## 이분법

```
5xe^x = 1의 근사해를 이용해 구해보자.
(a : 0  b : 5)

x 1 = 2.500000000000000000000000
x 2 = 1.250000000000000000000000
x 3 = 0.625000000000000000000000
x 4 = 0.312500000000000000000000
x 5 = 0.156250000000000000000000

x 6 = 0.234375000000000000000000
x 7 = 0.195312500000000000000000
x 8 = 0.175781250000000000000000
x 9 = 0.166015625000000000000000
x 10 = 0.170898437500000000000000
```

```
x 51 = 0.168915973499108584832129 1
x 52 = 0.168915973499109695055153 7
x 53 = 0.168915973499109139943641 4
x 54 = 0.168915973499109417499397 5
x 55 = 0.168915973499109556277275 6

x 56 = 0.168915973499109473010548 8
x 57 = 0.168915973499109528521700 0
x 58 = 0.168915973499109528521700 0

57번의 연산결과 근사해는 0.168916 입니다.
```

**[ 초기 값 설정 ]**

a : 0

b : 5

c : (a+b)/2

**[ 종료조건 ]**

컴퓨터의 연산결과가
이전 연산과 같을 시

**[ 결과 ]**

연산 결과 : 0.168916

실제 값 : 0.16892

**연산 차수 : 57 회**

# 오차조정법

```
==========================================================

  5xe^x = 1의 근사해를 오차조정법을 이용해 구해보자.
  (a : 0  b : 5)

==========================================================

  x 1 = 0.00134758939981709344353 71
  x 2 = 0.00268572825375440905198 99
  x 3 = 0.00401446340856183650491 75
  x 4 = 0.00533384176193689527739 62
  x 5 = 0.00664391025778036416193 92

  x 6 = 0.00794471588150026490626 75
  x 7 = 0.00923630565536455450104 11
  x 8 = 0.01051872663390288412110 78
  x 9 = 0.01179202589935773697049 64
  x 10 = 0.01305625055718524080150 54
```

```
  x 3566 = 0.16891597349910786318716 30
  x 3567 = 0.16891597349910789094273 87
  x 3568 = 0.16891597349910791869831 43
  x 3569 = 0.16891597349910794645388 99
  x 3570 = 0.16891597349910797420946 55

  x 3571 = 0.16891597349910800196504 11
  x 3572 = 0.16891597349910802972061 67
  x 3573 = 0.16891597349910805747619 24
  x 3574 = 0.16891597349910805747619 24

==========================================================

  3573번의 연산결과 근사해는 0.168916 입니다.

==========================================================
```

**[ 초기 값 설정 ]**

a : 0

b : 5

**[ 종료조건 ]**

컴퓨터의 연산결과가

이전 연산과 같을 시

**[ 결과 ]**

연산 결과 : 0.168916

실제 값 : 0.16892

**연산 차수 : 3573 회**

## 뉴턴방법


**[ 초기 값 설정 ]**

x : 5

**[ 종료조건 ]**

컴퓨터의 연산결과가
이전 연산과 같을 시

**[ 결과 ]**

연산 결과 : 0.168916

실제 값 : 0.16892

**연산 차수 : 12 회**

## 할선법

```
5xe^x = 1의 근사해를 할선법을 이용해 구해보자.
(x1 : 4 x0 : 5)

x 1 = 3.5833420498558847 8110
x 2 = 2.9832625758632826 9511
x 3 = 2.4802024276876104 5582
x 4 = 1.9750884621953990 1945
x 5 = 1.5143813041977560 7294

x 6 = 1.0953199355450542 4626
x 7 = 0.7383584134130722 8980
x 8 = 0.4608569537840689 9003
x 9 = 0.2800291346055086 0059
x 10 = 0.1944054874676213 1849

x 11 = 0.1713929202320885 7474
x 12 = 0.1689738323108392 8923
x 13 = 0.1689161063090294 2347
x 14 = 0.1689159735062383 8159
x 15 = 0.1689159734991095 2852

x 16 = 0.1689159734991095 8403
x 17 = 0.1689159734991095 5628
x 18 = 0.1689159734991095 5628

17번의 연산결과 근사해는 0.168916 입니다.
```

**[ 초기 값 설정 ]**

x0 : 5

x1 : 4

**[ 종료조건 ]**

컴퓨터의 연산결과가
이전 연산과 같을 시

**[ 결과 ]**

연산 결과 : 0.168916

실제 값 : 0.16892

**연산 차수 : 17 회**

## 분석

연산 결과를 도식화 하여 나타내었다.

| 사용 기법 | 초기 값 | 오차 | 연산횟수 |
|---|---|---|---|
| 이분법 | a : 0, b : 5 | 0 | 57 |
| 오차조정법 | a : 0, b : 5 | 0 | 3573 |
| 뉴턴 방법 | x : 5 | 0 | 12 |
| 할선법 | x0 : 5, x1 : 4 | 0 | 17 |

**초기 값** : 각 연산의 식에 따라 달라질 수밖에 없었고, 특성에 맞추어 비슷한 값으로 설정을 하였다.
**연산의 종료 시점** : '컴퓨터가 이전연산과 같은 값을 도출하였을 때'로 설정하였다.

**연산횟수에 따른 분석**

연산횟수가 연산식의 효율성을 좌우한다는 가정 하에 성능 순으로 나열한다면,
뉴턴방법 > 할선법 > 이분법 > 오차조정법
으로 표현할 수 있다.

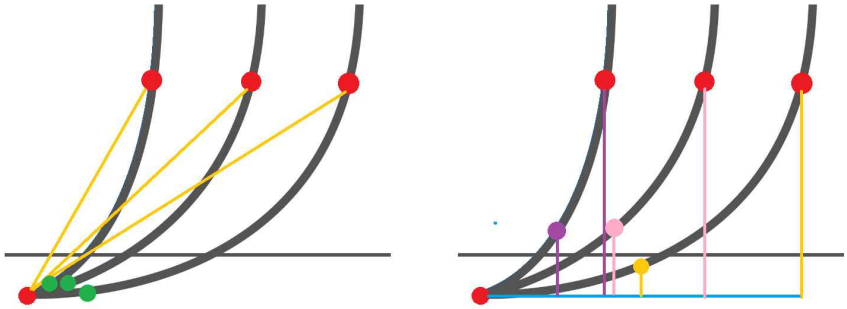
**결과의 오류 발생점**

1) c언어의 double은 16자리까지 정확한 값을 도출해내기 때문에, 만약 연산결과가 정확히 되었을 경우 더 빨리 연산이 끝날 수 있었지만 정확하지 않은 값으로 인해 더 많은 연산을 하였을 수 있다.
2) 초기값 설정 시 임의로 넣은 값이 특정식에는 유리한 결과를 도출해냈을 수 있다. 만약 이분법의 경우, 첫 구간의 a,b 값이 바로 c를 도출해낸 경우 가장 빠른 방법으로 도출 될 수 있다.

**오차조정법의 높은 연산횟수**

오차조정법의 경우 다른 연산에 비해 월등히 많은 연산수를 보인다. 이것은 위의 1)의 경우에 해당한다고 생각 할 수도 있지만, 여러 번 같은 연산을 해 본 결과 모두 같은 횟수를 도출해냈다.
이에 입력 범위를 달리 하여 연산을 해 본 결과,
0.0~0.5(3573번) / 0.0~0.4(1355번) / 0.0~0.3(509번) / 0.0~0.2(186번) / 0.0~0.1(60번)
으로 오차조정법 자체가 높은 연산결과를 도출해낸다는 것을 알 수 있었다.

이의 원인으로는 값의 전개 속도를 들 수 있다.



위는 이분법과 오차조정법의 전개를 그림으로 표현한 것이다.
이분법은 현재 거리를 기준으로 절반을 이동하는 것이 보장되어있다. 따라서 초기 a,b의 x좌표의 거리가 10이라고 하면 절반인 5의 범위를 줄이는 것이 확정적이다.
하지만 오차조정법의 경우에는 줄어드는 범위가 모호하다. 위의 그림에서 다양한 곡률에 따른 전개를 비교해 보면 값의 줄이는 범위가 확연히 다름이 느껴진다.

**뉴턴방법과 할선법**

뉴턴방법과 할선법의 차이는 기준점을 놓을 때 '그래프'를 보는가와 '기존의 데이터'를 보는가이다. 사람을 판단할 때조차 그 사람 자체를 내가 보는 것과 그 사람의 주변 평판을 보는 것의 차이는 존재한다.
**뉴턴 방법**은 그래프를 기준으로 본다. 그래프가 현재 주어진 기준점의 접점의 기울기가 낮을수록 기준점이 많이 이동하고 높을수록 적게 이동한다.
**할선법**은 기준점이 x축에 맞닿는 점이 그래프를 타고 올라간다. 이 방식은 기울기의 크기를 기존보다 크게 표현한다. 기울기가 낮을수록 기준점의 이동이 빠르게 되어 값을 도출해낼 수 있는데 이런 이점이 줄어드는 것이다.
이런 이유로 뉴턴 방법이 할선법보다 조금 더 빠르게 결과를 도출했다고 생각한다.