공학 수치해석

Numerical Analysis for Engineers

박은천

단국대학교 건축대학

LECTURE NOTE
NOTE-2012-ver-20121113
13 November 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

박은천
단국대학교 건축대학

Dept. of Architectural Engineering, Dankook Univ. School of Architecture, NOTE-2012-ver-20121113

13 November 2012

Table of Contents

I 모델링, 컴퓨터, 오차해석 **4**

1 수학적 모델링과 공학 문제의 해결 (Mathmetical Modeling and Engineering Problem Solving) **4**
1.1 단순화된 수학적 모델 5
1.1.1 해석해(analytic solution) 예제 . 6
1.1.2 수치해법(numerical method) 예제) 8

2 프로그래밍과 소프트웨어 (Programming and Software) **10**
2.1 프로그래밍 10
2.1.1 구조화된 프로그램(structured program) 10
2.1.2 논리적 표현 10
2.2 모듈프로그래밍 11
2.3 라이브러리 & 툴박스(Toolbox) 11

3 근사값과 반올림오차 (Approximations and Round-off Errors) **12**
3.1 유효숫자(Significant figures) 12
3.2 오차의 정의 13
3.3 컴퓨터상에서 수의 표현 13

4 절단오차와 Taylor 급수 (Truncation Errors and The Taylor Series) **15**
4.1 Taylor 급수 15
4.2 Talyor 급수전개에서의 나머지 항 18
4.3 수치미분 19
4.3.1 1차 도함수의 전진차분 근사 . . . 20
4.3.2 1차 도함수의 후진차분 근사 . . . 20
4.3.3 1차 도함수의 중심차분 근사 . . . 20
4.3.4 고차 도함수의 유한차분근사 . . . 22
4.4 오차의 전파 22
4.4.1 단일 변수 함수 22
4.4.2 다중 변수 함수 23
4.4.3 안정성과 조건수 24

5 구간법 (Bracketing Method) **26**
5.1 도식적 방법 26
5.2 이분법 28
5.2.1 종료 판정 기준과 오차추정 29
5.2.2 함수 계산의 최소화 29
5.3 가위치법 30

5.4 수정된 가위치법 32

6 개구간법
(Open Method) **35**
6.1 고정점 반복법(fixed point iteration) . . 35
6.2 Newton-Raphson 법 (Newton-Raphson method) 37
6.3 할선법 (secant method) 39
6.4 수정된 할선법 (modified secant method) 40

II 선형대수 방정식 (Linear Algebraic Equations) **41**

7 Gauss 소거법
(Gauss elimination) **41**
7.1 도식적 방법 42
7.2 행렬식과 Cramer 공식 43
7.2.1 행렬식 43
7.2.2 Cramer 공식 44
7.3 Gauss 소거법 45
7.3.1 미지수의 전진소거(forward elimination) 45
7.3.2 후진대입(back substitution) . . 46
7.4 피벗화 46

8 LU분해법과 역행렬
(LU Decomposition and Matrix Inversion) **49**
8.1 역행렬 51

III 최적화 (Optimization) **53**

9 비구속 최적화
(Unconstrained Optimization) **58**
9.1 황금분할법 58

10 구속 최적화
(Constrained Optimization) **60**
10.1 등식 구속조건 Equality constraints 60
10.1.1 Lagrangian Methods 60
10.2 등식과 부등식 구속조건 (Equality and Inequality Constraints) 62
10.3 선형계획모형 해법 64
10.4 패키지를 사용한 최적화 (MATLAB Optimization Toolbox Functions) 67
10.4.1 Linear programming `lingprog()` 67

IV **69**

Appendices **69**

A MATLAB m-code **69**
A.1 Figure 1.3의 낙하산병 문제 정확해와 수치해 69
A.2 무한히 미분 가능한 함수를 근사하기 위한 Taylor 급수전개 70
A.3 예제풀이 5.6 : 이분법 및 가위치법 71

B 과제풀이 **74**
B.1 과제 1 74
B.1.1 낙하산병의 초기속도 $v(0)$가 0이 아닌 경우, $v(0)$ 상수를 도입하여 정밀해를 구하라 74
B.1.2 낙하하는 낙하산병에게 작용하는 힘을 계산하는 과정에서, 식(B.1)로 주어진 선형관계식 대신, 다음과 같은 2차식을 사용하여 정밀해 혹은 수치해를 구하여라 75
B.1.3 테일러급수(Taylor series)를 증명하라. 77

C 중간고사 예제 **79**
C.1 CHAPTER 1. 수학적 모델링과 공학 문제의 해결 79
C.1.1 연습문제 1.18 79

C.2 CHAPTER 4. 절단오차와 Taylor 급수 . 79
C.2.1 연습문제 4.5 79
C.2.2 연습문제 4.10 79
C.3 CHAPTER 5. 구간법 79
C.3.1 연습문제 5.2 79
C.3.2 연습문제 5.7 80
C.3.3 연습문제 5.20 80
C.4 CHAPTER 6. 개구간법 80
C.4.1 연습문제 6.2 80
C.4.2 연습문제 6.11 81
C.5 문제 답안 83
C.5.1 연습문제 1.18(a) 83
C.5.2 연습문제 1.18(b) 83
C.5.3 연습문제 4.5 83
C.5.4 연습문제 4.10(a) 84
C.5.5 연습문제 4.10(b) 84
C.5.6 연습문제 5.2(a) 84
C.5.7 연습문제 5.2(b) 85
C.5.8 연습문제 5.7(b) 85
C.5.9 연습문제 5.20 85
C.5.10 연습문제 6.2(a) 86
C.5.11 연습문제 6.2(b) 86
C.5.12 연습문제 6.2(c) 86
C.5.13 연습문제 6.11 86
C.5.14 알고리즘 테이블 88

Part I

모델링, 컴퓨터, 오차해석

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

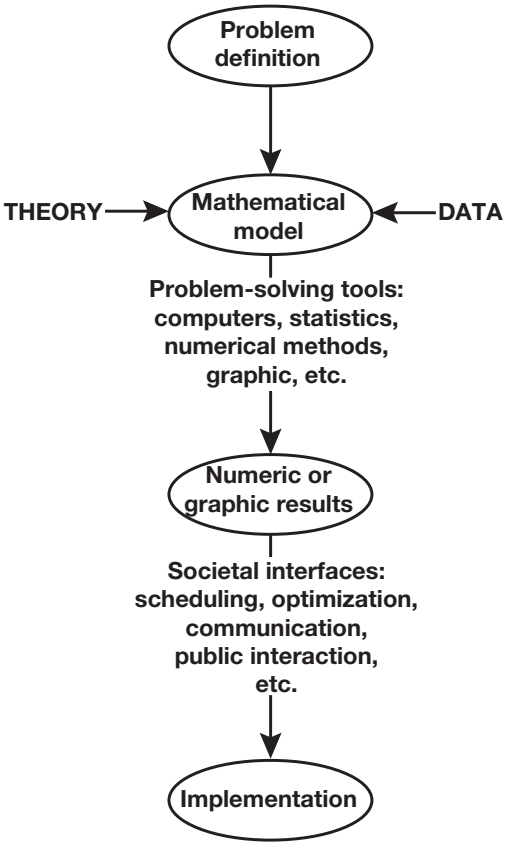


Figure 1.1: 공학적 문제의 해법과정

1.1 단순화된 수학적 모델

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

$$\text{종속변수} = f(\text{독립변수들}, \text{매개변수들}, \text{강제함수들}) \tag{1.1}$$

- 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(1.1)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.

$$F = ma \tag{1.2}$$

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

$$a = \frac{F}{m} \tag{1.3}$$

- 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(1.3)은 물리적 시스템을 기술하는 전형적인 수학적 모델이다.

1.1.1 해석해(**analytic solution**) 예제

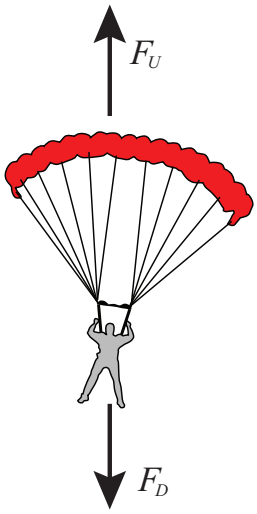


Figure 1.2: 낙하산병에 작용하는 힘에 대한 개략도. F_D는 중력에 의해 아래로 작용하는 힘이고, F_U는 공기의 저항에 의하여 위로 작용하는 힘이다.

낙하산병 문제로 정확해(**exact solution**) 구하기

$$\frac{dv}{dt} = \frac{F}{m} \tag{1.4}$$

$$F = F_D + F_U \tag{1.5}$$

$$F_U = -cv \tag{1.6}$$

$$\frac{dv}{dt} = \frac{mg - cv}{m} \tag{1.7}$$

$$\frac{dv}{dt} = g - \frac{c}{m}v \tag{1.8}$$

식 1.7 혹은 식 1.8의 엄밀해(exact solution)을 구하기 위해서는 미분방정식의 변수분리(**separation of variables**)[1] 방법등이 사용되어야 한다.

$$\frac{m}{mg - cv}dv = 1 \cdot dt \tag{1.9}$$

양변을 적분하면 식(1.9)은 시각 T에서 다음과 같이 변형할 수 있다.

$$\int_{v(0)}^{v(T)} \frac{m}{mg - cv}dv = \int_0^T 1 \cdot dt \tag{1.10}$$

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

$$mg - cv = X \tag{1.11}$$

$$\frac{dX}{dv} = -c \tag{1.12}$$

$$dv = -\frac{1}{c}dx \tag{1.13}$$

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

치환변수 X 의 초기값과 T 에서의 값은 각각, $X(0) = mg$ 그리고 $X(T) = mg - cv(T)$ 가 된다. 다시 식(1.10)에 대입하면,

$$-\frac{m}{c}\int_{X(0)}^{X(T)}\frac{1}{X}dv = \int_0^T 1\cdot dt \tag{1.14}$$

식(8.9)을 계산하면,

$$-\frac{m}{c}\ln X|_{X(0)}^{X(T)} = T \tag{1.15}$$

치환된 변수를 환원하면서 전개하면,

$$-\frac{m}{c}\left[\ln\{mg - cv(T)\} - \ln(mg)\right] = T \tag{1.16}$$

정리하면

$$\ln\left(1 - \frac{c}{mg}v(T)\right) = -\frac{c}{m}T \tag{1.17}$$

양변에 ln을 취하면,

$$e^{-(c/m)T} = 1 - \frac{c}{mg}v(T) \tag{1.18}$$

결국 시각 T 에서 엄밀해는 식(1.21)과 같이 계산된다.

$$v(T) = \frac{mg}{c}\left(1 - e^{-(c/m)T}\right) \tag{1.19}$$

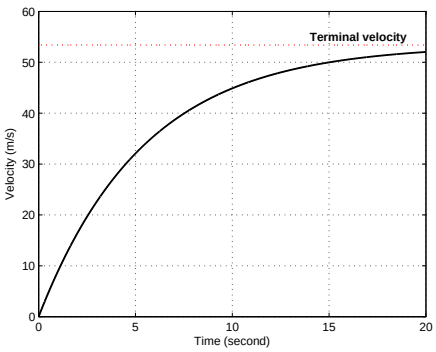


Figure 1.3: 낙하산병 문제의 해석해, 시간에 따라 속도가 증가하여 종단속도로 접근

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

$$\frac{dv}{dt} = \lim_{\Delta t \to 0} \frac{\Delta v}{\Delta t} \tag{1.20}$$

앞서 세운 수학적 모델 식(1.8)를 생각해보면 다음과 같이 근사화 될 수 있다.

$$\frac{dv}{dt} \cong \frac{\Delta v}{\Delta t} = \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i} = g - \frac{c}{m} v(t_i) \tag{1.21}$$

위의 식(1.21)을 다시 정리하면

$$v(t_{i+1}) = v(t_i) + \left[g - \frac{c}{m} v(t_i) \right] (t_{i+1} - t_i) \tag{1.22}$$

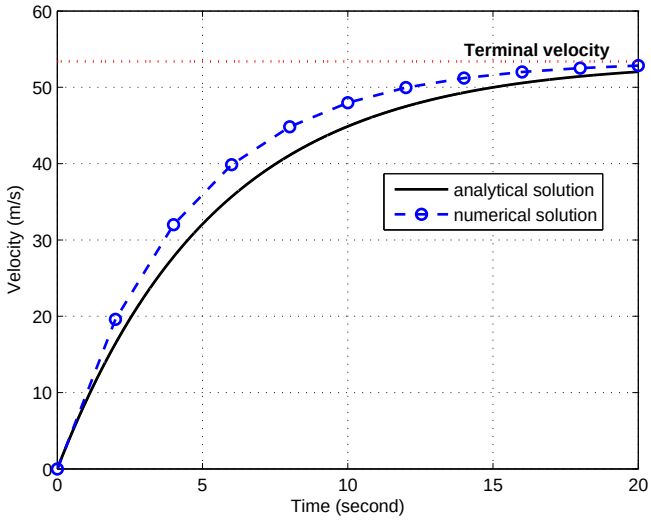


Figure 1.4: 낙하산병 문제에서의 수치해와 해석해의 비교

★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

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

Theorem 1.1. *$n \geq 0$인 정수 n에 대하여, 폐구간$[a,x]$에서 n번 미분가능하고 개구간(a,x)에서 $(n+1)$번 미분가능한 함수f는*

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \tag{1.23}$$

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* 선형근사는 $n=1$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(1.8)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

$$\text{참값} = \text{근사값} + \text{오차} \tag{3.1}$$

식(3.1)을 다시 배열하면,

$$E_t = \text{참값} - \text{근사값} \tag{3.2}$$

이러한 오차의 정의는 수치의 크기에 대한 고려가 없다는 단점이 있다. 이때 다루고 있는 양의 크기를 고려하여 오차를 정의하는 방법중에 식(3.3)와 같이 오차를 참값으로 정규화하는 방법이다.

$$\text{참상대오차} = \frac{\text{참오차}}{\text{참값}} \tag{3.3}$$

참값의 백분율 상대오차(true percent relative error)는

$$\varepsilon_t = \frac{\text{참오차}}{\text{참값}} \times 100(\%) \tag{3.4}$$

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

$$\varepsilon_a = \frac{\text{현재의근사값} - \text{이전의근사값}}{\text{현재의근사값}} \times 100(\%) \tag{3.5}$$

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 연산에 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

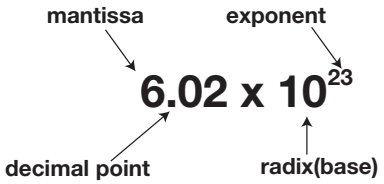


Figure 3.1: Scientific notation

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 3.2와 같다.

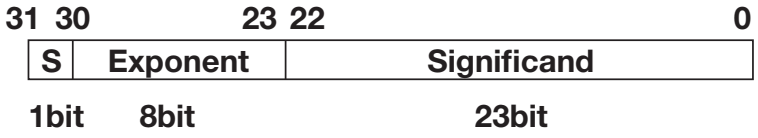


Figure 3.2: 32bit floating point

IEEE 754 Standard

$$FPnumber = (-1)^S \cdot (1 + Significand) \cdot 2^{Exponent} \tag{3.6}$$

$$FPnumber = (-1)^S \cdot (1 + Significand) \cdot 2^{Exponent - Bias} \tag{3.7}$$

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

4 절단오차와 **Taylor** 급수
(Truncation Errors and The Taylor Series)

4.1 **Taylor** 급수

Taylor 정리(Taylor theorem)와 이에 관련된 Taylor급수는 수치해법의 학습에 매우 중요한 가치를 갖는다. **Taylor** 급수는 함수값 및 그의 도함수값들을 사용하여 예측하는 방법을 제공한다.

만일 함수 f와 그것의 처음 $(n+1)$차까지의 미분이 a와 x를 포함하는 구간에서 연속적이라면, x에서의 함수값은 다음과 같이 주어진다.

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 + \cdots + \frac{f^{(n)}}{n!}(x-a)^n + R_n \tag{4.1}$$

여기서 나머지(remainder) R_n은 다음과 같이 정의된다.

$$R_n = \int_a^x \frac{(x-t)^n}{n!} f^{(n+1)}(t)dt \tag{4.2}$$

여기서 t는 모조변수(dummy variable)이다.

Taylor 급수를 통해 통찰력을 얻는 유용한 방법은 항을 추가해보면서 급수를 구성해보는 방법이다.
Taylor급수에서 x를 이전 함수값에 의한 새로운 위치 즉 x_{i+1}, 그리고 a를 이전 함수값의 위치 x_i로 생각해보자.

$$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1}-x_i) + \frac{f''(x_i)}{2!}(x_{i+1}-x_i)^2 + \frac{f^{(3)}(x_i)}{3!}(x_{i+1}-x_i)^3 + \cdots + \frac{f^{(n)}}{n!}(x_{i+1}-x_i)^n + R_n \tag{4.3}$$

- 0차근사(zero-order approximation) : $\boxed{f(x_{i+1}) \cong f(x_i)}$
  새로운 위치에서의 함수값은 이전의 위치에서의 함수값과 같다.
- 1차근사(first order approximation) : $\boxed{f(x_{i+1}) \cong f(x_i) + f'(x_i)(x_{i+1}-x_i)}$
  이전의 위치에서의 함수값과 기울기 $f'(x_i)$에 이전위치 x_i와 현재 위치 x_{i+1}사이의 거리가 곱해져서 얻어진다. 즉 이 표현은 직선의 형태를 가지며, x_i와 x_{i+1}사이의 구간에서 함수의 증가나 감소를 예측할 수 있다.
- 2차근사(second order approximation) : $\boxed{f(x_{i+1}) \cong f(x_i) + f'(x_i)(x_{i+1}-x_i) + \frac{1}{2!}f''(x_i)(x_{i+1}-x_i)^2}$
  1차근사는 직선 또는 선형의 경우에만 정확하다. 2차항을 추가하면서 함수가 가지는 곡률을 잡아낼 수 있다.

완전한 Taylor 급수의 전개는 식(4.3)이고 나머지 항은 $(n+1)$부터 무한대까지의 항을 나타낸다.

$$R_n = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x_{i+1}-x_i)^{n+1} \tag{4.4}$$

구간의 크기 $h = x_{i+1} - x_i$의 정의로 Taylor 급수를 단순화하여 식(4.3)을 다음과 같이 나타내면 편리할 때가 많다.

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f^{(3)}(x_i)}{3!}h^3 + \cdots + \frac{f^{(n)}}{n!}h^n + R_n \tag{4.5}$$

여기서 나머지 항은 다음과 같이 나타낸다.

$$R_n = \frac{f^{(n+1)}(\xi)}{(n+1)!}h^{n+1} \tag{4.6}$$

예제 다항식에 대한 **Taylor** 급수 근사

0차부터 4차까지의 Taylor 급수전개를 사용하여, $x_i = 0$에서부터 $h = 1$로 하여 다음의 함수를 근사하라

$$f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2 \tag{4.7}$$

즉, $x_{i+1} = 1$에서의 함수값을 예측하라.

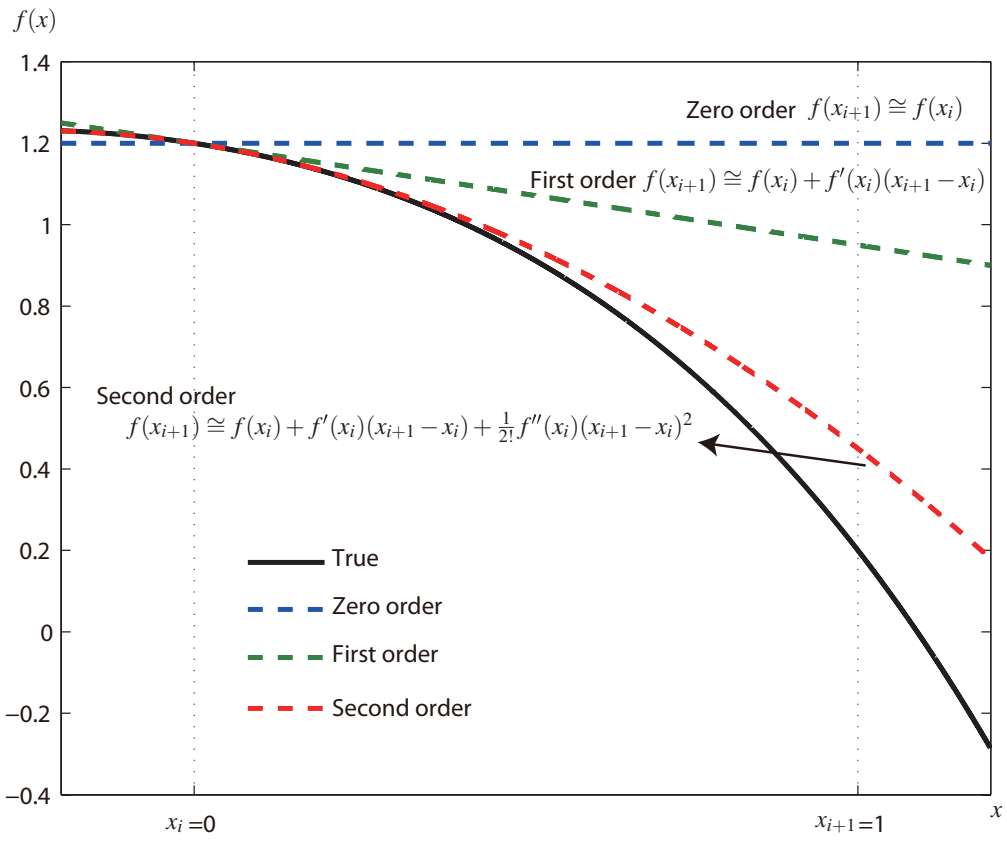


Figure 4.1: 0차, 1차, 2차 Taylor 급수전개를 이용한 $x = 1$에서의 함수 $f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$의 근사

해 함수의 형태를 알 고 있으므로, $f(0) = 1.2$로부터 시작하고 $f(1) = 0.2$의 값을 얻는다. 따라서 예측하고자 하는 참값은 0.2가 된다.

zero-order

$n=0$인 경우 Taylor 급수 근사

$$f(x_{i+1}) \cong 1.2$$

즉, $f(1) \cong 1.2$가 된다. 절단오차를 계산하면 $x=1$에서

$$E_t = 0.2 - 1.2 = -1.0$$

first order

$n=1$인 경우, $x=0$에서의 1차 도함수를 구하면,

$$\begin{aligned} f'(x) &= -0.4x^3 - 0.45x^2 - 1.0x - 0.25 \\ f'(0) &= -0.25 \end{aligned}$$

따라서 1차 근사식은

$$f(x_{i+1}) \cong 1.2 - 0.25h$$

즉, $f(1) \cong 0.95$가 된다. 절단오차를 계산하면 $x=1$에서

$$E_t = 0.2 - 0.95 = -0.75$$

second order

$n=2$인 경우, $x=0$에서의 2차 도함수를 구하면,

$$\begin{aligned} f''(x) &= -1.2x^2 - 0.9x - 1.0 \\ f''(0) &= -1.0 \end{aligned}$$

따라서 2차도함수값에 $1/2!$를 곱하여 추가한 2차 근사식은

$$f(x_{i+1}) \cong 1.2 - 0.25h - 0.5h^2$$

즉, $f(1) \cong 0.45$가 된다. 절단오차를 계산하면 $x=1$에서

$$E_t = 0.2 - 0.45 = -0.25$$

third order

$n=3$인 경우, $x=0$에서의 3차 도함수를 구하면,

$$\begin{aligned} f^{(3)}(x) &= -2.4x - 0.9 \\ f^{(3)}(0) &= -9.0 \end{aligned}$$

따라서 3차도함수값에 $1/3!$를 곱하여 추가한 3차 근사식은

$$f(x_{i+1}) \cong 1.2 - 0.25h - 0.5h^2 - 0.15h^3$$

즉, $f(1) \cong 0.3$이 된다. 절단오차를 계산하면 $x=1$에서

$$E_t = 0.2 - 0.3 = -0.1$$

fourth order

$n=4$인 경우, $x=0$에서의 4차 도함수를 구하면,

$$\begin{aligned} f^{(4)}(x) &= -2.4 \\ f^{(4)}(0) &= -2.4 \end{aligned}$$

따라서 4차도함수값에 $1/4!$를 곱하여 추가한 4차 근사식은

$$f(x_{i+1}) \cong 1.2 - 0.25h - 0.5h^2 - 0.15h^3 - 0.1h^4$$

즉, $f(1) \cong 0.2$이 된다. 절단오차를 계산하면 $x=1$에서

$$E_t = 0.2 - 0.2 = 0$$

이때 나머지항을 살펴보자.

$$R_4 = \frac{f^{(5)}(\xi)}{5!}h^5 = 0$$

이는 4차 다항식의 5차 미분항은 항상 0이 되기 때문에, 정확한 값을 얻을 수 있다.

4.2 Talyor 급수전개에서의 나머지 항

많은 경우에서 Taylor 급수의 실용적 가치는, 단지 몇 개의 항들만을 포함시키는 것만으로도 참값에 충분히 근접한 결과(공학적 측면에서)를 얻을 수 있다는 것이다. 충분히 근접한값을 얻기 위해 얼마나 많은 항들이 필요한 지에 대한 평가는, 급수의 나머지 항을 기초로 하여 결정된다.

$$R_n = \frac{f^{(n+1)}(\xi)}{(n+1)!}h^{n+1}$$

그러나 식(4.4)를 찾으려면, x_i와 x_{i+1} 사이에 놓인ξ를 알아야 하고, $f(x)$의 $(n+1)$차 도함수를 알아야하는데, 이것을 안다면 굳이 Taylor 급수전개를 할 필요가 없다. 모순

$x_i = \pi/4$에서의 $f(x)$값을 이용하여, $x_{i+1} = \pi/3$에서의 $f(x) = \cos x$를 근사하기 위해 Taylor급수전개를 6차까지 수행하면, (여기서 $h = \pi/3 - \pi/4$) **MATLAB** 코드 **A.2**장 **(70**페이지**)**

Order n	$f^{(n)}(x)$	$f(\pi/3)$	E_t
0	$\cos x$	0.707106781	-41.4
1	$-\sin x$	0.521986659	-4.4
2	$-\cos x$	0.497754491	0.449
3	$\sin x$	0.499869147	2.62×10^{-2}
4	$\cos x$	0.500007551	-1.51×10^{-3}
5	$-\sin x$	0.500000304	-6.08×10^{-5}
6	$-\cos x$	0.499999988	2.44×10^{-6}

4.3 수치미분

Taylor급수전개 식(4.3)에서 x를 시간함수로 생각해보면,

$$f(t_{i+1}) = f(t_i) + f'(t_i)(t_{i+1}-t_i) + \frac{f''(t_i)}{2!}(t_{i+1}-t_i)^2 + \frac{f^{(3)}(t_i)}{3!}(t_{i+1}-t_i)^3 + \cdots + \frac{f^{(n)}}{n!}(t_{i+1}-t_i)^n + R_n \tag{4.8}$$

1차 도함수의 항 뒤에 있는 모든 항을 R_1로 표현하면 다음과 같다.

$$f(t_{i+1}) = f(t_i) + f'(t_i)(t_{i+1}-t_i) + R_1 \tag{4.9}$$

따라서 식(4.9)을 사용하여 다음 도함수의 근사식을 구할 수 있다.

$$\star f'(t_i) = \frac{f(t_{i+1})-f(t_i)}{t_{i+1}-t_i} - \frac{R_1}{t_{i+1}-t_i} \tag{4.10}$$

Tayler급수의 절단오차 R_1은 식(4.4)에 의해,

$$R_1 = \frac{f''(\xi)}{2!}(t_{i+1}-t_i)^2 \tag{4.11}$$

따라서, 수치미분의 절단오차를 추정값은,

$$\frac{R_1}{t_{i+1}-t_i} = \frac{f''(\xi)}{2!}(t_{i+1}-t_i) \tag{4.12}$$

$$= O(t_{i+1}-t_i) \tag{4.13}$$

즉, 절단오차는 시간간격에 비례하게 된다.

식(4.10)는 수치해법에서 유한제차분 **(finite divided difference)**이라고 불린다. 일반적으로 표현하면,

$$f'(x_i) = \frac{f(x_{i+1})-f(x_i)}{x_{i+1}-x_i} + O(x_{i+1}-x_i) \tag{4.14}$$

$$= \frac{\Delta f_i}{h} + O(h) \tag{4.15}$$

4.3.1 1차 도함수의 전진차분 근사

여기서, Δf_i는 전진차분**(first forward difference)**이고, h는 구간 간격의 크기. 즉 근사값이 구해지는 구간의 길이이다. h를 구간의 크기(절대값)로 표시되기 때문에 i를 시작으로 $i+1$에 도달하게 된다. 전체 항 $\Delta f/h$는 **1**차 유한제차분**(first finite divided difference)**라고 한다.

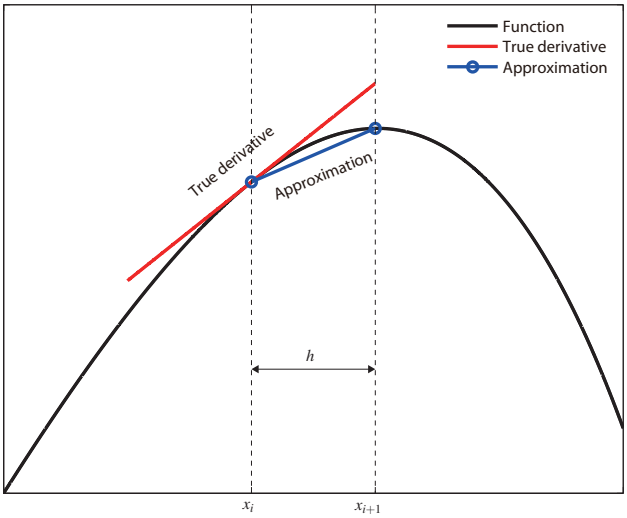


Figure 4.2: 전진 유한제차분 근사

4.3.2 1차 도함수의 후진차분 근사

현재 위치에서의 값을 기초로 하여 이전 값을 계산하기 위해 Taylor 급수를 다음과 같이 후진전개(backward expansion)한다.

$$f(x_{i-1}) = f(x_i) - f'(x_i)h + \frac{f''(x_i)}{2!}h^2 - \cdots \tag{4.16}$$

1차 도함수 이상의 항들을 절단하고, 이를 재배열하면,

$$f'(x_i) \cong \frac{f(x_i) - f(x_{i-1})}{h} = \frac{\nabla f_1}{h} \tag{4.17}$$

여기서 오차는 $O(h)$이고 ∇f_1는 **1**차 후진차분**(backward difference)**라고 한다.

4.3.3 1차 도함수의 중심차분 근사

1차 도함수를 근사적으로 구하기 위한 세번째 방법은 다음과 같이 주어진

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \cdots \tag{4.18}$$

전진 Taylor 급수전개에서 식(4.16)를 뺌으로써 얻을 수 있다.

$$f(x_{i+1}) = f(x_{i-1}) + 2f'(x_i)h + \frac{2f^{(3)}(x_i)}{3!}h^3 + \cdots \tag{4.19}$$

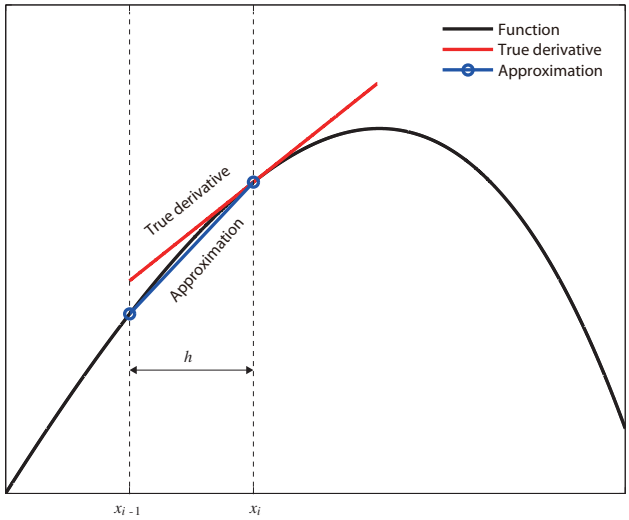


Figure 4.3: 후진 유한제차분 근사

이를 $f'(x_{i+1})$에 대해 정리하면

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - \frac{f^{(3)}(x_i)}{6}h^2 - \cdots \tag{4.20}$$

$$= \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - O(h^2) \tag{4.21}$$

식(4.21)는 1차 도함수에 대한 중심차분(centered difference)의 표현이다. 절단오차의 경우. $O(h)$인 전진 또는 후진 차분과는 달리 $O(h^2)$가 된다. 따라서 중심차분이 수치미분값을 나타내는데 더 정확한 표현이라는 것을 나타내준다.

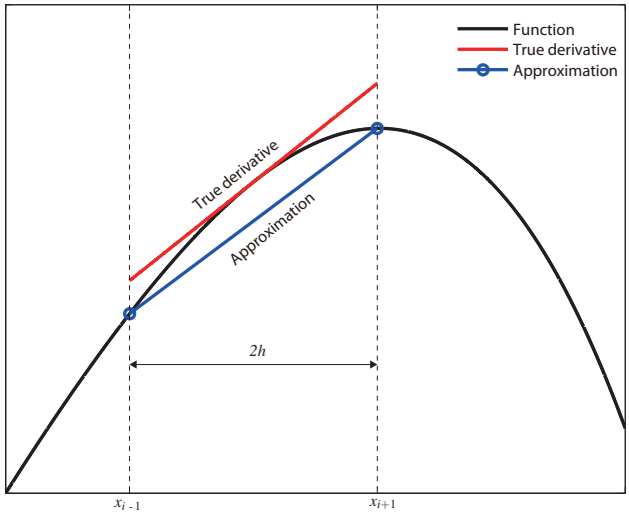


Figure 4.4: 중심 유한제차분 근사

4.3.4 고차 도함수의 유한차분근사

Taylor 급수전개를 사용하여 고차 도함수의 수치적 근사식을 유도할 수 있다. $f(x_{i+2})$를 위한 Taylor 급수전개를 $f(x_i)$을 기준으로 수행하면,

$$f(x_{i+2}) = f(x_i) + f'(x_i)(2h) + \frac{f''(x_i)}{2!}(2h)^2 + \cdots \tag{4.22}$$

식(4.18)에 2를 곱한후, 식(4.22)에서 빼면,

$$f(x_{i+2}) - 2f(x_{i+1}) = -f(x_i) + f''(x_i)h^2 + \cdots \tag{4.23}$$

즉 함수 $f(x)$에 대하여 x_i에서의 2차도함수는

$$f''(x_i) = \frac{f(x_{i+2}) - 2f(x_{i+1}) + f(x_i)}{h^2} + O(h) \tag{4.24}$$

이 관계식은 **2**차 전진유한제차분**(second forward finite divided difference)**이라고 한다. 비슷한 과정을 통해 다음과 같은 후진차분식과,

$$f''(x_i) = \frac{f(x_i) - 2f(x_{i-1}) + f(x_{i-2})}{h^2} + O(h) \tag{4.25}$$

다음의 중심차분식을 얻을 수 있다.

$$f''(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2} + O(h^2) \tag{4.26}$$

1차 유한제차분식 처럼 절단오차의 Big-O notation $O(h^2)$으로 쓰기때문에 2차도함수의 중심차분근사가 더 정확한 결과를 얻는다.

4.4 오차의 전파

근사값에 의해 함수값들의 오차가 어떻게 전파되는지 파악하기 위해 오차의 전파를 해석해야 한다.

4.4.1 단일 변수 함수

단일 독립변수 x의 함수 $f(x)$가 있다고 가정하자. $\tilde{x}$가 x의 근사값이라고 가정할 때, x와 $\tilde{x}$의 오차가 함수값에 미치는 영향을 구해보자.

$$\Delta f(\tilde{x}) = |f(x) - f(\tilde{x})| \tag{4.27}$$

$\Delta f(\tilde{x})$의 값을 구하기 위한 문제는 x를 모르기 때문에 $f(x)$를 알 수 없는 문제가 있다 하지만, $\tilde{x}$가 x에 매우 가깝고, $f(\tilde{x})$가 연속이며 미분가능하면 Taylor급수 전개($f(\tilde{x})$에 인접한 $f(x)$)를 통해 함수값의 오차를 계산할 수 있다.

$$f(x) = f(\tilde{x}) + f'(\tilde{x})(x - \tilde{x}) + \frac{f''(\tilde{x})}{2!}(x - \tilde{x})^2 + \cdots \tag{4.28}$$

2차와 그 이상의 고차항들을 절단하고 결과를 재배열하면,

$$f(x) - f(\tilde{x}) \cong f'(\tilde{x})(x - \tilde{x}) \tag{4.29}$$

또는

$$\Delta f(\tilde{x}) = \left|f'(\tilde{x})\right| \Delta\tilde{x} \tag{4.30}$$

여기서, $\Delta f(\tilde{x}) = |f(x) - f(\tilde{x})|$는 함수의 추정오차이며, $\Delta\tilde{x} = |x - \tilde{x}|$는 x에 대한 오차의 추정값이다. 식(4.30)는 함수의 도함수 및 독립변수의 추정오차값이 주어진 경우, $f(x)$의 오차의 근사값을 계산할 수 있도록 해준다.

4.4.2 다중 변수 함수

같은 방식으로 독립변수가 1개 이상의 경우의 함수에도 일반화될 수 있다. 즉, Taylor급수의 다중 변수 형태를 사용하여 얻을 수 있다. 함수가 두 개의 독립변수 u와 v의 함수라면, Taylor 급수는 다음과 같이 쓸 수 있다.

$$\begin{aligned} f(u_{i+1},v_{i+1}) =& f(u_i,v_i)+\frac{\partial f}{\partial u}(u_{i+1}-u_i)+\frac{\partial f}{\partial v}(v_{i+1}-v_i) \\ &+\frac{1}{2!}\left[\frac{\partial^2 f}{\partial u^2}(u_{i+1}-u_i)^2+2\frac{\partial^2 f}{\partial u\partial v}(u_{i+1}-u_i)(v_{i+1}-v_i)+\frac{\partial^2 f}{\partial v^2}(v_{i+1}-v_i)^2\right]+\cdot \end{aligned} \tag{4.31}$$

여기서 모든 편도함수들을 기준점 i에서 구한다. 만일 2차와 그 이상의 고차 항들을 버리면, 식(4.31)은 다음의 식으로 풀 수 있다.

$$\Delta f(\tilde{u},\tilde{v}) = \left|\frac{\partial f}{\partial u}\right|\Delta\tilde{u}+\left|\frac{\partial f}{\partial v}\right|\Delta\tilde{v} \tag{4.32}$$

여기서 $\Delta\tilde{u}$와 $\Delta\tilde{v}$는 각각 u와 v의 추정오차값이다. n개의 독립변수, $\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_n$가 $\Delta\tilde{x}_1,\Delta\tilde{x}_2,\cdots,\Delta\tilde{x}_n$의 오차를 갖는다면, 다음과 같은 일반적인 관계식으로 쓸 수 있다.

$$\Delta f(\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_n) \cong \left|\frac{\partial f}{\partial x_1}\right|\Delta\tilde{x}_1+\left|\frac{\partial f}{\partial x_2}\right|\Delta\tilde{x}_2+\cdots+\left|\frac{\partial f}{\partial x_n}\right|\Delta\tilde{x}_n \tag{4.33}$$

예제 다중 변수 함수에서의 오차의 전파

배의 돛대의 위 끝에서의 휨(deflection) y는 다음과 같다.

$$y=\frac{FL^4}{8EI}$$

여기서 F는 균일한 측면 하중(N/m), L은 높이(m), E는 탄성계수(N/m^2), I는 관성모멘트(m^4)이다. 다음의 데이터를 사용하여, y의 오차를 추측하라.

$$\begin{aligned} \tilde{F} &= 750N/m & \Delta\tilde{F} &= 30N/m \\ \tilde{L} &= 9m & \Delta\tilde{L} &= 0.03m \\ \tilde{E} &= 7.5\times10^9 N/m^2 & \Delta\tilde{E} &= 5\times10^7 N/m^2 \\ \tilde{I} &= 0.0005m^4 & \Delta\tilde{I} &= 0.000005m^4 \end{aligned}$$

해 원래 휨식에 측정값을 대입하여 구한 근사값은,

$$\begin{aligned} \tilde{y} &= \frac{750\times9^4}{8\times(7.5\times10^9)\times0.0005} \\ &= 1.64025\times10^{-1} \end{aligned}$$

추정된 변형의 오차를 구하기 위해, 식(4.33)을 사용하면,

$$\begin{aligned} \Delta y(\tilde{F},\tilde{L},\tilde{E},\tilde{I}) &= \left|\frac{\partial y}{\partial F}\right|\Delta\tilde{F}+\left|\frac{\partial y}{\partial L}\right|\Delta\tilde{L}+\left|\frac{\partial y}{\partial E}\right|\Delta\tilde{E}+\left|\frac{\partial y}{\partial I}\right|\Delta\tilde{I} \\ \Delta y(\tilde{F},\tilde{L},\tilde{E},\tilde{I}) &\cong \frac{\tilde{L}^4}{8\tilde{E}\tilde{I}}\Delta\tilde{F}+\frac{\tilde{F}\tilde{L}^3}{2\tilde{E}\tilde{I}}\Delta L+\frac{\tilde{F}\tilde{L}^4}{8\tilde{E}^2\tilde{I}}\Delta\tilde{E}+\frac{\tilde{F}\tilde{L}^4}{8\tilde{E}\tilde{I}^2}\Delta\tilde{I} \end{aligned}$$

값을 대입하여 추정된 오차를 구하면

$$\Delta y = 0.006561 + 0.002187 + 0.001094 + 0.00164 = 0.011482$$

즉, $y = 0.164025 \pm 0.011482$의 결과가 나온다. y의 최소값은 0.152543, y의 최대값은 0.175507라고 추정할 수 있는데, 이 값이 타당한지 알아보기 위해 변형식의 분자에 최대값들을 대입하고 분모에 최소값들을 대입하여 최대값 $y_{\max}$와 분자에 최소값들을 대입하고 분모에 최대값들을 대입하여 최소값 $y_{\min}$을 찾아보면

$$y_{\min} = \frac{720 \times 8.97^4}{8 \times (7.55 \times 10^9) \times 0.000505} = 0.152818$$
$$y_{\max} = \frac{780 \times 9.03^4}{8 \times (7.45 \times 10^9) \times 0.000495} = 0.175790$$

휨변형에 대한 오차영향 검토

$$\begin{aligned}\Delta y(\tilde{F},\tilde{L},\tilde{E},\tilde{I}) &\cong \frac{\tilde{L}^4}{8\tilde{E}\tilde{I}}\Delta\tilde{F} + \frac{\tilde{F}\tilde{L}^3}{2\tilde{E}\tilde{I}}\Delta L + \frac{\tilde{F}\tilde{L}^4}{8\tilde{E}^2\tilde{I}}\Delta\tilde{E} + \frac{\tilde{F}\tilde{L}^4}{8\tilde{E}\tilde{I}^2}\Delta\tilde{I} \\ &= 2.187 \times 10^{-4}\Delta\tilde{F} + 7.29 \times 10^{-2}\Delta\tilde{L} + 2.187 \times 10^{-11}\Delta\tilde{E} + 3.2805 \times 10^{2}\Delta\tilde{I}\end{aligned}$$

4.4.3 안정성과 조건수

대체적으로 수학문제의 조건(condition)은 입력값의 변화에 대한 민감도와 관련되어 있다. 입력값의 불확실성이 수치해법에 의해 크게 확대되면 계산이 수치적으로 불안정(numerically unstable)하다고 말한다. 함수$f(x)$의 오차를 1차 Taylor급수로 확인해보자.

$$f(x) = f(\tilde{x}) + f'(\tilde{x})(x - \tilde{x}) \tag{4.34}$$

위의 식을 함수값$f(x)$의 상대오차를 구하기위해 변형하면

$$\varepsilon_f = \frac{f(x) - f(\tilde{x})}{f(x)} \cong \frac{f'(\tilde{x})(x - \tilde{x})}{f(\tilde{x})} \tag{4.35}$$

x의 상대오차는 $\varepsilon_x = (x - \tilde{x})/\tilde{x}$이고, 조건수(condition number)[4]는 이들 상대오차의 비로 정의 된다.

$$\kappa = \frac{\varepsilon_f}{\varepsilon_x} = \frac{\tilde{x} f'(\tilde{x})}{f(\tilde{x})} \tag{4.36}$$

이 조건수는 x의 불확실성이 함수값$f(x)$에 얼마나 영향을 미치는지에 대한 척도를 제공한다.

수치해석 분야에서 함수의 조건수(condition number)는 argument에서 의 작은 변화의 비율에 대해 함수가 얼마나 변화할 수 있는지에 대한 argument measure이다. 여기서 "함수"는 문제의 해를 의미하며, "argument"는 문제에서의 데이터를 의미한다. 작은 조건수를 갖는 문제를 "well-conditioned"라고 하며, 큰 조건수를 갖는 문제를 "ill-conditioned"라고 한다. 조건수는 문제의 고유한 성질이다.

[4] http://en.wikipedia.org/wiki/Condition_number

과제2 (제출기한 10월9일)

1. $\cos x$의 Maclaurin 급수는 다음과 같이 주어진다.

$$\begin{aligned}\cos x &= \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} \\ &= 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \cdots\end{aligned}$$

가장 간단한 형태인 $\cos x = 1$로부터 시작하여 항들을 추가해 가면서 $\cos(\pi/3)$의 값을 구하라. 각각의 항이 추가될 때마다, 절단오차(상대오차)를 계산하라.

2. $x \in \mathbb{R}$에 대하여 $|x| > 1$의 개구간 미분이 가능한 다음 함수$f(x)$의 Taylor 급수를 전개하고, $x = 3$에서 5차 근사값을 구하고, 절단오차 E_t를 각각의 차수에 대하여 표시하라.

$$f(x) = \frac{x}{x-1}$$

3. 다음 함수 $f(x) = \ln(\cos x), x \in (-\pi/2, \pi/2)$를 7차이상의 다항식으로 전개하라

참고 Maclaurin 급수

$$\begin{aligned}\ln(1-x) &= -\sum_{n=1}^{\infty} \frac{x^n}{n}, (|x| \leq 1, x \neq 1) \\ \ln(1+x) &= \sum_{n=1}^{\infty} (-1)^{n+1} \frac{x^n}{n}, (|x| \leq 1, x \neq -1)\end{aligned}$$

4. 정확도(accuracy)와 정밀도(precision)에 대하여 조사하라[5] 그리고 각 전공영역에 맞추어 특정한 공학적 이슈를 예를들어 불확실성(uncertainty)에 대하여 자신의 생각을 쓰시오.

[5] http://en.wikipedia.org/wiki/Accuracy_and_precision

5 구간법 (Bracketing Method)

구간법은 함수가 근의 근처에서 부호(sign)가 변한다는 사실에 기초한다. 이 방법을 사용하여 근을 구하기 위해서는 두 개의 초기 가정값이 필요하기 때문에 구간법(bracketing method)이라고 부른다.

5.1 도식적 방법

방정식 $f(x)=0$에 대한 근사값을 구하기 위한 간단한 방법은 함수를 그려서 x축과 만나는 곳을 찾아보는 것이다. $f(x)=0$이 되게 하는 x값을 근의 대략적인 근사값으로 간주할 수 있다.

예제 도식적 방법

도식적 방법을 사용하여 질량 $m=68.1\text{kg}$인 낙하산병이 자유 낙하 10초 후에 $40m/s$의 속도를 갖도록 하는 제동계수를 구하라(단, 중력에 의한 가속도는 $9.8m/s^2$이다).

해 식(1.21)에서 제동계수인 c를 내재적(implicit) 매개변수로 포함시키면

$$f(c)=\frac{gm}{c}\left(1-e^{-(c/m)t}\right)-v \tag{5.1}$$

과 같이 쓸 수 있다. 즉, 매개변수 $t=10, g=9.8, v=40, m=68.1$ 이 주어진 c에 대한 함수$f(c)$로 쓸 수 있다.

$$f(c)=\frac{9.8\times 68.1}{c}\left(1-e^{-(c/68.1)\times 10}\right)-40 \tag{5.2}$$

다양한 c의 값을 대입하여 $f(c)$ 값을 계산한다. (MATLAB 활용)

```
clear all; clc; close all;
c=4:1:20;

for kk=1: length(c)
    f(kk)=9.8*68.1/c(kk)*(1-exp(-10*c(kk)/68.1)) -40;
end

figure
plot(c,f,'-ok')
xlabel('c')
ylabel('f(c)')
xlim([0 24])
ylim([-10 40])
grid on
```

Code [MATLAB] 1: 도식적 방법으로 방정식의 근 구하기

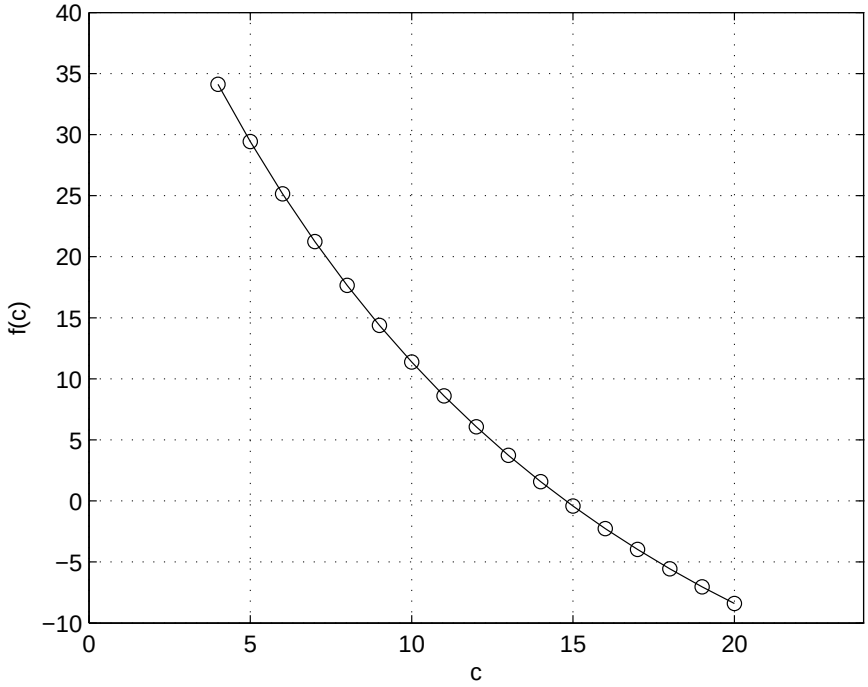


Figure 5.1: 방정식의 근을 구하기 위하여 도식적(그래프를 사용하는) 방법

5.2 이분법

일반적으로 $f(x)$가 구간 x_l과 x_u사이에서 실수이고 연속이며, $f(x_l)$과 $f(x_u)$가 반대 부호를 갖는다면, 즉

$$f(x_l)(x_u) < 0 \tag{5.3}$$

이면 x_l과 x_u사이에는 적어도 하나 이상의 실근(real root)이 존재한다.

- 증분탐색법(incremental search method) : 함수의 부호가 변화하는 구간을 찾아내어 근을 구하는 방법. 부호가 변하는 위치는 구간을 보다 작은 소구간으로 나눔으로써 정밀하게 식별할 수 있다.
- 이분법(bisection method) : 이진 버림(binary chopping), 구간 반분법(interval halving) 또는 Bolzano 법이라고도 부른다. 이분법은 증분탐색법의 하나로 구간을 항상 반으로 나눈다. 만일 함수의 부호가 구간내에서 바뀐다면 구간의 중간점에서 함수값을 계산한다.

Step 1 Choose lower x_l and upper x_u guesses for the root such that the function changes sign over the interval. This can be checked by ensuring that $f(x_l)f(x_u) < 0$.

Step 2 An estimate of the root x_r is determined by

$$x_r = \frac{x_l + x_u}{2}$$

Step 3 Make the following evaluations to determine in which subinterval the root lies:

(a) if $f(x_l)f(x_r) < 0$, the root lies in the lower subinterval. Therefore, set $x_u = x_r$ and return to **Step 2**.

(b) if $f(x_l)f(x_r) > 0$, the root lies in the upper subinterval. Therefore, set $x_l = x_r$ and return to **Step 2**.

(c) if $f(x_l)f(x_r) = 0$, the root equals x_r; terminate the computation.

도식적 방법의 예제에서 식(5.2)를 이분법을 이용해서 풀어보자.

예제 이분법

```
function y=f(x)

% y=(9.8*68.1)/x*(1-exp(-(x/68.1) *10))-40;
y=x^10-1;
```

Code [MATLAB] 2: 식(5.2)의 함수 정의

Code 2와 같이 식(5.2)을 함수화 시키자. 그리고 이분법 연산은

```
clear all; clc; close all;

```

```
xl=12;   % lower bound
xu=16;   % upper bound
ii=0;    % initial iteration value
xt=1.478020383166106e+001; % true x
while 1
    ii=ii+1;          % increment iteration value
    xr=(xl+xu)/2;     % bisection
    id=f(xl)*f(xr);
    if id<0
        xu=xr;        % assume xu to xr
    elseif id>0
        xl=xr;        % assume xu to xl
    else
        break;        % break loop
    end
    x(ii)=xr;         % calculated x vector
    et(ii)=(xt-xr)/xt;% calculated error vector
end
```

Code [MATLAB] 3: 식(5.2)의 이분법 코드

Code 3과 같이 반복문(loop)와 조건문(selection)으로 이뤄진다. 여기서는 `while`문을 사용하여 코딩을 하였는데, 이것은 반복문의 반복횟수로 종료판정을 하지 못하기 때문이다. `while`문은 `while` 이후에 조건을 입력해야 하지만 여기서는 여러조건이 함께 존재하기 때문에 `1` 즉, 무한반복을 선택하였다. 대신 `if`문과 `elseif` 그리고 `else` 문을 사용하였고 종료판정은 `else`에 `break`를 사용하여 반복을 종료하게 된다.

5.2.1 종료 판정 기준과 오차추정

실제로 예제의 정확한 값을 얻기 위해서 이분법을 반복시킬 수 있지만, 실제로 컴퓨터 연산은 일반적인 경우 $f(x_r) = 0$이 되는 경우까지 하는것은 매우 비효율 적이고, 복잡한 수치모델의 경우 반복문이 끝나지 않을 수가 있다. 따라서 언제 이 반복문을 끝낼 것인지에 대한 객관적인 판정 기준을 제시해야한다.
예를들어 오차가 미리 정한 정밀도 이하로 떨어지면 계산을 종료한다고 생각해보자. 상대오차가 0.1% 이하로 떨어지면 계산을 종료한다고 가정하면 문제가 발생한다. 왜냐하면 우리는 참값을 알지 못하기 때문에 상대오차를 계산할 수 없디 때문이다. 참값을 안다면 굳이 이분법 같은 수치해법으로 근을 구할 필요가 없기 때문이다. 하지만 13페이지 3.2절에서 식(3.5)를 통해 근사화된 상대오차를 구한적이 있다. 즉, 식(5.4)와 같이 반복계산문에서 이전의 값과 현재의 값의 변화로 판별하여 근사상대오차를 구한다.

$$e_a = \left| \frac{x_r^{new} - x_r^{old}}{x_r^{new}} \right| 100(\%) \tag{5.4}$$

5.2.2 함수 계산의 최소화

Code 3과 같은 알고리즘은 계산하기 쉬운 함수의 단일근을 찾는 경우에는 유용하게 사용될 수 있으나 실제 공학 문제에서 이와 같은 경우는 드물다. 예를들어 많은 근을 찾아야하는 프로그램을 개발하는 경우, Code 2와 같은 알고리즘을 수천, 수백만번 수행할 수 도 있다. 또한 함수의 경우 간단하게 처리하는(수행시간이 짧은) 함수가 아닌 경우가 더 많다. 특히 연산 시간이 많이 소요되는 경우 우리는 함수계산을 최소화 시켜야 한다. 변경된 코드를 보자

```
clear all; clc; close all;

xl=12;   % lower bound
xu=16;   % upper bound
ii=0;    % initial iteration value
xt=1.478020383166106e+001; % true x
fl=f(xl);
while 1
    ii=ii+1;          % increment iteration value
    xr=(xl+xu)/2;     % bisection
    fr=f(xr);
    id=fl*fr;
    if id<0
        xu=xr;        % assume xu to xr
    elseif id>0
        xl=xr;        % assume xl to xr
        fl=fr;        % replace fl to fr
    else
        break;        % break loop
    end
    x(ii)=xr;         % calculated x vector
    et(ii)=(xt-xr)/xt;% calculated error vector
end
```

Code [MATLAB] 4: 함수계산을 최소화한 이분법 코드

즉 Code 3에서 $2n$번 수행이 되는 함수연산을 Code 4에서 $n+1$으로 축소시켰다. (여기서 n은 반복문의 반복횟수 `ii`)

5.3 가위치법

이분법이 단일근을 찾는데에는 우수한 기법이지만 근에 접근하는 방식은 상당히 비효율적이다. 이분법의 단점은 구간 x_l과 x_u 사이를 항상 반으로 나누고, 함수값 $f(x_l)$과 $f(x_u)$의 크기를 고려하지 않는다. 가위치법은 이러한 단점을 고려하여 $f(x_l)$과 $f(x_u)$ 사이의 곡선을 직선으로 연결시켜 x축과 만나는 교점을 찾아 그 교점이 추정값이 되는 방식이다. 곡선을 직선으로 대치하여 근의 "가위치법"(false position method), 또는 라틴어로 "정규거짓"(regula falsi)이라고 하며, "선형보간법"(linear interpolation method)라고도 한다.

$$\frac{f(x_l)}{x_r - x_l} = \frac{f(x_u)}{x_r - x_u} \tag{5.5}$$

위의 식(5.5)은 다음과 같이 정리할 수 있다.

$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)} \tag{5.6}$$

즉, 식(5.6)로 x_r의 값을 계산한 후, 두 개의 초기경계값 x_l 또는 x_u 중 하나로 대치하고, $f(x_r)$과 같은 부호를 가진 함수값을 산출한다. 이 방법으로 x_l과 x_u의 사이에는 항상 참근이 존재하게 된다. 정확한 근을 구할 때 까지 이 과정을 반복한다. 이 알고리즘은 이분법의 **Step 2**에서 식(5.6)을 사용하는 것을 제외하고 이분법과 같다.

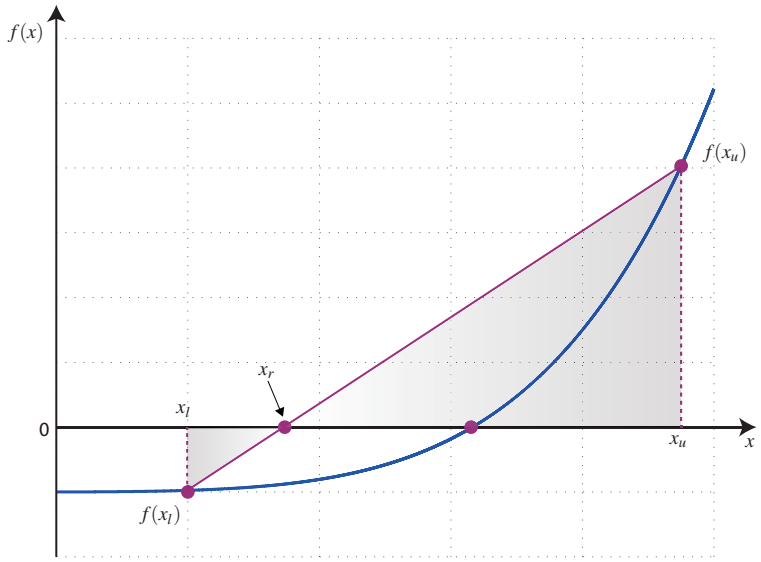


Figure 5.2: 가위치법의 도식적 묘사

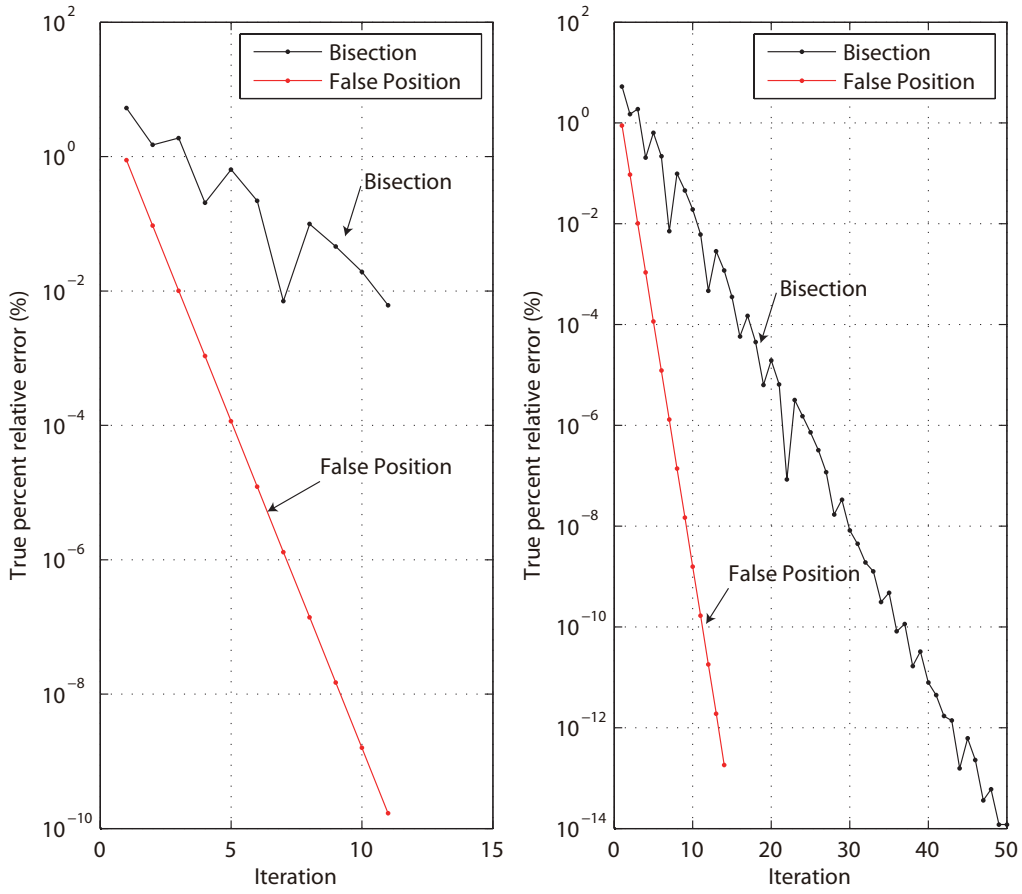
이분법과 같은 방식으로 낙하산병 예제를 풀어보자

```
clear all; clc; close all;

xl=12;   % lower bound
xu=16;   % upper bound
ii=0;    % initial iteration value
xt=1.478020383166106e+001; % true x
fl=f(xl);
fu=f(xu);
while 1
    ii=ii+1;          % increment iteration value
    xr=xu-(fu*(xl-xu))/(fl-fu);
    fr=f(xr);
    id=fl*fr;
    if id<0
        xu=xr;        % assume xu to xr
        fu=f(xu);     % replace fu to f(xu)
    elseif id>0
        xl=xr;        % assume xu to xl
        fl=f(xl);     % replace fl to f(xl)
    else
        break;        % break loop
    end
    x(ii)=xr;         % calculated x vector
```

```
        et(ii)=(xt-xr)/xt;% calculated error vector
25  end
```

Code [MATLAB] 5: 식(5.2)의 가위치법 코드

이분법과 가위치법의 상대적인 효율성을 비교해보자, 반복횟수에 대한 참상대오차가 줄어드는 경향을 판단해볼 수 있다.



(a) Short Iteration **(b)** Full Iteration

Figure 5.3: 이분법과 가위치법 상대오차 비교

5.4 수정된 가위치법

$$f(x) = x^{10} - 1 \tag{5.7}$$

식(5.7)의 경우 이분법이 가위치법보다 빠르게 수렴한다. 이것은 근을 찾는 방법을 일반화하는 것은 불가능하다는 것을 보여준다. 즉, 이것은 수정된 가위치법을 통해서 한쪽 경계가 변하지 않는 경우 알아내는 알고리즘을 포함시키는 것이다.

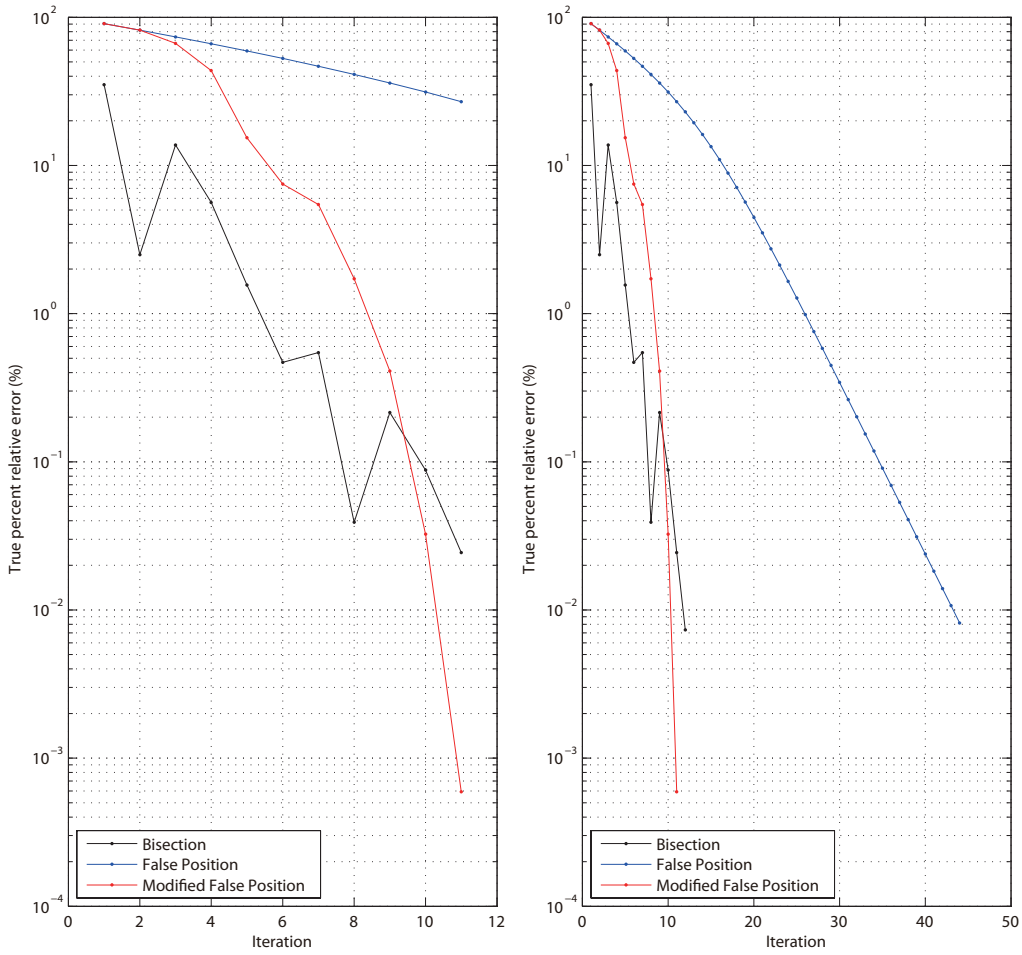
```
1  clear all; clc; close all;

3  xl=0;    % lower bound
   xu=1.3;    % upper bound
```

```
ii=0;    % initial iteration value
xt=1; % true x
il=0;iu=0;
fl=f(xl);
fu=f(xu);
while 1
    ii=ii+1;          % increment iteration value
    xr=xu-(fu*(xl-xu))/(fl-fu);
    fr=f(xr);
    id=fl*fr;
    if id<0
        xu=xr;        % assume xu to xr
        fu=f(xu);     % replace fu to f(xu)
        iu=0;         % set zero iteration of lower bound
        il=il+1;      % count iteration of lower bound
        if il≥2, fl=fl/2; end % modify lower function value
    elseif id>0
        xl=xr;        % assume xu to xl
        fl=f(xl);     % replace fl to f(xl)
        il=0;         % set zero iteration of lower bound
        iu=iu+1;      % count iteration of upper bound
        if iu≥2, fu=fu/2; end % modify upper function value
    end
    x(ii)=xr;               % calculated x vector
    et(ii)=abs((xt-xr)/xt); % calculated error vector
    if et(end)≤0.0001
        break;        % break loop
    end
end
```

Code [MATLAB] 6: 식(5.7)의 수정된 가위치법 코드

이러한 기존의 조건문에 또다른 조건문을 포함시켜 알고리즘을 추가하는 방식은 추후 증분탐색과 초기가정값의 크기에 따라 탐색을 놓칠 경우가 있기 때문에, 완전한 방법이라고 볼 수는 없다.

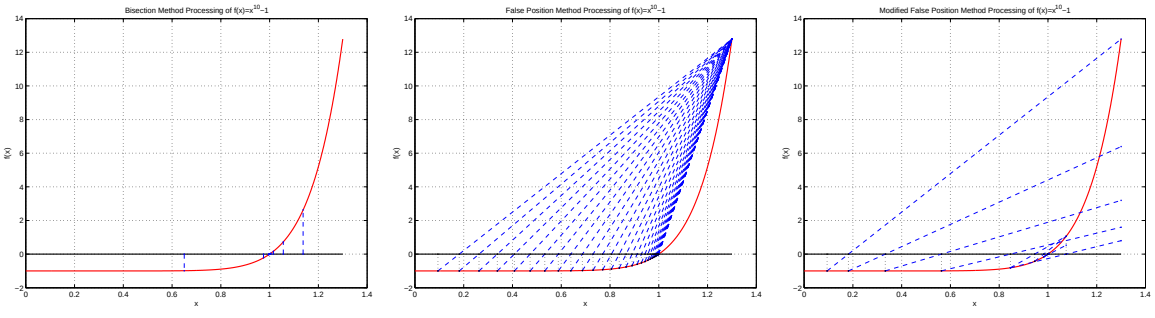
Figure 5.4을 통해 식(5.7)의 이분법, 가위치법, 수정된 가위치법의 참상대오차의 수렴과정을 살펴보자. 식(5.7) 문제의 이분법, 가위치법의 각각의 MATLAB 코드는 71페이지의 부록A.3장을 참고.



(a) Short Iteration **(b)** Full Iteration

Figure 5.4: 이분법과 가위치법 및 수정된 가위치법 상대오차 비교

Figure 5.5에서 이분법, 가위치법 그리고 수정된 가위치법이 단계별로 어떻게 근으로 수렴하는지 알 수 있다.



(a) Bisection method **(b)** False position method **(c)** Modified false position method

Figure 5.5: 이분법과 가위치법 및 수정된 가위치법의 계산과정

6 개구간법 (Open Method)

구간법은 미리 정의된 하한치(lower bound)와 상한치(upper bound)의 경계값으로 이루어지는 구간 내에서 근을 찾았다. 반면에 이장에서 다룰 개구간법(open methods)은 한 개의 초기값에서 시작하거나 구간 내에 근을 포함하지 않을 수도 있는 두 개의 초기값들로부터 시작하는 방법들이다. 이러한 개구간법은 종종 발산(diverge)하거나 근에서 멀어지기도 한다. 그러나 수렴(convergent)할 경우 빠르게 수렴한다.

6.1 고정점 반복법(fixed point iteration)

함수 $f(x)=0$의 형태를 다음 식과 같이 변형할 수 있다.

$$g(x)=x \tag{6.1}$$

이 변형된 함수는 함수 g에 대한 fixed point의 해가 된다. 이러한 함수 $f(x)=0$에서 단순한 해법으로 재구성하는 방법은 고정점 반복(fixed point iteration)으로 이루어진다. 그러나 이러한 방법을 수행하기 이전에 우리는 변형된 함수 $g(x)=x$의 해의 존재와 유일성을 검토해야한다. 다음 정리가 이러한 질문에 대한 해를 제시한다.

Theorem 6.1 (Fixed-Point Iteration). *함수 $g(x)$가 구간 $[a,b]$에서 연속의 함수일 때, $\forall x\in[a,b]$에 대해 $g(x)\in[a,b]$이면, g는 $[a,b]$에서 고정점(fixed point)를 갖는다. 또한, 함수 $g(x)$가 미분가능한 함수이고 아래의 식(6.2)과 같은 상수 $k<1$가 존재한다면,*

$$\left|g'(x)\right|\le k, x\in(a,b) \tag{6.2}$$

g는 정확하게 $[a,b]$에서 하나의 고정점(fixed point)를 갖는다.

구간 $[a,b]$에서 고정점을 갖는 연속함수 g가 주어진다면, 그 고정점(fixed point)은 구간 $[a,b]$에서 $g(x)=x$를 만족하는 x를 무수히 반복하여 찾을 수 있다.

Algorithm 6.1 Fixed-Point Iteration

Let g be a continuous function defined on the interval $[a,b]$. The following algorithm computes a number $x_r\in(a,b)$ that is a solution to the equation $g(x)=x$.
Choose an initial guess x_0 in $[a,b]$.

for $i=0,1,2,\cdots$ **do**
  $x_{i+1}=g(x_i)$
  if $|x_{i+1}-x_i|$ is sufficiently small **then**
    $x_r=x_{i+1}$
    return x_r
  end if
end for

어떠한 환경에서 고정점 반복이 수렴해서 x_r을 찾을 수 있을까? 단순히 오차를 $e_i=x_i-x_r$로 가정한다면, Taylor 급수에서 $e_{i+1}\approx g'(x_r)e_i$일 때, $g(x_r)=x_r$임을 찾을 수 있다. 즉, $k<1$일 때, $|g'(x_r)|\le k$ 라면, 고정점 반복(fixed-point iteration)은 부분수렴(locally convergent)한다고 한다. 즉, 초기설정값 x_0가 x_r에 충분히 근접한 값을 선택했을 때 수렴한다는 말이다. 이러한 결과는 다음 정리를 통해 알 수 있다.

Theorem 6.2 (Fixed-Point Theorem). *함수 $g(x)$ 가 구간 $[a,b]$ 에서 연속의 함수일 때, $\forall x \in [a,b]$ 에 대해 $g(x) \in [a,b]$ 이고, 다음 식과 같이 상수 $k < 1$ 인 점이 존재한다면,*

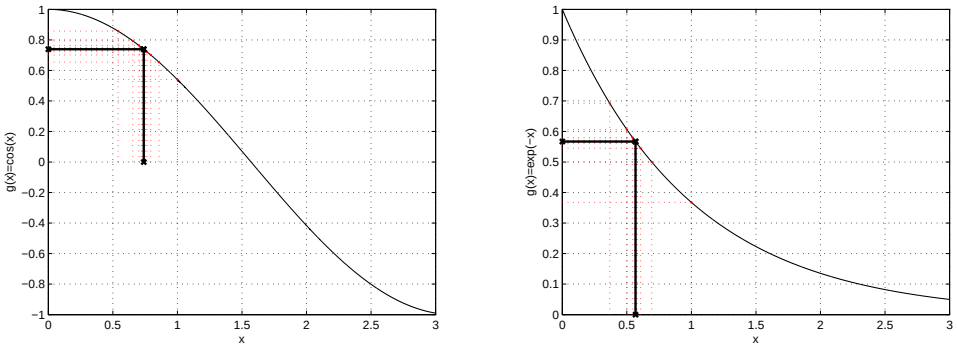
$$\left|g'(x)\right| \leq k, x \in (a,b), \tag{6.3}$$

반복문(the sequence of iterates) $\{x_i\}_{i=0}^{\infty}$ 는 어떤 초기가정 $x_0 \in [a,b]$ 에도, 특정 고정값 x_r 으로 수렴하게 된다.

함수 $g'(x)$ 가 수렴의 약조건 $|g'(x)| < 1$ 에 비하여 (a,b) 에서 1 과 떨어지도록 경계지어야하는지에 대하여, 만약 $g'(x)$ 가 x 가 어느 한점 $c \in (a,b)$ 로 접근할 때 1 로의 접근을 허용한다면, 오차 e_i 가 i 가 증가할 수록 0 으로 수렴하지 않을 수가 있다. 이러한 경우 고정점 반복법(fixed-point iteration)은 수렴하지 않는다.

일반적으로 고정점 반복(fixed-point iteration)이 수렴할 때, $|g'(x)|$ 의 경계치는 상수 k 의 역으로 변화하면서 수렴한다.

한편, $f(x) = 0$ 를 $g(x) = x$ 로 변환하는 식은 많은 방법이 존재하지만, 단순화된 방법으로 특정함수 $\phi(x)$ 를 추가한 $g(x) = x - \phi(x)f(x)$ 방법을 사용한다. 그러나 이러한 함수변형에 있어 가장 중요한 요소는 함수 $g(x)$ 의 수렴성이다.



(a) $g(x) = \cos(x)$ **(b)** $g(x) = \exp(-x)$

Figure 6.1: 고정점 반복법의 수렴과정

6.2 Newton-Raphson법 (Newton-Raphson method)

구간법의 문제점은 적어도 한 근을 포함하는 구간을 알고있어야 한다는 점이다. 즉, 이러한 방법은 필요한 구간을 상정하기 위해서 무수히 많이 반복을 수행해야하기 때문에 실용적이지 못하다. 좀더 효과적으로 근을 구하는 방법을 찾기 위해서는 다음과 같은 질문을 해결해야 한다.

- Are there cases in which the problem easy to solve, and if so, how do we solve it in such cases?
- Is it possible to apply our method of solving the problem in these "easy" cases to more general cases?

근을 구하는 공식들 중에서 가장 일반적으로 사용하는 것이 Newton-Raphson 법이다. 근에 대한 초기가정값이 x_i라면, 점 $[x_i, f(x_i)]$에 접하는 접선을 구할 수 있고, 이 접선이 x축과 교차하는 점이 개선된 근이 된다. 4.1장 15페이지 식(4.3)의 Taylor 급수를 통해 1차항으로 구성된 다음 식으로 근사화 시킬 수 있다.

$$f(x_{i+1}) \cong f(x_i) + f'(x_i)(x_{i+1} - x_i) \tag{6.4}$$

즉, x축과의 교점에서 $f(x_{i+1}) = 0$이 되므로 다음과 같이 근사할 수 있다.

$$0 = f(x_i) + f'(x_i)(x_{i+1} - x_i) \tag{6.5}$$

x_{i+1}에 대하여 정리하면 다음식(6.6)과 같다.

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \tag{6.6}$$

Algorithm 6.2 Newton-Raphson Method

Let $f : \mathbb{R} \to \mathbb{R}$ be a differentiable function. The following algorithm computes an approximate solution x_r to the equation $f(x) = 0$.

Choose an initial guess x_0.

for $i = 0, 1, 2, \cdots$ **do**
  if $f(x_i)$ is sufficiently small **then**
    $x_r = x_i$
    return x_r
  end if
  $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$
  if $|x_{i+1} - x_i|$ is sufficiently small **then**
    $x_r = x_{i+1}$
    return x_r
  end if
end for

Newton-Raphson법이 수렴할때, 상당히 빨리 수렴한다. 그러나, 수렴 여부에 대해 찾기가 쉽지 않을 수 있다. 특히, 함수 $f(x)$가 근의 부근 x_r에서 수평기울기(horizontal tangent)을 가지게 될 때 반복구간에서 값이 커질 수가 있다. 이러한 경우 일반적으로 시작지점 x_0을 x_r에 가장 가까운 곳에서 선택할 필요가 있다.

Theorem 6.3 (Convergence of Newton-Raphson's Method). *함수 $f(x)$를 구간 $[a,b]$에서 연속이며 미분가능하다고 가정하고, 특정한 값 $c \in [a,b]$에서 $f(c)=0$와 $f'(c)=0$을 만족한다고 가정하면, 함수 $f(x)$에 Newton-Raphson 법을 적용했을때 $[c-\delta, c+\delta]$ 구간에서 수렴하는 어떤 초기값 x_0을 갖는 $\delta > 0$인 δ가 존재한다.*

예제 **Newton-Raphson법 예제 ($\sqrt{2}$ 찾기)**

다음 함수 $f(x)$를 통해 $\sqrt{2}$의 값을 Newton-Raphson법을 이용하여 예측하라.

$$f(x) = x^2 - 2 \tag{6.7}$$

해 $f'(x) = 2x$이기 때문에, Newton-Raphson법에서 x_i에 대한 x_{i+1}은 다음식과 같다.

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{x_i^2 - 2}{2x_i} = \frac{x_i}{2} + \frac{1}{x_i} \tag{6.8}$$

x_i	$f(x_i)$	$f'(x_i)$	**Tangent line**
$x_0 = 0.5$	$f(x_0) = -1.75$	$f'(x_0) = 1$	$y = f'(x_0)(x-x_0) + f(x_0)$
$x_1 = 2.25$	$f(x_1) = 3.0625$	$f'(x_1) = 4.5$	$y = f'(x_1)(x-x_1) + f(x_1)$
$x_2 = 1.5694$	$f(x_2) = 0.463155$	$f'(x_2) = 3.13889$	$y = f'(x_2)(x-x_2) + f(x_2)$
$x_3 = 1.42189$	$f(x_3) = 2.177221 \times 10^{-2}$	$f'(x_3) = 2.84378$	$y = f'(x_3)(x-x_3) + f(x_3)$
$x_4 = 1.414234$	$f(x_4) = 5.861552 \times 10^{-5}$	$f'(x_4) = 2.82847$	$y = f'(x_4)(x-x_4) + f(x_4)$

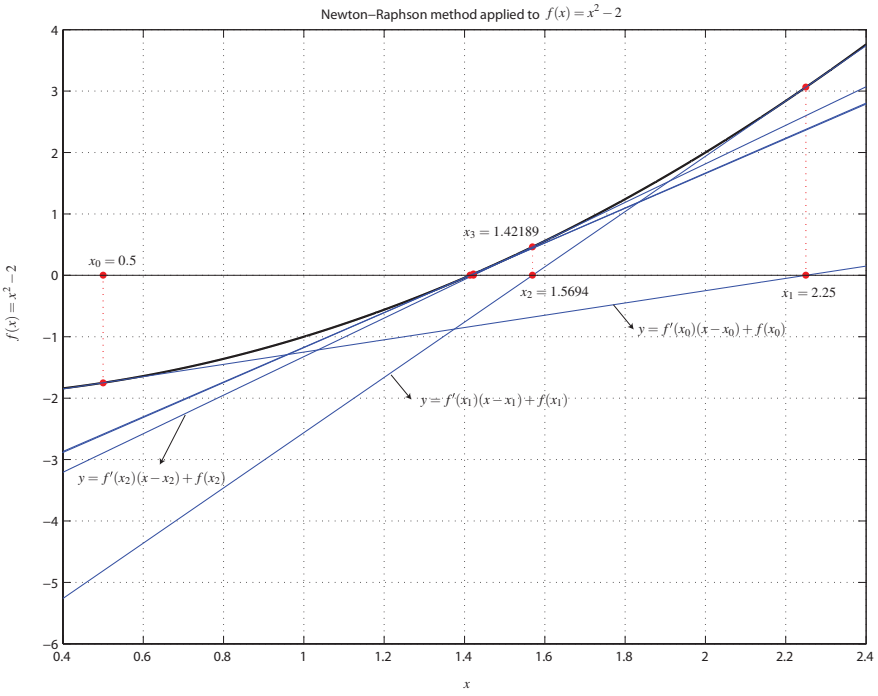


Figure 6.2: Newton-Raphson Method

6.3 할선법 (secant method)

Newton-Raphson법을 수행하는데 어려운점은 바로 도함수의 계산이다. 어떤 함수에서는 도함수를 계산하는 것이 매우 어려울 수 있다. 그러한 경우 도함수는 4.3장에서 학습한 수치미분중 후진유한제차분으로 근사시킬 수 있다. 후진차분근사법은 식(4.17)에서 도함수를 가져오기로 하자.

$$f'(x_i) \cong \frac{f(x_{i-1}) - f(x_i)}{x_{i-1} - x_i} \tag{6.9}$$

도함수의 후진차분근사 식(6.9)을 Newton-Raphson법 식(6.6)에 대입시킨다.

$$x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)} \tag{6.10}$$

식(6.10)과 같이 할선법(secant method)는 두개의 초기값 x_{i-1}, x_i이 필요하다. 즉, x_2부터 추적해나가기 때문에 x_0과 x_1의 초기값이 필요하다.

Algorithm 6.3 Secant Method

Let $f : \mathbb{R} \to \mathbb{R}$ be a differentiable function. The following algorithm computes an approximate solution x_r to the equation $f(x) = 0$.

Choose an initial guess x_0 and x_1.

for $i = 0, 1, 2, \cdots$ **do**
  if $f(x_i)$ is sufficiently small **then**
    $x_r = x_i$
    return x_r
  end if
  $x_{i+2} = x_{i+1} - \frac{f(x_{i+1})(x_i - x_{i+1})}{f(x_i) - f(x_{i+1})}$
  if $|x_{i+2} - x_{i+1}|$ is sufficiently small **then**
    $x_r = x_{i+2}$
    return x_r
  end if
end for

6.4 수정된 할선법 (modified secant method)

도함수를 계산하기 위하여 임의의 두 값을 사용하기보다 $f'(x)$를 추정하기 위하여 독립변수에 약간의 변동을 주는 방법을 고려할 수 있다.

$$f'(x_i) \cong \frac{f(x_i + \delta x_i) - f(x_i)}{\delta x_i} \tag{6.11}$$

여기서 δ는 약간의 변동량이다. 식(6.6)에 위의 식을 대입하면 다음과 같은 반복계산식을 얻을 수 있다.

$$x_{i+1} = x_i - \frac{\delta x_i f(x_i)}{f(x_i + \delta x_i) - f(x_i)} \tag{6.12}$$

Algorithm 6.4 Modified Secant Method

Let $f : \mathbb{R} \to \mathbb{R}$ be a differentiable function. The following algorithm computes an approximate solution x_r to the equation $f(x) = 0$.

Choose an initial guess x_0 and δ.

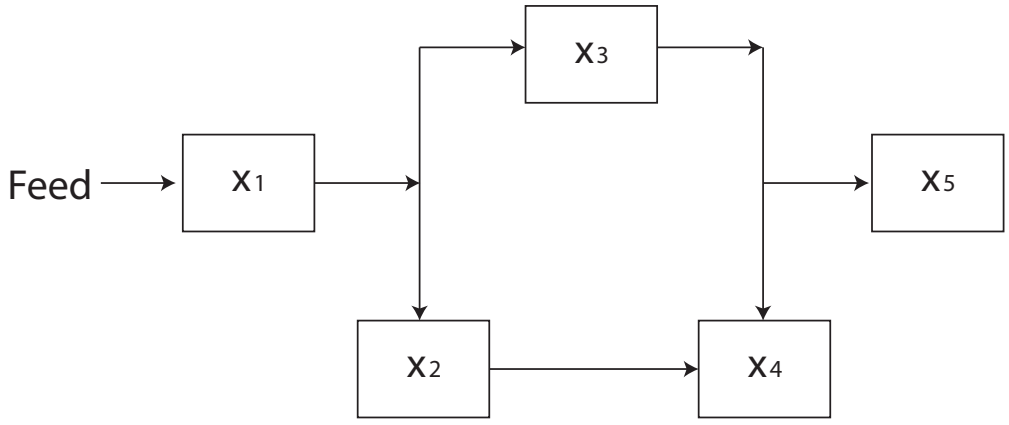
for $i = 0, 1, 2, \cdots$ **do**
  if $f(x_i)$ is sufficiently small **then**
    $x_r = x_i$
    return x_r
  end if
  $x_{i+1} = x_i - \frac{\delta x_i f(x_i)}{f(x_i + \delta x_i) - f(x_i)}$
  if $|x_{i+1} - x_i|$ is sufficiently small **then**
    $x_r = x_{i+1}$
    return x_r
  end if
end for

이 수정된 할선법 알고리즘에서 δ의 선택이 이슈가 되는데 적절하게 δ를 선택하는 방법밖에는 없다. δ가 작으면 식(6.12)의 분모에서 발생하는 반올림 오차가 커지게되고, 또한 δ가 커지게 되면 발산할 수도 있다. 그러나 적절한 δ가 선택되면 도함수를 계산하기 어려운 함수의 근이나 두개의 초기가정을 결정하는 것이 어려운 경우 근을 찾는 좋은 방법이 될 수 있다.

Part II

선형대수 방정식
(Linear Algebraic Equations)

공학적 문제에서 만나는 많은 함수혹은 방정식들은 보존 법칙(conservation law)에 기초를 두고 있다. 이와 같은 법칙이 적용되는 양으로는 질량, 에너지 그리고 운동량 등이 있다. 이들 원리들은 수학적 용어를 사용하여 모델링되어지는 양(quantity)의 수준(level) 또는 응답(response)으로 표현되는 시스템의 거동(behavior)을, 시스템의 성질(property) 또는 특성(characteristic)과 외부 자극(stimuli) 또는 시스템에 작용하는 강제함수(forcing function)등에 관련시키는 평형(balance) 또는 연속방정식(continuity equation)을 이끌어내게 된다. I부에서, 단일요소 시스템은 근-위치 방법들을 사용하여 풀 수 있는 단일 방정식이었다면, 다중요소 시스템은 동시에 연립하여 풀어야 할 상호 연관된 방정식으로 표현된다. 이들 방정식들은 서로 연관되어 상호 작용을 하는데 그 이유는 시스템의 한 부분이 다른 부분에 의하여 영향을 받기 때문이다. 이러한 다중요소 문제들은 집중(lumped) 또는 분산(distributed) 변수의 수학적 모델 모두에서 발생하게 된다.



(a) 구성 요소들이 상호 연관되어 있는 집중 변수시스템(lumped variable system)

(b) 연속체로 이루어진 분산 변수 시스템(distributed variable system

Figure 6.3: 선형대수 방정식을 사용하여 모델링할 수 있는 두가지 시스템

7 Gauss 소거법 (Gauss elimination)

Gauss 소거법은 연립 방정식의 해를 구하는 최초의 방법 중의 하나이지만, 현재에도 매우 중요한 알고리즘의 하나로 남아 있으며, 널리 알려져 있는 상용소프트웨어 패키지상에서 선형 방정식의 해를 구하는데 기초가 되고 있다.

7.1 도식적 방법

좌표의 한 축이 x_1 이고, 다른 축이 x_2 인 직교좌표계상에서 두 방정식을 그래프로 그린 후 그 교점으로부터 해를 얻는다.

$$a_{11}x_1 + a_{12}x_2 = b_1$$
$$a_{21}x_1 + a_{22}x_2 = b_2$$

두 식을 x_2 항으로 표현하여 기울기와 x_2 절편을 얻는다.

$$x_2 = -\left(\frac{a_{11}}{a_{12}}\right)x_1 + \frac{b_1}{a_{12}}$$
$$x_2 = -\left(\frac{a_{21}}{a_{22}}\right)x_1 + \frac{b_2}{a_{22}}$$

이 직선들을 직교좌표계상에서 x_2 를 세로축, x_1 을 가로축으로 하여 그리면, 두 선들이 만나는 점의 좌표값 x_1 과 x_2 가 해가 된다. 도식적 방법을 사용하여 다음 방정식을 풀면,

$$3x_1 + 2x_2 = 18$$
$$-x_1 + 2x_2 = 2$$

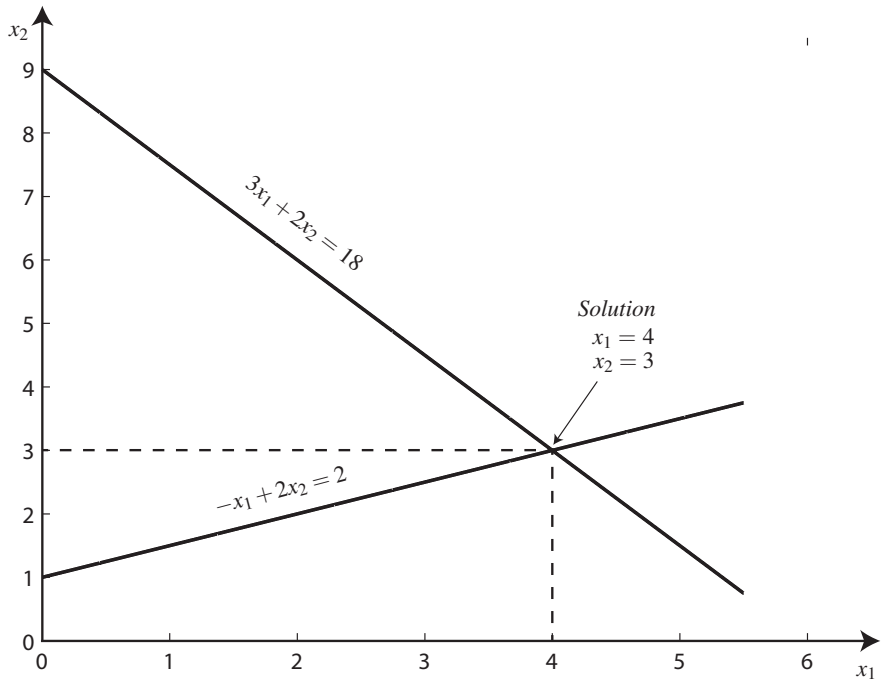
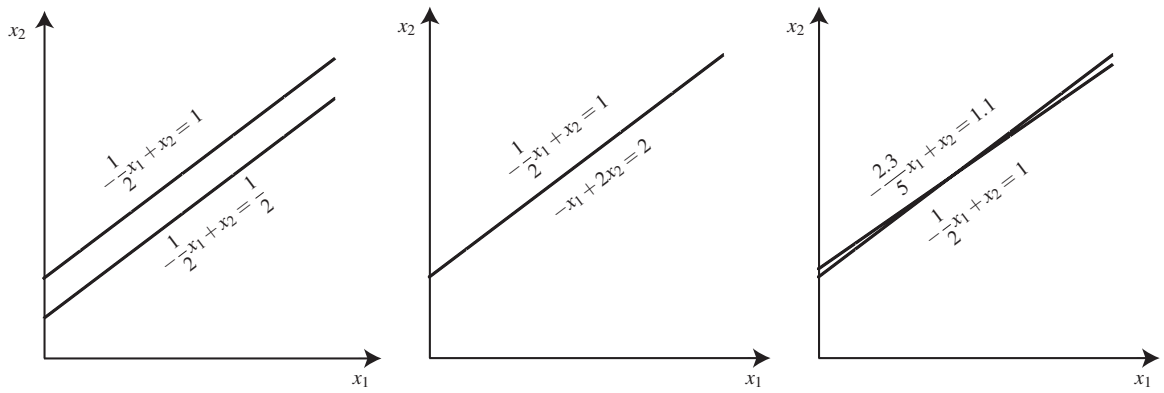


Figure 7.1: 2개의 연립 선형대수 방정식의 도식적 방법

Figure 7.2는 연립 선형 방정식을 풀 때 발생할 수 있는 세가지 경우를 보여주고 있다. Figure 2(a)는 두 개의 직선이 평행한 경우로, 두 직선이 서로 교차하지 않아 해가 존재하지 않는다. Figure 2(b)는 두 개의 직선이 일치하는 경우. 이 두 경우의 시스템을 특이(singular)하다고 한다. 특이한 경우에 매우 가까운 시스템인 Figure 2(c)의 경우는 시스템에 문제를 발생시킬 수 있는 시스템으로 불량조건(ill-conditioned)을 갖는다고 말한다.



(a) 해가 존재하지 않는 경우 **(b)** 해가 무한히 많은 경우 **(c)** 불량조건 시스템

Figure 7.2: 특이하고 불량조건을 갖는 시스템

7.2 행렬식과 Cramer 공식

Cramer 공식(Cramer's rule)은 방정식의 수가 적을 때 매우 적합한 또 다른 해법이다.Cramer 식을 수행하는데 사용될 행렬식(determinant)의 대하여 알아야 한다. 행렬식은 행렬이 불량조건인지의 여부를 판단하는데도 사용된다.

7.2.1 행렬식

행렬식을 다음과 같은 세 개의 연립방정식의 경우에 대하여 설명하면,

$$\mathbf{AX} = \mathbf{B}$$

여기서 **A**는 계수행렬(coefficient matrix)로써

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

이 시스템의 행렬식 D는 다음과 같이 방정식의 계수로부터 얻어진다.

$$D = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \tag{7.1}$$

행렬식D와 계수행렬**A**는 같은 요소들로 구성되어 있지만, 완전히 다른 수학적 개념을 가지고 있다. 이 때문에 행렬은 bold 폰트를 사용하여 표시하고 구성요소는 대괄호를 사용하여 나타내지만 , 행렬식은 스칼라값이기 때문에 normal 폰트를 사용하고 구성요소는 절대값 괄호로 표시한다. 2차 행렬식은 다음 식(7.3)과같이 표현한다.

$$D = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \tag{7.2}$$

$$= a_{11}a_{22} - a_{12}a_{21} \tag{7.3}$$

3차 행렬식[식(7.1)]의 경우, 행렬식의 값은 다음과 같이 계산한다.

$$D = a_{11}\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12}\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13}\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \tag{7.4}$$

여기서 2×2의 행렬식을 소행렬식(minor)라고 한다.

7.2.2 Cramer 공식

이 공식은 연립 선형대수 방정식에서의 각각의 미지수는 두개의 행렬식의 비로 표시될 수 있다는 것을 말해준다. 다음의 $n\times n$ 연립 선형대수 방정식에서,

$$\mathbf{AX} = \mathbf{B}$$

$n\times n$크기의 행렬 $\mathbf{A}$의 행렬식이 0을 가지지 않는다면, 벡터 $\mathbf{X} = (x_1, \cdots, x_n)^T$의 x_i는 다음 식을 통해 구할 수 있다.

$$x_i = \frac{|\mathbf{A}_i|}{|\mathbf{A}|} \tag{7.5}$$

여기서, $\mathbf{A}_i$는 행렬 A의 i번째 열을 벡터 $\mathbf{B}$로 대체한 행렬이다.

예제 **Cramer 공식**

Cramer 공식을 사용하여 다음의 방정식을 풀어라.

$$0.3x_1 + 0.52x_2 + x_3 = -0.01$$
$$0.5x_1 + x_2 + 1.9x_3 = 0.67$$
$$0.1x_1 + 0.3x_2 + 0.5x_3 = -0.44$$

해 행렬식 D는 다음과 같이 쓸 수 있다.

$$D = D = \begin{vmatrix} 0.3 & 0.52 & 1 \\ 0.5 & 1 & 1.9 \\ 0.1 & 0.3 & 0.5 \end{vmatrix}$$

소행렬식을 구하면,

$$D_1 = \begin{vmatrix} 1 & 1.9 \\ 0.3 & 0.5 \end{vmatrix} = 1(0.5) - 1.9(0.3) = -0.07$$
$$D_2 = \begin{vmatrix} 0.5 & 1.9 \\ 0.1 & 0.5 \end{vmatrix} = 0.5(0.5) - 1.9(0.1) = 0.06$$
$$D_3 = \begin{vmatrix} 0.5 & 1 \\ 0.1 & 0.3 \end{vmatrix} = 0.5(0.3) - 1(0.1) = 0.05$$

$\mathbf{A}$의 행렬식은

$$D = 0.3(-0.07) - 0.52(0.06) + 1(0.05) = -0.0022$$

Cramer 공식을 적용하면, 해는 다음과 같이 구할 수 있다.

$$x_1 = \frac{\begin{vmatrix} -0.01 & 0.52 & 1 \\ 0.67 & 1 & 1.9 \\ -0.44 & 0.3 & 0.5 \end{vmatrix}}{-0.0022} = \frac{0.03278}{-0.0022} = 14.9$$

$$x_2 = \frac{\begin{vmatrix} 0.3 & -0.01 & 1 \\ 0.5 & 0.67 & 1.9 \\ 0.1 & -0.44 & 0.5 \end{vmatrix}}{-0.0022} = \frac{0.0649}{-0.0022} = -29.5$$

$$x_3 = \frac{\begin{vmatrix} 0.3 & 0.52 & -0.01 \\ 0.5 & 1 & 0.67 \\ 0.1 & 0.3 & -0.44 \end{vmatrix}}{-0.0022} = \frac{-0.04356}{-0.0022} = 19.8$$

방정식의 개수가 세 개 이상이 되면, 방정식의 개수가 증가함에 따라 행렬식을 수작업(또는 컴퓨터에 의하여)에 의하여 구하는 데에 매우 많은 시간이 소비되므로, Cramer 공식은 비현실적이라고 할 수 있다.

7.3 Gauss 소거법

Gauss 소거법은 전진소거(forward elimination)와 후진대입(back substitution)의 방법으로 나뉜다. 이 방법은 컴퓨터상에서 구현하기에 이상적이지만, 신뢰성 있는 알고리즘을 위하여 약간의 수정이 필요하다. 특히, 컴퓨터 프로그램에서는 0으로 나누는 경우를 피해야한다. 다음과 같이 일반적인 n개의 방정식을 풀고자 한다.

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \qquad (7.6)$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = b_2 \qquad (7.7)$$

$$\vdots \qquad = \vdots$$

$$a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n = b_n \qquad (7.8)$$

7.3.1 미지수의 전진소거(forward elimination)

첫 번째 단계는 연립방정식을 상부삼각(upper triangular) 시스템으로 만드는 것이다. 2번째 x_1 항을 소거하기 위하여 식(7.6)에 a_{21}/a_{11}을 곱하면,

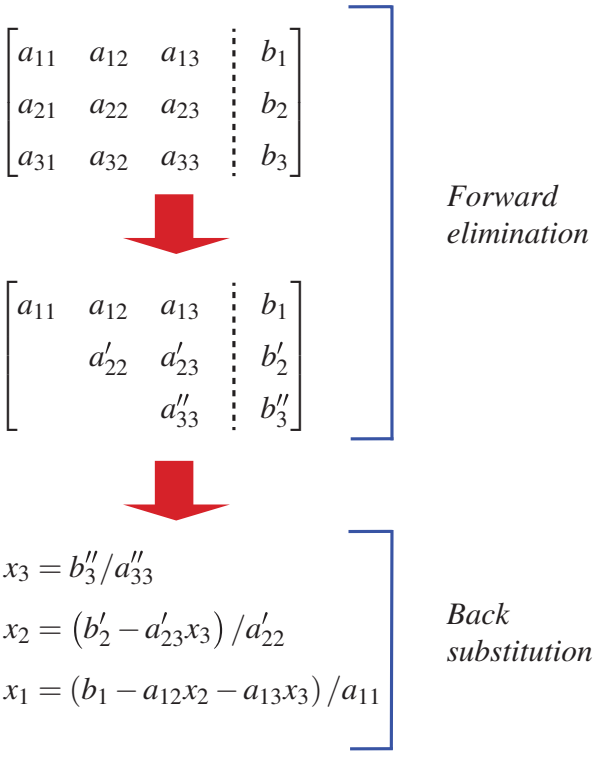


Figure 7.3: Gauss 소거법의 두단계

$$a_{21}x_1 + \frac{a_{21}}{a_{11}}a_{12}x_2 + \frac{a_{21}}{a_{11}}a_{13}x_3 + \cdots + \frac{a_{21}}{a_{11}}a_{1n}x_n = \frac{a_{21}}{a_{11}}b_1 \tag{7.9}$$

이 방정식을 식(7.7)에서 빼면,

$$\left(a_{22} - \frac{a_{21}}{a_{11}}a_{12}\right)x_2 + \cdots + \left(a_{2n} - \frac{a_{21}}{a_{11}}a_{1n}\right)x_2 = b_2 - \frac{a_{21}}{a_{11}}b_1 \tag{7.10}$$

$$a'_{22}x_2 + \cdots + a'_{2n}x_n = b'_2 \tag{7.11}$$

이러한 방식으로 나머지 방정식에 대해서도 같은 절차를 반복한다. 그 절차를 $n-1$ 번을 반복수행하면,

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \tag{7.12}$$

$$a'_{22}x_2 + a'_{23}x_3 + \cdots + a'_{2n}x_n = b'_2 \tag{7.13}$$

$$a'_{32}x_2 + a'_{33}x_3 + \cdots + a'_{3n}x_n = b'_3 \tag{7.14}$$

$$\vdots \qquad = \vdots$$

$$a'_{n2}x_2 + a'_{n3}x_3 + \cdots + a'_{nn}x_n = b'_n \tag{7.15}$$

식(**??**)를 피벗방정식(pivot equation)이라고 하고, a_{11} 을 피벗계수(pivot coefficient) 또는 피벗요소(pivot element)라고 한다. 계속해서 식(7.13)를 시작으로 나머지 피벗방정식을 아래로 내리면서 소거를 해나갈 수 있다. 이 과정에서 마지막 작업은 n번째 방정식으로부터 x_{n-1} 을 소거하기 위하여 $(n-1)$번째 방정식을 사용하는 것이다. 이 시점에서 시스템은 상부삼각(upper triangular) 시스템으로 변환된다.

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \tag{7.16}$$

$$a'_{22}x_2 + a'_{23}x_3 + \cdots + a'_{2n}x_n = b'_2 \tag{7.17}$$

$$a''_{33}x_3 + \cdots + a''_{3n}x_n = b''_3 \tag{7.18}$$

$$\ddots \qquad = \vdots$$

$$a_{nn}^{(n-1)}x_n = b_n^{(n-1)} \tag{7.19}$$

7.3.2 후진대입(**back substitution**)

식(7.19)는 x_n 값을 얻는데 사용할 수 있다.

$$x_n = \frac{b_n^{(n-1)}}{a_{nn}^{(n-1)}} \tag{7.20}$$

이 결과를 $(n-1)$번째 방정식에 대입하여 x_{n-1} 을 얻게 된다. 나머지 x들의 값을 얻기 위하여 반복되는 이 후진대입의 과정은 다음 식과 같다.

$$x_i = \frac{b_i^{(i-1)} - \sum_{j=i+1}^{n} a_{ij}^{(i-1)}x_j}{a_{ii}^{(i-1)}} \quad \text{for } i = n-1, n-2, \cdots, 1 \tag{7.21}$$

7.4 피벗화

피벗요소가 0인 경우에는 정규화 과정에서 0으로 나누는 경우가 발생하고 0이 아닌경우에도 0에 가까운 피벗일 수록 반올림오차가 생겨날 수 있기 때문에 문제가 발생할 수 있다. 따라서, 각 행들이 정규화되기 전에, 피벗요소 아래에 열중에 가장

Algorithm 7.1 Forward Elimination

Let loop k control the elimination step, loop i control $i-$th row accessing and loop j control $j-$th column accessing, the sequential Gaussian Elimination alogithms is decribed as follow(in numerical program, vector **B** is stored as $(n+1)-$th column of matrix **A**):

for $k = 1$ to $n-1$ **do**
  for $i = k+1$ to n **do**
    $a_{ik} = a_{ik}/a_{kk}$;
    for $j = k+1$ to $n+1$ **do**
      $a_{ij} = a_{ij} - a_{ik} \times a_{kj}$;
    end for
  end for
end for

Note that since the entries in the lower triangluar matrix vanish after elimination, their space is used to store multipliers $a_{ik} = a_{ik}/a_{kk}$

Algorithm 7.2 Backward Substitution

for $i = n$ to 1 **do**
  for $j = i+1$ to n **do**
    $x_i = x_i - a_{ij} \times x_j$;
  end for
  $x_i = x_i/a_{ii}$;
end for

Note that x_i is stored in the space of $a_{i,n+1}$.

큰 계수를 찾은 후, 가장 큰 요소가 피벗요소가 되도록 행의 위치를 교환하여야 한다. 이를 부분피벗화(partial pivoting)라고 한다. 만일 행뿐만 아니라 열에서도 가장 큰 요소를 찾아 위치교환을 한다면, 완전피벗화(complete pivoting)라고 부른다.

8 LU분해법과 역행렬 (LU Decomposition and Matrix Inversion)

LU분해법의 주요 장점은, 많은 시간이 소비되는 소거 단계를 공식화함으로써 계수행렬 **A**에만 연산을 수행한다는 것이다. 따라서, 이 방법은 **A**가 주어지고, 여러가지 다른 값을 가지는 우변벡터 **B**를 사용하여 반복적으로 해를 구하는 경우에 적합하다.

$$\mathbf{Ax} = \mathbf{B} \tag{8.1}$$

Gauss소거법은 전진소거와 후진대입이라는 두 단계로 이루어져 있는데, 이 중에 전진소거 단계에서 많은 계산시간이 요구된다. 이는 특히 방정식의 수가 많은 경우에 더욱 그러하다. LU분해법(LU decomposition method)으로 시간이 많이 걸리는 **A**의 소거 과정과 우변벡터 **B**에 대한 소거 과정이 분리되게 된다. 따라서, **A**가 처음에 한번만 "분해된다면(decompose)", 우변벡터가 계속 바뀌는 경우에 대해서도 효율적으로 해를 구할 수 있게 된다.

$$\mathbf{Ax} - \mathbf{B} = 0 \tag{8.2}$$

식(8.2)가 상부삼각 시스템으로 표시될 수 있다고 가정하면,

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} \tag{8.3}$$

이는 Gauss 소거법의 첫 번째 단계에서 수행되는 과정과 유사하다. 즉, 소거 과정을 통해 주어진 시스템이 상부삼각 형태로 변환된다. 식(8.3)은 다음과 같이 표시될 수 있다.

$$\mathbf{Ux} - \mathbf{D} = 0 \tag{8.4}$$

여기서 다음과 같이 대각선에 1의 값을 가지는 하부삼각행렬이 있다고 가정하고

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \tag{8.5}$$

이를 식(8.4)에 곱하면 식(8.2)의 결과가 얻어진다고 가정하자. 즉,

$$\mathbf{L}(\mathbf{Ux} - \mathbf{D}) = \mathbf{Ax} - \mathbf{B} \tag{8.6}$$

만일 위 방정식이 성립한다면, 행렬의 곱셈 법칙으로부터 다음과 같은 결과를 얻는다.

$$\mathbf{LU} = \mathbf{A} \tag{8.7}$$

$$\mathbf{LD} = \mathbf{B} \tag{8.8}$$

식(8.4),(8.7),(8.8)을 기초로 하여, 해를 구하기 위한 두 단계 전략을 구성한다.

1 **LU**분해단계 **:** **A**를 하부삼각행렬 **L**와 상부삼각행렬 **U**로 분해한다.

2 대입단계 **:** **L**과 **U**를 사용하여 주어진 우변벡터 **B**에 대한 해 **x**를 구한다. 이 과정은 다시 두 단계로 구성된다. 먼저, 식(8.8)을 이용하여 벡터 **D**를 전진대입하여 구한다. 그리고, 그 결과를 식(8.4)에 대입한 후 후진대입으로 **x**를 구한다.

Algorithm 8.1 LU Decompose

```
function DECOMPOSE(a ∈ A,n)
    for k = 1 to n − 1 do
        for i = k + 1 to n do
            a_ik = a_ik / a_kk;
            for j = k + 1 to n + 1 do
                a_ij = a_ij − a_ik × a_kj;
            end for
        end for
    end for
end function
```

Algorithm 8.2 Substitution

```
function SUBSTITUTE(a ∈ A,n,b ∈ B,x ∈ x)                ▷ Forward substitution
    for i = 2 to n do
        sum = b_i
        for j = 1 to i − 1 do
            sum = sum − a_ij × b_i
        end for
        b_i = sum
    end for                                              ▷ Back substitution
    x_n = b_n / a_nn
    for i = n − 1 : −1 : 1 do
        sum = 0
        for j = i + 1 to n do
            sum = sum + a_ij × x_j
        end for
        x_i = (b_i − sum) / a_ij
    end for
end function
```

8.1 역행렬

LU분해의 가장 큰 강점은 여러개의 우변벡터에 대하여 효율적으로 해를 구할 수 있다는 것이다. 즉, 이방법은 역행렬을 구하기 위하여 여러개의 단위벡터를 가지고 계산하는 경우에 이상적이라고 할 수 있다.

예제 역행렬

LU분해를 사용하여 다음 **A** 행렬의 역행렬을 구하라.

$$\mathbf{A} = \begin{bmatrix} 3 & -0.2 & -0.2 \\ 0.1 & 7 & -0.3 \\ 0.3 & -0.2 & 10 \end{bmatrix}$$

LU분해를 하여 다음과 같은 상부, 하부삼각행렬을 얻을 수 있다.

$$\mathbf{U} = \begin{bmatrix} 3 & -0.1 & -0.2 \\ 0 & 7.00333 & -0.293333 \\ 0 & 0 & 10.0120 \end{bmatrix}$$

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 0.0333333 & 1 & 0 \\ 0.100000 & -0.0271300 & 1 \end{bmatrix}$$

해 단위벡터(첫 번째 행에 1이 있음)를 우변벡터로 놓고, 전진대입하여 해를 구함으로써 역행렬의 첫 번째 열이 계산된다. 즉, 하부삼각행렬인 식(8.8)을 다음과 같이 놓을 수 있다.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0.0333333 & 1 & 0 \\ 0.100000 & -0.0271300 & 1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix}$$

전진대입으로 해를 구하면 $\mathbf{D}^T = [1, -0.03333, -0.1009]$ 가 된다. 이 벡터를 식(8.3)의 우변으로 사용하면,

$$\begin{bmatrix} 3 & -0.1 & -0.2 \\ 0 & 7.00333 & -0.293333 \\ 0 & 0 & 10.0120 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 1 \\ -0.03333 \\ 0.1009 \end{Bmatrix}$$

위 식을 후진대입으로 풀면, $\mathbf{x}^T = [0.33249, -0.00518, -0.01008]$ 을 얻게 되어 역행렬의 첫 번째 열은,

$$\mathbf{A}^{-1} = \begin{bmatrix} 0.33249 & 0 & 0 \\ -0.00518 & 0 & 0 \\ -0.01008 & 0 & 0 \end{bmatrix}$$

역행렬의 두 번째 열을 결정하기 위하여, 식(8.8)을 다음과 같이 공식화한다.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0.0333333 & 1 & 0 \\ 0.100000 & -0.0271300 & 1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 1 \\ 0 \end{Bmatrix}$$

위 식을 $\mathbf{D}$에 대하여 풀고, 식(8.3)을 사용하여 $\mathbf{x}^T = [0.3004944, 0.142903, 0.00271]$을 구해, 역행렬의 두 번째 열은 다음과 같이 된다.

$$\mathbf{A}^{-1} = \begin{bmatrix} 0.33249 & 0.004944 & 0 \\ -0.00518 & 0.142903 & 0 \\ -0.01008 & 0.00271 & 0 \end{bmatrix}$$

마지막으로 $\mathbf{B} = [0, 0, 1]$을 사용하여 전진, 후진대입으로 해를 구하면, $\mathbf{x}^T = [0.006798, 0.004183, 0.09988]$을 얻게 되므로, 역행렬의 마지막 열은 다음과 같이 된다.

$$\mathbf{A}^{-1} = \begin{bmatrix} 0.33249 & 0.004944 & 0.006798 \\ -0.00518 & 0.142903 & 0.004183 \\ -0.01008 & 0.00271 & 0.09988 \end{bmatrix}$$

이 결과에 대한 검증은 $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$가 되는 것을 보임으로써 증명할 수 있다.

Part III

최적화
(Optimization)

최적화는 5장과 6장과 같이 둘 다 함수 위의 한 점을 추측하거나 찾는다는 점에서 관련성이 있다. 두 방법의 근본적인 차이점으로 구간법은 함수들의 근을 구하는 반면 최적화(optimization)은 최소값 혹은 최대값을 찾는 것이다.
최적점은 곡선 위의 평탄한 점으로, 수학적 용어로 나타내면 도함수 $f'(x)$가 0이 되는 점에 해당한다. 또한 2차도함수 $f''(x)$의 부호에 의해 최적점이 최소값인지 최대값인지를 나타낸다. 하지만 $f'(x)$가 손쉽게 구해지지 않는 경우 이러한 최적화 방법은 종종 복잡해지게 된다. 이 경우 도함수를 근사하기 위하여 유한자분 근사법을 사용하여야 한다.
최적화는 근 구하는 방법 이상으로 단순히 근 구하기가 아닌 다른 수학적 접근을 포함한다. 이러한 접근은 다차원의 최적화를 더욱 가능하게 한다.

- 최소 중량 및 최대 강도의 항공기 설계
- 우주선의 최적궤도
- 최소 비용의 토목공학 구조물 설계
- 최대 수력에너지를 내면서 호수 때 손실을 최소화하는 댐의 수자원 프로젝트
- 위치에너지를 최소화하는 구조물 거동의 예측
- 최소 비용의 재료 절삭 방법
- 최대 효율의 펌프와 열전달 장치의 설계
- 전기회로와 기계의 열발생을 최소화하면서 최대 출력설계
- 한 번의 판촉 여행으로 여러 도시를 방문시 가장 짧은 경로 계산
- 최적 계획과 스케줄링(scheduling)
- 최소 오차를 갖는 통계분석과 모델
- 최적 파이프라인 회로 설계
- 재고 제어
- 비용을 최소화하는 보수계획
- 대기시간과 준비시간의 최소화
- 최소 비용으로 수질표준을 만족하는 폐기물 처리 시스템 설계

지금까지 다룬 대부분의 공학적 모델들은 서술적(descriptive) 모델이었다. 즉, 공학적 장치나 시스템의 거동을 나타내기 위하여 유도된 모델이다. 반면에 최적화는 문제의 "최상의 결과" 혹은 최적해를 구하는 것을 다루는 일련의 과정, 혹은 가장 좋은 설계를 설정하게 되는 설정적(prescriptive) 모델이라는 용어를 사용한다. 즉, 엔지니어는 효율적인 방법으로 프로젝트를 수행하거나 제품을 개발하여야 하는데 그 과정중에 실제 문제들에 의한 물리적 제약이 존재하게 된다. 또는 비용을 계속적으로 줄여야만 한다. 따라서 성능과 제약조건 사이의 균형을 구하는 최적화 문제에 항상 부딪히게 된다.

예제 낙하산 비용의 최적화

전쟁지역의 피난민에게 물자들을 공수하는 기관에 속한 엔지니어라고 가정하자. 물자들을 가능한 한 낮은 고도에서 떨어뜨려 관측되지 않으면서 가능한 피난민 캠프에 근접하려고 한다. 낙하산은 수송기에서 떨어지자마자 펼쳐지며, 손상을 줄이기 위하여 지면에 도착하는 충격속도가 적어도 임계속도 $v_c = 20m/sec$ 보다는 작아야한다. 낙하용 낙하산의 단면적은 반구의 단면적과 같다.

$$A = 2\pi r^2$$

질량을 낙하산에 연결하는 16개의 줄이 길이는 다음 식과 같이 낙하산 반지름으로 표시된다.

$$l = \sqrt{2}r$$

낙하산에 걸리는 항력계수는 다음 식과 같이 단면적의 선형함수로 표현된다.

$$c = k_c A$$

여기서, c는 항력계수(kg/s), k_c는 항력에 대한 면적의 영향을 나타내는 비례상수로 $[kg/(s \cdot m^2)]$의 단위를 갖는다. 또한 전체 하중을 여러 개의 뭉치로 나누면, 각각의 뭉치 덩어리 질량은 다음과 같이 계산된다.

$$m = \frac{M_t}{n}$$

여기서 m은 각 뭉치의 질량(kg)이며, M_t는 낙하된 전체질량(kg), 그리고 n의 전체 뭉치수이다. 끝으로 각 낙하산의 가격은 다음 식과 같이 비선형식으로 표시된다.

$$p = c_0 + c_1 l + c_2 A^2$$

여기서, p는 대당가격, c_0, c_1, c_2는 가격 계수들이다. c_0는 낙하산들의 기본 가격이며, 큰 낙하산은 작은 크기의 낙하산에 비해 제작하기가 훨씬 어렵기 때문에 면적에 대하여 비선형적 가격증가를 보인다.
충분히 작은 낙하 충격속도를 가지면서 최소의 비용이 되도록 낙하산 크기 r과 개수 n을 결정하라.
목적함수를 정의하면 다음식으로 표현된다.

$$\underset{n,r}{\text{minimize}} \quad C(n,r) = np$$

$$\text{subject to} \quad v \le v_c,\ n \ge 1, n \in \mathbb{Z}$$

이 문제는 비선형 구속화 문제가 된다. 넓은 의미로는 수식화 되었어도 어떻게 충격속도 v를 구할 것이가라는 다른 문제가 남는다. 낙하하는 물체의 속도는 다음과 같이 계산되었다.

$$v = \frac{gm}{c}\left(1 - e^{-(c/m)t}\right) \tag{8.9}$$

여기서 v는 속도(m/sec), g는 중력가속도 (m/sec^2), m은 질량(kg), 그리고 t는 시간(sec)이다. 속도와 시간의 관계가 주어져있지만 물체가 떨어지는데 걸리는 시간 t를 결정해야하고, 지면에 닿을때의 속도가 임계속도 v_c 이하여야 한다.

낙하위치에 따라 지면에 도달할때의 속도와 시간이 달라지기 때문에 식(8.9)을 적분하여 얻어야 한다. 낙하높이 z와 낙하시간 t사이의 관계는

$$z = \int_0^t \frac{gm}{c}\left(1 - e^{-(c/m)t}\right) dt \tag{8.10}$$

$$= z_0 - \frac{gm}{c}t + \frac{gm^2}{c^2}\left(1 - e^{-(c/m)t}\right) \tag{8.11}$$

여기서 z_0는 낙하 최초의 높이(m)다. 즉 이식을 통해 시간 t가 주어지면 높이 z를 예측할 수 있다. 그러나 이 문제는 높이 z_0만큼 떨어지는데 걸리는 시간을 계산하여야 하기 때문에 식(8.11)의 근을 구하눈 문제로 다시 수식화 하여야 한다. 즉, 높이 z가 0이 되는데 필요한 시간에 대한 식을 쓰면,

$$f(t) = z_0 - \frac{gm}{c}t + \frac{gm^2}{c^2}\left(1 - e^{-(c/m)t}\right) \tag{8.12}$$

$$= 0 \tag{8.13}$$

$$root\left[z_0 - \frac{gm}{c}t + \frac{gm^2}{c^2}\left(1 - e^{-(c/m)t}\right)\right] \tag{8.14}$$

으로 정리가 된다. 이때 시간을 구하기 위해 구간법(수치해법)을 사용하여 근을 구하여야 한다. 그 이후 시간t가 계산이 되면, 지면에 닿을때의 속도v를 구할 수 있다. 이 속도가 임계속도 이내인지 판별해야하는 조건이 추가가 된다.

다음 조건으로 총가격을 최소화 시키는 최적의 뭉치의 개수 n과, 최적의 낙하산의 반지름 r을 찾아라.

parameter name	parameters	value	unit
초기 낙하높이	z_0	100	m
면적당 항력비례상수	k_c	200	$kg/(s\cdot m)^2$
수송할 물자의 전체 질량전체질량	M_t	100	ton
낙하산 개당 기본 제작단가	c_0	1,000	KRW
낙하산 줄의 길이당 단가	c_1	300	KRW$/m$
낙하산 섬유 면적당 단가	c_2	50	KRW$/m^2$

이 계산을 수계산 혹은 컴퓨터계산을 통해 결과를 얻어내는데 어떠한 수치해석 방법을 사용하여도 무방하다. 또한 최적의 결과(가격)가 아니어도 조건은 만족시켜야 한다. 단, 수계산의 경우 최소한 10번의 시행오차과정을 컴퓨터계산의 경우 20번의 시행오차과정을 수행하고, 최적의 단가, 개수, 전체가격, 지면에 도달할 때의 시간, 지면에 도달할 때의 속도를 테이블로 작성하라.

해 낙하산 최적화 문제는 "구속 비선형 다변수 함수 최적화(constrained nonlinear multivariable function minimization)" 방식을 사용한다. 이 함수는 구속최적화 67페이지 10.4장에서 다룬다. 일반적으로 본 강의에서는 MATLAB의 `fmincon()`을 사용하도록 한다. 우선 낙하산이 지면에 도달할 때의 시간 t를 구하기 위한 낙하거리의 함수를 정의하자.

```
1 function y=fdist(t,g,m,c,z0)

3 y=z0-(g*m)/c*t+(g*m^2)/(c^2)*(1-exp(-(c/m)*t));
```

Code [MATLAB] 7: 낙하 거리 함수 `fdist(t,g,m,c,z0)`

구간법 fminbnd()함수를 이용하여 근을 구하고, 도달거리의 조건에 대한 함수를 정의하자, 이 때, 구속조건은 부등식구속조건 con과 등식구속조건 ceq가 사용이 되는데, 등식구속조건은 사용되지 않기 때문에 ceq의 리턴값은 없다. 구속조건함수를 pricecon(x)로 정의하자.

```
function [con,ceq]=pricecon(x)

n=x(1);
r=x(2);
z0=100;

kc=200;     % kg/(s*m)^2
Mt=100000; % kg
va=20;      % m/sec
g=9.8;      % m/sec^2

A=2*pi*r^2;

m=Mt/n;
c=kc*A;
T=fminbnd(@(t) abs(fdist(t,g,m,c,z0)),0,10000);
gl=fdist(T,g,m,c,z0);
v=g*m/c*(1-exp(-(c/m)*T));
con(1)=abs(v)-va;
con(2)=abs(gl)-0.001;
ceq=[];
```

Code [MATLAB] 8: 구속조건함수 pricecon(x)

그리고 마지막으로 목적함수인 총 낙하산 가격에 대한 함수를 정의한다.

```
function C = pricep(x)

n=x(1);
r=x(2);

A=2*pi*r^2;
l=sqrt(2)*r;

c0=1000; % KRW
c1=300;  % KRW/m
c2=50;   % KRW/m^2

C=n*(c0+c1*l+c2*A^2);
```

Code [MATLAB] 9: 목적함수 pricep(x)

구속최적화 방법을 수행하는데 낙하거리의 함수는 근을 구하기 위한 함수였고 두가지함수(목적함수, 구속조건함수)만 필요로한다. 이를 통해 총 낙하산 가격의 최소화 구문은

[x,fval]=fmincon(@pricep,x0,[],[],[],[],lb,ub,@pricecon,options);를 통해 구하게 된다. 메인프로그램은 다음 코드와 같다,

```
clear all; clc; close all;

lb=[2 0.01];
ub=[100000 100000];
x0=lb;

options = optimset('Algorithm','active-set','MaxFunEvals',10000);

[x,fval]=fmincon(@pricep,x0,[],[],[],[],lb,ub,@pricecon,options);

n=ceil(x(1));
r=x(2);
z0=100;      % m
g=9.8;       % m/sec^2
kc=200;      % kg/(s*m)^2
Mt=100000;   % kg
A=2*pi*r^2; % m^2
m=Mt/n;
c=kc*A;
C=pricep([n,r]);
t=fminbnd(@(t) abs(fdist(t,g,m,c,z0)),0,10000);
gl=fdist(t,g,m,c,z0);
v=g*m/c*(1-exp(-(c/m)*t));

disp(['n : ',num2str(n)])
disp(['Radius of parachute (m) : ',num2str(r)])
disp(['Drag coefficient (kg/sec) : ',num2str(c)])
disp(['Mass per each (kg) : ',num2str(m)])
disp(['Arrival time (sec) : ',num2str(t)])
disp(['Arrival velocity (m/sec) : ',num2str(v)])
disp(['Unit price (KRW/EA) : ',num2str(ceil(C/n))])
disp(['Total price (KRW) : ',num2str(ceil(C))])
```

Code [MATLAB] 10: 최적화 실행프로그램

메인프로그램을 수행했을 때, 결과 값을 보자. 이 문제에는 공학적인 접근에서 일반적으로 접하게 되는 최적화 문제들의 기초적인 요소들을 대부분 포함하고 있다. 이러한 최적화 문제들은

- 우리의 목표를 표현하는 "목적함수"를 포함한다.
- 많은 "설계변수"를 포함하며, 실수 혹은 정수가 될 수도 있다. 낙하산 예제에서 r은 실수이며, n은 정수이다.
- 설계에 대한 제약을 반영하는 "구속조건"을 포함한다.

위의 과제와 예제를 통해 우리는 최적화를 수행하는데 필요한 함수를 공부할 것이다. 컴퓨터 프로그램을 수행하는데 있어서, 논리적으로 최적화 문제의 해를 구하는 것도 중요하지만 이론적인 배경도 함께 숙지하고 있어야한다. 도구에

Parameters	Value	Unit
Number of parachutes (n)	49	EA
Radius of parachute (r)	0.88057	meter
Drag coefficient (c)	974.3905	kg/sec
Mass per each (m)	2040.8163	kg
Arrival time (t)	6.8883	sec
Arrival velocity (v)	19.7601	m/sec
Unit price (p)	2,561	KRW/EA
Total price (C)	125,460	KRW

종속되어 공학적인 문제들을 수행하게 되면 그 컴퓨터 연산결과에 대하여 신하게 되며 공학적인 고찰또한 결여되게 된다. 가능한 해 집단에서 최종적으로 우리는 특정한 해를 사용하게 되기 때문에, 이러한 고찰이 매우 중요하다.

최적화(optimization) 문제는 일반적으로 다음과 같이 표현된다.
$f(x)$를 최소화 혹은 최대화하며 다음의 조건을 만족하는 변수 x를 구하라.

$$d_i(\mathbf{x}) \leq a_i \qquad i = 1,2,\cdots,m \tag{8.15}$$

$$e_i(\mathbf{x}) = b_i \qquad i = 1,2,\cdots,p \tag{8.16}$$

여기서 $\mathbf{x}$는 n차원의 설계벡터(design vector)이고, $f(\mathbf{x})$는 목적함수이며, $d_i(\mathbf{x})$는 부등식구속조건(inequality constraints), $e_i(\mathbf{x})$는 등식구속조건(equality constraints), 그리고 a_i와 b_i는 상수들이다.
최적화 문제들은 $f(\mathbf{x})$의 형태에 따라 다음과 같이 분류된다.

- $f(\mathbf{x})$와 구속조건이 선형적이면, 선형프로그래밍 문제이다.
- $f(\mathbf{x})$가 2차 함수이고 구속조건이 선형적이면, 2차 프로그래밍 문제이다.
- $f(\mathbf{x})$가 선형도 2차 함수도 아니거나, 혹은 구속조건이 비선형이면 비선형 프로그래밍 문제 이다.

또한 식(8.15)과 식(8.16)이 포함되면 구속최적화(constrained optimization) 문제이며, 그렇지 않으면 비구속최적화(unconstrained optimization) 문제이다.

9 비구속 최적화 (Unconstrained Optimization)

9.1 황금분할법

한 개의 비선형 방정식의 근을 구하는데 있어서의 목표는 $f(x)$가 0이 되는 x의 값을 찾는 것이다. 단일변수 최적화 문제(single-variable optimization)는 $f(x)$의 최대값과 최소값을 나타내는 극값인 x를 구하는 것이 목적이다.
황금분할탐색법은 단순하며 일반적으로 적용 가능한 단일변수 탐색방법이다. 이 때 근을 구하기 위해 구간법에서 학습한 이분법과 의미를 같이한다.

$$x_r = \frac{x_l + x_u}{2}$$

이분법에서와 같이 한 개의 해를 담는 구간을 먼저 정의한다. 즉, 그 구간에는 한개의 최대값이 담겨 있어야 한다. 이러한 경우를 단모드(unimodal)이라고 한다.
두 개의 함수값을 사용하는 것(부호의 변화를 탐색하여 근을 찾음)과 다르게 최대값의 발생여부를 탐색하기 위하여 세 개의 함수값이 필요하다. 따라서 구간 내의 또 다른 하나의 점을 선택하여야 한다. 다음으로는 네 번째 점을 구간 내에 적절하게 배치하여 최대값이 앞의 세 점 사이에 나타났는지, 나중 세 점에서 나타났는지를 확인하는 것이다. 이러한 방법이 효율적으로 진행되기 위하여 중간점들의 현명한 선택이 중요하다. 이러한 방법이 효율적으로 진행되기 위하여 중간점들의 현명한 선택이 중요하다. 이분법에서처럼 과거의 값들을 새로운 값들로 치환하여 함수값 계산을 줄이는 것이 목표이다.

$$l_0 = l_1 + l_2 \tag{9.1}$$

$$\frac{l_1}{l_0} = \frac{l_2}{l_1} \tag{9.2}$$

첫 번째 조건은 두 개의 부가적 길이인 l_1 과 l_2 의 합이 원래 구간의 길이가 되도록 한다. 두 번째 조건은 길이의 비가 같도록 함을 나타낸다.

$$\frac{l_1}{l_1 + l_2} = \frac{l_2}{l_1} \tag{9.3}$$

위 식의 역수를 취한 후 $R = l_2/l_1$ 으로 하면, 다음 식을 얻게 된다.

$$1 + R = \frac{1}{R} \tag{9.4}$$

$$R^2 + R - 1 = 0 \tag{9.5}$$

여기서 양수의 근을 구하면 다음과 같다.

$$R = \frac{-1 + \sqrt{1 - 4(-1)}}{2} = \frac{\sqrt{5} - 1}{2} = 0.61803\cdots \tag{9.6}$$

이 값은 고대로부터 황금비(golden ratio)라고 알려져 왔다. 이것을 이용하여 최적값을 효율적으로 얻을 수 있기 때문에 우리가 지금까지 개념적으로 전개해온 황금분할법의 핵심요소가 된다. 이 방법을 컴퓨터에서 구현되도록 알고리즘을 유도해보자. 이 방법은 $f(x)$ 의 국부적 극점을 포함하는 두 개의 초기추측값 x_l 과 x_u 로 시작한다. 다음은 두 개의 내부점 x_1 과 x_2 를 황금비를 이용하여 구한다.

$$d = \frac{\sqrt{5} - 1}{2}(x_u - x_l)$$

$$x_1 = x_l + d$$

$$x_2 = x_u - d$$

두 개의 내부점에서 함수값을 구하면 두 가지 결과를 얻을 수 있다.

1 $f(x_1) > f(x_2)$ 이면 x_2 의 왼쪽에 있는 x 의 영역, 즉, x_l 로 부터 x_2 까지의 구간을 제거할 수 있다. 왜냐하면 그 구간에는 최대값이 없기 때문이다. 이 경우 x_2 가 다음 반복 시행의 새로운 x_l 이 된다.

2 $f(x_2) > f(x_1)$ 인 경우 x_1 의 오른쪽에 있는 영역, 즉 x_1 부터 x_u 까지의 구간이 제거된다. 이 경우 x_1 이 다음 반복시행의 새로운 x_u 가 된다.

여기서 황금비 사용에 따른 실제적인 이점을 생각해 보자. 최초의 x_1 과 x_2 가 황금비를 따라 선택되었기 때문에 다음 반복수행시 모든 "함수값"을 다시 계산할 필요가 없다. 알고리즘을 마루리하기 위하여 새로운 x_1 을 결정하여야 한다. 이것은 전과 같은 비례상수를 사용하여 얻어진다.

10 구속 최적화
(Constrained Optimization)

목적함수와 구속조건이 선형인 문제를 이장에서 논의 한다. 이 경우 주어진 함수의 선형성을 이용하는 특별한 방법들이 존재한다. 선형프로그래밍(linear programming)법이라고 불리는 알고리즘은 수천 개의 변수와 구속조건을 갖는 대형 문제를 매우 효율적으로 계산한다. 이 방법은 공학과 경영 분야에 걸쳐 많은 문제들에 적용된다. 이후 비선형 구속 최적화의 일반적인 문제들을 간단히 다룬 후에 MATLAB 라이브러리의 사용법에 대해 개략적으로 설명한다. 선형프로그래밍(LP)은 제한된 자원 같은 구속조건을 갖는 경우, 이윤을 최대화하거나 비용을 최소화하여 원하는 목적을 만족시키는 최적화 문제를 다룬다. 선형이라는 용어는 목적함수나 구속조건을 표현하는 수학적 함수들이 모두 선형(linear)임을 나타낸다. 프로그래밍이라는 용어는 "컴퓨터 프로그래밍"을 의미하기보다는 "스케쥴"을 정하거나 과제목록을 정하는 의미를 함축하고 있다.

10.1 등식 구속조건
Equality constraints

10.1.1 Lagrangian Methods

일반적인 최적화 문제를 가정하자.

$$\begin{aligned} P: \underset{x}{\text{maximize}} \quad & f(x) \\ \text{subject to} \quad & g(x) = b, x \in X, \end{aligned}$$

여기서, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ (n개의 변수(variables)와 m개의 구속조건(constraints)). 라그랑지안(Lagrangian)은

$$L(x, \lambda) = f(x) + \lambda^T (b - g(x)) \tag{10.1}$$

여기서, $\lambda \in \mathbb{R}^m$(각 구속조건의 한개의 요소). λ의 각 요소를 라그라지안 승수(Lagrangian multiplier)라고 한다.

다음 일반적인 문제를 가정하자.

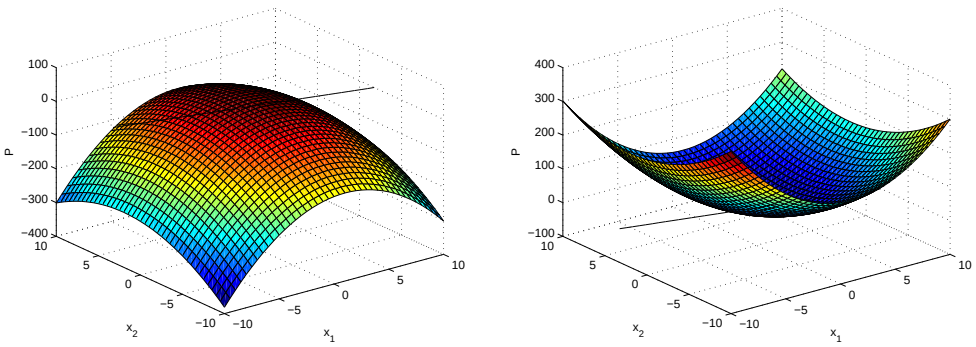
$$\begin{aligned} \underset{x_1, x_2}{\text{maximize}} \quad & 5 - (x_1 - 2)^2 - 2(x_2 - 1)^2 \\ \text{subject to} \quad & x_1 + 4x_2 = 3 \end{aligned}$$

만약 구속조건을 무시한다면, 우리는 이문제의 해를 쉽게 풀 수 있다. $x_1 = 2$, $x_2 = 1$ 그렇지만 구속조건에는 큰값으로 만족하지 않는다. 구속조건의 값을 크게 만드는데에 있어 λ를 도입하여 손실가중치를 정의하자. 위의 함수를 다음과 같이 쓸 수 있다.

$$L(x_1, x_2, \lambda) = 5 - (x_1 - 2)^2 - 2(x_2 - 1)^2 + \lambda(3 - x_1 - 4x_2) \tag{10.2}$$

이러한 함수를 문제의 라그랑지안(Lagrangian of the problem)이라고 한다. 결국 식의 오른쪽에 구속조건의 값이 추가된 λ값을 조정하는 문제로 귀결된다. λ의 값에 따른 답을 찾아보자.

- $\lambda = 0$일 때, $(x_1, x_2) = (2, 1)$.
- $\lambda = 1$일 때, $(x_1, x_2) = (3/2, 0)$.
- $\lambda = 2/3$일 때, $(x_1, x_2) = (5/3, 1/3)$. (구속조건을 정확히 만족시킴)



(a) Concave problem **(b)** Convex problem

Figure 10.1: Convex and concave problem

여러개의 구속조건을 갖는 최적화 방식을 일반화 시키면

$$\begin{aligned} P: \underset{x}{\text{maximize}} \quad & f(x) \\ \text{subject to} \quad & g_1(x) = b_1 \\ & g_2(x) = b_1 \\ & \vdots \\ & g_m(x) = b_m \end{aligned}$$

라그랑지안(Lagrangian)을 사용한 해는 다음 식으로 찾을 수 있다.

$$L(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i \left(b_i - g_i(x)\right) \tag{10.3}$$

여기서, λ_i는 i번째 구속조건에 관련한 가중치(가격)으로 생각할 수 있다.

Theorem 10.1 (Lagrangian multiplier)**.** *최적화 문제 P를 만족시키는 x^*에 대하여 x^*와 λ^*가 존재 하고, $x^* = (x_1^*, x_2^*, \cdots, x_n^*)$를 함수 $f(x)$를 구속조건 $g_i(x) = b_i$, for $i = 1, 2, \cdots, m$을 만족시키면서 최대화(maximizes) 시키거나 최소화(minimizes) 시키는 값을 가지는 벡터로 가정하면,*

(i) 벡터 $\nabla g_1(x^*), \nabla g_2(x^*), \cdots, \nabla g_m(x^*)$가 모두 선형적으로 독립이거나,

(ii) $\nabla L(x^*, \lambda^*) = 0$을 만족시키는 벡터 $\lambda^* = (\lambda_1^*, \lambda_2^*, \cdots, \lambda_m^*)$가 존재한다.

여기서 ∇ 연산자(Nabla-operator)는 벡터의 편미분 연산자로 다음 식처럼 쓰인다.

$$\nabla = \left(\frac{\partial}{\partial x_1}, \cdots, \frac{\partial}{\partial x_n} \right) \tag{10.4}$$

즉, 위의 라그랑지안 정리는 다음식처럼 쓸 수 있다.

$$\frac{\partial L}{\partial x_1}(x^*, \lambda^*) = \frac{\partial L}{\partial x_2}(x^*, \lambda^*) = \cdots = \frac{\partial L}{\partial x_n}(x^*, \lambda^*) = 0 \tag{10.5}$$

$$\frac{\partial L}{\partial \lambda_1}(x^*, \lambda^*) = \frac{\partial L}{\partial \lambda_2}(x^*, \lambda^*) = \cdots = \frac{\partial L}{\partial \lambda_m}(x^*, \lambda^*) = 0 \tag{10.6}$$

10.2 등식과 부등식 구속조건 (Equality and Inequality Constraints)

등식 구속조건과 부등식 구속조건이 포함된 함수의 최적화는 어떻게 두가지의 조건들을 사용할 수 있느냐에 달려있다. 일반화된 다음 식을 보자.

$$\begin{aligned} P: \underset{x}{\text{maximize}} \quad & f(x) \\ \text{subject to} \quad & g_1(x) = b_1 \\ & \vdots \\ & g_m(x) = b_m \\ & h_1(x) \le d_1 \\ & \vdots \\ & h_p(x) \le d_p \end{aligned}$$

만약 이 조건이 아닌 $\ge$조건의 경우 $\le$를 만들어주기 위해 양변에 -1을 곱하여 위와 동일한 조건을 만들어준다. 또한 같은 방식으로 최소값 문제는 최대값 문제로 변형할 수 있다. 이 때 라그랑지안은

$$L(x,\lambda,\mu) = f(x) + \sum_{i=1}^{m} \lambda_i (b_i - g_i(x)) + \sum_{j=1}^{p} \mu_j (d_i - h_j(x)) \tag{10.7}$$

정리 10.1와 유사하게 다음과 같이 정리한다. $x^* = (x_1^*, x_2^*, \cdots, x_n^*)$를 함수 $f(x)$를 구속조건 $g_i(x) = b_i,$ for $i = 1,2,\cdots,m$과 $h_j(x) \le d_j,$ for $j = 1,2,\cdots,p$을 만족시키면서 최대화(maximizes)시키는 값을 가지는 벡터로 가정하면,

(*i*) 벡터 $\nabla g_1(x^*), \cdots, \nabla g_m(x^*)$와 $\nabla h_1(x^*), \cdots, \nabla h_p(x^*)$가 모두 선형적으로 독립이거나,

(*ii*) 아래의 식을 만족시키는 벡터 $\lambda^* = (\lambda_1^*, \cdots, \lambda_m^*)$와 $\mu^* = (\mu_1^*, \cdots, \mu_p^*)$가 존재한다.

$$\nabla f(x^*) - \sum_{i=1}^{m} \lambda_i^* \nabla g_i(x^*) - \sum_{j=1}^{p} \mu_j^* \nabla h_j(x^*) = 0$$

$$\mu_j^* (d_j - h_j(x^*)) = 0 \text{ (Complementarity)}$$

$$\mu_j^* \ge 0$$

일반적으로 부등식조건이 포함된 해를 구하기 위해서는 우리는 μ_j^*가 0이 되야하는 조건과 $d_j - h_j(x_*) = 0$이 되는 상보성(complementarity)조건을 검토해야한다. 이 조건에 만족하는 여러가지 경우의 수에 근거하여 우리는 적절한 하나의 해 혹은 후보해들을 선택해야한다. 만약 최적의 해가 존재한다면, 하나의 해 아니면 후보해들이 그 답이 될 것이다. 위의 조건들은 Kuhn-Tucker(혹은 Karush-Kuhn-Tucker)조건이라고 말한다. KKT조건에서 말하는 것은 최적의 해 x^*는 부등식 조건에서 꽉 찬 조건이거나 그렇지 않은 경우의 수가 있다. 즉, 부등식 조건에 꽉 찬 조건이 아닌 경우는 무시할 수 있다(해당 경우엔 $\mu_j^* = 0$이 되므로). 그러나 부등식 조건에 꽉 찬 경우는 Lagrangian의 등식조건으로 생각할 수 있기 때문이다.

$$\mu_j^* (d_j - h_j(x^*)) = 0 \text{ (Complementarity)}$$

결국 이 식이 의미하는 것은 최적화 문제에서 부등식조건의 가중치 μ_j^*를 0으로 잡거나, 구속조건을 꽉 채운 등식조건으로 인식하게 한다.

예제 **Complementarity**

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & x^3 - 3x \\ \text{subject to} \quad & x \leq 2 \end{aligned}$$

라그랑지안은

$$L = x^3 - 3x + \mu(2-x)$$

풀면

$$\begin{aligned} \frac{\partial L}{\partial x} = 3x^2 - 3 - \mu = 0 \\ x \leq 2 \\ \mu(2-x) = 0 \\ \mu \geq 0 \end{aligned}$$

2가지의 상보성 조건이 존재한다.

- 만약 $\mu = 0$의 경우에 $3x^2 - 3 = 0$이 되어 $x = \pm 1$이 되고 $f(1) = -2$, $f(-1) = 2$가 되기 때문에 해로 적합하다.
- 만약 $x = 2$라면 $\mu = 9$가 된다. 이때도 마찬가지로 $f(2) = 2$가 되기 때문에 해로 적합하다.

따라서 최대값을 갖는 경우는 2가지의 해를 가지게 된다. $x = -1$과 $x = 2$.

예제 **Standard form**

$$\begin{aligned} \underset{x,y}{\text{minimize}} \quad & (x-2)^2 + 2(y-1)^2 \\ \text{subject to} \quad & x + 4y \leq 3 \\ & x \geq y \end{aligned}$$

위의 식을 최대값을 갖는 형태와 표준형태로 변형하면,

$$\begin{aligned} \underset{x,y}{\text{maximize}} \quad & -(x-2)^2 - 2(y-1)^2 \\ \text{subject to} \quad & x + 4y \leq 3 \\ & -x + y \leq 0 \end{aligned}$$

라그랑지안은

$$L(x, y, \mu_1, \mu_2) = -(x-2)^2 - 2(y-1)^2 + \mu_1(3 - x - 4y) + \mu_2(0 + x - y)$$

풀면 다음 조건이 된다.

$$\begin{aligned}
\frac{\partial L}{\partial x} &= -2(x-2) - \mu_1 + \mu_2 = 0 \\
\frac{\partial L}{\partial y} &= -4(y-1) - 4\mu_1 - \mu_2 = 0 \\
\frac{\partial L}{\partial \mu_1} &= \mu_1(3-x-4y) = 0 \\
\frac{\partial L}{\partial \mu_2} &= \mu_1(x-y) = 0 \\
x+4y &\le 3 \\
-x+y &\le 0 \\
\mu_1, \mu_2 &\ge 0
\end{aligned}$$

여기서 총 2개의 상보성 조건이 존재하기 때문에, 우리는 4가지의 경우를 체크해야한다.

- $\mu_1 = 0$, $\mu_2 = 0$의 경우 $x = 2$, $y = 1$이 되기 때문에 해로 부적합하다.
- $\mu_1 = 0$, $x - y = 0$의 경우 $x = 4/3$, $y = 4/3$, $\mu_2 = -4/3$이 되기 때문에 해로 부적합하다.
- $\mu_2 = 0$, $3 - x - 4y = 0$의 경우 $x = 5/3$, $y = 1/3$, $\mu_1 = 2/3$이 되므로 해로 적합하다.
- $3 - x - 4y = 0$, $x - y = 0$의 경우 $x = 3/5$, $y = 3/5$, $\mu_1 = 22/25$, $\mu_2 = -48/25$가 되므로 해로 부적합하다.

따라서 최적해는 $x = 5/3$, $y = 1/3$임을 알 수 있다.

10.3 선형계획모형 해법

선형계획모형의 해를 구하는 해법으로 도해법(graphical solution)과 대수적 방법(algebraic method)이 있다. 도해법은 의사결정변수가 2개인 경우에 2차원 평면에서 적용될 수 있어 유용하지만, 3개이상인 경우 적용되는데 어려움이 발생하여 일반적인 해법이라고 보기 어렵다. 일반적으로 2개 이상의 변수에서 도해법이 유용한 경우는 고정된 조건에 의해 특성을 알아보려할 때 유용하다. 또한 최적해를 구하는 해법의 특성을 이해하는데에도 필요하다. 보통 변수가 2개를 초과하면 대수적인 방법으로 해를 구해야한다. 대표적인 대수적 방법으로는 Simplex법, 타원체해법(ellipsoid algorithm), 그리고 Karmarkar의 알고리즘이 있다. Simplex법은 1947년 George B, Danzig에 의하여, 타원체해법은 1979년 Khachian에 의하여, Karmarker의 알고리즘은 1984년 Karmarkar에 의하여 개발되었으며 내부점해법(interior-point algorithm)으로도 불린다. 타원체해법은 현재 거의 사용되지 않고 있고, 규모가 큰(large scale) 선형계획모형에 있어서는 근사치를 제고하는 내부점해법(interior-point method)이 Simplex법보다 복잡성 측면에서 우수한 것으로 알려져 있으나 일반적인 규모에서는 Simplex법이 효용성이 인정되어 아직도 널리 사용되고 있다. MATLAB Optimization Toolbox 패키지의 `linprog`의 함수에서도 규모를 분리하여 내부점해법과 Simplex법을 모두 차용하고 있다.

문제 **LP**문제 세우기

다음의 문제는 화학공학, 혹은 석유공학분야에서 발생하는 문제이다. 그러나 유한한 자원을 가지고 제품을 생산하는 모든 공학분야에도 적용할 수 있다.

가스를 정제하는 공장이 매주 일정한 양의 원료가스를 받는다고 가정한다. 원료가스는 난방용 가스로 사용되는 보통 및

우수 품질의 두 가지 등급으로 정제된다. 이러한 등급들의 가스는 매우 수요가 높으며(항상 잘 팔림), 회사에 각각 다른 이윤을 가져다 준다. 그렇지만 이것들의 생산은 시간과 공장 내의 저장이라는 구속조건을 갖는다. 예를 들면 한 번에 한 개 등급의 제품만이 생산되며, 시설물은 단지 주당 80시간만 사용된다. 또한 각각의 제품들에 대하여 공장 내 보관장소의 제약이 따른다. 이러한 모든 인자들을 아래의 표로 정리되어 있다.

	Product		
Resource	Regular	Premium	Resource Availability
Raw gas	$7m^3$/tonne	$11m^3$/tonne	$77m^3$/week
Production time	$10hr$/tonne	$8hr$/tonne	$80hr$/week
Storage	9tonnes	6tonnes	-
Profit	150/tonne	175/tonne	-

위의 선형프로그래밍문제는 교재에서는 Simplex법을 기초로 설명하고 있다. 일단 이 서술형 문제에서 선형프로그래밍식으로 작성해보자. 공장을 가동하는 기술자는 이윤을 극대화하기 위하여 각각 얼마만큼의 가스들을 생산할지를 결정하여야 한다. 주당 생산되는 보통 등급과 우수 등급의 제품의 양을 각각 x_1 과 x_2 라고 한다면, 주당 전체 이윤은 다음과 같이 계산된다.

$$\text{total profit per week} = 150x_1 + 175x_2$$

구속조건들은 비슷한 방법으로 구할 수 있다. 예를 들면 전체 원료가스의 사용량은 다음과 같이 계산된다.

$$\text{total gas used amount per week} = 7x_1 + 11x_2$$

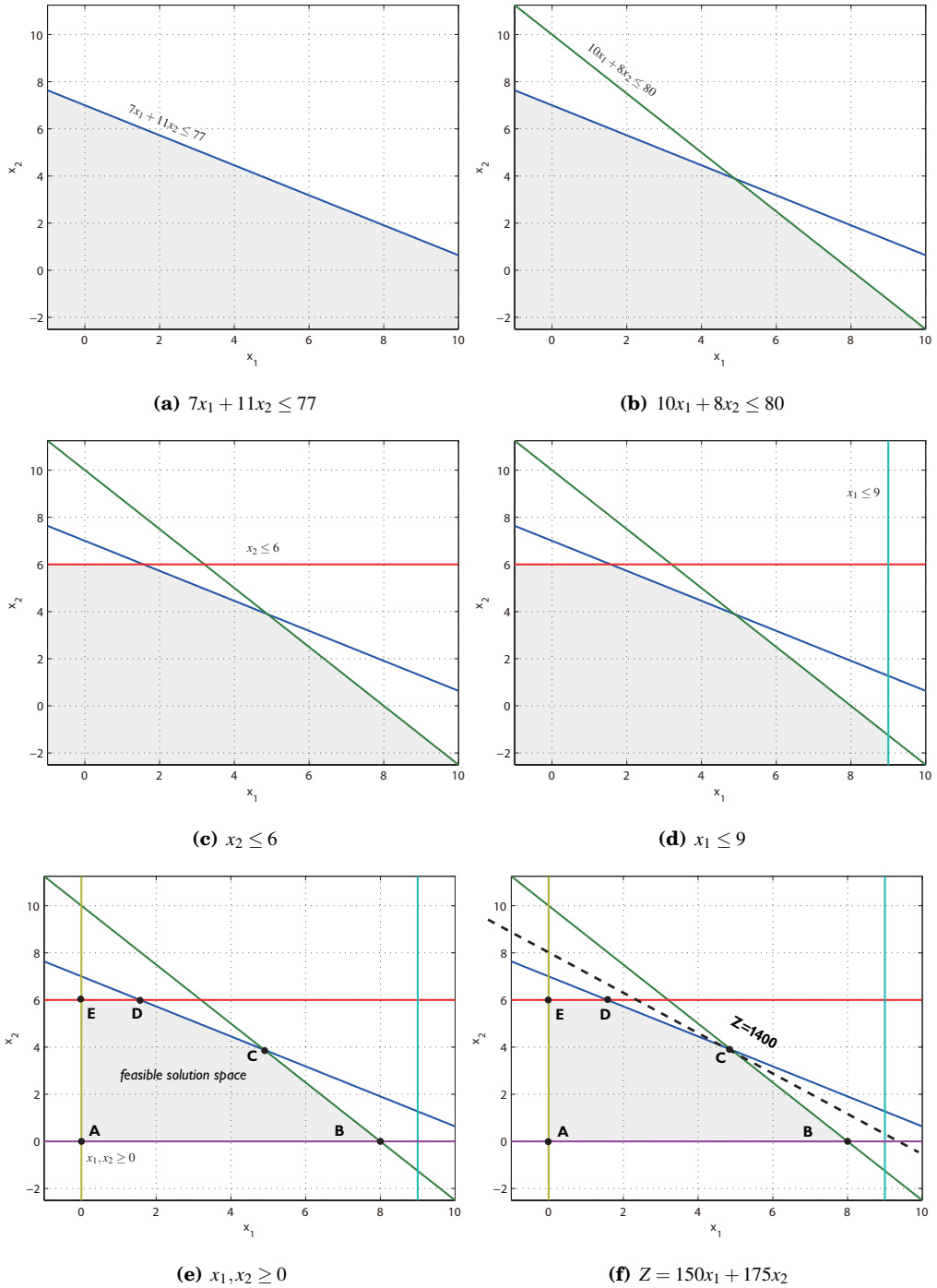
이 함계는 주당 가용한 양인 $77m^3/week$를 넘지 않아야한다. 따라서 구속조건은 다음 과 같이 표시된다.

$$7x_1 + 11x_2 \leq 77$$

남은 구속조건들도 같은 방법으로 구해지며, 이를 종합하면 선형프로그래밍식은 다음과 같다.

$$\begin{aligned}
\underset{x_1,x_2}{\text{maximize}} \quad & Z = 150x_1 + 175x_2 && \text{이윤의 극대화} \\
\text{subject to} \quad & 7x_1 + 11x_2 \leq 77 && \text{재료의 구속조건} \\
& 10x_1 + 8x_2 \leq 80 && \text{시간의 구속조건} \\
& x_1 \leq 9 && \text{"보통등급"의 보관 구속조건} \\
& x_2 \leq 6 && \text{"우수등급"의 보관 구속조건} \\
& x_1, x_2 \geq 0 && \text{양수의 구속조건}
\end{aligned}$$

도해법으로 해를 구하면



(a) $7x_1 + 11x_2 \leq 77$

(b) $10x_1 + 8x_2 \leq 80$

(c) $x_2 \leq 6$

(d) $x_1 \leq 9$

(e) $x_1, x_2 \geq 0$

(f) $Z = 150x_1 + 175x_2$

Figure 10.2: Graphical solution

10.4 패키지를 사용한 최적화 (MATLAB Optimization Toolbox Functions)

다음 테이블은 MATLAB에서 사용할 수 있는 최적화 함수들을 나타낸다. 이 함수들은 최적화 툴박스에서 사용되어지지만 근을 구하거나, 최소화, 그리고 여러개의 목적함수를 둔 최적화 그리고 회귀분석에 이용될 수 있다.

Type	Formulation		Solver
Scalar minimization	$\min_x$	$f(x)$	`fminbnd()`
	subject to	$l < x < u, (x \text{ is scalar})$	
Unconstrained minimization	$\min_x$	$f(x)$	`fminunc()`,`fminsearch()`
	subject to	$(none)$	
Linear programming	$\min_x$	$\mathbf{f}^\top \mathbf{x}$	`linprog()`
	subject to	$\mathbf{Ax} \leq \mathbf{b}, \mathbf{A}_{eq}\mathbf{x} = \mathbf{b}_{eq}, l \leq x \leq u$	
Quadratic programming	$\min_x$	$\frac{1}{2}\mathbf{x}^\top \mathbf{Hx} + \mathbf{c}^\top \mathbf{x}$	`quadprog()`
	subject to	$\mathbf{Ax} \leq \mathbf{b}, \mathbf{A}_{eq}\mathbf{x} = \mathbf{b}_{eq}, l \leq x \leq u$	
Constrained minimization	$\min_{\mathbf{x}}$	$f(\mathbf{x})$	`fmincon()`
	subject to	$c(x) \leq 0, c_{eq}(x) = 0,$	
		$\mathbf{Ax} \leq \mathbf{b}, \mathbf{A}_{eq}\mathbf{x} = \mathbf{b}_{eq}, l \leq x \leq u$	

10.4.1 Linear programming `lingprog()`

MATLAB함수의 `lingprog()`는 선형계획법의 최적해를 구해준다. `linprog()`함수가 사용하는 알고리즘은 옵션에 선언해 주지 않는 이상 Large scale linear programming과 Medium scale linear programming 으로 자동선택되어 사용되어진다. 디폴트 옵션으로 사용되어지는 Large scale LP는 Mehrotra's predictor-corrector 알고리즘[6]의 변형인 primal-dual interior-point method를 사용한 LIPSOL[7]법을 사용한다. Medium scale LP는 1947년 George Dantzig[8]가 개발한 Simplex법(Simplex method)를 사용한다.

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \mathbf{f}^\top \\ \text{subject to} \quad & \mathbf{A} \cdot \mathbf{x} \leq \mathbf{b} \\ & \mathbf{A}_{eq} \cdot \mathbf{x} = \mathbf{b}_{eq} \\ & \mathbf{L}_b \leq \mathbf{x} \leq \mathbf{U}_b \end{aligned}$$

위의 수식은 MATLAB에서 사용할 수 있는 형태로 표현하기 위해, 행렬형태를 사용한다.
`[x,fval,exitflag,output,lambda] =` **linprog**`(f,A,b,Aeq,beq,lb,ub,x0,options)` 앞서 언급한대로 MATLAB프로그램을 수행하면, Large-scale optimization 방식으로 선형프로그래밍의 최적해를 구한다.

[6]Mehrotra, S., "On the Implementation of a Primal-Dual Interior Point Method," SIAM Journal on Optimization, Vol. 2, pp 575–601, 1992.

[7]Zhang, Y., "Solving Large-Scale Linear Programs by Interior-Point Methods Under the MATLAB Environment," Department of Mathematics and Statistics, University of Maryland, Baltimore County, Baltimore, MD, Technical Report TR96-01, July, 1995.

[8]George B. Dantzig, "Programming of Interdependent Activities: II Mathematical Model" Econometrica Vol. 17, No. 3/4, pp. 200-211, 1949.

f	Linear objective function vector f
Aineq	Matrix for linear inequality constraints
bineq	Vector for linear inequality constraints
Aeq	Matrix for linear equality constraints
beq	Vector for linear equality constraints
lb	Vector of lower boundsubVector of upper bounds
x0	Initial point for x, active set algorithm only
solver	'linprog'
options	Options structure created with optimset()

Table 1: Input Arguments

```
clear all; clc; close all;

f=[-150 -175];
A=[7 11;
    10 8];
b=[77 80];
ub=[9 6];
lb=[0 0];

[x,fval,exitflag,output,lambda] = linprog(f,A,b,[],[],lb,ub);
```

Code [MATLAB] 11: 가스정제공장 수익극대화의 최적해

Part IV

Appendices

A MATLAB m-code

A.1 Figure 1.3의 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)')
```

Code [MATLAB] 12: Comparison between exact and numerical solution

A.2 무한히 미분 가능한 함수를 근사하기 위한 **Taylor** 급수전개

```
clear all; clc; close all;

xi=pi/4; % initial position of x

%% Derivative values of Cosine function
deriv=[cos(xi) -sin(xi) -cos(xi) sin(xi)]; % derivative values of f(x)
derivs=repmat(deriv,1,10); % repeatation of derivative function values

%% Taylor Series Expansion
xd=pi/3; % desired position of x
N=7;
for kk=1:N

    terms(kk)=derivs(kk)*(xd-xi)^(kk-1)/factorial(kk-1); % Taylor expansion

    order(kk)=kk-1; % order
    estival(kk)=sum(terms); % approximation value
    et(kk)=(cos(xd)-estival(kk))/cos(xd); % truncation error
end
%% Prrint Result
fid=fopen('result.txt','w');
header={'Order','Approx. Val','Et'};
fprintf(fid,'%s \t %s \t %s \n','Order','Approx. Value','Et(%)');
for kk=1:N
    fprintf(fid,'\t %d \t %.9f \t %0.2e \n',order(kk),estival(kk),et(kk));
end
fclose(fid);
%% Show Result
type result.txt
```

Code [MATLAB] 13: Approximation $\cos(\pi/3)$ from $\cos(\pi/4)$ using Taylor series expansion

A.3 예제풀이 **5.6 :** 이분법 및 가위치법

```
function y=f(x)

y=x^10-1;
```

Code [MATLAB] 14: 예제 함수

```
clear all; clc; close all;

xl=0;   % lower bound
xu=1.3;   % upper bound
ii=0;    % initial iteration value
xt=1; % true x
fl=f(xl);
while 1
    ii=ii+1;          % increment iteration value
    xr=(xl+xu)/2;     % bisection
    fr=f(xr);
    id=fl*fr;
    if id<0
        xu=xr;        % assume xu to xr
    elseif id>0
        xl=xr;        % assume xl to xr
        fl=fr;        % replace fl to fr
    end
    x(ii)=xr;               % calculated x vector
    et(ii)=abs((xt-xr)/xt); % calculated error vector
    if et(end)≤0.0001
        break;        % break loop
    end
end
```

Code [MATLAB] 15: 이분법

```
clear all; clc; close all;

xl=0;   % lower bound
xu=1.3;   % upper bound
ii=0;    % initial iteration value
xt=1; % true x
fl=f(xl);
fu=f(xu);
while 1
    ii=ii+1;          % increment iteration value
```

```
    xr=xu-(fu*(xl-xu))/(fl-fu);
    fr=f(xr);
    id=fl*fr;
    if id<0
        xu=xr;          % assume xu to xr
        fu=f(xu);       % replace fu to f(xu)
    elseif id>0
        xl=xr;          % assume xu to xl
        fl=f(xl);       % replace fl to f(xl)
    end
    x(ii)=xr;                % calculated x vector
    et(ii)=abs((xt-xr)/xt); % calculated error vector
    if et(end)≤0.0001
        break;          % break loop
    end
end
```

Code [MATLAB] 16: 가위치법

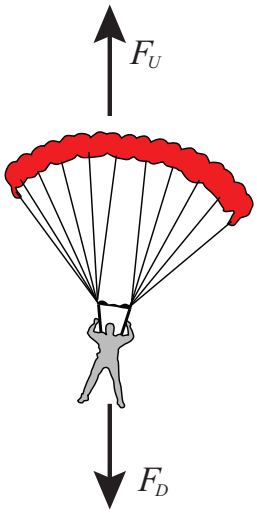
```
clear all; clc; close all;

xl=0;   % lower bound
xu=1.3;   % upper bound
ii=0;    % initial iteration value
xt=1; % true x
il=0;iu=0;
fl=f(xl);
fu=f(xu);
while 1
    ii=ii+1;          % increment iteration value
    xr=xu-(fu*(xl-xu))/(fl-fu);
    fr=f(xr);
    id=fl*fr;
    if id<0
        xu=xr;          % assume xu to xr
        fu=f(xu);       % replace fu to f(xu)
        iu=0;           % set zero iteration of upper bound
        il=il+1;        % count iteration of lower bound
        if il≥2, fl=fl/2; end % modify lower function value
    elseif id>0
        xl=xr;          % assume xu to xl
        fl=f(xl);       % replace fl to f(xl)
        il=0;           % set zero iteration of lower bound
        iu=iu+1;        % count iteration of upper bound
        if iu≥2, fu=fu/2; end % modify upper function value
    end
    x(ii)=xr;                % calculated x vector
    et(ii)=abs((xt-xr)/xt); % calculated error vector
```

```
30      if et(end)≤0.0001
            break;          % break loop
32      end
    end
```

Code [MATLAB] 17: 수정된 가위치법

B 과제풀이

B.1 과제 1



B.1.1 낙하산병의 초기속도 $v(0)$가 0이 아닌 경우, $v(0)$ 상수를 도입하여 정밀해를 구하라

$$\frac{dv}{dt} = g - \frac{c}{m}v \tag{B.1}$$

풀이 강의노트에서 해석해 풀이과정과 유사하게 변수분리방법을 사용하면

$$\frac{m}{mg-cv}dv = 1 \cdot dt \tag{B.2}$$

양변을 적분하면 식(B.2)은 시각 T 에서 다음과 같이 변형할 수 있다.

$$\int_{v(0)}^{v(T)} \frac{m}{mg-cv}dv = \int_0^T 1 \cdot dt \tag{B.3}$$

초기속도가 0이 아니기 때문에 즉, $t=0$일때, $v(0)$로 놓고, $mg-cv$를 시간의 종속변수X로 치환하면,

$$mg - cv = X \tag{B.4}$$

$$\frac{dX}{dv} = -c \tag{B.5}$$

$$dv = -\frac{1}{c}dx \tag{B.6}$$

치환변수 X의 초기값과 T에서의 값은 각각, $X(0) = mg - cv(0)$ 그리고 $X(T) = mg - cv(T)$가 된다. 다시 식(B.3)에 대입하면,

$$-\frac{m}{c}\int_{X(0)}^{X(T)} \frac{1}{X}dv = \int_0^T 1 \cdot dt \tag{B.7}$$

식(B.7)을 계산하면,

$$-\frac{m}{c}\ln X|_{X(0)}^{X(T)} = T \tag{B.8}$$

치환된 변수를 환원하면서 전개하면,

$$-\frac{m}{c}\left[\ln\{mg-cv(T)\}-\ln\{mg-cv(0)\}\right]=T \tag{B.9}$$

정리하면

$$\ln\left(\frac{mg-cv(T)}{mg-cv(0)}\right)=-\frac{c}{m}T \tag{B.10}$$

양변에 ln을 취하면,

$$e^{-(c/m)T}=\frac{mg-cv(T)}{mg-cv(0)} \tag{B.11}$$

함수$v(T)$를 구하기위해 정리하면

$$\{mg-cv(0)\}e^{-(c/m)T}=mg-cv(T) \tag{B.12}$$

$$v(T)=\frac{mg}{c}-\frac{\{mg-cv(0)\}}{c}e^{-(c/m)T} \tag{B.13}$$

결국 시각 T에서 엄밀해는 식(B.14)과 같이 계산된다.

$$v(T)=\frac{mg}{c}\left(1-e^{-(c/m)T}\right)+v(0)e^{-(c/m)T} \tag{B.14}$$

B.1.2 낙하하는 낙하산병에게 작용하는 힘을 계산하는 과정에서, 식**(B.1)**로 주어진 선형관계식 대신, 다음과 같은 **2**차식을 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.

풀이(정확해)

같은 방식으로 식(B.1)을 속도의 제곱항으로 변형하면,

$$\frac{dv}{dt}=g-\frac{c'}{m}v^2 \tag{B.15}$$

$$=\frac{mg-c'v^2}{m} \tag{B.16}$$

같은 방식으로 변수분리 방식을 사용하기 위해 식을 변형하면,

$$\frac{m}{mg-c'v^2}dv=1\cdot dt \tag{B.17}$$

양변의 시간 $t=0$에서 $t=T$까지 정적분을 취하면,

$$\int_{v(0)}^{v(T)}\frac{m}{mg-c'v^2}dv=\int_0^T dt \tag{B.18}$$

다음 식(B.19)과 같은 적분공식을 적용하기 위하여 식을 변형하고

$$\int\frac{1}{a^2-x^2}dx=\frac{1}{a}\tanh^{-1}\frac{x}{a}+C \tag{B.19}$$

적분식에서 발생하는 $\sqrt{c'}v$ 항을 X 로 치환하면

$$m\int_{v(0)}^{v(T)}\frac{1}{(\sqrt{mg})^2-(\sqrt{c'}v)^2}=\int_0^T dt \tag{B.20}$$

$$\frac{m}{\sqrt{c'}}\int_{X(0)}^{X(T)}\frac{1}{\sqrt{mg}^2-X^2}dX=\int_0^T dt \tag{B.21}$$

여기서, 치환변수 $X=\sqrt{c'}v$는, $dX/dv=\sqrt{c'}$, $dv=(1/\sqrt{c'})dX$ 로 변형되고, 적분구간은 $X(0)=v(0)=0$, $X(T)=\sqrt{c'}v(T)$ 로 변형된다. 정적분을 풀고 이항하여 $v(T)$ 로 정리하면,

$$\frac{m}{\sqrt{c'}}\left[\frac{1}{\sqrt{mg}}\tanh^{-1}\frac{X}{\sqrt{mg}}\right]_{X(0)}^{X(T)}=T \tag{B.22}$$

$$\frac{m}{\sqrt{mgc'}}\tanh^{-1}\left(\sqrt{\frac{c'}{mg}}v(T)\right)=T \tag{B.23}$$

$$\tanh^{-1}\left(\sqrt{\frac{c'}{mg}}v(T)\right)=\frac{\sqrt{gc'}m}{T} \tag{B.24}$$

$$\sqrt{\frac{c'}{mg}}v(T)=\tanh\left(\sqrt{\frac{gc'}{m}}T\right) \tag{B.25}$$

$$\therefore v(t)=\sqrt{\frac{mg}{c'}}\tanh\left(\sqrt{\frac{gc'}{m}}T\right) \tag{B.26}$$

풀이(수치해)

부록1장에 수록된 MATLAB Code에서 `v(kk)`를 `v(kk)^2`로 수정한다.

```
clear all; clc; close all;

%% Parameters
m=68.1; % kg
c=0.232; % kg/m
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=sqrt((m*g)/c)*tanh(sqrt((g*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)^2)*dt;
end

ref=ones(length(t),1)*v(end); % 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)')
```

Code [MATLAB] 18: 2번문제 수치해

B.1.3 테일러급수 (Taylor series) 를 증명하라.

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

풀이 Taylor급수를 증명하기 위해서는 미적분학 제1기본정리와 제2기본정리를 알아야한다.

Theorem B.1 (미적분학 제1기본정리). *함수 f 가 폐구간 $[a,b]$ 에서 적분가능할 때, 함수 F 를 $F(x) = \int_a^x f(t)dt$ 라 하면, 이때 다음이 성립한다. f 가 $[a,b]$ 상의 점 c 에서 연속이면, F 는 $[a,b]$ 에서 미분가능하고, $F'(c) = f(c)$ 이다. 즉,*

$$F'(x) = \frac{d}{dx}\int_a^x f(t)dt = f(x) \tag{B.27}$$

Proof $S(x) = \int_a^x f(t)dt$ 함수 $f(t)$ 에 대해 $[a,b]$ 에서 연속이고, (a,b) 에서 미분가능하므로 함수 $S(x)$ 도 $[a,b]$ 에서 연속하고 (a,b) 에서 미분가능하다. 최대최소 정리에 의해 $h>0$ 일 때 $[x,x+h]$ 에서 $f(t)$ 는 최대값 M 과 최소값 m 을 가진다. 여기서 $mh < S(x+h) - S(x) < Mh$ 이므로

$$\lim_{h\to 0} m \le \lim_{h\to 0} \frac{S(x+h)-S(x)}{h} \le \lim_{h\to 0} M$$

이다, 즉 압착정리에 의해 $\lim_{h\to 0} m = \lim_{h\to 0} M = f(x) = S'(x)$ 이 되므로, $S'(x) = f(x)$ 가 성립한다.

Theorem B.2 (미적분학 제2기본정리). *f 가 폐구간 $[a,b]$ 에서 적분가능한 함수이고, 함수 F 를 f 의 임의의 역도함수라 하면, 다음식*

$$\int_a^b f(t)dt = F(b) - F(a) \tag{B.28}$$

이 성립한다. 즉, 정적분은 임의의 역도함수의 차로 계산할 수 있다.

Proof 미적분학 제1기본정리에서 $S(x) = F(x) + C$(단, C 는 적분상수)에서, $S(x) = \int_a^x f(t)dt$ 로 정의되었으므로,

$$S(a) = F(a) + C = 0 \tag{B.29}$$

위 식에서 $C = -F(a)$, $S(x) = \int_a^x f(t)dt = F(x) - F(a)$ 이다. 따라서,

$$\int_a^b f(x)dx = F(b) - F(a) \tag{B.30}$$

이 성립한다.

Taylor series 미적분학 제2기본정리로부터,

$$\int_a^x f'(t)dt = f(x) - f(a) \tag{B.31}$$

위의 식(B.31)을 부분적분을 하기 위해 다음 식(B.32)로 변형하자

$$\int_a^x f'(t)dt = \int_a^x (-1)\left\{-f'(t)\right\}dt \tag{B.32}$$

$f(t)$는 무한번 미분가능하면 부분적분을 무한번 수행할 수 있으므로, -1을 적분할 함수, $-f'(t)$를 미분할 함수로 두고, 부분적분을 수행하면[9],

$$\int_a^x (-1)\left\{-f'(t)\right\}dt = \left[-(x-t)f'(t) - \frac{(x-t)^2}{2}f''(t) - \frac{(x-t)^3}{6}f'''(t) - \cdots\right]_a^x \tag{B.33}$$

이제 식(B.33)을 풀게되면,

$$\begin{aligned}\left[-(x-t)f'(t) - \frac{(x-t)^2}{2}f''(t) - \frac{(x-a)^3}{6}f'''(t) - \cdots\right] &= (x-a)f'(a) + \frac{(x-a)^2}{2!}f'''(a) + \frac{(x-a)^3}{3!}f''''(a) + \cdots \\ &= f(x) - f(a)\end{aligned} \tag{B.34}$$

그러므로, 무한번 미분가능한 함수 $f(x)$에 대하여

$$f(x) = f(a) + (x-a)f'(a) + \frac{(x-a)^2}{2!}f''(a) + \frac{(x-a)^3}{3!}f'''(a) + \cdots \tag{B.35}$$

$$= \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \tag{B.36}$$

---

[9]단, 여기서 -1을 계속 적분할 때 -1의 한 부정적분을 구해서 써주면 되는데, 적분변수 t와 관계없는 값x를 상수취급하여 $x-t$를 부정적분으로서 구했다.

C 중간고사 예제

C.1 CHAPTER 1. 수학적 모델링과 공학 문제의 해결

C.1.1 연습문제 1.18

속도는 다음 식과 같이 거리 $x(m)$에 대한 시간의 변화량과 같다.

$$\frac{dx}{dt} = v(t) \tag{C.1}$$

(a) 다음식을 대입하여 시간의 함수로 거리를 표현할 수 있도록 해석해를 구하라.

$$v(t) = \frac{gm}{c}\left(1 - e^{-(c/m)t}\right)$$

(b) 낙하산병문제와 같은 매개변수를 사용하고, Euler법을 사용하여 수치적으로 적분함으로써 처음 10초 동안의 속도와 낙하한 거리를 시간의 함수로 구하라.

C.2 CHAPTER 4. 절단오차와 Taylor 급수

C.2.1 연습문제 4.5

다음과 같은 함수에서 $f(3)$을 계산하기 위해, $x = 1$을 기준점으로 0차부터 3차까지의 Taylor급수전개를 사용하여 $f(3)$의 값을 구하라. 참백분율 상대오차 ε_t를 구하라

$$f(x) = 25x^3 - 6x^2 + 7x - 88$$

C.2.2 연습문제 4.10

낙하산병의 속도는 다음과 같이 주어졌다.

$$v(t) = \frac{gm}{c}\left(1 - e^{-(c/m)t}\right)$$

(a) $g = 9.8$,$m = 50$,$c = 12.5 \pm 1.5$일 때, $t = 6$에서 1차 오차해석으로 속도의 추정오차값을 구하도록 하라.

(b) $g = 9.8$,$t = 6$,$c = 12.5 \pm 1.5$,$m = 50 \pm 2$로 주어졌을때 1차 오차해석으로 속도의 추정오차값을 구하도록 하라.

C.3 CHAPTER 5. 구간법

C.3.1 연습문제 5.2

다음 식의 실근을 구하라

$$f(x) = 4x^3 - 6x^2 + 7x - 2.3$$

(a) 이분법을 사용하여 근을 구하라, $x_l = 0$과 $x_u = 1$을 초기구간으로 가정하고, 근사오차 ε_a가 10% 이하로 떨어질 때까지 반복하라

(b) 가위치법을 사용하여 근을 구하라, 조건은 (a)와 동일

C.3.2 연습문제 **5.7**

다음 식의 실근을 구하라

$$f(x) = (0.8 - 0.3x)/x$$

(a) 해석적인 방법으로 구하라

(b) 1과 3을 초기구간으로 하고 가위치법을 세 번 반복해서 실근을 구하라. 각 반복계산 후 근사오차 ε_a와 참오차 ε_t를 구하라.

C.3.3 연습문제 **5.20**

Figure C.1(a)는 선형적으로 증가하는 분포하중을 받고 있는 보를 나타낸 것이다. 이에 따른 탄성곡선의 방정식은 다음 식(C.2)과 같다.(결과 Figure C.1(b) 참조)

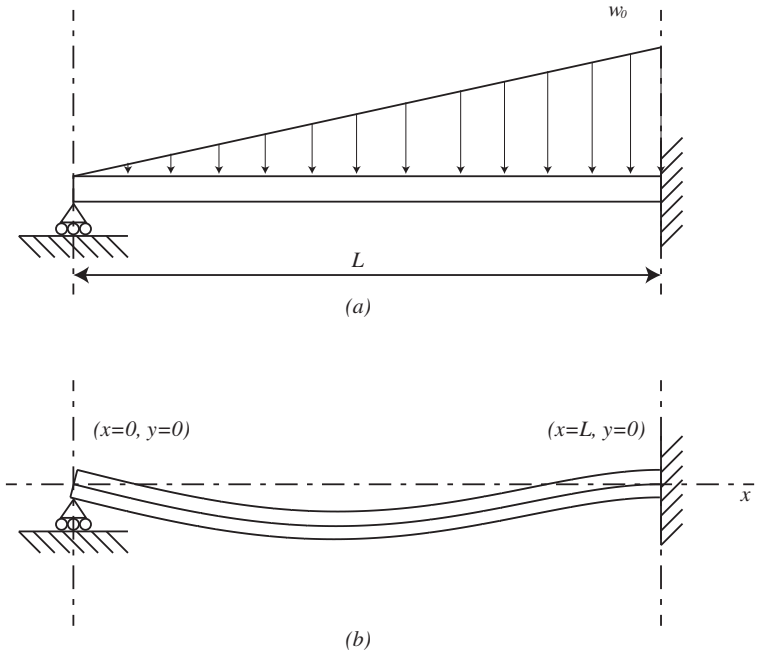


Figure C.1: 선형적으로 증가하는 분포하중을 받고 있는 보

$$y = \frac{w_0}{120EIL}(-x^5 + 2L^2x^3 - L^4x) \tag{C.2}$$

이분법을 사용하여 최대 처짐이 발생하는 지점을 구하라(즉, $dy/dx = 0$인 x의 값). 이 값을 식(C.2)에 대입하여 최대 처짐값을 정하라. 매개변수들의 값은 다음과 같다. $L = 600cm$, $E = 50{,}000kN/cm^2$, $I = 30{,}000cm^4$, $w_0 = 2.5kN/cm$.

C.4 CHAPTER 6. 개구간법

C.4.1 연습문제 **6.2**

다음 함수의 가장 큰 근을 구하라.

$$f(x) = 2x^3 - 11.7x^2 + 17.7x - 5$$

(a) Newton-Raphson법을 사용하라. 초기가정으로 $x_0 = 3$을 사용하고 세번 반복하라.

(b) 할선법을 사용하라. 초기가정으로 $x_0 = 3$, $x_1 = 4$를 사용하고 세번 반복하라.

(c) 수정된 할선법을 사용하라. 초기가정으로 $x_0 = 3$과 $\delta = 0.01$을 사용하고 세번 반복하라.

(Newton-Raphson법, 할선법 그리고 수정된 할선법 알고리즘은 각각 아래를 참조.)

C.4.2 연습문제 6.11

Newton-Raphson법을 사용하여 다음 식의 근을 구하라

$$f(x) = e^{-0.5x}(4-x) - 2$$

초기가정값으로 (a) 2, (b) 6 그리고 (c) 8 을 사용하라. (Newton-Raphson법 알고리즘은 아래를 참조.)

Algorithm C.1 Newton-Raphson Method

Let $f : \mathbb{R} \to \mathbb{R}$ be a differentiable function. The following algorithm computes an approximate solution x_r to the equation $f(x) = 0$.

Choose an initial guess x_0.

for $i = 0, 1, 2, \cdots$ **do**
  if $f(x_i)$ is sufficiently small **then**
    $x_r = x_i$
    return x_r
  end if
  $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$
  if $|x_{i+1} - x_i|$ is sufficiently small **then**
    $x_r = x_{i+1}$
    return x_r
  end if
end for

Algorithm C.2 Secant Method

Let $f : \mathbb{R} \to \mathbb{R}$ be a differentiable function. The following algorithm computes an approximate solution x_r to the equation $f(x) = 0$.

Choose an initial guess x_0 and x_1.

for $i = 0, 1, 2, \cdots$ **do**
  if $f(x_i)$ is sufficiently small **then**
    $x_r = x_i$
    return x_r
  end if
  $x_{i+2} = x_{i+1} - \frac{f(x_{i+1})(x_i - x_{i+1})}{f(x_i) - f(x_{i+1})}$
  if $|x_{i+2} - x_{i+1}|$ is sufficiently small **then**
    $x_r = x_{i+2}$
    return x_r
  end if
end for

Algorithm C.3 Modified Secant Method

Let $f : \mathbb{R} \to \mathbb{R}$ be a differentiable function. The following algorithm computes an approximate solution x_r to the equation $f(x) = 0$.

Choose an initial guess x_0 and δ.

for $i = 0, 1, 2, \cdots$ **do**
  if $f(x_i)$ is sufficiently small **then**
    $x_r = x_i$
    return x_r
  end if
  $x_{i+1} = x_i - \frac{\delta x_i f(x_i)}{f(x_i + \delta x_i) - f(x_i)}$
  if $|x_{i+1} - x_i|$ is sufficiently small **then**
    $x_r = x_{i+1}$
    return x_r
  end if
end for

C.5 문제 답안

C.5.1 연습문제 1.18(a)

$$\frac{dx}{dt} = \frac{gm}{c}\left(1 - e^{-1(c/m)t}\right) \tag{C.3}$$

$$\int_{x(0)}^{x(T)} dx = \frac{gm}{c}\int_0^T \left(1 - e^{-(c/m)t}\right) dt \tag{C.4}$$

$$x(T) - x(0) = \frac{gm}{c}\left[t + \frac{m}{c}e^{-(c/m)t}\right]_0^T \tag{C.5}$$

$$= \frac{gm}{c}\left\{T + \frac{m}{c}\left(e^{-(c/m)T} - 1\right)\right\} \tag{C.6}$$

C.5.2 연습문제 1.18(b)

$g = 9.8$, $m = 68.1$, $c = 12.6$을 사용하여 Euler식을 적용하면

$$x_{i+1} = x_i + (t_{i+1} - t_i)\frac{gm}{c}\left(1 - e^{-(c/m)t}\right) \tag{C.7}$$

Time(second)	Distance(m)
0	0
1	0
2	8.9468
3	25.3292
4	47.8912
5	75.5889
6	107.555
7	143.0683
8	181.5297
9	222.4413
10	265.3892

C.5.3 연습문제 4.5

$$f(x) = 25x^3 - 6x^2 + 7x - 88$$

참값은 $f(3) = 554$

Order	Equation	Result	ε_t
0	$f(3) \cong f(1)$	$f(3) \cong -62$	111%
1	$f(3) \cong f(1) + f'(1)(3-1)$	$f(3) \cong 78$	86.9%
2	$f(3) \cong f(1) + f'(1)(3-1) + \frac{1}{2!}f''(1)(3-1)^2$	$f(3) \cong 354$	36.10%

C.5.4 연습문제 **4.10(a)**

$g = 9.8, m = 50, c = 12.5 \pm 1.5$일 때, $t = 6$에서 속도의 1차오차해석에 의한 추정오차값은

$$\Delta v(t) \cong \left|\frac{\partial v(t)}{\partial c}\right| \Delta c \tag{C.8}$$

$$= \left|\frac{gm}{c^2}\left(e^{-(c/m)t} - 1\right) + \frac{gt}{c}\left(e^{-(c/m)t}\right)\right| \Delta c \tag{C.9}$$

$$= \left|-\frac{gm}{c} + \frac{g}{c}(\frac{m}{c} + t)e^{-(c/m)t}\right| \Delta c \tag{C.10}$$

$$= |-1.2807| \Delta c \tag{C.11}$$

$$= 1.2807 \cdot 1.5 \tag{C.12}$$

$$\therefore v(t) = 35.5133 \pm 1.9212 \tag{C.13}$$

C.5.5 연습문제 **4.10(b)**

$g = 9.8, t = 6, c = 12.5 \pm 1.5, m = 50 \pm 2$일 때, $t = 6$에서 속도의 1차오차해석에 의한 추정오차값은

$$\Delta v(t) \cong \left|\frac{\partial v(t)}{\partial c}\right| \Delta c + \left|\frac{\partial v(t)}{\partial m}\right| \Delta m \tag{C.14}$$

$$= \left|-\frac{gm}{c} + \frac{g}{c}(\frac{m}{c} + t)e^{-(c/m)t}\right| \Delta c + \left|\frac{g}{c} - \left(\frac{g}{c} + \frac{gt}{m}\right)e^{-(c/m)t}\right| \Delta m \tag{C.15}$$

$$= |-1.2807| \Delta c + |0.2370| \Delta m \tag{C.16}$$

$$= 1.2807 \cdot 1.5 + 0.2370 \cdot 2 \tag{C.17}$$

$$\therefore v(t) = 35.5133 \pm 2.3951 \tag{C.18}$$

C.5.6 연습문제 **5.2(a)**

이분법을 사용하여 근을 구하라, $x_l = 0$과 $x_u = 1$을 초기구간으로 가정하고, 근사오차 ε_a가 10% 이하로 떨어질 때까지 반복하라

Iter.	x_l	x_u	x_r	$f(x_r)$	ε_a
1	0	1	0.5	0.2	-
2	0	0.5	0.25	-0.8625	100%
3	0.25	0.5	0.375	-0.30781	33.3333%
4	0.375	0.5	0.4375	-0.050977	14.2857%
5	0.4375	0.5	0.46875	0.074878	6.6667%

C.5.7 연습문제 **5.2(b)**

가위치법을 사용하여 근을 구하라, $x_l = 0$과 $x_u = 1$을 초기구간으로 가정하고, 근사오차 ε_a가 10% 이하로 떨어질 때까지 반복하라

Iter.	x_l	x_u	x_r	$f(x_r)$	ε_a
1	0	1	0.46	0.039744	-
2	0	0.46	0.45219	0.0083076	1.728%

C.5.8 연습문제 **5.7(b)**

Iter.	x_l	x_u	x_r	$f(x_r)$	ε_a	ε_t
1	1	3	2.875	-0.021739	-	0.078125%
2	1	2.875	2.7969	-0.013966	2.7933%	0.048828%
3	1	2.7969	2.748	-0.0088842	1.7768%	0.030518%

C.5.9 연습문제 **5.20**

최대처짐이 발생하는 지점은 $dy/dx = 0$인 지점이므로 주어진 식의 1차도함수를 구한다.

$$\frac{dy}{dx} = \frac{w_0}{120EIL}\left(-5x^4 + 6L^2x^2 - L^4\right) \tag{C.19}$$

즉, 처짐에 대한 함수의 근을 이분법으로 찾는다.

$$f(x) = \frac{w_0}{120EIL}\left(-5x^4 + 6L^2x^2 - L^4\right) \tag{C.20}$$

Iter.	x_l	x_u	x_r	$\frac{dy}{dx}$	y	ε_t
1	0	600	300	0.0005625	-0.50625	-
2	0	300	150	-0.0019336	-0.39551	100%
3	150	300	225	-0.00076538	-0.4985	33.3333%
4	225	300	262.5	-0.00010423	-0.51489	14.2857%
5	262.5	300	281.25	0.00023088	-0.5137	6.6667%
6	262.5	281.25	271.875	6.3442×10^{-5}	-0.51508	3.4483%
7	262.5	271.875	267.1875	-2.0405×10^{-5}	-0.51518	1.7544%
8	267.1875	271.875	269.5313	2.1521×10^{-5}	-0.51518	0.86957%

C.5.10 연습문제 **6.2(a)**

Newton-Raphson법을 사용하라. 초기가정으로 $x_0 = 3$을 사용하고 세번 반복하라.

Iter.	x_i	$f(x_i)$	$f'(x_i)$	ε_a
0	3	-3.2	1.5	41.5584%
1	5.1333	48.0901	55.6867	20.2256%
2	4.2698	12.9562	27.1724	12.5712%

C.5.11 연습문제 **6.2(b)**

할선법을 사용하라. 초기가정으로 $x_0 = 3$을 사용하고 세번 반복하라.

Iter.	x_i	x_{i+1}	$f(x_i)$	$f(x_{i+1})$	ε_a
0	3	4	-3.2	6.6	20.2454%
1	4	3.3265	6.6	-1.9689	4.445%
2	3.3265	3.4813	-1.9689	-0.79592	2.9279%

C.5.12 연습문제 **6.2(c)**

수정된 할선법을 사용하라. 초기가정으로 $x_0 = 3$와 $\delta = 0.01$을 사용하고 세번 반복하라.

Iter.	x_i	$f(x_i)$	$f(x_i + \delta x_i)$	ε_a
0	3	-3.2	-3.1493	38.6828%
1	4.8926	35.7632	38.0973	18.0945%
2	4.1429	9.7305	10.7367	10.7055%

C.5.13 연습문제 **6.11**

Newton-Raphson법을 사용하여 다음 식의 근을 구하라

$$f(x) = e^{-0.5x}(4 - x) - 2$$

초기가정값으로 (a) 2, (b) 6 그리고 (c) 8 을 사용하라. (Newton-Raphson법 알고리즘은 아래를 참조.)

Iter.	x_i	$f(x_i)$	$f'(x_i)$	ε_a
0	2	-1.2642	-0.73576	609.9294%
1	0.28172	1.2297	-2.4835	63.7376%
2	0.77689	0.18563	-1.7709	11.8884%
3	0.88171	0.0065795	-1.6468	0.45109%

Table 2: 초기가정값 $x_0 = 2$

Iter.	x_i	$f(x_i)$	$f'(x_i)$	ε_a
0	6	-2.0996	0	NaN%

Table 3: 초기가정값 $x_0 = 6$

Iter.	x_i	$f(x_i)$	$f'(x_i)$	ε_a
0	8	-2.0733	0.018316	93.3991%
1	121.1963	-2	2.7731e-25	100%
2	7212131452880262800000000	-2	0	NaN%

Table 4: 초기가정값 $x_0 = 8$

C.5.14 알고리즘 테이블

Algorithm C.4 이분법(Bisection Method)

Let $f : \mathbb{R} \to \mathbb{R}$ be a differentiable function. The following algorithm computes an approximate solution x_r to the equation $f(x) = 0$.

Choose lower x_l and upper x_u guesses for the root such that the function changes sign over the interval. This can be checked by ensuring that $f(x_l)f(x_u) < 0$.

while $f(x_r)$ is not sufficiently small **do**
  $x_r = \frac{x_l + x_u}{2}$
  if $f(x_l)f(x_r) < 0$ **then**
    $x_u = x_r$
  else
    $x_l = x_r$
  end if
  if $\varepsilon_a = \left| \frac{x_{new} - x_{old}}{x_{new}} \right|$ is sufficiently small **then**
    $x_r = x_{new}$
    return x_r
  end if
end while

Algorithm C.5 가위치법(False Position Method)

Let $f : \mathbb{R} \to \mathbb{R}$ be a differentiable function. The following algorithm computes an approximate solution x_r to the equation $f(x) = 0$.

Choose lower x_l and upper x_u guesses for the root such that the function changes sign over the interval. This can be checked by ensuring that $f(x_l)f(x_u) < 0$.

while $f(x_r)$ is not sufficiently small **do**
  $x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)}$
  if $f(x_l)f(x_r) < 0$ **then**
    $x_u = x_r$
  else
    $x_l = x_r$
  end if
  if $\varepsilon_a = \left| \frac{x_{new} - x_{old}}{x_{new}} \right|$ is sufficiently small **then**
    $x_r = x_{new}$
    return x_r
  end if
end while

Algorithm C.6 Newton-Raphson Method

Let $f : \mathbb{R} \to \mathbb{R}$ be a differentiable function. The following algorithm computes an approximate solution x_r to the equation $f(x) = 0$.

Choose an initial guess x_0.

for $i = 0, 1, 2, \cdots$ **do**
  if $f(x_i)$ is sufficiently small **then**
    $x_r = x_i$
    return x_r
  end if
  $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$
  if $|x_{i+1} - x_i|$ is sufficiently small **then**
    $x_r = x_{i+1}$
    return x_r
  end if
end for

Algorithm C.7 Secant Method

Let $f : \mathbb{R} \to \mathbb{R}$ be a differentiable function. The following algorithm computes an approximate solution x_r to the equation $f(x) = 0$.

Choose an initial guess x_0 and x_1.

for $i = 0, 1, 2, \cdots$ **do**
  if $f(x_i)$ is sufficiently small **then**
    $x_r = x_i$
    return x_r
  end if
  $x_{i+2} = x_{i+1} - \frac{f(x_{i+1})(x_i - x_{i+1})}{f(x_i) - f(x_{i+1})}$
  if $|x_{i+2} - x_{i+1}|$ is sufficiently small **then**
    $x_r = x_{i+2}$
    return x_r
  end if
end for

Algorithm C.8 Modified Secant Method

Let $f : \mathbb{R} \to \mathbb{R}$ be a differentiable function. The following algorithm computes an approximate solution x_r to the equation $f(x) = 0$.

Choose an initial guess x_0 and δ.

for $i = 0, 1, 2, \cdots$ **do**
  if $f(x_i)$ is sufficiently small **then**
    $x_r = x_i$
    return x_r
  end if
  $x_{i+1} = x_i - \frac{\delta x_i f(x_i)}{f(x_i + \delta x_i) - f(x_i)}$
  if $|x_{i+1} - x_i|$ is sufficiently small **then**
    $x_r = x_{i+1}$
    return x_r
  end if
end for