Computational Physics
計算物理

https://ceiba.ntu.edu.tw/1071Phys7030_

Professor Ting-Wai Chiu 趙挺偉
Email: twchiu@phys.ntu.edu.tw
Physics Department
National Taiwan University

COURSE OUTLINES

This course introduces the numerical methods for solving problems in sciences and engineering whose complexity or difficulty is beyond analytic solution or human endurance.

Since computer simulation has become an integral part of basic and applied sciences and has been serving as a bridge between experimental and theoretical sciences, this course focuses on the Monte Carlo simulation of physical systems, and the related algorithms.

It is assumed that students have learned the basic programming techniques in C and/or Fortran, and C is the primary programming language in this course.

Topics

- Basic Mathematical Operations
- Differentiation and Integration
- Monte Carlo Integration
- Monte Carlo Simulation of Spin Systems
- Probability and Statistics, Random Number Generators
- System of Linear Equations
- Differential Equations
- Partial Differential Equations
- Introduction to Quantum Field Theory
- Path Integral Formulation of QFT
- Monte Carlo Simulation of QFT

REFERENCES

1. B. Kernigan and D. Ritchie, "C Programming Language", 2nd Edition, Prentice Hall (1988).

2. R. Landau, M. Paez and C. Bordeianu, "Computational Physics", 2nd Edition, Wiley (2007).

3. Press, W.H., et. al., "Numerical Recipes, The Art of Scientific Computing", Cambridge (1992).

Grading

Homework	70%
Term Project	30%
Class Participation and Discussions	10%

Guidelines

- You are allowed to discuss HW assignments with other students in the class.
- However, copying other's code/HW is not acceptable.
- Also, copying some library code that directly gives the solution of a HW/Project is not acceptable.
- Two students can collaborate on one term project, and submit a joint report.

This Lecture will cover:

1. Introduction
2. Numerical Differentiation
3. Numerical Integration

Platforms for computations

If you have a MacBook, you can use it to work out the problem sets and the term project. On the other hand, if you have a Window system, you may set up cygwin in your Window system.

cygwin - Linux-like environment for Windows making it possible to port software running on POSIX systems (such as Linux, BSD, and Unix systems) to Windows.

http://www.cygwin.com/

Some useful commands in Linux

man	format and display the on-line manual pages
ls	list directory contents
cp	copy files and directories
mkdir	make directories
pwd	print name of current/working directory
ps	report process status
vi	text editor

The Editor - vi

You may use vi to edit your programs as well as any files.

For tutorials on vi, see, for example,
http://www.oualline.com/vim-cook.html
http://www.study-area.org/cyril/opentools/opentools/editor.html

The nature of any program

A program is an implementation of an algorithm to perform the target task, which can be executed by a computer system with a single or many CPUs.

Basic questions

1. What is the target task ?
2. Which algorithm for the target task ?
3. How much resources does it take ?

The basic features of a program

1. Declarations - Variables, Data, …
2. Inputs and Outputs
3. Arithmetic Operations on Scalar and Vectors
4. Loops, and Loops within a Loop
5. Conditional Statements
6. Subroutines and Functions
7. Main Program

Programming Languages

For this course, we use C and F77
which are available in cygwin

F77 Fortran is a programming language designed for numerical computations in sciences and engineering. Many libraries in the public domain.

C A modern and versatile programming language for system programming, and numerical computations.

Program Development

In this course, we will discuss how to develop programs effectively, with examples, from the simplest program to the state-of-the-art ones for Monte Carlo simulations of QCD.

Data Representation (IEEE Standard)

Integer - 4 bytes

32 bits = 1 sign bit + 31 bits

Single precision real number – 4 bytes

32 bits = 1 bit (sign) + 8 bits (exponent)
+ 23 bits (fraction)

Double precision real number – 8 bytes

64 bits = 1 bit (sign) + 11 bits (exponent)
+ 52 bits (fraction)

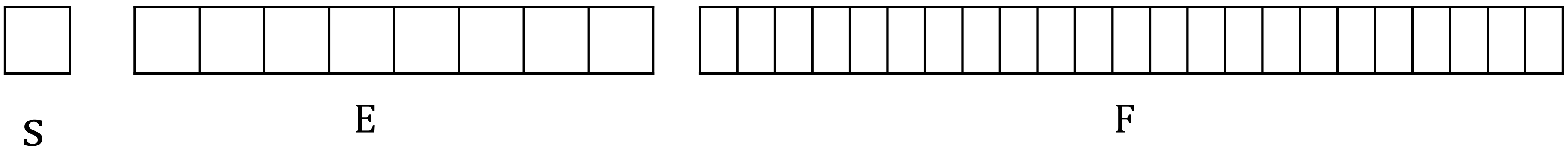
Integer - 4 bytes

4 bytes = 32 bits = 1 sign bit + 31 bits

0	1	` ` ` ` ` ` ` ` `		1	$= 2147483647 = 2^{31} - 1$	
		⋮				
0	0	` ` ` ` ` ` ` `	1	0	= 2	
0	0	` ` ` ` ` ` ` `	0	1	= 1	
0	0	` ` ` ` ` ` ` `	0	0	= 0	
1	1	` ` ` ` ` ` ` `	1	1	= -1	2's complement plus 1
1	1	` ` ` ` ` ` ` `	1	0	= -2	
		⋮				
1	0	` ` ` ` ` ` ` `	0	0	$= -2^{31}$	

Single precision real number – 4 bytes

32 bits = 1 sign bit + 8 exponent bits + 23 mantisa (fraction) bits



$$|R| = 1.F \text{ x } 2^{E\text{-bias}} \qquad \max(E) = 255 = 2^8 - 1 = \frac{2^8 - 1}{2 - 1}$$

bias = 127

$$1.175494 \text{ x } 10^{-38} \leq |R| \leq 3.402823 \text{ x } 10^{38}$$

Double precision real number - 8 bytes

1	bit	for	sign
11	bits	for	exponent
52	bits	for	fraction

$$|R| = 1.F \text{ x } 2^{\text{E-bias}} \qquad \text{bias} = 1023$$

$$2.225047 \text{ x } 10^{-308} \leq |R| \leq 1.797693 \text{ x } 10^{308}$$

Machine Precision

Machine precision is the smallest number ε such that the difference between 1 and 1 + ε is nonzero. In other words, ε is the smallest difference between two numbers that can be recognized by the computer. A pseudocode to determine ε in the single/double precison arithematics is given as follows.

```
void main() {
 float x, eps=1.0;
 int   i, N=100;
 for (i=1; i<N; i++){
   eps = eps/2.0;
   x = 1.0 + eps;
   printf("i=%d  x =%25.20e  eps=%25.20e \n", i, x, eps);
   if(x == 1.0) exit(0);
 }
}
```

```
void main() {
 double x, eps=1.0;
 int   i, N=100;
 for (i=1; i<N; i++){
   eps = eps/2.0;
   x = 1.0 + eps;
   printf("i=%d  x =%25.20e  eps=%25.20e \n", i, x, eps);
   if(x == 1.0) exit(0);
 }
}
```

Fortran versus C

FORTRAN	C
Program Structure	
program f	void main( )
function f(x)	double f(double x)
subroutine abc(x,y,z)	void abc(int x, float y, double z)
Data Type Declarations	
real*8 x,y	double x,y;
integer i,j	int i,j;
real*8 z(100,100)	double z[100][100];
data y/0.0/	#define y 0.0;
character abc	char abc;

(Note that in Fortran, all variables are case insensitive,
but in C, everything is case sensitive, e.g., Abc is not the same as abc)

Fortran versus C (cont)

FORTRAN	C

==

Operations

FORTRAN	C
`x * y`	`x * y`
`x ** y`	`pow(x,y)`
`x = y`	`x = y`

```
do k=1,kmax
   x = x + k
   y = y + k**2
end do
```

```
for(k=1;k<=kmax;k++) {
   x = x + k;
   y = y + pow(k,2);
}
```

```
do k=kmax,1,-1
   x = x + k
end do
```

```
for(k=kmax; k>=1;k--) {
   x = x + k;
}
```

Fortran versus C (cont)

FORTRAN	C
Input and Output to screen	
write(*,*) 'Enter angle :’	printf("Enter angle :");
read(*,*) x	scanf(“%lf”,x);
write(*,*) 'angle is ', x	printf("angle is %f ",x);
Input and Output to Files	
	FILE *fout, *fin;
open(7,file='data.dat',status='new')	fout = fopen("data.dat“, "w");
write(7,F10.2) angle	fprintf(fout,"%lf“,angle);
open(8,file='input.dat',status='old')	fin = fopen("input.dat“, "r");
read(8,*) x	fscanf(fin,"%lf“, x);

Fortran versus C (cont)

FORTRAN C

==

Control Structure : if ... else

FORTRAN	C

`if( ab .eq. 0 ) then`
`   print *,'Error !'`
`end if`
	`if( ab == 0 ) {`
	`  printf("Error !");`
	`}`

`if( ab .eq. 0 ) then`
`   print *,'Error !'`
`else`
`   x = x + 1.0`
`end if`
	`if( ab == 0 ) {`
	`  printf("Error !");`
	`}`
	`else {`
	`    x = x + 1.0;`
	`}`