Homework2

과목명.	수치해석
담　당.	박 종 일 교수님
제출일.	2021년 9월 17일
공과대학	컴퓨터공학과
학　번.	2016025205
이　름.	김 범 진

HANYANG UNIVERSITY

Summary Numerical Recipes in C Chapter1

Program Organization and Control Structures

대부분의 경우 프로그램의 커뮤니케이션 대상은 컴퓨터이지만 재사용가능하고 유지 관리 가능하게 프로그램을 작성하려면 프로그램의 커뮤니케이션 대상은 인간이다.
프로그램이 무슨 일을 하는지 아는 것뿐만 아니라 어떻게 동작하는지 아는 것도 중요하다. 왜냐하면 동작을 이해해야 프로그램을 수정하고 특정 응용프로그램에 맞게 수정할 수 있다.
좋은 프로그래밍 연습을 하는 방법 중 한가지 중요한 것은 프로그래밍이 계층구조를 갖고있는 것을 아는 것이다.
첫번째로는 constants, identifiers, operands, operators등을 이해하는 것이 중요하고 두번째로는 control structure를 이해해야한다. 그리고 마지막으로는 function, module 등을 이해하는 것이 중요하다.

Some C Conventions for Scientific Computing

C언어는 원래 과학계산이 아닌 시스템 프로그래밍 작업을 위해 만들어졌다. 따라서 높은 수준의 과학계산을 위해 C를 사용하는 것은 일반적으로 매우 느리게 개발되었다. 하지만 ANSI C Standard가 나오면서 여러 어려움들이 해결되었고 이 책을 통해 그 방법들을 소개한다.

prototype이 있는 함수 선언이나 정의가 보인다면, 컴파일러는 주어진 함수 호출이 올바른 인수 타입으로 호출하는지 알 수 있다.
만약 C 프로그램이 여러 소스파일로 이루어져있다면 다음과 같은 안전한 방법들을 사용하는 것이 좋다.

- 모든 외부 함수는 헤더파일에 하나의 프로토타입 선언이 있어야한다.
- 함수의 정의가 있는 소스파일은 헤더파일을 포함해야한다.
- 모든 소스파일은 적절한 헤더파일을 포함해야한다.

이러한 루틴들에 따라서 Numerical Recipes에 있는 모든 프로토타입은 nr.h에 선언되어 있다.
이 외에 반복적으로 사용하는 소수의 유틸리티 루틴이 nrutil.h에 선언되어있다.

C에서 pointer와 array의 관계는 매우 가깝다. C에서 Array는 zero-origin으로 0부터 시작이 된다.
한가지 문제는 많은 알고리즘들이 0부터 M-1이 아닌 1부터 M까지로 사용한다. 이것을 바꾸기 위해 다음과 같은 방법을 사용한다.

```
float b[4], *bb;
bb = b-1;
```

이렇게 정의하면 bb의 범위는 [1-4]가 된다. 이를 unit-offset vector라고 부르고 이 책에서는 argument로 array가 전달될 때 index range를 적어준다.
C에서는 zero-offset array가 일반적인데 이를 unit-offset vector로 만들어주는 몇가지 루틴들이 있다.

이차원 배열에서도 역시 동일한 문제가 있다. 2차원 배열은 과학 계산의 핵심인데 시스템 프로그래밍에서는 가변적이고 런타임시에만 알려진 2차원 배열을 다루지 않는다.
이차원 배열은 선언되는 방식에 따라서 계산을 하는 방식이 다른데 array로 선언되는 것보다 double pointer로 선언하는 것이 더 빠르고 다양한 일을 할 수 있다.
따라서 크기가 고정된 2차원 배열을 사용하지 않고 double pointer로 2차원 배열을 다루게 된다.

C에서는 복소수를 위한 데이터 타입이 없다. 이를 위해 complex.c 복소수 계산을 위한 데이터 타입을 정의해두었다.

그리고 double 대신 float를 사용하였는데 이는 계산 속도의 손실을 방지하기 위해서이다. Double을 사용하고 싶다면 float를 double로 변경하여 사용하면 된다.

Error, Accuracy, and Stability

컴퓨터에서 수는 fixed-point 혹은 floating-point로 표현하는데 integer를 표현하는 것은 정확하다. 하지만 floating-point 표현에서는 약간의 오차가 발생한다. 1.0에 매우작은 수를 더해도 그 결과가 1.0이 되는 수를 machine accuracy라고 한다. 32bit에서는 그 값이 약 3*10^-9 정도이다. 이러한 오차를 roundoff error라고 한다. Machine accuracy는 floating point number에서 표현될 수 있는 가장 작은 수가 아니고 roundoff error는 계산이 많아질수록 그 크기가 증가한다. Roundoff error는 컴퓨터 계산의 특성이다. 또 다른 error는 program 혹은 alogorithm의 특성으로 생기는 truncation error이다.

Pointer to function & pointers for memory allocation

Numerical Analysis를 하려면 여러 함수들의 값을 계산하고 측정해야 하는데 이러한 함수들을 모두 하드코딩한다면 재사용률이 매우 떨어져 사용하기 어려울 것이다. 따라서 Numerical Analysis Method들을 이용할 때는 우리가 분석하고자 하는 함수들을 인자로 받아서 사용할 수 있도록 해야한다. 이를 위해 함수 포인터 개념이 사용된다. 포인터는 메모리를 저장하는 공간인데 이 메모리 값을 함수의 메모리로 저장한다면 함수를 인자로 전잘할 수 있게 된다.
Memory allocation을 위해서도 포인터를 사용해야 하는데 그 이유는 위에서 언급한 것처럼 array로 선언했을 때 보다 pointer로 선언했을 때 더 빠르고 다양한 일들을 할 수 있기 때문이다.
Numerical recipes를 사용할 때는 nrutil.c에 선언된 배열을 alloc해주고 free해주는 여러 유틸들을 사용한다면 memory관리를 효율적으로 할 수 있게 된다.

3.6

1) $e^{-x} = 1 - x + \frac{x^2}{2} - \frac{x^3}{3!} + \frac{x^4}{4!} - \frac{x^5}{5!} + \cdots$

$e^{-5} = 1 - 5 + \frac{5^2}{2} - \frac{5^3}{3!} + \frac{5^4}{4!} - \frac{5^5}{5!} + \cdots$

$\doteq 6.74554 \times 10^{-3}$

relative error $= \frac{6.73744 \times 10^{-3} - 6.74554 \times 10^{-3}}{6.73744 \times 10^{-3}}$

$\doteq -1.12690 \times 10^{-3}$

2) $e^{-x} = \frac{1}{e^x} = \frac{1}{1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \cdots}$

$e^{-5} = \frac{1}{e^5}$

$\doteq 6.73748 \times 10^{-3}$

relative error $= \frac{6.73744 \times 10^{-3} - 6.74748 \times 10^{-3}}{6.73744 \times 10^{-3}}$

$\doteq 1.48432 \times 10^{-7}$

3.7 $f'(x) = \frac{6x}{(1-3x^2)^2}$

$f'(0.5\pi) = \frac{6 \cdot 0.5\pi}{(1-3 \cdot 0.5\pi^2)^2}$

$= 2.352.911$

3-digit

$6x = 3.462 \doteq 3.46$

$x^2 = 0.332424 \doteq 0.332$

$f'(0.5\pi) = \frac{3.46}{1-0.498^2} \doteq 216.250$

4-digit

$6x = 3.462$

$x^2 = 0.3329$

$f'(0.5\pi) = \frac{3.462}{1-0.4998^2} \doteq 2.006.521$

3-digit은 오차가 매우 큼을 알수 있다.

4.2

i) $\cos\frac{\pi}{3} = 1$

$$\frac{0.5 - 1}{0.5} = 1$$

ii) $\cos\frac{\pi}{3} = 1 - \frac{\left(\frac{\pi}{3}\right)^2}{2}$

$$\doteq 0.451689$$

$$\frac{0.5 - 0.451689}{0.5} \doteq 0.096622$$

iii) $\cos\frac{\pi}{3} = 1 - \frac{\left(\frac{\pi}{3}\right)^2}{2} + \frac{\left(\frac{\pi}{3}\right)^4}{4!}$

$$\doteq 0.501046$$

$$\frac{0.5 - 0.501046}{0.5} \doteq 0.002092$$

4.5 $f(3)=?$

i) 0-order

$f(3) \approx f(1)$

$= -62$

$e_t = \frac{54-(-62)}{54} = 2.11$

ii) 1-order

$f(3) \approx f(1) + f'(1)\cdot 2$

$= -62 + 140 = 78$

$e_t = 0.859$

iii) 2-order

$f(3) \approx f(1) + f'(1)\cdot 2 + \frac{f''(1)}{2!}\cdot 2^2$

$= 78 + \frac{138}{2}\cdot 2^2$

$= 354$

$e_t = 0.361$

iv) 3-order

$f(3) \approx f(1) + f'(1)\cdot 2 + \frac{f''(1)}{2!}\cdot 2^2 + \frac{f'''(1)}{3!}\cdot 2^3$

$= 354 + \frac{150}{6}\cdot 8$

$= 754$

$e_t = 0.$

4.12

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

$$v(6) \approx v(5) + v'(5)$$

$$= \frac{gm}{c}\left(1 - e^{-\frac{c}{m}\cdot 5}\right) + g \cdot e^{-\frac{c}{m}\cdot 5}$$

$$= \frac{9.81 \times (50 \pm 2)}{12.5 \pm 1.5}\left(1 - e^{-\frac{12.5 \pm 1.5}{50 \pm 2}\cdot 5}\right)$$

$$+ 9.81 \cdot e^{-\frac{12.5 \pm 1.5}{50 \pm 2}\cdot 5}$$

$$\approx -122.153$$