Simulation Tutorials
Lecture 2

Instructor: Takeshi Kawasaki

D3 center, Osaka University

Last update: April 18, 2025

Contents

Contents

1. Explanation of Assignment 1

Convergence of π (Monte Carlo Simulation)

(1) Regarding the sample program in List 2, consider the convergence of the calculated value of π as the number of random number generations n is varied. Plot the relationship between n on the horizontal axis and $\pi(n)$ on the vertical axis by taking various values of the numerical calculation of $\pi(n)$ with respect to n. Also, evaluate and plot how the error $\delta(n) = |\pi(n) - \pi|$ changes with respect to n.

(2) (Advanced Problem) In large-scale numerical calculations, the pseudo-random number generator rand() may cause problems due to its short random number cycle. On the other hand, the Mersenne Twister, developed by Makoto Matsumoto, is famous as a high-quality pseudo-random number generator [**Reference Link**][1]. Now, rewrite the sample program in List 2 to use the Mersenne Twister [**Reference Link**][2]. You may use the header file for the Mersenne Twister placed in the specified GitHub repository [**Link**][3].

1. Explanation of Assignment 1 (2)

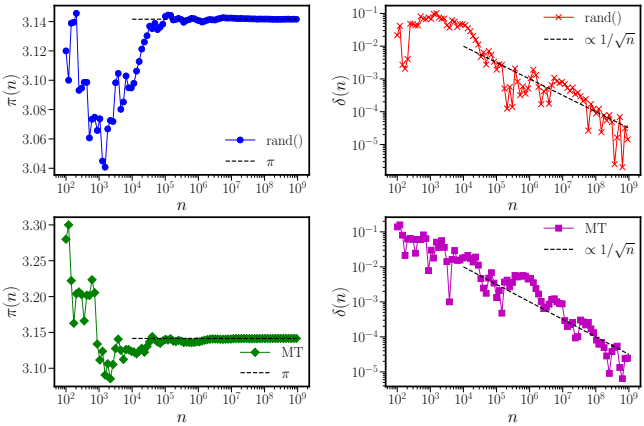

図 1: Calculated $\pi(n)$ and $\delta(n)$ using the standard random number generator rand() (upper) and Mersenne Twister (lower).

- Below is the calculation program using C(C++).

List 1: Assignment 1 Sample Calculation Program “pi_error.cpp”. Available in the following GitHub repository: **[Link]**

```
1
2  #include <stdio.h> // for printf, etc
3  #include <stdlib.h> // for rand(), etc
4  #include <math.h> // for sin(),cos(), etc
5  #include <iostream>// for std::cout, etc
6  #include <iomanip>//  for std::setprecision()
7  #include <fstream> // for ifstream/ofstream
8  #include <time.h>// for time(NULL), etc
9  #include "MT.h"// for MT
10
11 int main(void){
12   int i, count = 0, max = 1e+8;
13   double x,y,z,pi, out=1.e+2;
14   char fname[128];
15   std::ofstream file;
16   srand(time(NULL));
17   sprintf(fname,"pi-error.dat");
18   file.open(fname);
19   for(i=0;i<max;i++){
20     x = (double)rand()/RAND_MAX;
21     y = (double)rand()/RAND_MAX;
22     z = x*x + y*y;
23     if(z<=1.0)
24       count++;
```

```
25      if(i>=(int)out){    // (int)out: cast double to int
26        pi=(double)count /(double)i*4.0; // (double)count: cast int to double
27        file<<std::setprecision(16)<<i<<"\t"<<pi<<"\t"<<abs(pi-M_PI)<<std::endl;;
28        std::cout<<std::setprecision(16)<<i<<"\t"<<pi<<"\t"<<abs(pi-M_PI)<<std::endl;
29        out*=1.2;
30      }
31    }
32    file.close();
33    out=1e+2;
34    count=0;
35
36    sprintf(fname,"pi-error-MT.dat");
37    file.open(fname);
38    init_genrand(time(NULL));
39    for(i=0;i<max;i++){
40      x = genrand_res53();
41      y = genrand_res53();
42      z = x*x + y*y;
43      if(z<=1.0)
44        count++;
45      if(i>=(int)out){
46        pi=(double)count /(double)i*4.0;
47        file<<std::setprecision(16)<<i<<"\t"<<pi<<"\t"<<abs(pi-M_PI)<<std::endl;;
48        std::cout<<std::setprecision(16)<<i<<"\t"<<pi<<"\t"<<abs(pi-M_PI)<<std::endl;
49        out*=1.2;
50      }
51    }
52    file.close();
```

```
53   return 0;
54 }
55 }
```

- Below is a sample program for plotting.

List 2: Python Sample Program "pi_error.py" Available in the GitHub repository: **[Link]**. The equivalent program is also included in the jupyter notebook file "jupyter.ipynb".

```
1  import matplotlib
2  import matplotlib.pyplot as plt
3  %matplotlib inline
4  # Increases the resolution of the figures
5  %config InlineBackend.figure_format = 'retina'
6  import numpy as np
7
8  # Tex font (optional): uncomment the following if you want to use it
9  #plt.rcParams["text.usetex"] =True
10 plt.rcParams['font.family'] = 'Arial' # Font name to use
11 plt.rcParams["font.size"] = 25
12
13 # Change the overall size and aspect ratio of the figure
14 fig = plt.figure(figsize=(18,12))
15 # Adjust this when placing multiple plots
16
17  ############################
18 ax = fig.add_subplot(221)
19 # Change the file paths as needed
20 i, pi,error  = np.loadtxt("./Lecture2/pi-error-MT.dat", comments='#', unpack=True)
21 plt.plot(i, pi, "o-",markersize=10,color="b",label=r"rand()")
22 plt.xscale('log')
23 ###Drawing a line ######
24 x= np.linspace(1e4, 1e8, 100)
```

```
25 y= np.pi+0*x
26 plt.plot(x, y, "--",markersize=3,linewidth = 2.0, color="k",label=r"$\pi$")
27 #########
28 # Format the plot
29 plt.tick_params(which='major',width = 1, length = 10)
30 plt.tick_params(which='minor',width = 1, length = 5)
31 ax.spines['top'].set_linewidth(3)
32 ax.spines['bottom'].set_linewidth(3)
33 ax.spines['left'].set_linewidth(3)
34 ax.spines['right'].set_linewidth(3)
35 plt.xlabel(r"$n$",color='k', size=30)
36 plt.ylabel(r"$\pi(n)$",color='k', size=30)
37
38 # Adjust the legend position, size, and whether it appears
39 plt.legend(ncol=1, loc=4, borderaxespad=0, fontsize=25,frameon=False)
40 # Set the aspect ratio of each plot to 1:1
41 ################################
42
43  ############################
44 ax = fig.add_subplot(222)
45 # Change the file paths as needed
46 i, pi,error  = np.loadtxt("./Lecture2/pi-error.dat", comments='#', unpack=True)
47 plt.plot(i, error, "x-",markersize=10,color="r",label=r"rand()")
48 plt.xscale('log')
49 plt.yscale('log')
50
51 ###Drawing a line ######
52 x= np.linspace(1e4, 1e8, 100)
```

```
53 y=1/x**0.5
54 plt.plot(x, y, "--",markersize=3,linewidth = 2.0, color="k",label=r"$\propto 1/\sqrt{n}$")
55 #########
56
57
58 # Format the plot
59 plt.tick_params(which='major',width = 1, length = 10)
60 plt.tick_params(which='minor',width = 1, length = 5)
61 ax.spines['top'].set_linewidth(3)
62 ax.spines['bottom'].set_linewidth(3)
63 ax.spines['left'].set_linewidth(3)
64 ax.spines['right'].set_linewidth(3)
65 plt.xlabel(r"$n$",color='k', size=30)
66 plt.ylabel(r"$\delta(n)$",color='k', size=30)
67
68 # Adjust the legend position, size, and whether it appears
69 plt.legend(ncol=1, loc=1, borderaxespad=0, fontsize=25,frameon=False)
70 ################################
71
72  ############################
73
74 ax = fig.add_subplot(223)
75 # Change the file paths as needed
76 i, pi,error  = np.loadtxt("./Lecture2/pi-error-MT.dat", comments='#', unpack=True)
77 plt.plot(i, pi, "D-",markersize=10,color="g",label=r"MT")
78 plt.xscale('log')
79 ###Drawing a line ######
80 x= np.linspace(1e4, 1e8, 100)
```

```
81  y= np.pi+0*x
82  plt.plot(x, y, "--",markersize=3,linewidth = 2.0, color="k",label=r"$\pi$")
83  #########
84
85  # Format the plot
86  plt.tick_params(which='major',width = 1, length = 10)
87  plt.tick_params(which='minor',width = 1, length = 5)
88  ax.spines['top'].set_linewidth(3)
89  ax.spines['bottom'].set_linewidth(3)
90  ax.spines['left'].set_linewidth(3)
91  ax.spines['right'].set_linewidth(3)
92  plt.xlabel(r"$n$",color='k', size=30)
93  plt.ylabel(r"$\pi(n)$",color='k', size=30)
94
95  # Adjust the legend position, size, and whether it appears
96  plt.legend(ncol=1, loc=4, borderaxespad=0, fontsize=25,frameon=False)
97  ################################
98
99  ############################
100
101 ax = fig.add_subplot(224)
102 # Change the file paths as needed
103 i, pi,error  = np.loadtxt("./Lecture2/pi-error-MT.dat", comments='#', unpack=True)
104 plt.plot(i, error, "s-",markersize=10,color="m",label=r"MT")
105
106 ###Drawing a line ######
107 x= np.linspace(1e4, 1e8, 100)
108 y=1/x**0.5
```

```
109 plt.plot(x, y, "--",markersize=3,linewidth = 2.0, color="k",label=r"$\propto 1/\sqrt{n}$")
110 #########
111 plt.xscale('log')
112 plt.yscale('log')
113 # Format the plot
114 plt.tick_params(which='major',width = 1, length = 10)
115 plt.tick_params(which='minor',width = 1, length = 5)
116 ax.spines['top'].set_linewidth(3)
117 ax.spines['bottom'].set_linewidth(3)
118 ax.spines['left'].set_linewidth(3)
119 ax.spines['right'].set_linewidth(3)
120 plt.xlabel(r"$n$",color='k', size=30)
121 plt.ylabel(r"$\delta(n)$",color='k', size=30)
122
123 # Adjust the legend position, size, and whether it appears
124 plt.legend(ncol=1, loc=1, borderaxespad=0, fontsize=25,frameon=False)
125 ################################
126
127 # Adjust the margins of the plot
128 plt.subplots_adjust(wspace=0.3, hspace=0.3)
129
130 # Change the file paths as needed.
131 plt.savefig('./Lecture2/pi-error.png')
132 plt.savefig('./Lecture2/pi-error.pdf')
```

Contents

2.Basics of Numerical Computation

We now delve into the main topic of this lecture. Here, we summarize essential concepts that must be considered when performing numerical computations. Specifically, we will cover:

- **Floating Point Numbers and Rounding Errors (Information Loss and Cancellation)**
- **Discretization**
- **Non-dimensionalization**

2.1. Floating Point Numbers and Rounding Errors (Information Loss and Cancellation)

In this section, we discuss floating point numbers, which often introduce issues in numerical computation—such as rounding errors, information loss, and cancellation. We also explain why these problems arise.

Information Loss

When adding a large value and a small value, the information in the smaller value may be lost. For instance, since the precision of a float type is about 6 – 7 digits:

$$77777.7 + 1.23456 = 77778.9 \tag{1}$$

The lower 4 digits of 1.23456 are effectively discarded.

Cancellation

When subtracting two nearly equal numbers, significant digits may cancel out, leading to reduced accuracy:

$$77777.7 - 77777.6 = 0.1 \tag{2}$$

- These issues originate from how floating point numbers are represented in computers.

2.1. Floating Point Numbers and Rounding Errors (Information Loss and Cancellation) (2)

- A floating point number consists of a **mantissa**, which stores the significant digits, and an **exponent**, which determines the decimal point's position.
- This format is the most widely used for representing real numbers with decimals [4].

Floating Point Numbers and Machine Epsilon

Floating point representation follows the IEEE 754 standard.
Single-precision (float): 32 bits **Double-precision (double)**: 64 bits
Structure (binary):

Sign bit: 0 for positive, 1 for negative (1 bit)
Mantissa: Significant digits (float: 23 bits, double: 52 bits)
Exponent: Signed integer in offset form (float: 8 bits with bias 127, double: 11 bits with bias 1023)

Machine epsilon: the smallest increment distinguishable by the mantissa: For double precision: $2^{-52} \approx 2.22 \times 10^{-16}$
Note: The mantissa is normalized as 1.xxx, and the leading 1 is omitted (hidden bit).

2.2. Discretization

This section introduces **discretization**—the method of replacing derivatives and integrals with finite differences to enable numerical computation.

- As a simple example, consider the motion of a sphere in a fluid.
- Let a sphere of diameter a [m] and mass m [kg] move in one dimension through a fluid with drag coefficient ζ [kg/s] and initial velocity v_0 [m/s]. The equation of motion is:

$$m\frac{\mathrm{d}v(t)}{\mathrm{d}t} = -\zeta v(t) \tag{3}$$

- To solve this numerically, we discretize the time and apply a numerical integration method.
- First, we introduce the Euler method, a simple but low-accuracy method.
- The exact first derivative of a function $f(t)$ is defined by:

$$\frac{\mathrm{d}f(t)}{\mathrm{d}t} = \lim_{\Delta t \to 0} \frac{f(t+\Delta t) - f(t)}{\Delta t} \tag{4}$$

- In numerical computation, we approximate this using a small but finite time step Δt:

$$\boxed{\frac{\mathrm{d}f(t)}{\mathrm{d}t} \approx \frac{f(t+\Delta t) - f(t)}{\Delta t} + O(\Delta t)} \tag{5}$$

2.2. Discretization (2)

- The Euler method accumulates errors of order Δt, so it may be unsuitable for certain problems.
- However, for dissipative systems like our example, it provides reasonable accuracy (unlike for conservative systems such as harmonic oscillators).
- Discretizing Eq. (3) using the Euler method:

$$m\frac{v(t+\Delta t)-v(t)}{\Delta t} = -\zeta v(t) \tag{6}$$

leads to:

$$v(t+\Delta t) = v(t) - \frac{\zeta v(t)\Delta t}{m} \tag{7}$$

- For dissipative systems, the Euler method is often sufficient. For conservative systems, higher-order methods like the Velocity Verlet or Runge-Kutta are preferable.

2.3. Non-dimensionalization

In this section, we discuss **non-dimensionalization**, which simplifies equations by removing physical units.

- **All numbers handled by computers are inherently dimensionless.**
- Therefore, when solving equations with physical units, we must separate physical units from the dimensionless variables.
- We now apply non-dimensionalization to the differential equation introduced previously.
- Assume the following variable transformations: $t \to t_0\tilde{t}$, $v = v_0\tilde{v} = \frac{a}{t_0}\tilde{v}$.
- Substituting into Eq. (7) gives:

$$\frac{a}{t_0}\tilde{v}(\tilde{t}+\Delta\tilde{t}) = \frac{a}{t_0}\tilde{v}(\tilde{t}) - \frac{\zeta a \Delta\tilde{t}}{m}\tilde{v}(\tilde{t}) \tag{8}$$

$$\Rightarrow \tilde{v}(\tilde{t}+\Delta\tilde{t}) = \tilde{v}(\tilde{t}) - \frac{\zeta t_0}{m}\tilde{v}(\tilde{t})\Delta\tilde{t} \tag{9}$$

- Here, $\frac{\zeta t_0}{m}$ is a dimensionless parameter. Although any value is allowed in principle, we simplify the equation by setting:

$$\boxed{t_0 = \frac{m}{\zeta}} \tag{10}$$

which corresponds to the **relaxation time** for velocity damping.

2.3. Non-dimensionalization (2)

- While the choice of t_0 is arbitrary, poor choices can lead to **cancellation errors** due to overly large or small dimensionless parameters.
- With $\frac{\zeta t_0}{m} = 1$, the recurrence equation becomes:

$$\boxed{\tilde{v}(\tilde{t} + \Delta\tilde{t}) = \tilde{v}(\tilde{t}) - \tilde{v}(\tilde{t})\Delta\tilde{t}} \tag{11}$$

  which is extremely simple.
- Solving this equation numerically involves iterating the recurrence relation for $\tilde{v}(\tilde{t})$.

Contents

3. Assignment 2

Numerical Computation and Analytical Comparison of Damped Motion

Consider the motion of a particle in a solvent in one dimension. The equation of motion for this particle is:

$$m\frac{\mathrm{d}v(t)}{\mathrm{d}t} = -\zeta v(t),$$

and at time $t = 0$, it is assumed to be moving with a speed of $10a\zeta/m$. Answer the following questions based on this setup. When non-dimensionalizing, take the unit of length as a and the unit of time as $t_0 = m/\zeta$.

(1) Find the analytical solution of this differential equation. Further, non-dimensionalize it using the units specified in the problem statement.

(2) Numerically compute the time evolution of the particle's velocity over the time interval $[0, 10^4 t_0]$ and compare it with the analytical solution. For simplicity, use the Euler method for discretizing the equation of motion and compare results for different time resolutions: $\Delta t/t_0 = 0.1, 0.01, 0.001$.

Contents

4.1. Common Commands Used on Unix-based Terminals

Here are some commonly used commands on Unix-based terminals, as requested.

- Common Unix Commands [5]

List 3: Common Unix Commands:

```
pwd    Display the full path of the current directory
cd     Change directory
cd /   Move to the home directory
mkdir  Create a new directory
rm -r  Delete a specified directory along with its contents
cp -r (dirA) (dirB)   Copy a directory
mv (file) (dir)   Move a file to a directory
ls     Display the list of files in the current directory
cat (file)   Display the contents of a file
rm (file)   Delete a file
cp (fileA) (fileB)   Copy a file
mv (fileA) (fileB)   Rename a file
touch (file)   Create an empty file
find (dir)   List files under a specified directory
more (file)   Display the contents of a file (pauses after each page)
```

Contents

References

[1] Makoto Matsumoto.
Mersenne Twister: A random number generator (since 1997/10).
http://www.math.sci.hiroshima-u.ac.jp/m-mat/MT/mt.html.

[2] Takahiro Ohmi.
C 言語による乱数生成.
https://omitakahiro.github.io/random/random variables generation.html.

[3] Takeshi Kawasaki.
2024-simulation-tutorial (GitHub), April 2022.

[4] 浮動小数点数とは - IT 用語辞典.
https://e-words.jp/w/%E6%B5%AE%E5%8B%95%E5%B0%8F%E6%95%B0%E7%82%B9%E6%95%B0.html.

[5] Takayuki Omori.
UNIX コマンド集.
https://g-omr.github.io/unix.html, 2021.