수치해석

7. 2 비선형 방정식의 해

7.2.0 서론
7.2.1 고정점 반복법
7.2.2 Newton-Raphson 법
7.2.3 이분법
7.2.4 가위치법

7.2.0 서론

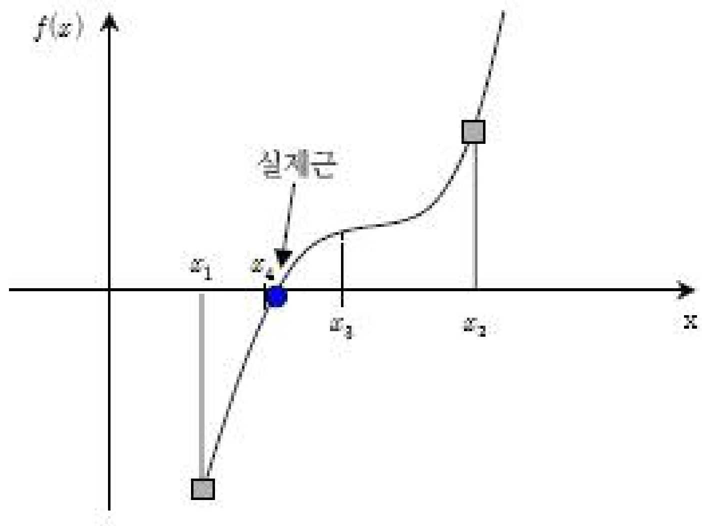
- 방정식의 근을 찾는 방법
  - 근의 공식을 이용한 해석적인 방법
  - 근사해
    - 그래프 이용법
      - 함수식을 그래프로 그려 x축과 만나는 점, 즉 f(x)=0이 되는 근 x를 찾음
      - 정확도가 떨어짐
    - 시행 착오법(trial and error)
      - 많은 임의의 수를 대입하여 그 중 조건에 맞는 값을 근으로 취함
      - 비효율적이고 계산 시간이 오래 걸림
    - 구간법 및 개방법
      - 개방법 : 근을 포함하는 구간의 양 끝이 아닌 한 개 이상의 초기값을 이용
        - 고정점 반복법, Newton-Raphson법, 할선법
      - 구간법 : 근을 포함하는 구간의 양 끝을 초기값으로 이용
        - 이분법, 가위치법

- 반복 종료를 위한 임계값 결정

$$E_r = \frac{x^{now} - x^{old}}{x^{now}} \times 100\%$$

x^{now} : 현재의 반복 계산에서 구한 근

x^{old} : 이전의 반복 계산에서 구했던 근



7.2.1 고정점 반복법 (Fixed-point iteration)

- 방정식의 형태를 변형하여 근을 구하는 방법

$$x = g(x)$$

- 이전 계산 $g(x)$에서 얻은 x의 값을 이용하여 새로운 x 값을 예측

$$x_{i+1} = g(x_i)$$

- 예제) 고정점 반복법을 이용하여 $f(x) = x - e^{-x}$ 의 근을 구하라.

풀이. 주어진 함수를 변형하면 $x_{i+1} = e^{-x}$ 이다.
따라서 초기값 $x_0 = 0$ 을 사용하여 2회 반복 계산하면 다음과 같다.

$$x_1 = e^0 = 1$$

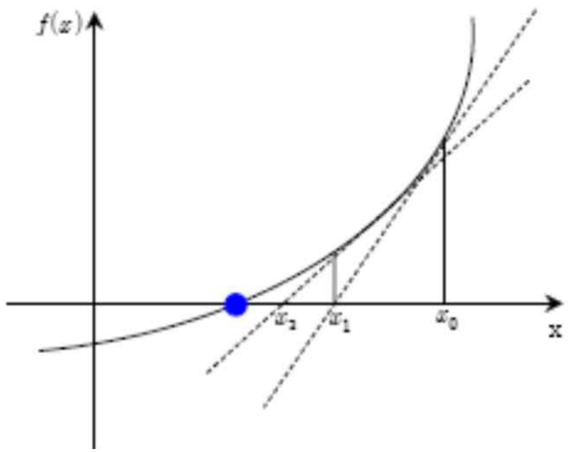
$$x_2 = e^1 = 0.3679$$

상대오차가 0.0001이 될 때까지 반복한 결과 실제값인 0.56714329에 수렴함을 알 수 있음

i(반복 횟수)	x_i	상대 오차(%)
0	0.0000	
1	1.0000	100
2	0.3679	171.828
3	0.6922	46.856
4	0.5005	38.309
5	0.6062	17.447
6	0.5454	11.157
7	0.5796	5.903
8	0.5601	3.481
9	0.5711	1.931
10	0.5649	1.098

7.2.2 Newton-Raphson법

- 근을 구하기 위한 가장 효율적인 방법
  - 초기값 x_0 이 주어졌을 때 점 $(x_0, f(x_0))$ 에 접하는 접선을 구하고, 이 접선이 x 축과 만나는 점이 새로운 근 x 가 됨



| Newton-Raphson법의 원리

미분은 특정 지점에서 나타나는 접선의 기울기를 뜻한다

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

- 예제) Newton-Raphson법을 사용해서 $f(x) = \log(x+5.0) + x$ 의 근을 구하라. 초기값은 $x_0 = 7.0$ 으로 한다.

풀이. $f(x) = \log(x+5.0) + x$ 이고 $f'(x) = \dfrac{1}{x+5.0} + 1$ 이므로

$$x_1 = 7 - \frac{8.0792}{1.0833} = -0.4580$$ 이 되며,

$$x_1 = -0.4580 - \frac{0.1992}{1.2202} = -0.6213$$ 이 된다.

상대오차가 0.0001이 될 때까지 반복한 결과 고정점 반복법보다 훨씬 더 빨리 근에 수렴함을 알 수 있음

i(반복 횟수)	x_i	상대 오차(%)
0	7.0000	
1	−0.4577	1629.4000
2	−0.6213	26.3270
3	−0.6376	2.5640
4	−0.6393	0.2684
5	−0.6395	0.0283

MATLAB Code Example

```
% Experiment 4
% Newton Raphson Method
        clear all
        close all
        clc
% Change here for different functions
    f=@(x) cos(x)-3*x+1
%this is the derivative of the above function
    df=@(x) -sin(x)-3
% Change lower limit 'a' and upper limit 'b'
    a=0; b=1;
    x=a;
for i=1:1:100
    x1=x-(f(x)/df(x));
    x=x1;
end
sol=x;
fprintf('Approximate Root is %.15f',sol)

a=0;b=1;
x=a;
er(5)=0;
for i=1:1:5
    x1=x-(f(x)/df(x));
    x=x1;
    er(i)=x1-sol;
end
plot(er)
xlabel('Number of iterations')
ylabel('Error')
title('Error Vs. Number of iterations')


f =

    @(x)cos(x)-3*x+1


df =

    @(x)-sin(x)-3

Approximate Root is 0.607101648103123
```

Python Code Example

```
def derivative(f, x, h):
      return (f(x+h) - f(x-h)) / (2.0*h)  # might want to return a small non-zero if ==0

def quadratic(x):
    return 2*x*x-5*x+1     # just a function to show it works

def solve(f, x0, h):
    lastX = x0
    nextX = lastX + 10* h  # "different than lastX so loop starts OK
    while (abs(lastX - nextX) > h):  # this is how you terminate the loop - note use of abs(
        newY = f(nextX)                     # just for debug... see what happens
        print "f(", nextX, ") = ", newY     # print out progress... again just debug
        lastX = nextX
        nextX = lastX - newY / derivative(f, lastX, h)  # update estimate using N-R
    return nextX

xFound = solve(quadratic, 5, 0.01)    # call the solver
print "solution: x = ", xFound        # print the result
```

output:

```
f( 5.1 ) =  27.52
f( 3.31298701299 ) =  6.38683083151
f( 2.53900845771 ) =  1.19808560807
f( 2.30664271935 ) =  0.107987672721
f( 2.28109300639 ) =  0.00130557566462
solution: x =  2.28077645501
```

7.2.3 이분법 (Bisection method)

- 중간값 정리(intermedia value theorem)을 토대
  - 함수 $f(x)$가 구간 (x_1, x_2) 에서 연속이고, $f(x_1)$과 $f(x_2)$의 부호가 서로 반대이면 x_1과 x_2 사이에 적어도 한 개의 근이 존재한다는 원리
  - 이때 두 점 사이를 다시 이등분한 점 x_3을 근으로 가정

$$x_3 = \frac{x_1 + x_2}{2}$$

  - if $f(x_1)f(x_3) < 0$ 이면 근은 x_1과 x_3 사이에 존재

    else 근은 x_2과 x_3 사이에 존재
  - 이를 다시 이등분 반복하여 $x_4, \ldots, x_n$ 을 구해 가다가 임계값에 도달시 이때의 x_m을 근으로 추정

- 예제)구간 (0,1)에서 $f(x) = x - e^{-x}$ 의 실근을 이분법을 이용하여 구하라.

풀이. 초기값을 $x_1 = 0$ 과 $x_2 = 1$ 로 가정시,

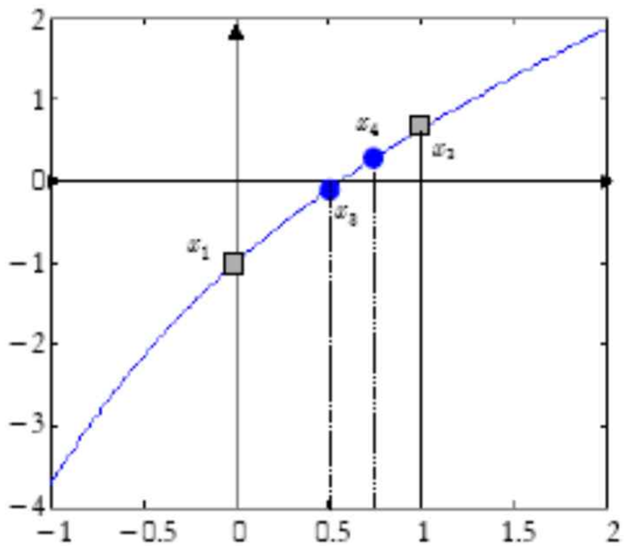
$f(0)f(1) < 0$ 이므로 근은 0과 1 사이에 존재한다.

따라서

$$x_3 = \frac{x_1 + x_2}{2} = \frac{0+1}{2} = 0.5$$

또한 $f(0)f(0.5) > 0, f(0.5)f(1) < 0$ 이므로 근이 0.5와 1 사이에 존재함을 알 수 있다.

반복의 종료를 위한 임계치를 0.0001로 했을 때 반복 결과 10회의 반복으로 실제값인 0.5671에 가까워짐을 알 수 있음



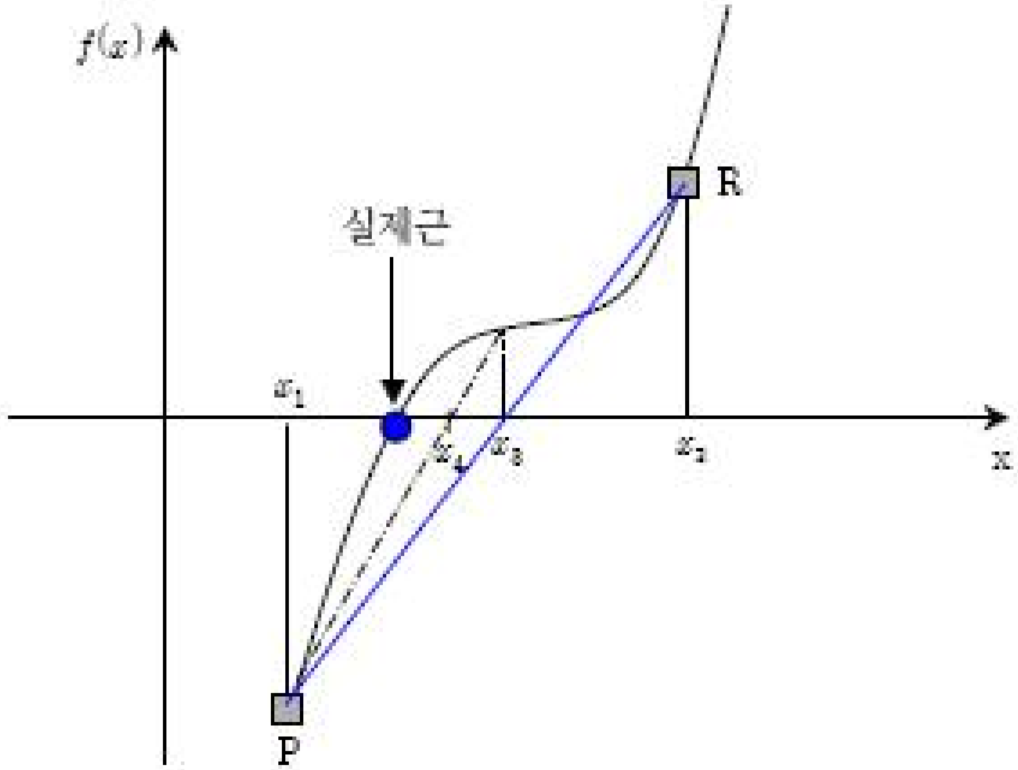
i(반복 횟수)	x_i	상대 오차(%)
0	0.0000	
1	0.5000	100
2	0.7500	33.3333
3	0.6250	20
4	0.5625	11.1111
5	0.5938	5.2632
6	0.5781	2.7027
7	0.5703	1.3699
8	0.5664	0.6897
9	0.5685	0.3436
10	0.5674	0.1721

7.2.4 가위치법 (false position method)

- 이분법의 수렴 속도를 개선한 것
- 이분법과 같이 구간을 절반으로 나누지 않고 $f(x_1)$과 $f(x_2)$을 직선으로 연결시켜 이 직선과 x축이 만나는 교점 x_3을 새로운 근으로 추정

$$x_3 = x_2 - f(x_2)\frac{x_2 - x_1}{f(x_2) - f(x_1)}$$

- if $f(x_1)f(x_3) < 0$ 이면 근은 x_1 와 x_3 사이에 존재
  else 근은 x_2 와 x_3사이에 존재
- 다시 두 점을 지나는 직선을 그어 x축과 만나는 점을 다음 반복의 시작점으로 둠



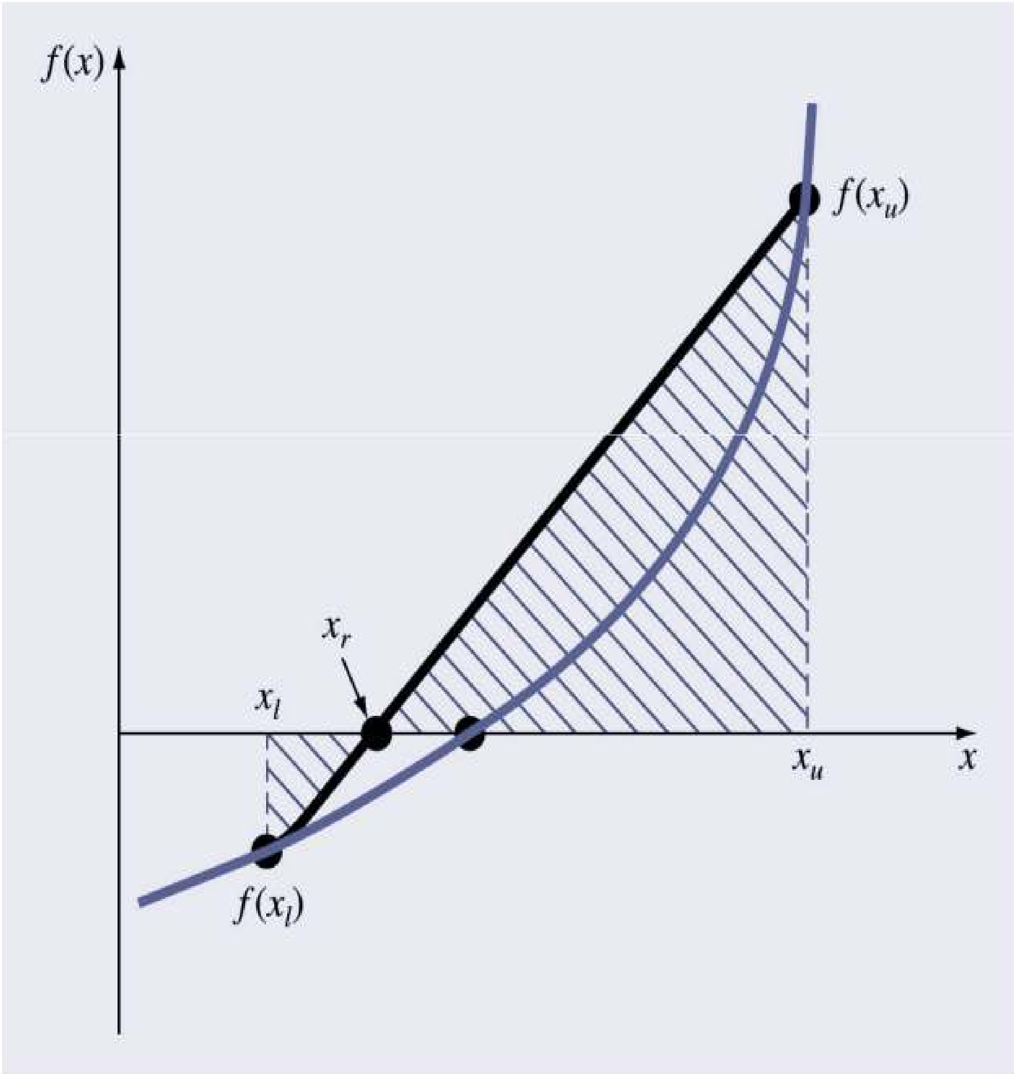
가위치법의 원리

- 이분법과 매우 유사하다.

- $f(x_l)$과 $f(x_u)$를 연결하는 직선과 x축의 교점을 찾아 개선된 추정값으로 이용.

- 가위치법 공식

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



- 예제)구간 (0,1)에서 $f(x) = x - e^{-x}$의 실근을 가위치법을 이용하여 구하라.

풀이. 초기값을 $x_1 = 0$ 과 $x_2 = 1$ 로 가정시,

$f(0)f(1) < 0$ 이므로 근은 0과 1 사이에 존재한다.

따라서

$$x_3 = 1 - 0.6321\left(\frac{1}{1.6321}\right) = 0.6127$$

또한 $f(0)f(0.6127) > 0, f(0.6127)f(1) < 0$ 이므로

근이 0.6127과 1 사이에 존재함을 알 수 있다.

0.6127과 1을 초기값으로 가정하고 임계치를 0.0001로 했을 때 4회 반복 결과

i(반복 횟수)	x_i	상대 오차(%)
0	0.0000	
1	0.6127	100
2	0.5722	7.0814
3	0.5677	0.7888
4	0.5672	0.0877

7.3 연립 방정식의 해

7.3.1 연립 방정식의 행렬 표현
7.3.2 소거법

7.3.1 연립 방정식의 행렬 표현

- n개의 변수 $x_1, x_2, \cdots, x_n$을 갖는 m개의 연립 방정식의 형태

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
&\vdots \\
a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m
\end{aligned}
$$

- 행렬 방정식의 형태

$$
\begin{bmatrix} a_{11} & a_{12} & a_{1n} \\ a_{21} & a_{22} & a_{2n} \\ & & \\ a_{m1} & a_{m2} & a_{mn} \end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}
=
\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}
$$

또는 $Ax = b$

$\therefore$ A : 계수행렬
x : 미지 변수 벡터
b : 상수항 벡터

7.3.2 소거법

(1) Gauss 소거법

- 등가 변환에 의해 상삼각(upper triangle) 연립 방정식으로 만든 후 역진 대입법으로 미지수를 구하는 방법

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

- 예. 다음의 1차 연립 방정식의 해를 Guass 소거법을 이용하여 구하라.

$$3x_1 - x_2 + 2x_3 = 12$$
$$x_1 + 2x_2 + 3x_3 = 11$$
$$2x_1 - 2x_2 - x_3 = 2$$

풀이. 첫번째 식에 −1을 곱하고 두번째 식에 3을 곱하여 첫번째 식을 두번째 식에 더하면 다음과 같다.

$$3x_1 - x_2 + 2x_3 = 12$$
$$7x_2 + 7x_3 = 21$$
$$2x_1 - 2x_2 - x_3 = 2$$

이제 위의 첫번째 식에 −2를 곱하고 세번째 식에 3을 곱하여 두 식을 더하면 다음과 같다.

$$\begin{aligned} 3x_1 - x_2 + 2x_3 &= 12 \\ 7x_2 + 7x_3 &= 21 \\ -4x_2 - 7x_3 &= -18 \end{aligned}$$

두번째 식과 세번째 식에서 x_1이 소거되며, x_2를 소거하기 위해 두번째 식에 4를 곱하고, 세번째 식에 7을 곱하여 더하면 다음과 같다.

$$\begin{aligned} 3x_1 - x_2 + 2x_3 &= 12 \\ 7x_2 + 7x_3 &= 21 \\ -21x_3 &= -42 \end{aligned}$$

결국 세번째 식으로부터 $x_3 = 2$

이를 두번째 식에 대입하면 $x_2 = 1$

x_3, x_2 를 첫번째 식에 대입하여 $x_1 = 3$ 을 구하게 된다.

- 별해
  - 확대 행렬(augment matrix) $A \vdots b$에 행 연산을 수행하면서 계수 행렬을 상삼각 행렬로 만듬

$$A \vdots B = \begin{bmatrix} 3 & -1 & 2 & \vdots & 12 \\ 1 & 2 & 3 & \vdots & 12 \\ 2 & -2 & -1 & \vdots & 2 \end{bmatrix} \begin{matrix} \leftarrow R1 \\ \leftarrow R2 \\ \leftarrow R3 \end{matrix}$$

$$\Big\downarrow \begin{matrix} R2 \leftarrow 3R2 - R1 \\ R3 \leftarrow 3R3 - 2R1 \end{matrix}$$

$$\begin{bmatrix} 3 & -1 & 2 & 12 \\ 0 & 7 & 7 & 21 \\ 0 & -4 & -7 & -18 \end{bmatrix}$$

$$\Big\downarrow R3 \leftarrow 7R3 + 4R2$$

$$\begin{bmatrix} 3 & -1 & 2 & 12 \\ 0 & 7 & 7 & 21 \\ 0 & 0 & -21 & -42 \end{bmatrix}$$

후진 대입법을 사용하면

$$x_3 = 2, x_2 = 1, x_1 = 3$$

- 예제1) Gauss 소거법을 이용하여 연립 방정식의 해를 구하라.

$$
\begin{aligned}
x_1 - 4x_2 - 2x_3 &= 21 \\
2x_1 + x_2 + 2x_3 &= 3 \\
3x_1 + 2x_2 - x_3 &= -2
\end{aligned}
$$

풀이. 확대 행렬을 만들고 행 연산을 하면서 상삼각 행렬을 만듬

$$\begin{bmatrix} 1 & -4 & -2 & 21 \\ 2 & 1 & 2 & 3 \\ 3 & 2 & -1 & -2 \end{bmatrix} \quad \begin{matrix} \leftarrow R1 \\ \leftarrow R2 \\ \leftarrow R3 \end{matrix}$$

$$\Big\downarrow \quad \begin{matrix} R2 \leftarrow R2 - 2R1 \\ R3 \leftarrow R3 - 3R1 \end{matrix}$$

$$\begin{bmatrix} 1 & -4 & -2 & 21 \\ 0 & 9 & 6 & -39 \\ 0 & 14 & 5 & -65 \end{bmatrix}$$

$$\Big\downarrow \quad R3 \leftarrow R3 - \frac{14}{9}R1$$

$$\begin{bmatrix} 1 & -4 & -2 & 21 \\ 0 & 9 & 6 & -39 \\ 0 & 0 & -\frac{13}{3} & -\frac{13}{3} \end{bmatrix} \quad \begin{matrix} \leftarrow R1 \\ \leftarrow R2 \\ \leftarrow R3 \end{matrix}$$

후진 대입법을 사용하면 $x_3 = 1, x_2 = -5, x_1 = 3$ 을 얻게 됨

(2) Gauss-Jordan 소거법

- 대각 원소의 윗부분을 0으로 하는 대각 행렬을 만들어 계산의 효율성을 높이는 방법
- Gauss 소거법보다 연산 횟수가 많으므로 전파오차 발생 가능

$$\begin{bmatrix} 3 & -1 & 2 & 12 \\ 0 & 7 & 7 & 21 \\ 0 & 0 & -21 & -42 \end{bmatrix} \begin{array}{l} \leftarrow R1 \\ \leftarrow R2 \\ \leftarrow R3 \end{array}$$

$$\Big\downarrow \; \begin{array}{l} R1 \leftarrow 7R1 + R2 \\ R2 \leftarrow 3R2 + R3 \end{array}$$

$$\begin{bmatrix} 21 & 0 & 21 & 105 \\ 0 & 21 & 0 & 21 \\ 0 & 0 & -21 & -42 \end{bmatrix}$$

$$\Big\downarrow \; R1 \leftarrow R1 + R3$$

$$\begin{bmatrix} 21 & 0 & 0 & 63 \\ 0 & 21 & 0 & 21 \\ 0 & 0 & -21 & -42 \end{bmatrix}$$

$$\therefore x_1 = 3, x_2 = 1, x_3 = 2$$

- 예제 2) 예제 1)의 연립 방정식의 해를 Guass-Jordan 소거법을 이용하여 구하라.

풀이. 상위 대각 성분을 0으로 만들기 위해 다음과 같은 행 연산을 수행한다.

$$\begin{bmatrix} 1 & -4 & -2 & 21 \\ 0 & 9 & 6 & -39 \\ 0 & 0 & -\frac{13}{3} & -\frac{13}{3} \end{bmatrix} \begin{matrix} \leftarrow R1 \\ \leftarrow R2 \\ \leftarrow R3 \end{matrix}$$

$$\Big\downarrow \quad \begin{matrix} R1 \leftarrow 9R1 + 4R2 \\ R2 \leftarrow \frac{13}{3}R2 + 6R3 \end{matrix}$$

$$\begin{bmatrix} 9 & 0 & 6 & 33 \\ 0 & 39 & 0 & -195 \\ 0 & 0 & -\frac{13}{3} & -\frac{13}{3} \end{bmatrix} \begin{matrix} \leftarrow R1 \\ \leftarrow R2 \\ \leftarrow R3 \end{matrix}$$

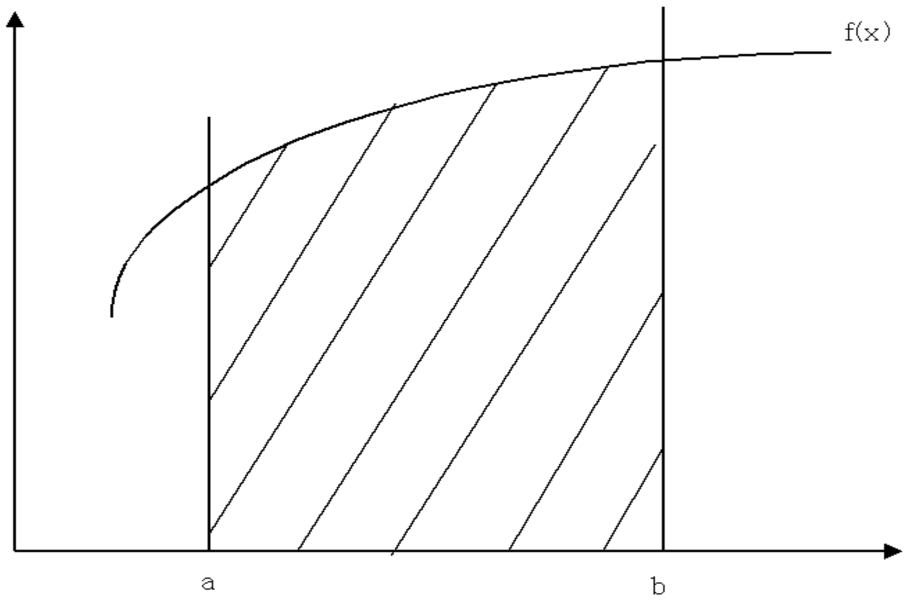
$$\Big\downarrow \quad R1 \leftarrow \frac{13}{3}R1 + 6R3$$

$$\begin{bmatrix} 39 & 0 & 6 & 117 \\ 0 & 39 & 0 & -195 \\ 0 & 0 & -\frac{13}{3} & -\frac{13}{3} \end{bmatrix} \begin{matrix} \leftarrow R1 \\ \leftarrow R2 \\ \leftarrow R3 \end{matrix}$$

$$\therefore x_1 = 3, x_2 = -5, x_3 = 1$$

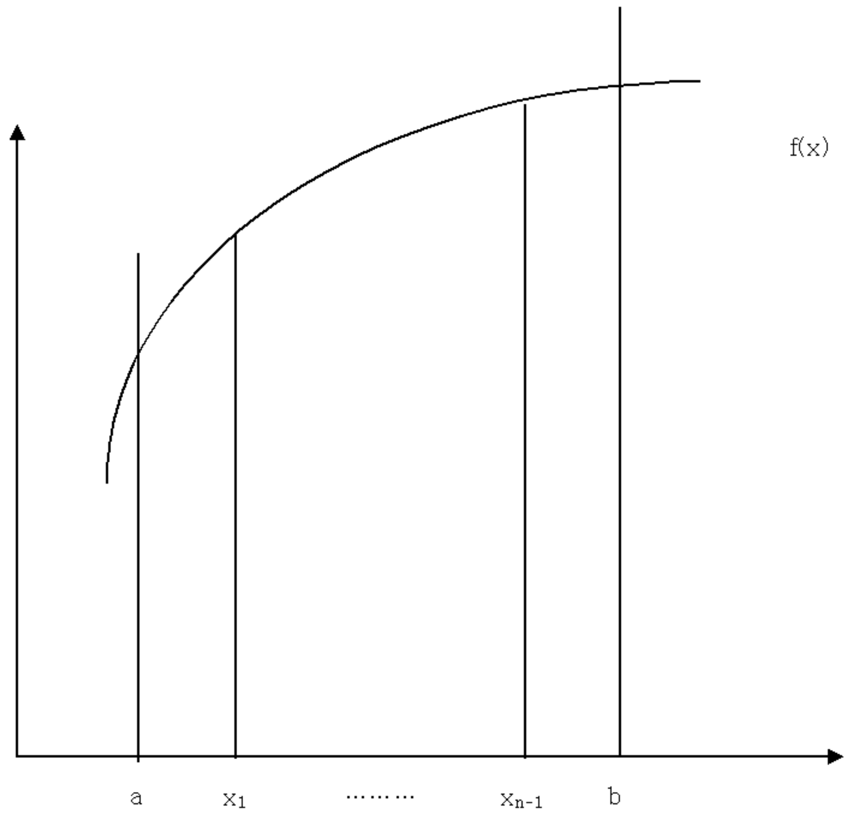
7.4 적분법

$\int_a^b f(x)dx$: 함수 $f(x)$ 에 대하여 a부터 b 사이의 면적



(1) 사다리꼴 공식(Trapezoidal rule)

➪ a부터 b 사이를 n 개로 분할



Let $h = \dfrac{b-a}{n}$

$a \sim x_1,\ x_1 \sim x_2,\ \ldots\ldots\ldots\ldots,\ x_{n-2} \sim x_{n-1},\ x_{n-1} \sim b$: n 개의 구간

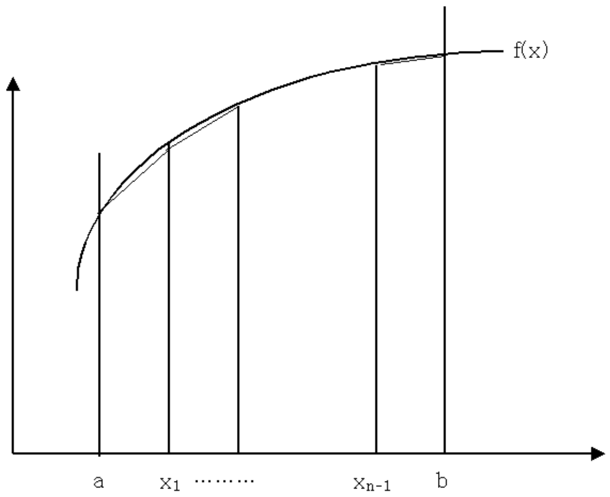
⇨ $a - x_1 - f(x_1) - f(a)$

$x_1 - x_2 - f(x_2) - f(x_1)$: n 개의 사다리꼴

$\cdots\cdots\cdots$

$x_{n-1} - b - f(b) - f(x_{n-1})$

⇨ 이 n 개의 사다리꼴 면적의 합 = $f(x)$의 적분값 (즉, 면적)



⇨ 첫 사다리꼴 $a - x_1 - f(x_1) - f(a)$ 의 면적 :

$$\frac{f(a) + f(x_1)}{2} \times h$$

두 번째 사다리꼴 $x_1 - x_2 - f(x_2) - f(x_1)$ 의 면적 :

$$\frac{f(x_1) + f(x_2)}{2} \times h$$

....

마지막 사다리꼴 $x_{n-1} - b - f(b) - f(x_{n-1})$ 의 면적 :

$$\frac{f(x_{n-1}) + f(b)}{2} \times h$$

이 된다. 이를 다 더하면 ⟹ 함수의 면적

$$S=\frac{f(a)+f(x_1)}{2}\times h+\frac{f(x_1)+f(x_2)}{2}\times h+\cdots\cdots\frac{f(x_{n-1})+f(b)}{2}\times h$$

$$=\frac{h}{2}\{f(a)+f(x_1)+f(x_1)+f(x_2)+\cdots\cdots\cdots+f(x_{n-2})+f(x_{n-1})+f(x_{n-1})+f(b)\}$$

$$=\frac{\mathrm{h}}{2}\{f(a)+2f(x_1)+2f(x_2)+\cdots\cdots\cdots+2f(x_{n-2})+2f(x_{n-1})+f(b)\}$$

$$=\frac{\mathrm{h}}{2}\left\{f(a)+2\sum_{i=1}^{n-1}f(x_i)+f(b)\right\}$$

여기서

$$
\begin{aligned}
x_1 &= a + h \\
x_2 &= a + 2h \\
x_3 &= a + 3h \\
&\vdots \\
&\vdots \\
x_{n-2} &= a + (n-2)h \\
x_{n-1} &= a + (n-1)h
\end{aligned}
$$

$$
\therefore \; S = \frac{h}{2}\left\{ f(a) + 2\sum_{i=1}^{n-1} f(a+ih) + f(b) \right\}
$$

[프로그램]

$S = \int_0^1 e^{-x^2} dx$ 을 구하라.

```
C    Program for Integration of F(X) = EXP(-X**2)

     N = 100
     A = 0.0
     B = 1.0

     H = (B - A) / FLOAT(N)

     AREA = F(A) + F(B)

     DO  10  I = 1, N-1
     AREA = AREA + 2. * F(A + I * H)
10   CONTINUE

     AREA = AREA * H / 2.0

     WRITE(*, 30) AREA
30   FORMAT(//10X, 'INTEGRATION OF F(X) = ', F10.6)
     STOP
     END

     FUNCTION   F(X)
     F = EXP(- X**2)
     RETURN
     END
```

[C 프로그램]

```c
#include <stdio.h>
#include <math.h>

// 테스트함수01, y = x^3 - 5x^2 + 2x + 1
double function01(double x)
{
    return (x*x*x - 5*x*x + 2*x + 1);
}

// 함수 func에 대해 구간 a에서 b까지 n개의 사다리꼴로 나누어 면적을 구한다.
// n이 커질수록 정확도가 증가한다.
double integral_trapezoidal(double (*func)(double), double a, double b, int n)
{
    int i;
    double result, top_edge, bot_edge;
    double height = (b-a) / n;
    double height_half = height / 2;

    top_edge = func(a);
    bot_edge = func(a+height);
    result = (top_edge+bot_edge) * height_half;

    for ( i=2; i<=n; i++ )
    {
        top_edge = bot_edge;
        bot_edge = func( a + i*height );
        result += (top_edge+bot_edge) * height_half;
    }

    return result;
}

int main()
{
    double a, b;
    int n;

    printf("입력[a, b, n]: ");
    scanf("%lf %lf %d", &a, &b, &n);

    printf("구간 [%.2lf, %.2lf] n=%d\n 적분결과: %.6lf\n",
           a, b, n, integral_trapezoidal(function01, a, b, n));

    return 0;
}
```