データサイエンス系科目群（理・博）：シミュレーション実習
第 2 回　講義資料

担当：川﨑猛史

名古屋大学大学院理学研究科理学専攻物理科学系・非平衡物理研究室 (R 研)

Last update: April 18, 2022

目次

目次

1. 講義のスケジュール

- 当実習は春 1 期にて実施する.
- 講義資料は各講義予定日当日朝 **11** 時迄にアップロードする.
- スケジュール
  1. 4/11：第 1 回
  2. **4/18：第 2 回**
  3. 4/25：第 3 回
  4. 5/02：第 4 回 (中間レポート課題公開)
  5. 5/09：第 5 回
  6. 5/16：第 6 回 (中間レポート課題提出期限予定)
  7. 5/23：休講
  8. 5/30：第 7 回
  9. 6/06：第 8 回 (期末レポート課題公開)
  10. 6/13
  11. 6/20：　　(期末レポート課題提出期限予定)

1.1. シラバス

当実習では以下の内容を扱う予定である（進捗に合わせ変更する可能性がある）.

1. 導入
   - C(C++) の使い方 (主に数値計算)
   - Python の使い方 (データ解析と作図)
   - 数値計算の理念
   - 桁落ち
   - 科学計算における無次元化
2. 常微分方程式の数値解法：減衰振動や調和振動子を例に
   - 微分方程式の数値積分
   - オイラー法
   - ~~蛙飛び差分法（速度ベルレ法）~~
   - ~~軌道の安定性と保存則~~
3. １粒子系のブラウン運動
   - ランジュバン方程式（確率微分方程式）
   - 正規乱数の生成法
   - オイラー・丸山法
   - 時間平均とアンサンブル平均
   - 拡散係数の計算
4. 多粒子系のブラウン運動
   - 相互作用力の計算方法
   - 非平衡系のシミュレーション：相分離現象を例に
5. 多粒子系の分子動力学シミュレーション
   - 位置ベルレ法と速度ベルレ法
   - 多粒子系における保存則（運動量・エネルギー・角運動量）
6. モンテカルロ法
   - 統計力学の復習
   - マルコフ連鎖モンテカルロ法
   - メトロポリス判定法

目次

2.1. C(C++) の環境構築 (前回までの作業)

- C(C++) 環境構築について：
  - **Windows:** MinGW(Minimalist GNU for Windows) のインストール [R 研浅谷氏作成マニュアル（**4/13** 修正）：[参考リンク]
  - どうしても上手くいかない場合は，VScode など，その他ソフトウェアでプログラムを動かしてもよい．

目次

3. 第 1 回自主課題解説

π の収束性（モンテカルロシミュレーション）

(1) リスト 2 のサンプルプログラムに関して，乱数の発生回数を変化させた際の円周率の収束性について考える．乱数の発生回数 n に対する数値計算上の円周率 $\pi(n)$ を様々とることで，横軸に n，縦軸に $\pi(n)$ の関係を作図せよ．また，理論値 π との誤差 $\delta(n) = |\pi(n) - \pi|$ が n に対してどのように変化するか図示し評価せよ．

(2) （発展問題）大規模な数値計算においては時として，擬似乱数生成器 rand() では乱数周期が短いため不具合が生じることがある．一方，高品質な擬似乱数生成器として松本眞氏により開発されたメルセンヌツイスタが有名である [参考リンク][1]．いま，リスト 2 のサンプルプログラムを，メルセンヌツイスタを用いたものに書き換えよ [参考リンク][2]．指定 GitHub リポジトリ [リンク][3] に，メルセンヌツイスタのヘッダファイルを置いたのでこれを用いてもよい．

3. 第 1 回自主課題解説 (2)

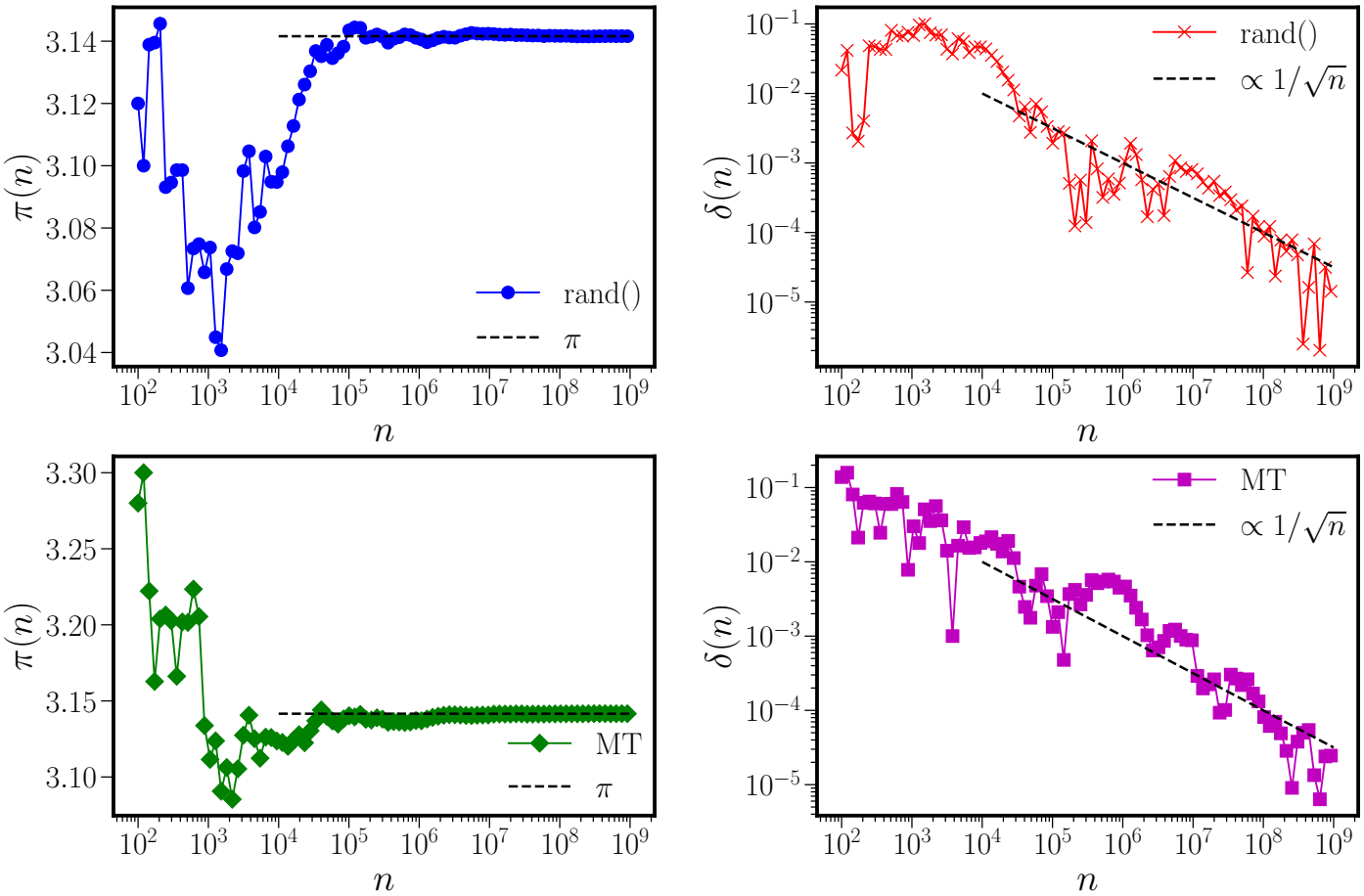

図 1: 標準乱数生成器 rand()（上段），メルセンヌツイスタ（下段）によって計算した $\pi(n)$ と $\delta(n)$.

- 以下，C(C++) を用いた計算プログラムである．

リスト 1: 第１回自主課題　主計算用サンプルプログラム “pi_error.cpp”．以下の GitHub リポジトリより取得可能．　[リンク]

```
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  #include "MT.h"// for MT
9  using namespace std;
10 int main(void){
11   int i, count = 0, max = 1e+9, out=1e+2;
12   double x,y,z,pi;
13   char fname[128];
14   ofstream file;
15   srand(time(NULL));
16   sprintf(fname,"pi-error.dat");
17   file.open(fname);
18   for(i=0;i<max;i++){
19     x = (double)rand()/RAND_MAX;
20     y = (double)rand()/RAND_MAX;
21     z = x*x + y*y;
22     if(z<1)
23       count++;
24     if(i>=(int)out){
25       pi=(double)count /i*4;
26       file<<setprecision(16)<<i<<" "<<pi<<" "<<abs(pi-M_PI)<<endl;
```

```
27        cout<<setprecision(16)<<i<<" "<<pi<<" "<<abs(pi-M_PI)<<endl;
28        out*=1.2;
29      }
30    }
31    file.close();
32    out=1e+2;
33    count=0;
34
35    sprintf(fname,"pi-error-MT.dat");
36    file.open(fname);
37    init_genrand(time(NULL));
38    for(i=0;i<max;i++){
39      x = genrand_res53();
40      y = genrand_res53();
41      z = x*x + y*y;
42      if(z<1)
43        count++;
44      if(i>=(int)out){
45        pi=(double)count /i*4;
46        file<<setprecision(16)<<i<<" "<<pi<<" "<<abs(pi-M_PI)<<endl;;
47        cout<<setprecision(16)<<i<<" "<<pi<<" "<<abs(pi-M_PI)<<endl;
48        out*=1.2;
49      }
50    }
51    file.close();
52    return 0;
53  }
```

- 以下，図示の際のサンプルプログラムである．

リスト 2: Python サンプルプログラム “pi_error.py” GitHub リポジトリより取得可能．[リンク] jupyter notebook ファイル “2022_simulation.ipynb”にも同等のプログラムを掲載．

```
1  import matplotlib
2  import matplotlib.pyplot as plt
3  %matplotlib inline
4  #図の解像度が上がる
5  %config InlineBackend.figure_format = 'retina'
6  import numpy as np
7  #Tex フォント
8  plt.rcParams["text.usetex"] =True
9  #図全体のサイズやアスペクト比を変える
10 fig = plt.figure(figsize=(18,12))
11 #複数の図を並べる時ここを変える
12  ############################
13 plt.subplot(221)
14 ax1 = fig.add_subplot(221)
15 #各自ファイルのパスを変えること
16 i, pi,error  = np.loadtxt("./Documents/GitHub/2022-simulation-training-
       main/Lecture2/pi-error.dat", comments='#', unpack=True)
17 plt.plot(i, pi, "o-",markersize=10,color="b",label=r"rand()")
18 plt.xscale('log')
19 ###Drawing a line ######
20 x= np.linspace(1e4, 1e9, 100)
21 y= np.pi+0*x
22 plt.plot(x, y, "--",markersize=3,linewidth = 2.0, color="k",label=r"$\pi$
       ")
23 #########
```

```
24  #図の書式設定
25  plt.tick_params(which='major',width = 1, length = 10)
26  plt.tick_params(which='minor',width = 1, length = 5)
27  ax1.spines['top'].set_linewidth(3)
28  ax1.spines['bottom'].set_linewidth(3)
29  ax1.spines['left'].set_linewidth(3)
30  ax1.spines['right'].set_linewidth(3)
31  plt.xlabel(r"$n$",color='k', size=30)
32  plt.ylabel(r"$\pi(n)$",color='k', size=30)
33  plt.xticks(color='k', size=25)
34  plt.yticks(color='k', size=25)
35  #図の凡例の有無や位置，サイズを調整
36  plt.legend(ncol=1, loc=4, borderaxespad=0, fontsize=25,frameon=False)
37  #各グラフのアスペクト比をにする1:1
38  ##################################
39
40   ############################
41  plt.subplot(222)
42  ax2 = fig.add_subplot(222)
43  #各自ファイルのパスを変えること
44  i, pi,error  = np.loadtxt("./Documents/GitHub/2022-simulation-training-
        main/Lecture2/pi-error.dat", comments='#', unpack=True)
45  plt.plot(i, error, "x-",markersize=10,color="r",label=r"rand()")
46  plt.xscale('log')
47  plt.yscale('log')
48
49  ###Drawing a line ######
50  x= np.linspace(1e4, 1e9, 100)
51  y=1/x**0.5
52  plt.plot(x, y, "--",markersize=3,linewidth = 2.0, color="k",label=r"$\
```

```
     propto 1/\sqrt{n}$")
#########

#図の書式設定
plt.tick_params(which='major',width = 1, length = 10)
plt.tick_params(which='minor',width = 1, length = 5)
ax2.spines['top'].set_linewidth(3)
ax2.spines['bottom'].set_linewidth(3)
ax2.spines['left'].set_linewidth(3)
ax2.spines['right'].set_linewidth(3)
plt.xlabel(r"$n$",color='k', size=30)
plt.ylabel(r"$\delta(n)$",color='k', size=30)
plt.xticks(color='k', size=25)
plt.yticks(color='k', size=25)
#図の凡例の有無や位置，サイズを調整
plt.legend(ncol=1, loc=1, borderaxespad=0, fontsize=25,frameon=False)
##################################

 ############################
plt.subplot(223)
ax3 = fig.add_subplot(223)
#各自ファイルのパスを変えること
i, pi,error  = np.loadtxt("./Documents/GitHub/2022-simulation-training-
     main/Lecture2/pi-error-MT.dat", comments='#', unpack=True)
plt.plot(i, pi, "D-",markersize=10,color="g",label=r"MT")
plt.xscale('log')
###Drawing a line ######
x= np.linspace(1e4, 1e9, 100)
y= np.pi+0*x
plt.plot(x, y, "--",markersize=3,linewidth = 2.0, color="k",label=r"$\pi$
```

```
      ")
81  #########
82
83  #図の書式設定
84  plt.tick_params(which='major',width = 1, length = 10)
85  plt.tick_params(which='minor',width = 1, length = 5)
86  ax3.spines['top'].set_linewidth(3)
87  ax3.spines['bottom'].set_linewidth(3)
88  ax3.spines['left'].set_linewidth(3)
89  ax3.spines['right'].set_linewidth(3)
90  plt.xlabel(r"$n$",color='k', size=30)
91  plt.ylabel(r"$\pi(n)$",color='k', size=30)
92  plt.xticks(color='k', size=25)
93  plt.yticks(color='k', size=25)
94  #図の凡例の有無や位置，サイズを調整
95  plt.legend(ncol=1, loc=4, borderaxespad=0, fontsize=25,frameon=False)
96  ##################################
97
98  ############################
99  plt.subplot(224)
100 ax4 = fig.add_subplot(224)
101 #各自ファイルのパスを変えること
102 i, pi,error  = np.loadtxt("./Documents/GitHub/2022-simulation-training-
      main/Lecture2/pi-error-MT.dat", comments='#', unpack=True)
103 plt.plot(i, error, "s-",markersize=10,color="m",label=r"MT")
104
105 ###Drawing a line ######
106 x= np.linspace(1e4, 1e9, 100)
107 y=1/x**0.5
108 plt.plot(x, y, "--",markersize=3,linewidth = 2.0, color="k",label=r"$\
```

```
     propto 1/\sqrt{n}$")
#########
plt.xscale('log')
plt.yscale('log')
#図の書式設定
plt.tick_params(which='major',width = 1, length = 10)
plt.tick_params(which='minor',width = 1, length = 5)
ax4.spines['top'].set_linewidth(3)
ax4.spines['bottom'].set_linewidth(3)
ax4.spines['left'].set_linewidth(3)
ax4.spines['right'].set_linewidth(3)
plt.xlabel(r"$n$",color='k', size=30)
plt.ylabel(r"$\delta(n)$",color='k', size=30)
plt.xticks(color='k', size=25)
plt.yticks(color='k', size=25)
#図の凡例の有無や位置，サイズを調整
plt.legend(ncol=1, loc=1, borderaxespad=0, fontsize=25,frameon=False)
##################################

#図のマージン設定
plt.subplots_adjust(wspace=0.3, hspace=0.25)

#各自ファイルのパスを変えること.
plt.savefig('./Documents/GitHub/2022-simulation-training-main/Lecture2/pi
    -error.png')
plt.savefig('./Documents/GitHub/2022-simulation-training-main/Lecture2/pi
    -error.pdf')
```

目次

4. 数値計算の基礎

以下，今回の本題に入る．ここでは，数値計算をするにあたり抑えておくべき基本事項をまとめる．特に，

- 桁落ちと浮動小数点
- 離散化
- 無次元化

について説明する．

4.1. 桁落ちと浮動小数点数

本節では，数値的総和計算においてしばしば問題となる桁落ちと，桁落ちが生じる原因となる浮動小数点数 (floating point number) について導入する．

桁落ち (loss of significant digits)/丸め誤差 (round off error)

絶対値が大きな値と小さな値とを加えた場合，小さい方の数値がもつ情報が失われる． たとえば， float 型の精度は約 6-7 桁であるから，float 型の数値 77777.7 と 1.23456 とを加えると， 結果も 6-7 桁であるから，77778.9 となる．すなわち，1.23456 がもつ情報のうちの下 4 桁の情報が失われている．

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

- 次に桁落ちの原因となる浮動小数点数の性質について考える．
- 浮動小数点数とは，コンピュータにおける数値の表現形式の一つで、数値を桁の並びを表す仮数部と小数点の位置を表す指数部に分割して表現する方式である．

4.1. 桁落ちと浮動小数点数 (2)

- 小数点以下の値を含む数値の表現法として最も広く利用されている [4].

浮動小数点数 (floating point number) と計算機イプシロン

単精度浮動小数点数 (float 型): 32-bit, 倍精度浮動小数点数 (double 型) 64-bit 浮動小数の構成–2 進数の場合（基数 2）:

符号部　正なら 0　負なら 1　（1-bit）
仮数部　（整数 1 桁（基数未満）＋小数）の絶対値　（float: 23-bit, double 52-bit）
指数部　符号付き整数（float: 8-bit, double 11-bit）　オフセットバイナリー値 (負の値を表現するためビット数の原点をバイアス値だけ下駄をはかせる): float の場合 $2^7 - 1 = 127$, double の場合 $2^{10} - 1 = 1023$

計算機エプシロン **(**仮数の最小単位**)**：　double の場合，$2^{-53} = 1.110 \times 10^{-16}$.
IEEE（アイトリプルイー）**754** 浮動小数点：仮数部 1.xxx　ただし 1 は省略（けちビット or hidden bit という）

4.2. 離散化

次に本節では，微積分の数値計算において必要になる離散化（差分化）の方法について説明する．

- ここでは例として，簡単な微分方程式を数値的に解いてみる．
- いま，直径 a[m] 質量 m[kg] の球が，抵抗係数 ζ[kg/s] の一次元の流体中を初速度 v_0[m/s] で運動する場合を考えよう．まず球の運動方程式は

$$m\frac{\mathrm{d}v(t)}{\mathrm{d}t} = -\zeta v(t) \tag{2}$$

  である．
- 数値計算でこの運動方程式を解くためには，これを離散化し，数値積分する必要がある．
- 数値積分の精度を上げることは，問題によっては非常に重要になるが，まずは精度が低いが非常に簡便である Euler 法を紹介する．
- まず，関数 $f(t)$ の時間に関する一階微分は厳密に

$$\frac{\mathrm{d}f(t)}{\mathrm{d}t} = \lim_{\Delta t\to 0}\frac{f(t+\Delta t)-f(t)}{\Delta t} \tag{3}$$

  である．

4.2. 離散化 (2)

- 一方，数値計算では、$\Delta t \to 0$ の極限を厳密にとることはできず，各時間ステップで Δt にできるだけ小さな値を当てはめ，以下のように近似計算を行う．特に，以下方法を Euler 法 と呼ぶ.

$$\boxed{\frac{\mathrm{d}f(t)}{\mathrm{d}t} = \frac{f(t+\Delta t) - f(t)}{\Delta t} + \mathcal{O}((\Delta t))} \tag{4}$$

- Euler 法は厳密な微分と比較して Δt のオーダーの誤差を累積するため，問題によっては機能しない離散手法である．
- 一方，今回扱う問題では，保存系ではないので，ある程度の精度を得ることができる（調和振動ではダメ）．
- Euler 法を用いることで式 (2) は

$$m\frac{v(t+\Delta t) - v(t)}{\Delta t} = -\zeta v(t). \tag{5}$$

と離散化することができ，運動方程式は

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

となる．

4.2. 離散化 (3)

- この手の減衰型の微分方程式の数値積分においては Euler 法で十分な精度が得られるが調和振動のように減衰項を含まない系においては，エネルギーや運動量などの保存則を厳密に満たす必要があり，このような場合は速度 Verlet 法や Runge-Kutta 法など高次の数値積分を要する．

4.3. 無次元化

ここでは，物理変数（定数）の無次元化について扱う．

- 計算機が扱うことが出来る数は基本的に無次元数である．
- そのため，物理次元をもつ方程式を計算機で計算させるためには，物理的単位と無次元数に分ける必要がある．
- そこで，前節で扱った微分方程式を計算機で解くために，無次元化する．
- 　ここでは，まず時間を $t \to t_0\tilde{t}$，速度 $v = v_0\tilde{t} = \frac{a}{t_0}\tilde{t}$ と置こう．
- すると，式 (6) は，

$$\begin{aligned} \frac{a}{t_0}\tilde{v}(\tilde{t}+\tilde{\Delta t}) &= \frac{a}{t_0}\tilde{v}(\tilde{t}) - \frac{\zeta a \Delta\tilde{t}}{m}\tilde{v}(\tilde{t}) \\ \tilde{v}(\tilde{t}+\tilde{\Delta t}) &= \tilde{v}(\tilde{t}) - \frac{\zeta t_0}{m}\tilde{v}(\tilde{t})\Delta\tilde{t} \end{aligned} \tag{7}$$

となる．

- ここで，$\frac{\zeta t_0}{m}$ は無次元パラメータであり，基本的にどんな数字を当てはめることも原理的には可能であるが，ここでは簡単のために 1 と置こう．この様にするためには，時間の単位を

$$\boxed{t_0 = \frac{m}{\zeta}} \tag{8}$$

と選んであげればよいことが分かる．この時間スケールは，速度減衰の緩和時間となる！

4.3. 無次元化 (2)

- t_0 の選び方は任意であるが，これを上手く選ばないとパラメータ $\frac{\zeta t_0}{m}$ の値が大きすぎたり小さすぎたりすることになり桁落ちの原因となるので注意せよ．
- さて，$\frac{\zeta t_0}{m} = 1$ として得られた差分方程式は

$$\boxed{\tilde{v}(\tilde{t} + \Delta\tilde{t}) = \tilde{v}(\tilde{t}) - \tilde{v}(\tilde{t})\Delta\tilde{t}} \tag{9}$$

  という極めて単純なものとなる．
- これを数値的に解くことは，$\tilde{v}(\tilde{t})$ に関する漸化式を解くことである．

目次

5. 第 2 回自主課題

減衰運動の数値計算と解析計算の比較

１次元中を運動する溶媒中の粒子の運動を考える．この粒子の運動方程式は

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

であり，時刻 $t = 0$ において速度 $10a\zeta/m$ の速さで運動しているとする. この時，以下の各問に答えよ．なお，無次元化に際し，長さの単位は a, 時間の単位は $t_0 = m/\zeta$ とおくこと．

(1) この微分運動方程式の解析解を求めよ．さらに問題文中にて指定した単位系を用いることで無次元化せよ．

(2) この粒子の速度の時間発展 (時間区間 $[0, 10^4 t_0]$) を数値的に計算し解析解と比較せよ．ただし運動方程式の離散化は簡単のため Euler 法を用いるものとし，離散化における時間分解能を $\Delta t/t_0 = 0.1, 0.01, 0.001$ と変えたものを比較してみよ．

目次

6.1. 端末上でよく使うコマンド集（Unix 系および Windows）

端末上で用いるコマンドについて，纏めて欲しいというリクエストがあったので，以下に示す．

- Unix 環境よく使うコマンド [5]

リスト 3: Unix 環境よく使うコマンド：

```
1
2  pwd 現在ディレクトリのフルパス表示
3  cd ディレクトリ移動
4  cd/ ホームディレクトリ移動
5  mkdir 新規ディレクトリを作成
6  rm -r 指定ディレクトリを中のファイルごと削除
7  cp -r (dirA) (dirB)   ディレクトリのコピー
8  mv (file) (dir)   ファイルをディレクトリへ移動
9  ls   現在ディレクトリのファイル一覧表示
10 cat (file)   ファイルの内容表示
11 rm (file)   ファイルを削除
12 cp (fileA) (fileB)   ファイルコピー
13 mv (fileA) (fileB)   ファイル名変更
14 touch (file)   空のファイルを作成
15 find (dir)   指定ディレクトリ以下のファイルを列挙
16 more (file)   ファイルの内容表示（ページごとに止まる）
17 diff (fileA) (fileB)   ファイル間の差分（変更点）を表示
18 history コマンドの履歴リストを表示
```

6.1. 端末上でよく使うコマンド集（Unix 系および Windows） (2)

- Windows 環境よく使うコマンド [6]：

リスト 4: Windows 環境よく使うコマンド：

```
1  dir 現在ディレクトリのファイル一覧表示
2  cd ディレクトリへ移動
3  mkdir フォルダーの作成
4  del(file)  ファイル削除
5  ren   (fileA) (fileB)   ファイル名変更
6  copy (fileA) (fileB)   ファイル名コピー
7  move  (fileA) (fileB/dirC)   ファイルの移動、もしくはディレクトリ名の変更
8  type テキストファイルの内容を表示する。
9  find ファイル内やコマンドの出力結果に含まれる文字列を検索する。
10 where ファイルの場所を表示する
```

目次

参考文献・ウェブサイト

[1] Makoto Matsumoto.
Mersenne Twister: A random number generator (since 1997/10).

[2] Takahiro Ohmi.
C 言語による乱数生成.

[3] Takeshi Kawasaki.
2022-simulation-training (GitHub), April 2022.
original-date: 2021-08-18T02:15:01Z.

[4] 浮動小数点数とは - IT 用語辞典.

[5] Takayuki Omori.
UNIX コマンド集, 2021.

[6] ITSakura.
Windows コマンドプロンプトのコマンド一覧 | ITSakura, 2020.