurogu, Seoul (152-842)
PHN: +82-2-830-7071 (236)
SEL: +82-10-4499-6420
FAX: +82-2-830-5256

공학 수치해석

Numerical Analysis for Engineers

구조해석 및 동역학 연구실
(Structural Analysis & Dynamics Laboratory)
단국대학교 건축대학

Dept. of Architectural Engineering, Dankook Univ. School of Architecture, NOTE-2012-ver-20120918

18 September 2012

Contents

1 수학적 모델링과 공학 문제의 해결 (Mathmetical Modeling and Engineering Problem Solving)

- 경험적 방법과 이론적 방법의 고찰
- 수학적 모델의 중요성
- 일반화 작업(generalization)
- 컴퓨터의 활용

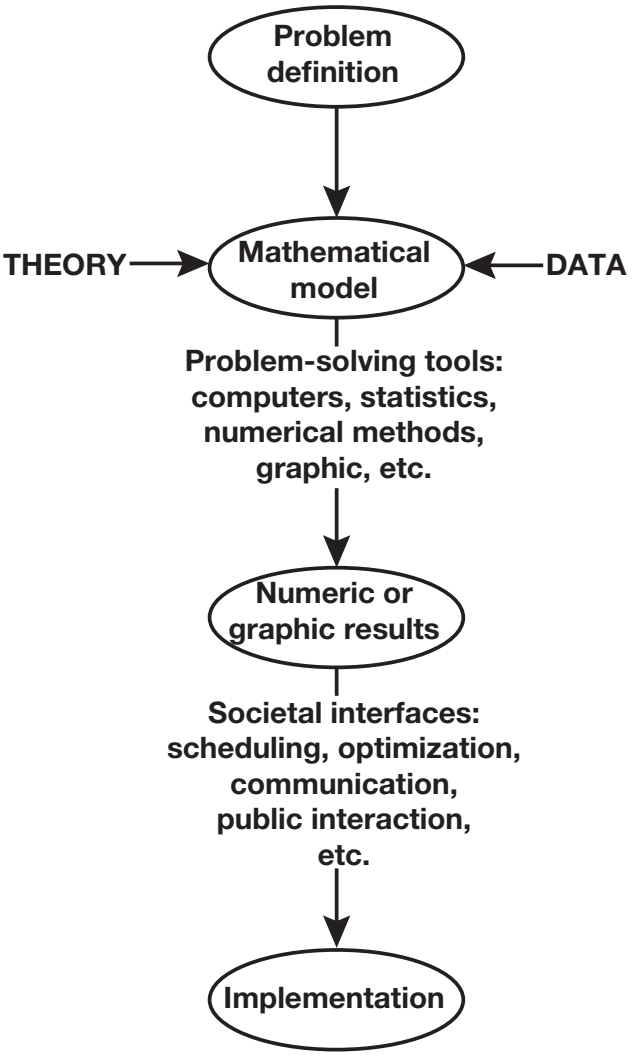

Figure 1: 공학적 문제의 해법과정

1.1 단순화된 수학적 모델

- **수학적 모델(mathematical model)** 물리적 시스템이나 과정의 이해에 필수적으로 요구되는 특징을 수학적으로 표현한 함수의 관계식

$$\text{종속변수} = f\left(\text{독립변수들}, \text{매개변수들}, \text{강제함수들}\right) \tag{1}$$

- **종속변수(dependent variables)** 시스템의 거동이사 상태를 기술하는 특성량
- **독립변수(independent variables)** 시스템의 거동을 결정하는데 사용되는 시간과 공간과 같은 차원
- **매개변수(parameters)** 시스템의 성질이나 구성
- **강제함수(forcing functions)** 시스템에 작용되는 외부의 영향

식(1)의 수학적 표현은, 단순한 대수적 관계식으로부터 매우 복잡한 연립미분방정식에 이르기까지 매우 다양한 적용범위를 가진다. 예를 들어, Newton은 그의 경험에 근거하여, 물체가 가지는 운동량의 시간에 따른 변화는 이에 작용되는 외력의 합과 같다는 제2 운동법칙을 공식화하였다.

$$F = ma \tag{2}$$

여기서 F는 물체에 주어진 유효힘$(N$ 또는 $kg \cdot m/s^2)$이며, m은 물체의 질량(kg)이며 a는 가속도(m/s^2)이다. 특정한 물체에 F의 힘을 가했을 경우 물체의 가속도 a는

$$a = \frac{F}{m} \tag{3}$$

- a : 시스템의 거동을 기술하는 종속변수(dependent variable)
- F : 시스템에 작용되는 외력을 나타내는 강제함수(forcing function)
- m : 시스템의 특성을 나타내는 매개변수(parameter)

a는 시간에따라 공간상에서 어떻게 변화할 것인가?
식(3)은 물리적 시스템을 기술하는 전형적인 수학적 모델이다.

1.1.1 해석해(analytic solution) 예제

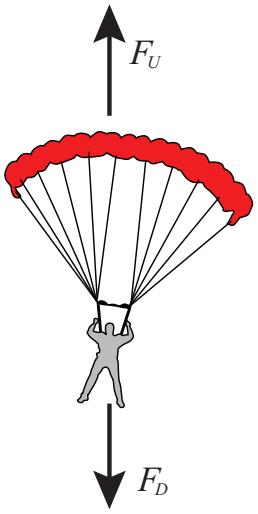

Figure 2: 낙하산병에 작용하는 힘에 대한 개략도. F_D는 중력에 의해 아래로 작용하는 힘이고, F_U는 공기의 저항에 의하여 위로 작용하는 힘이다.

낙하산병 문제로 정확해(exact solution) 구하기

$$\frac{dv}{dt} = \frac{F}{m} \tag{4}$$

$$F = F_D + F_U \tag{5}$$

$$F_U = -cv \tag{6}$$

$$\frac{dv}{dt} = \frac{mg - cv}{m} \tag{7}$$

$$\frac{dv}{dt} = g - \frac{c}{m}v \tag{8}$$

식 7 혹은 식 8의 엄밀해(exact solution)을 구하기 위해서는 미분방정식의 **변수분리(separation of variables)**[1] 방법등이 사용되어야 한다.

$$\frac{m}{mg - cv}dv = 1 \cdot dt \tag{9}$$

양변을 적분하면 식(9)은 시각 T에서 다음과 같이 변형할 수 있다.

$$\int_{v(0)}^{v(T)} \frac{m}{mg - cv}dv = \int_0^T 1 \cdot dt \tag{10}$$

[1] 종속변수에 관련된 항과 독립변수에 관련된 항을 분리시킬 수 있을 때, 그 각각의 변수에 관해 적분하여 해를 구하는 미분방정식 풀이법의 일종

초기속도가 0 즉, $t=0$일때, $v(0)=0$로 가정하고, $mg-cv$를 시간의 종속변수X로 치환하면,

$$mg - cv = X \tag{11}$$

$$\frac{dX}{dv} = -c \tag{12}$$

$$dv = -\frac{1}{c}dx \tag{13}$$

치환변수 X의 초기값과 T에서의 값은 각각, $X(0)=mg$ 그리고 $X(T)=mg-cv(T)$가 된다. 다시 식(10)에 대입하면,

$$-\frac{m}{c}\int_{X(0)}^{X(T)}\frac{1}{X}dv = \int_0^T 1\cdot dt \tag{14}$$

식(14)을 계산하면,

$$-\frac{m}{c}\ln X|_{X(0)}^{X(T)} = T \tag{15}$$

치환된 변수를 환원하면서 전개하면,

$$-\frac{m}{c}\left[\ln\{mg-cv(T)\}-\ln(mg)\right] = T \tag{16}$$

정리하면

$$\ln\left(1-\frac{c}{mg}v(T)\right) = -\frac{c}{m}T \tag{17}$$

양변에 ln을 취하면,

$$e^{-(c/m)T} = 1-\frac{c}{mg}v(T) \tag{18}$$

결국 시각 T에서 엄밀해는 식(21)과 같이 계산된다.

$$v(T) = \frac{mg}{c}\left(1-e^{-(c/m)T}\right) \tag{19}$$

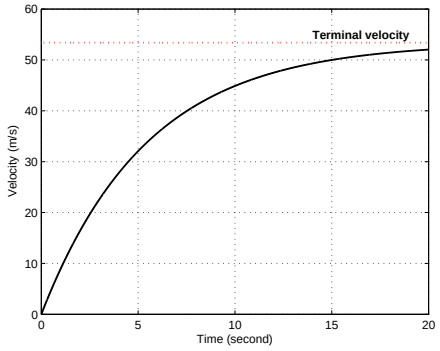

Figure 3: 낙하산병 문제의 해석해, 시간에 따라 속도가 증가하여 종단속도로 접근

1.1.2 수치해법(numerical method) 예제)

수치해법은 단순히 산술연산을 이용하여 해를 얻을 수 있도록 수학적 문제를 재공식화 하는 것이다. 근사화 문제를 생각하면 시간의 증분 즉, Δ가 유한하기 때문에 $dv/dt \cong \Delta v/\Delta t$는 근사식이 된다. 미적분학에서는

$$\frac{dv}{dt} = \lim_{\Delta t \to 0} \frac{\Delta v}{\Delta t} \tag{20}$$

앞서 세운 수학적 모델 식(8)를 생각해보면 다음과 같이 근사화 될 수 있다.

$$\frac{dv}{dt} \cong \frac{\Delta v}{\Delta t} = \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i} = g - \frac{c}{m} v(t_i) \tag{21}$$

위의 식(21)을 다시 정리하면

$$v(t_{i+1}) = v(t_i) + \left[g - \frac{c}{m} v(t_i) \right] (t_{i+1} - t_i) \tag{22}$$

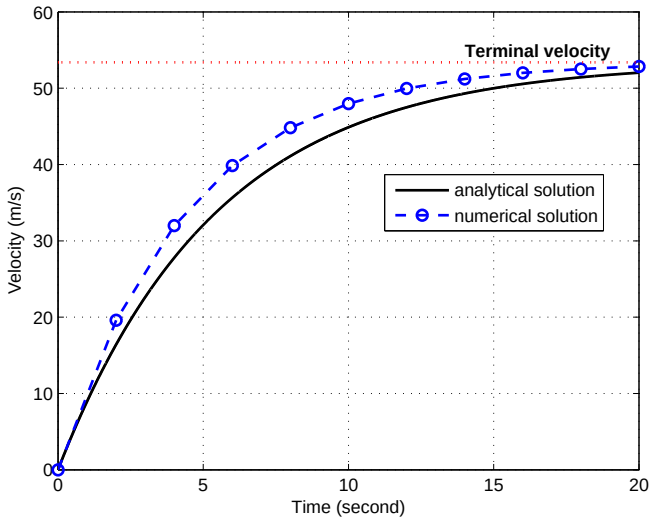

Figure 4: 낙하산병 문제에서의 수치해와 해석해의 비교

★테일러 급수(Taylor Series) 테일러 급수(Taylor Series)는 수치해석에서 매우 중요한 역할을 한다.(4장에서 학습)

Leonhard Euler 1707-1783

$$e^{\pi i} + 1 = 0$$

Isaac Newton 1642-1727

$$\begin{aligned} e &= 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots \\ &\cong 2.718281828459046\cdots \end{aligned}$$

정리 1 *$n \geq 0$ 인 정수 n 에 대하여, 폐구간 $[a,x]$ 에서 n 번 미분가능하고 개구간 (a,x) 에서 $(n+1)$ 번 미분가능한 함수 f 는*

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

로 나타내어질 수 있다. 처음 몇 항까지를 선택함으로써 $x=a$ 주변에서의 $f(x)$ 의 근사식으로 사용할 수 있는데, 이를 **테일러 다항식** ***(Taylor polynomial)*** *이라고 한다. 특히* **선형근사**는 *$n=1$ 인 테일러 급수로 볼 수 있다. $a=0$ 인 경우를 특별히* **맥로린 급수** ***(Maclaurin series)*** *라고 부른다.*

과제1 (제출기한 9월18일)

1 초기속도 $v(0)$ 가 0이 아닌 경우, $v(0)$ 상수를 도입하여 정밀해를 구하라

2 낙하하는 낙하산병에게 작용하는 힘을 계산하는 과정에서, 식 (8) 로 주어진 선형관계식 대신, 다음과 같은 2차식을 사용하여 정밀해 혹은 수치해를 구하여라

$$F_U = -c'v^2$$

단, c' 은 2차항력계수 (kg/m) 이며, 수치해를 구할땐, 10초후의 낙하산병의 속도를 구하라. 여기서, 낙하산병의 질량은 68.1kg이고 2차항력계수는 $0.232kg/m$ 이다.

3 테일러급수 (Taylor series) 를 증명하라.

2 프로그래밍과 소프트웨어 (Programming and Software)

2.1 프로그래밍

컴퓨터 프로그램이란 컴퓨터가 특정 작업을 수행하도록 지시하기 위한 명령어의 집합이다. 많은 개인들이 다양한 종류의 문제를 해결하기 위해 프로그램을 작성하기 때문에, 고급 컴퓨터 언어들은 아주 다양한 기능들을 가지고 있다. 어떤 공학자들은 특정 작업을 수행하기 위해서 컴퓨터 언어가 가지고 있는 모든 기능을 섭렵하여야 하지만, 대부분의 사람들은 공학적 응용 위주의 수치계산을 수행할 수 있을 정도면 충분하다.

- 단순한 정보의 표현[상수, 변수, 형식 선언(type declarations)]
- 효과적인 정보의 표현[데이터 구조, 배열(array), 기록(record)]
- 수학적 공식[지정(assignment), 우선순위 규칙(priority rule), 기본적 함수(intrinsic function)]
- 입력/출력
- 논리적 표현[순차적 진행(sequence), 선택 및 조건분기(selection), 반복(repetition)]
- 모듈프로그래밍(module programming)[함수(function), 서브루틴(subroutine)]

2.1.1 구조화된 프로그램(structured program)

구조적 프로그래밍의 주요 아이디어는 어떤 수치알고리즘도 순차적 진행(sequence), 선택(selection), 반복(repetition)이라는 3개의 기본적 제어구조(control structure)로 이루어질 수 있다.

구조적 프로그래밍 혹은 프로그래밍에 있어 순서도(flowchart)는 알고리즘 논리의 이해를 돕지만, 본 강의에서는 다루지 않는다.

2.1.2 논리적 표현

- **순차적 진행(sequence)** 순차적 구조(sequence structure)는 한번에 하나의 명령을 수행함. 코딩에 있어 위에서 아래구문으로 수행됨
- **선택(selection)** 순차적 구조의 프로그램의 흐름을 논리적 결정(logical condition)으로 분리하거나 분기시킴. (IF/IF-ELSE/CASE등)
- **반복(repetition)** 명령을 반복적으로 수행하는 수단. 흔히 루프(loop)라고 불리는 구문은 decision loop이 일반적이고 반복 구문에서 조건에 의해 반복문을 빠져나가는 위치가 어디에 있는지에 따라 pretset loop, posttest loop으로 불리운다. 그러나 MATLAB은 while 구문으로 decision loop을 제어하며 그 확장성이 많이 좋은편은 아니다. 주로 횟수제어(for-loop)방식을 사용한다.

2.2 모듈프로그래밍

컴퓨터 프로그램은 각각 독립적을 개발되고 검증될 수 있는 여러개의 작은 부프로그램(subprogram)과 모듈(module)로 나눌 수 있다. MATLAB에서는 함수(function)이 그역할을 하며, 여러개의 입력변수에서 여러개의 출력변수를 내보내는등 독립적인 역할을 한다. 모듈프로그래밍은 여러가지 장점을 가지고 있다. 크기가 작고, 스스로 필요한 도구들을 갖춘 구성단위들을 이용하면 그 밑에 깔려있는 논리를 고안해내고, 개발자와 사용자 모두가 이를 쉽게 이해하게 된다. 각각의 모듈이 독립적이면서도 완벽하게 작동하기 때문에 프로그램의 개발 작업도 매우 쉽게 진전될 수 있다.

2.3 라이브러리 & 툴박스(Toolbox)

본 강의에서 쓰일 MATLAB은 쉬운 대화식으로 프로그래밍되고, 행렬연산, 기호연산, 수치적 함수등을 포함하여 공학자들에게 매우 필요한 구조적인 프로그램이다. 그리고 수치해석이나 새로운 공학적 이슈들이 툴박스(Toolbox)의 형태로 배포된다.

3 근사값과 반올림오차 (Approximations and Round-off Errors)

수치해석의 효과적인 사용을 위해서는 오차에 대한 개념을 이해하는 것이 매우 중요하다.

- 수치해법 자체가 근사값을 다루는 것이기 때문에 정확한 해와 불일치, 즉 오차(error)가 항상 발생하게 된다.
- 수치해법으로 완벽한 결과를 얻는 것은 모든 사람이 바라는 목표이나, 실제로 이루기는 매우 어렵다.

3.1 유효숫자(Significant figures)

유효숫자 또는 자릿수의 개념은 수치의 신뢰도를 공식적으로 나타내기 위해 개발되었다. 즉, 유효숫자는 신뢰를 가지고 사용할 수 있는 수치의 개수이며, 정확한 자릿수에 하나의 추정자릿수를 더한 것과 같다. 유효숫자(Significant figures)는 수의 정확도에 영향을 주는 숫자이다. 보통 다음의 경우를 제외하고 모든 숫자는 유효숫자이다.

- 0.00012의 1 앞에 있는 0들처럼 자리수를 표시하기 위한 0
- 유효숫자가 아닌 자리의 숫자와 연산하여 영향받은 자리의 숫자
- 측정 기구의 한계로 정확하지 않은 자리의 숫자

48.50	87,324.35	0.00001845	45,000

과학적 표시법(scientific notatioin) 자리수가 10의 지수로 표현되고 유효숫자만이 10^n 를 곱하는 수로 표현된다.

4.850×10^1	8.732435×10^4	1.845×10^{-5}	4.500×10^4

유효숫자의 계산

- 덧셈과 뺄셈에서, 계산된 결과는 원래 있던 수의 소수점 아래 자리보다 더 낮은 유효숫자를 가질 수 없다. 예를 들어, 유효숫자 세 개인 수 3.14와 유효숫자 5개인 8.9714를 더하면 산술적으로는 12.1114가 나오지만, 3.14에 의해 10^{-2} 자리까지만이 유효한 결과로 판단되어 결과는 12.11이 된다.
- 곱셈과 나눗셈에서, 계산된 결과는 두 측정치 중 유효숫자가 적은 쪽과 같은 유효숫자를 가진다. 예를 들어, 2.56×12.8690의 산술적 계산결과는 4.78464이지만, 2.56의 유효숫자가 3개이므로 유효한 결과는 4.78이다.
- 세 개 이상의 숫자를 연속적으로 계산할 때, 중간의 연산 결과는 그 중간 연산으로 계산이 끝날 때의 유효숫자 개수보다 한 개 더 많다.

3.2 오차의 정의

수치오차는 정확한 수학적 연산을 근사값을 사용하여 표현하기 때문에 발생된다. 이는 정확한 수학적 절차를 근사값으로 표현하기 때문에 발생되는 절단오차(truncation error)와 정확한 수치를 유한한 수의 유효숫자로 표시하기 때문에 발생되는 반올림오차(round-off error or rounding error)를 포함한다.

IEEE 표준연산 (IEEE standard arithmetic)[2]

- **truncation** 유효숫자 이후 단순 모두 버림
  $0.142857 \cong 0.142$ (dropping all significant digits after the third)
- **round to nearest** 유효숫자 이후 반올림
  $0.142857 \cong 0.143$ (rounding the fourth significant digit. This is rounded up because $8 > 5$)
- **round to** $-\infty$ 항상 왼쪽 값을 취함
- **round to** $+\infty$ 항상 오른쪽 값을 취함

참값과 근사값 사이의 관계식은

$$\text{참값} = \text{근사값} + \text{오차} \tag{23}$$

식(23)을 다시 배열하면,

$$E_t = \text{참값} - \text{근사값} \tag{24}$$

이러한 오차의 정의는 수치의 크기에 대한 고려가 없다는 단점이 있다. 이때 다루고 있는 양의 크기를 고려하여 오차를 정의하는 방법중에 식(25)와 같이 오차를 참값으로 정규화하는 방법이다.

$$\text{참상대오차} = \frac{\text{참오차}}{\text{참값}} \tag{25}$$

참값의 백분율 상대오차(true percent relative error)는

$$\varepsilon_t = \frac{\text{참오차}}{\text{참값}} \times 100(\%) \tag{26}$$

수치해석에서는 실제 응용문제의 참값의 알지 못하는 경우가 대부분이며 이와 같은 상황에서의 대안은 오차를 정의할 때 참값을 가장 잘 나타내는 수렴의 오차의 한계로 정규화 시킨다.

$$\varepsilon_a = \frac{\text{현재의근사값} - \text{이전의근사값}}{\text{현재의근사값}} \times 100(\%) \tag{27}$$

3.3 컴퓨터상에서 수의 표현

컴퓨터에서 숫자를 저장하는 방법과 오차는 직접적인 관계가 있다.

- 물리적 측량
- 동적 계측(data acquisition)

[2] 5가지의 표준 근사화 절차는 **부동소수점 연산**에 대한 표준 IEEE Standard for Floating-Point Arithmetic (IEEE 754)에 잘나타나 있다. `http://en.wikipedia.org/wiki/IEEE_754-2008`

- 음향 녹음(recording)

유동소수점 표현(floating point, FP)[3]

컴퓨터 내부에서 수의 표시는 주로 정보가 저장되는 작은 단위 1word의 구조를 살펴봐야한다. 과학적 표기법을 살펴보면

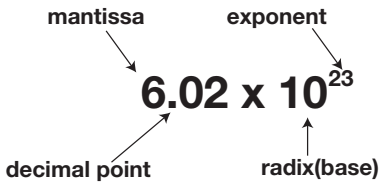

Figure 5: Scientific notation

가수(mantissa or significand)와 지수(exponent or characteristic) 그리고 기저(base)가 사용되며 이것을 2진법 과학적 표기법을 사용하여 컴퓨터 1word에 저장시킨다. 32bit의 유동소수점(floating point)의 표현은 Figure 6와 같다.

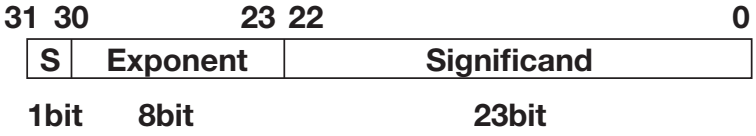

Figure 6: 32bit floating point

IEEE 754 Standard

$$FPnumber = (-1)^S \cdot (1 + Significand) \cdot 2^{Exponent} \tag{28}$$

$$FPnumber = (-1)^S \cdot (1 + Significand) \cdot 2^{Exponent - Bias} \tag{29}$$

[3]참조 `http://www.cise.ufl.edu/~mssz/CompOrg/CDA-arith.html`

A MATLAB m-code

A.1 Figure 3의 낙하산병 문제 정확해와 수치해

```
clear all; clc; close all;

%% Parameters
m=68.1; % kg
c=12.5; % m/sec^2
g=9.8;  % kg/s

%% Calculation
t=0:0.0001:20; % time of exact solution

dt=2;          % dt of numerical solution
t1=0:dt:20;    % time of numerical solution
v=g*m/c*(1-exp(-(c/m)*t)); % exact solution

v1(1)=0;       % initial velocity of numerical solution
for kk=1:length(t1)
    v1(kk+1)=v1(kk)+(g-c/m*v1(kk))*dt;
end

ref=ones(length(t),1)*g*m/c; % terminal velocity

%% Display result
figure
plot(t,v,'-k',t1,v1(1:end-1),'o--b',t,ref,':r','linewidth',1.5)
text(14,55,'Terminal velocity','FontWeight','bold')
grid on
legend('analytical solution','numerical solution','Location','best')
xlabel('Time (second)')
ylabel('Velocity (m/s)')
```

Listing 1: Comparison between exact and numerical solution