

# Data Science Course Group (Graduate School): Simulation Tutorials

## Lecture 1 Lecture Materials

Instructor: Takeshi Kawasaki

Nagoya University, Graduate School of Science, Department of Physics, Nonequilibrium Physics Research Group (R Lab)

Last update: August 23, 2024

# Contents

## 1 About This Lecture (Tutorial)

- Lecture Schedule
- Purpose and Important Points of This Tutorial
- Syllabus

## 2 Programming Setup

- Setting up the C(C++) Environment
- Compiling C(C++) Programs
- Setting up the Python Environment
- Running Python (matplotlib)

## 3 First Assignment

## 4 References

# Contents

## 1 About This Lecture (Tutorial)

- Lecture Schedule
- Purpose and Important Points of This Tutorial
- Syllabus

## 2 Programming Setup

- Setting up the C(C++) Environment
- Compiling C(C++) Programs
- Setting up the Python Environment
- Running Python (matplotlib)

## 3 First Assignment

## 4 References

## 1.1. Lecture Schedule

- This tutorial will be conducted during the first term of spring.
- Lecture materials will be uploaded to TACT by 11:00 AM on the day of each lecture.
- Schedule
  - 1 **April 15: Lecture 1**
  - 2 April 22: Lecture 2
  - 3 April 30 (Tue): Lecture 3
  - 4 May 13: Lecture 4 (**Midterm Report Assignment Released**)
  - 5 May 20: Lecture 5
  - 6 May 27: Lecture 6 (**Midterm Report Submission Deadline**)
  - 7 May 29: Lecture 7
  - 8 **June 3: No Lecture**
  - 9 June 10: Lecture 8 (**Final Report Assignment Released**)
  - 10 June 17: Lecture 9 (Make-up Lecture)

## 1.2. Purpose and Important Points of This Tutorial

### Purpose:

- The purpose of this tutorial is to acquire practical knowledge and skills in computer simulations, which are essential in various aspects of scientific research.
- To achieve this goal, the lectures and exercises will cover a wide range of practical numerical calculation methods, from basic programming for **analysis** and **plotting** using C (C++) and Python to practical numerical simulation techniques such as the Monte Carlo method and molecular dynamics, with a focus on their broad application in scientific research.

### Important Points:

- Attendance surveys will be conducted for each session (via TACT quizzes). Please feel free to write your questions and comments.
- A self-study assignment will be given after each tutorial session.
- In addition, two report assignments will be issued (via TACT assignments).
- The main computations in this tutorial will be demonstrated using C (C++), and the plotting will be done using Python, with explanations provided.
- However, you are free to use other languages or adjust the division of tasks (e.g., writing everything in Python) for the self-study and report assignments.

### Sample Programs:

- Sample programs will be stored in the following GitHub repository, so please download them as needed. [[Link](#)][1]

## 1.3. Syllabus

The following topics are planned to be covered in this tutorial (subject to change depending on the progress).

### 1 Introduction

- Basics of C (C++) (mainly for numerical calculations)
- Basics of Python (for data analysis and plotting)
- Concepts of numerical calculations
- Loss of significance
- Non-dimensionalization in scientific calculations

### 2 Numerical solutions of ordinary differential equations: Examples with damped oscillations and harmonic oscillators

- Numerical integration of differential equations
- Stability and conservation laws of orbits

### 3 Brownian motion of a single particle

- Langevin equation (stochastic differential equation)
- Generation of normal random numbers
- Euler-Maruyama method
- Time averaging and ensemble averaging

### 4 Brownian motion of multi-particle systems

- Calculation methods for interaction forces
- Non-equilibrium system simulations: Example with phase separation

### 5 Molecular dynamics simulation of multi-particle systems

- Position Verlet method and velocity Verlet method
- Conservation laws in multi-particle systems

### 6 Monte Carlo method

- Review of statistical mechanics
- Markov chain Monte Carlo method
- Metropolis algorithm

# Contents

## 1 About This Lecture (Tutorial)

- Lecture Schedule
- Purpose and Important Points of This Tutorial
- Syllabus

## 2 Programming Setup

- Setting up the C(C++) Environment
- Compiling C(C++) Programs
- Setting up the Python Environment
- Running Python (matplotlib)

## 3 First Assignment

## 4 References

## 2.1. Setting up the C(C++) Environment

- In this tutorial, the primary numerical calculations will be done using C(C++), and plotting will be done using Python.
- Setting up the C(C++) environment:
  - **Windows:** Installing Ubuntu Yoshii's manual: [[Link](#)] [2]
  - **mac:** Installing Homebrew [[Homebrew Official Link](#)] [2]

**List 1:** (On terminal) (For mac) Installing emacs (terminal editor) using Homebrew.

```
1 brew install emacs
```

### Common Unix Commands

- **cd** (directory name): Change directory
- **mkdir** (directory name): Create a directory
- **rm** (directory name): Remove a directory
- **touch** (file name): Create a file
- **ls**: List files and directories



## 2.2. Compiling C(C++) Programs

This section explains how to compile and run the following C(C++) program (pi.cpp) on your computer.

### ■ C Compiler:

#### Various Compilers

- gcc: GNU C compiler collection.
- g++: GNU C++ compiler collection.
- clang: An alternative to gcc — the default compiler on macOS.
- clang++: An alternative to g++ — the default compiler on macOS.
- icc: Intel C/C++ compiler — expensive but offers faster computation speed.

#### Compilation Options

- -O3: One of the optimization options (i.e., -O -O0 -O1 -O2 -O3 -Os -Ofast -Og). -O3 is most commonly used.
- -o: Specifies the name of the output file.

### ■ サンプルプログラム “pi.cpp”

## 2.2. Compiling C(C++) Programs (2)

List 2: サンプルプログラム “pi.cpp”

```
1  #include <stdio.h> // for printf, etc
2  #include <stdlib.h> // for rand(), etc
3  #include <math.h> // for sin(),cos(), etc
4  #include <iostream> // for cout, etc
5  #include <iomanip> // for setprecision()
6  #include <fstream> // for ifstream/ofstream
7  #include <time.h> // for time(NULL), etc
8
9  int main(void){
10     int i, count = 0, max = 1e+5;
11     double x,y,z,pi;
12     char fname[128];
13     std::ofstream file;
14     srand(time(NULL)); // "time(NULL)" as a seed of random number
15     sprintf(fname, "coord%d.dat", max); // Define the file name for fname[128]
16     file.open(fname); // "file" with the name of fname[128]
17     for(i=0; i<max; i++){
18         x = (double)rand()/RAND_MAX;
19         y = (double)rand()/RAND_MAX;
20         z = x*x + y*y;
21         if(z<1){
22             count++;
```

## 2.2. Compiling C(C++) Programs (3)

```
23     file << x << " " << y << std::endl;  
24 }  
25 }  
26 file.close();  
27 pi = (double)count / max * 4.;  
28 printf("%.20f\n", pi); //by C, %.20f -- Displaying with 20 decimal precision  
29 std::cout << std::setprecision(21) << pi << std::endl; // by C++  
30  
31 return 0;  
32 }
```

- Download the program from the designated GitHub repository [\[Link\]](#) [1].
- How to download from GitHub [\[Link\]](#) [3].
- After downloading, place 'pi.cpp' in an appropriate directory.
- In the directory where 'pi.cpp' is located, execute the following commands.
- Compilation method:

## 2.2. Compiling C(C++) Programs (4)

**List 3:** (On terminal) Example of code compilation: (**execute each**) . Use the `-o` option to specify the name of the output executable file. `a.out` is obtained in case of no `-o` option.

```
1 g++ -O3 pi.cpp
2 g++ -O3 pi.cpp -o pi.out
3 g++ -O3 -o pi.out pi.cpp
4 clang++ -O3 -o pi.out pi.cpp
```

### ■ Execution method:

**List 4:** (Linux/Mac terminal) Running the program (`pi.out`).

```
1
2 ./pi.out
```

### ■ Execution result:

**List 5:** (On terminal) Example of program execution result. Note that the values may vary each time as the random seed is set to the current time.

```
1
2 3.145360000000000015575
3 3.145360000000000015575
```

## 2.2. Compiling C(C++) Programs (5)

### Useful Editor (Optional): Visual Studio Code

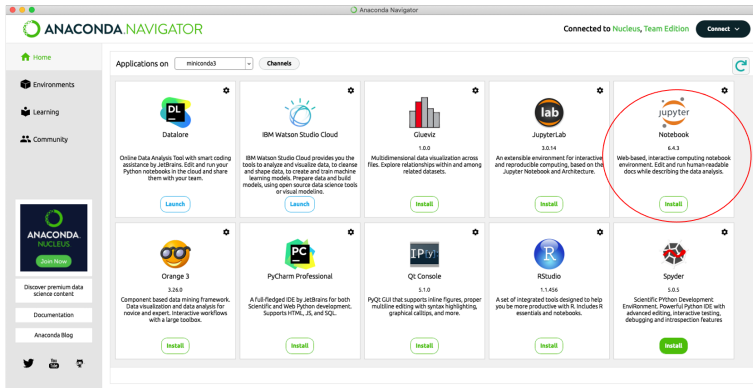
Setup Manual (Created by Mr. Ikeda, R Lab):

- Win: [\[Link\]](#)
- Mac: [\[Link\]](#)

## 2.3. Setting up the Python Environment

- In this course, Python programs will be executed using Jupyter Notebook.
- Jupyter Notebook is included in Anaconda Navigator ([[Official Link](#)], see Figure 1).
- Refer to the following for instructions on installing Anaconda Navigator.
  - **Windows/mac:** Installation of Anaconda Navigator  
[\TKWindows manual] [\TKMac manual]
- The installation method for Jupyter Notebook is shown in Figure 2.

## 2.3. Setting up the Python Environment (2)



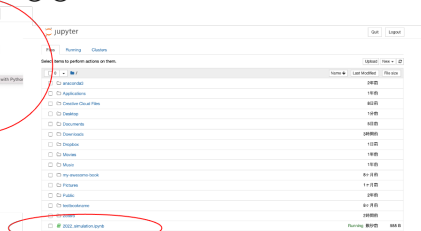
**Fig 1:** Install and run the Jupyter note book (circled in red) found in the Anaconda Navigator. Now you can use python.

## 2.3. Setting up the Python Environment (3)

### ① jupyterを開く



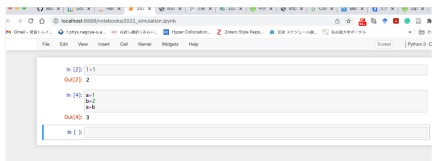
### ③ ②で作ったnotebookを開く



### ② Newのタブを開きpython3を選び、新規Notebookに名前をつける。



### ④ グレーの箱にプログラムを書き込み、shift+enterで実行



**Fig 2:** From the start of the Jupyter notebook to the execution of python..



## 2. 4. Running Python (matplotlib)

- Run the following program (plotting using matplotlib.pyplot) on Jupyter Notebook (see the execution result in Figure 3).
- (Reference) matplotlib manual [[Link](#)]

### Supplement on Plotting with Python [4]

- matplotlib: A graphics package for Python.
- matplotlib.pyplot: Provides various graph tools such as axes, figure, plot, scatter, etc.

#### List 6: Importing and using matplotlib.pyplot example

```
1 import matplotlib.pyplot as plt
2 fig = plt.figure(figsize=(7,7))
```

- NumPy: Provides fast computation. Many packages refer to NumPy. Offers various linear algebra operations, statistical operations, special functions, etc.

#### List 7: Importing NumPy

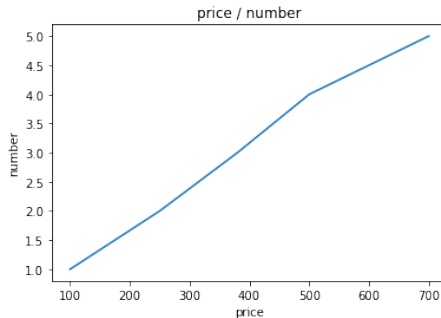
```
1 import numpy as np
```

## 2. 4. Running Python (matplotlib) (2)

List 8: Python Sample Program “matplot.py”

```
1 import matplotlib.pyplot as plt
2 %matplotlib inline
3 # Increase the resolution of the graph
4 %config InlineBackend.figure_format = 'retina'
5 price = [100, 250, 380, 500, 700]
6 number = [1, 2, 3, 4, 5]
7
8 # Plot the graph
9 plt.plot(price, number)
10
11 # Graph title
12 plt.title("price / number")
13
14 # x-axis label
15 plt.xlabel("price")
16
17 # y-axis label
18 plt.ylabel("number")
19
20 # Display the graph
21 plt.show()
```

## 2. 4. Running Python (matplotlib) (3)



**Fig 3:** Execution result of matplotlib.py.

- Next, create a scatter plot of “coord100000.dat” generated by running the program in List 2, as shown below.
- In the sample program “coord.py,” some advanced formatting is applied, such as adjusting the size of the figure, aspect ratio, axis thickness, font size, and font (TeX). Execution result (Figure 4):

## 2. 4. Running Python (matplotlib) (4)

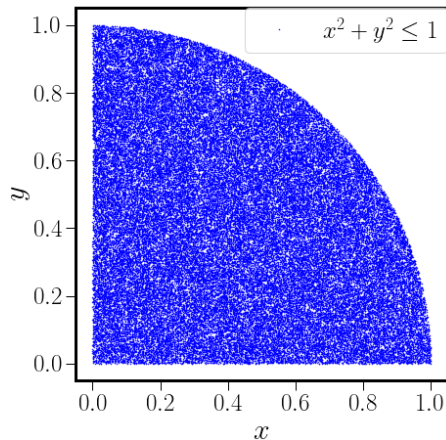
List 9: Python Sample Program “coord.py”

```
1  import matplotlib
2  import matplotlib.pyplot as plt
3  %matplotlib inline
4  # Increase the resolution of the graph
5  %config InlineBackend.figure_format = 'retina'
6  import numpy as np
7  # TeX font
8  #plt.rcParams["text.use_tex"] = True
9
10 # Change the overall size and aspect ratio of the figure
11 fig = plt.figure(figsize=(7,7))
12
13 # When placing multiple graphs, change this
14 ax = fig.add_subplot(111)
15
16 # Change the file path as needed
17 x, y = np.loadtxt("./Lecture1/coord100000.dat", comments='#', unpack=True)
18 plt.plot(x, y, "o", markersize=0.5, color="b", label=r"$x^2+y^2\leq 1$")
19
20 # Graph formatting
21 plt.tick_params(which='major', width = 1, length = 10)
22 plt.tick_params(which='minor', width = 1, length = 5)
```

## 2. 4. Running Python (matplotlib) (5)

```
23 ax.spines['top'].set_linewidth(3)
24 ax.spines['bottom'].set_linewidth(3)
25 ax.spines['left'].set_linewidth(3)
26 ax.spines['right'].set_linewidth(3)
27 plt.xlabel(r"$x$",color='k', size=30)
28 plt.ylabel(r"$y$",color='k', size=30)
29 plt.xticks(color='k', size=25)
30 plt.yticks(color='k', size=25)
31 # Adjust the presence, position, and size of the graph legend
32 plt.legend(ncol=1, loc=1, borderaxespad=0, fontsize=25,frameon=True)
33 # Set graph margins
34 plt.subplots_adjust(wspace=0.0, hspace=0.25)
35 # Set the aspect ratio of each graph to 1:1
36 ax.set_aspect('equal', adjustable='box')
37 # Change the file path as needed.
38 plt.savefig('./Lecture1/coord.png')
39 plt.savefig('./Lecture1/coord.pdf')
```

## 2. 4. Running Python (matplotlib) (6)



**Fig 4:** Execution result of coord.py.

# Contents

## 1 About This Lecture (Tutorial)

- Lecture Schedule
- Purpose and Important Points of This Tutorial
- Syllabus

## 2 Programming Setup

- Setting up the C(C++) Environment
- Compiling C(C++) Programs
- Setting up the Python Environment
- Running Python (matplotlib)

## 3 First Assignment

## 4 References

### 3. First Assignment

#### Convergence of $\pi$ (Monte Carlo Simulation)

- (1) Based on the sample program in List 2, examine the convergence of the calculated value of  $\pi$  as the number of random samples  $n$  changes. Create a graph with  $n$  on the horizontal axis and the calculated value of  $\pi(n)$  on the vertical axis. Also, plot the error  $\delta(n) = |\pi(n) - \pi|$  relative to the theoretical value of  $\pi$  as a function of  $n$ , and evaluate the results.
- (2) (Advanced Problem) In large-scale numerical simulations, the standard pseudo-random number generator `rand()` may occasionally cause issues due to its short period. On the other hand, the Mersenne Twister, developed by Makoto Matsumoto, is known for its high-quality pseudo-random number generation [\[Reference Link\]](#) [6]. Modify the sample program in List 2 to use the Mersenne Twister instead [\[Reference Link\]](#) [7]. The Mersenne Twister header file has been placed in the designated GitHub repository [\[Link\]](#) [1], so feel free to use it.



# Contents

## 1 About This Lecture (Tutorial)

- Lecture Schedule
- Purpose and Important Points of This Tutorial
- Syllabus

## 2 Programming Setup

- Setting up the C(C++) Environment
- Compiling C(C++) Programs
- Setting up the Python Environment
- Running Python (matplotlib)

## 3 First Assignment

## 4 References

## References and Websites

- [1] Takeshi Kawasaki.  
2024-simulation-tutorial (GitHub), April 2022.
- [2] Homebrew (The Missing Package Manager for macOS (or Linux)).  
[https://brew.sh/index\\_ja](https://brew.sh/index_ja).
- [3] GitHub からファイルをダウンロードする方法【2020 年 12 月追記】 .  
<https://tetsufuku-blog.com/github-download/>, April 2020.
- [4] Hiroshi Hashimoto and Koji Makino.  
Python コンピュータシミュレーション入門.  
オーム社, 2021.
- [5] Kotaro Matsumoto.  
早く知っておきたかった matplotlib の基礎知識、あるいは見た目の調整が捗る Artist の話.  
<https://qiita.com/skotaro/items/08dc0b8c5704c94eafb9>.
- [6] Makoto Matsumoto.  
Mersenne Twister: A random number generator (since 1997/10).  
<http://www.math.sci.hiroshima-u.ac.jp/m-mat/MT/mt.html>.

## References and Websites (2)

- [7] Takahiro Ohmi.  
C 言語による乱数生成.  
[https://omitakahiro.github.io/random/random\\_variables\\_generation.html](https://omitakahiro.github.io/random/random_variables_generation.html).