수치해석 NA-introduction

Numerical Analysis

- Formulation: 기본 법칙을 세심하게
- Solution: 기본적인Computer method를 다룬다.
- 계산은 비교적 쉽고, System sensitivity와 Error를 다룸

Representation of numbers

- Finite number of bits

$$s * M * B^{e-E}$$

Double precision real numbers (64 bits)

$$(-1)^s * 2^{c-1023} * (1 + f)$$

- c: 11 bit (0 ~ 2^11 - 1)
- f: 52 bit

Accuracy

- Finite number of bits 때문에 표현할 수 있는 실수가 제한된다.
- **Double precision real numbers**
  - **Underflow (**c=0): 일반적으로 0으로 세팅해버림
  - **Overflow (**c=$2^n - 1$): inf, NaN 등으로 인해 다음 계산에 영향이 간다. 다음 Step에서 계산이 멈춘다.
- **Smallest number** (c = 1)
  - $2^{-1022} \cdot (1 + 0) = 0.2225 * 10^{-307}$
- **Largest number** (c = $2^n - 2$)

- $2^{1023} \cdot (1 + (1 - 2^{-52})) = 0.17977 * 10^{309}$

Machine Accuracy

$$epsilon = b^{1-m}$$

- m: Mantissa의 비트 수
- Machine Accuracy: **1 + epsilon ≠ 1이 되는 최초의 epsilon 값**
- Epsilon값을 점점 키워가는 과정을 생각해보면 된다.

Accuracy vs Precision

- Accuracy는 목표 (중앙)에 얼마나 가까운지
- Precision은 샘플 값들이 서로 얼마나 모여있는지

Roundoff error

Machine Accuracy 때문에 발생하는 Error

- 실수 x를 floating point에 따른 fl(x)로 표현할 때의 표현 가능한 수 fl(x)와 실수 x의 차이
- Error: fl(x) - x

실수 x를 Floating point에 따라 변형할 때 방법이 두 가지 있다.

Δx = fl(x) - x라고 하자.

1. **Chopping: 뒷자리 수를 그냥 버린다.**
    a. 그냥 버리기 때문에 Roundoff error = Δ x이다.

2. **Symmertirc rounding: 가까운 값으로 반올림한다.**
    a. 가까운 값으로 반올림하기 때문에 Roundoff error = Δ x / 2

Minimizing roundoff errors

1. Overflow와 Underflow를 피하기 위해 **Intermediate value를 around 1로 유지**
    a. $x = 1.f * 2^e$에서 e가 [-1022, 1023] 사이이기 때문에 중간 값은 대충 $1.f * 2^0 = 1$이다.
2. 계산 오차가 누적되는 것을 방지하기 위해 **연산 횟수 자체를 줄인다.**
3. **Subtractive cancellation**을 피한다.
    a. 비슷한 숫자끼리 뺄셈을 하게 되면, 상위 비트의 수가 비슷해 상위 비트의 유효자릿수가 사라진다.
    b. Small number에서 시작한다.
4. **Double precision**을 사용하여 표현할 수 있는 숫자의 범위를 늘린다.

Truncation error

계산한 추정값과 실제 정답값의 차이

- 추정값을 계산할 때, Formular를 approximation하기 때문에 발생한다.
- **NA의 목표는 Truncation error**를 줄이는 것이**다!**

Approximation

Talyor series는 approximation의 일종이다.

- $f(x) = P_n(x) + R_n(x)$
    - $P_n(x) = f(x_0) + f'(x_0)(x - x_0) + f''(x_0)(x - x_0)^2/2! + ... + f^(n)(x_0)(x - x_0)^n/n!$
    - $R_n(x) = f^{(n+1)}(epsilon)(x - x_0)^{n+1}/(n+1)!$
    - $R_n(x)$는 Approximation으로 인한 Error이다.
- Taylor series는 **이미 알고 있는 값 x0을 통해 함수 f(x)를 Approximation하는 방법**이다.

Taylor series를 응용하여 우리가 모르는 다음 함수값 $f(x_{i+1})$을 $f(x_i)$로 근사할 수 있다.

- $f(x_{i+1}) = P_n(x_i) + R_n(x_i)$

Error

Absolute error: (true value - approximation)

Relative error: (true value - approximation) / true value

Approximate relative error: (가장 최근에 예측한 값 - 기존 예측값) / 가장 최근에 예측한 값

Iterative algorithm의 Stopping condition

- $e_a < e_s$
- e_s: Desired relative error

Data error

$$|f(x) - f(x^*)|$$

- x^*은 추정치이다.
- 테일러 급수에 따라 x*을 이용하여 f(x)를 근사해보면 아래와 같다.
  - $f(x) = f(x^*) + f'(x^*)(x - x^*) + \ldots$
  - $f(x) - f(x^*) = f'(x^*)(x - x^*)$
  - $|f(x) - f(x^*)| = f'(x^*)|(x - x^*)|$
- 따라서 Data error는 $f'(x^*)|(x - x^*)|$ 이다.

Data relative error

$$e_f = \frac{|f'(x)|(x - x^*)}{f(x^*)}$$

- Relative error 정의에 따르면 분모에는 f(x)가 와야 하지만, 추론 및 근사 과정에선 대부분 실제 함수 값 f(x)를 모르기 때문에 f(x*)을 사용한다.

Relative error of x

$$e_x = \frac{x - x^*}{x^*}$$

Condition number: 입력의 오차가 출력의 오차에 영향을 미치는 정도

$$\frac{e_f}{e_x} = \frac{x^* f'(x^*)}{f(x^*)}$$

- **If condition number < 1 →** $e_f < e_x$
    - Error reduction
    - 입력의 변화가 출력에 큰 변화를 가져오지 않는다.
    - Data error가 줄어든다.
- **If condition number >> 1 →** $e_f > e_x$
    - ill-conditioned
    - 입력의 변화가 출력에 큰 변화를 가져온다.

Forward: x를 이용하여 y=f(x)값을 찾는 문제

- 이 경우 Condition number가 $\frac{e_f}{e_x}$이다.

Inverse: f(x) 값을 이용하여 x = $f^{-1}(x)$를 찾는 문제

- Root finding이 예시이다.
- 이 경우 Condition number가 $\frac{e_x}{e_f}$이다.

Error propagation

- Data error $f'(x^*)|(x - x^*)|$ 은 **입력 오차 (x-x*)에 비례**한다.
- **입력 오차가 출력 오차로 전달된다는 것**이 Error propagation이다.

Total error: Roundoff error + Truncation error

- h: x - x*라고 하자. (step size)
- Step size가 커질수록 Truncation error는 커진다.
- Step size가 작을수록 계산 결과는 더 정밀한 값이 필요하기 때문에 Roundoff error는 커진다.

결국 Step size에 따라 Roundoff error와 Truncation error 중 하나는 포기해야 한다.

- Trade-off
- 둘을 동시에 줄이는 Step size는 없다.