# Data Science Course Group (Graduate): Simulation Exercise
# Lecture 3  Lecture Material

Instructor: Takeshi Kawasaki

Nagoya University, Graduate School of Science, Department of Physics, Non-equilibrium Physics Laboratory (R Lab)

Last update: August 23, 2024

## Contents

## Contents

## 1. Lecture Schedule

- This exercise will be conducted during the first half of the spring semester.
- Lecture materials will be uploaded by 11:00 AM on the scheduled lecture day.
- Schedule
    1. 4/15: Lecture 1
    2. 4/22: Lecture 2
    3. **4/30 (Tue): Lecture 3**
    4. 5/13: Lecture 4 (Midterm Report Assignment Released)
    5. 5/20: Lecture 5
    6. 5/27: Lecture 6 (Midterm Report Submission Deadline)
    7. 5/29: Lecture 7
    8. **6/03: No Class**
    9. 6/10: Lecture 8 (Final Report Assignment Released)
    10. 6/17: Lecture 9 (Makeup Class)

## 1.1. Syllabus

The following topics will be covered in this exercise (subject to change based on progress):

1. Introduction
   - Using C(C++) (primarily for numerical calculations)
   - Using Python (for data analysis and plotting)
   - Principles of numerical calculations
   - Loss of significance
   - Dimensional analysis in scientific calculations
2. Numerical solutions to ordinary differential equations: Examples with damped oscillations and harmonic oscillators
   - Numerical integration of differential equations
   - Stability and conservation laws in orbits
3. Brownian motion in single-particle systems
   - Langevin equation (stochastic differential equations)
   - Generation of normal random numbers
   - Euler-Maruyama method
   - Time averages and ensemble averages
4. Brownian motion in multi-particle systems
   - Calculation of interaction forces
   - Non-equilibrium simulations: Example of phase separation phenomena
5. Molecular dynamics simulations of multi-particle systems
   - Position Verlet and velocity Verlet methods
   - Conservation laws in multi-particle systems
6. Monte Carlo method
   - Review of statistical mechanics
   - Markov Chain Monte Carlo method
   - Metropolis criterion

## Contents

## 2. Explanation of the Second Assignment

### Comparison of Numerical and Analytical Calculations for Damped Motion

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

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

where at time $t = 0$, the particle is moving with a speed of $10a\zeta/m$. Answer the following questions. For non-dimensionalization, assume the units of length and time are $a$ and $t_0 = m/\zeta$, respectively.

(1) Derive the analytical solution for this differential equation. Furthermore, non-dimensionalize it using the units specified in the problem statement.

(2) Numerically compute the time evolution of the particle's velocity and compare it with the analytical solution. Use the Euler method for discretizing the equation of motion and compare the results for different time resolutions $\Delta t/t_0 = 0.1, 0.01, 0.001$.

[Explanation]

## 2. Explanation of the Second Assignment (2)

(1) The analytical solution for the differential equation is

$$v(t) = 10\frac{a\zeta}{m}\exp(-t\zeta/m) \tag{1}$$

Since the numerically obtained solution is in non-dimensional form, let us also non-dimensionalize this analytical solution. Dividing both sides of the equation by $\frac{a\zeta}{m}$ gives:

$$v(t)\frac{m}{a\zeta} = 10\exp(-t\zeta/m) \tag{2}$$

Given that $\tilde{v}(t) = v(t)\frac{m}{a\zeta}$ and $\tilde{t} = t\zeta/m$, we obtain:

$$\boxed{\tilde{v}(t) = 10\exp(-\tilde{t})} \tag{3}$$

Thus, the solution has been non-dimensionalized. The comparison should now be made between this solution and the numerical one.

## 2. Explanation of the Second Assignment (3)

(2) The main calculation program for the numerical computation is listed in ”damping.cpp”. It is available for download from the following GitHub repository: **[Link]**.

- (Point) The calculation is set to terminate when the velocity becomes small enough to be considered zero based on the machine epsilon value learned in the previous Lecture.
- (Point) By encapsulating the main calculations in a function (subroutine), the main script is reduced to just four lines. This allows for convenient initialization of variables within the function.

- The Python script for plotting the results is compiled in ”velo_plot.py” and is also available in the GitHub repository linked above.
  - When plotting, the use of for loops and if statements allowed the code to be streamlined, especially when arranging multiple graphs compared to the previous Lecture.

## 2. Explanation of the Second Assignment (4)

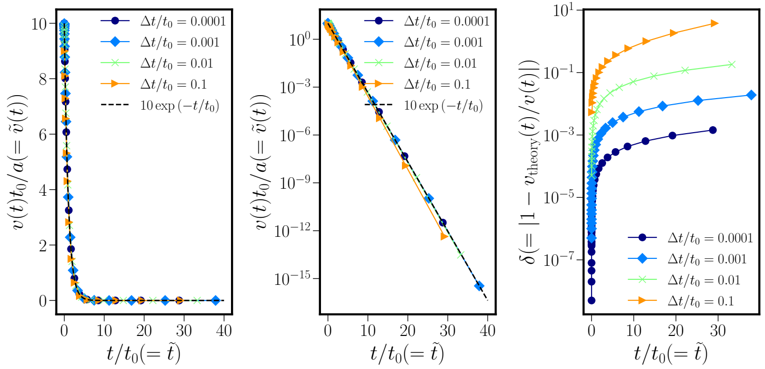

図 1: Time evolution of non-dimensionalized velocity for different time steps (time resolutions): (left) Linear plot of velocity, (middle) Semi-logarithmic plot of velocity, (right) Time evolution of velocity error.

## 2. Explanation of the Second Assignment (5)

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

リスト 1: Sample Program for Main Calculation in Assignment 2 “damping.cpp”. Available in the following GitHub repository: **[Link]**

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h> // for mathmatical functions
#include <iostream> // for input or output, i.e., std::cout
#include <fstream> // for  std::ofstream file
#include <cfloat> // for DBL_EPSILON, a.k.a. numerical epsilon
//using namespace std;

int damp(double dt){
  char filename[128];
  std::ofstream file;
  int i=0;
  double v=10.,out=1.;
  sprintf(filename,"velo_%.4f.dat",dt);
  file.open(filename);
```

## 2. Explanation of the Second Assignment (6)

```
  while(v > DBL_EPSILON){ //continue while v > 2.22044604925031e-16.
    v-=v*dt;  // as same as v = v - v*dt
    i++;      // as same as i += 1 or i =i + 1;
    if((double)i >= out){
      file << (double)i*dt << "    " << v <<std::endl;
      std::cout << (double)i*dt << "    " << v <<std::endl;
      out*=1.5;  // as same as out = out * 1.5
    }
  }
  file.close();  // should be closed when all input process has been finised.
  return 0;
}

int main(){
  double dt;
  for(dt=1.e-4;dt<=1.e-1;dt*=10.)
    damp(dt);
  return 0;
}
```

## 2. Explanation of the Second Assignment (7)

- Below is a sample program for plotting.

リスト 2: Python Sample Program “velo_plot.py”. Available in the following GitHub repository: **[Link]**. An equivalent program is also available in the Jupyter notebook file “_jupyter.ipynb”.

```


\item Below is the sample plotting program.
\begin{lstlisting}[basicstyle=\ttfamily\small, breaklines=true, frame=single,caption=
     Python Sample Program ‘‘velo\_plot.py". Available in the following GitHub repository:
      \href{https://github.com/TakeshiKawasaki/2024-simulation-tutorial/tree/main}{\TK{\
     underbar{[Link]}}}. An equivalent program is also available in the Jupyter notebook
     file ‘‘\_jupyter.ipynb".]

import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
import matplotlib.cm as cm  # colormap
import numpy as np
#plt.rcParams["text.usetex"] =True
plt.rcParams['font.family'] = 'Arial' # Set the font family
plt.rcParams["font.size"] = 25

```

## 2. Explanation of the Second Assignment (8)

```python
fig = plt.figure(figsize=(18,8))

dt=[0.0001,0.0010,0.0100,0.1000]
symbol =['o-','D-','x-','>-']

for j in range (1,4):
    #ax = fig.add_subplot("13{}".format(j))
    ax = fig.add_subplot(1,3,j)
    if(j>1):
        plt.yscale('log')

    for i in range (0,4):
        print(i,symbol[i],dt[i])  # For checking the arrays
        time,vel= np.loadtxt("./Lecture3/velo_{:.4f}.dat".format(dt[i]), comments='#',
             unpack=True)
        if(j!=3):
            plt.plot(time,vel, "{}".format(symbol[i]) ,markersize=10,color=cm.jet(i/4),
                 label=r"$\Delta t/t_0={}$".format(dt[i]))
        else:
            plt.plot(time,np.abs(1 -10.*np.exp(-time)/vel), "{}".format(symbol[i]) ,
                 markersize=10,color=cm.jet(i/4),label=r"$\Delta t/t_0={}$".format(dt[i]))



    ### Drawing a line ######
```

## 2. Explanation of the Second Assignment (9)

```
    if(j<3):
        time= np.linspace(1e-4, 4e1, 1000)
        vel= 10.*np.exp(-time)
        plt.plot(time,vel, "--",markersize=3,linewidth = 2.0, color="k",label=r"$10\exp{(-
            t/t_0)}$")
    #########
    # Formatting the plot
    plt.tick_params(which='major',width = 1, length = 10)
    plt.tick_params(which='minor',width = 1, length = 5)
    ax.spines['top'].set_linewidth(3)
    ax.spines['bottom'].set_linewidth(3)
    ax.spines['left'].set_linewidth(3)
    ax.spines['right'].set_linewidth(3)
    plt.xlabel(r"$t/t_0(=\tilde{t})$",color='k', size=30)
    if(j!=3)  :
        plt.ylabel(r"$v(t)t_0/a(=\tilde{v}(t))$",color='k', size=30)
    else:
        plt.ylabel(r"$\delta(=|1-v_{\rm theory}(t)/v(t)|)$",color='k', size=30)
    if(j<3):
        plt.legend(ncol=1, loc=1, borderaxespad=0, fontsize=20,frameon=False)
    else:
        plt.legend(ncol=1, loc=4, borderaxespad=0, fontsize=20,frameon=False)

################################
```

## 2. Explanation of the Second Assignment (10)

```
# Setting the plot margins
plt.subplots_adjust(wspace=0.5, hspace=0.25)
# Change the file path as needed.
plt.savefig('./Lecture3/velo_dt_lin_log.png')
plt.savefig('./Lecture3/velo_dt_lin_log.pdf')
plt.show()
```

## Contents

## 3. Assignment 3: Numerical Calculation of Damped Oscillations, Overdamping, and Critical Damping

Assignment 3

## 3. Assignment 3: Numerical Calculation of Damped Oscillations, Overdamping, and Critical Damping (2)

Consider a small ball with mass $m$, which can be treated as a point mass, attached to a spring with a spring constant $k$ on a smooth horizontal surface, where one end of the spring is fixed. Additionally, the ball is subjected to a viscous drag force with a drag coefficient $\zeta$. In this case, by setting up an appropriate coordinate system, the equation of motion for the position $x(t)$ and velocity $v(t)$ of the ball at time $t$ is given by:

$$m\frac{\mathrm{d}v(t)}{\mathrm{d}t} = -\zeta v(t) - kx(t) \tag{4}$$

Now, by varying the positive parameters $(m, \zeta, k)$, consider how the motion of the point mass changes significantly based on their relative magnitudes.
[**Questions**]

**(1)** Express analytically the conditions under which the motion of the ball exhibits **damped oscillations (underdamping), overdamping, and critical damping**, using $m$, $\zeta$, and $k$.

**(2)** Let $\tilde{x}$ and $\tilde{t}$ be dimensionless variables for length and time, respectively, and assume the relationships between the actual physical quantities given in the problem and the dimensionless variables are $x = a\tilde{x}$, $t = t_0\tilde{t}$, and $\dot{x} = v = (a/t_0)\tilde{v}$. Here, $a$ [m] and $t_0$ [s] are appropriately defined length and time scales. Furthermore, extract several physically meaningful time scales from the physical quantities given in the equation of motion. In this case, the time scale characterizing the damped motion is $t_d = \frac{m}{\zeta}$, and the time scale characterizing the oscillations of the spring is $t_s = \sqrt{m/k}$. Now, using these, nondimensionalize the equation of motion, and further discretize it using the Semi-implicit Euler method with a time step $\Delta\tilde{t}$ to derive the recurrence relations for $\tilde{v}(\tilde{t} + \Delta\tilde{t})$ and $\tilde{x}(\tilde{t} + \Delta\tilde{t})$.

**(3)** Now, if you set the time unit as $t_0 = t_d$, one of the coefficients in the recurrence relations derived in **(2)** will drop out, revealing that $\boxed{t_d/t_s}$ is the **only parameter that controls the characteristics of the motion** in this system. Therefore, based on the conditions estimated in **(1)**, substitute appropriate values for $\boxed{t_d/t_s}$ and reproduce the damped oscillations (underdamping), overdamping, and critical damping by solving the recurrence relations discussed in **(2)** numerically, and plot the graph of $x(t)$. Assume the initial conditions are $x(0) = a$ and $\dot{x}(0) = 0$. The calculation time for $t > 0$ can be set appropriately within the range where the characteristics of the motion can be observed. The size of the time step $\Delta\tilde{t}$ can also be set appropriately, provided the numerical solution converges.

[Partial Explanation]

## Contents

## 4.1. Evaluating the Convergence of the Euler Method

- For a damped differential equation:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \lambda x \tag{15}$$

When discretized using the Euler method, it can be written as:

$$x_{n+1} = (1 + \lambda \Delta t) x_n \tag{16}$$

In this case, the convergence condition for the sequence is:

$$|x_{n+1}/x_n| = |1 + \lambda \Delta t| < 1 \tag{17}$$

Thus, by choosing $\Delta t$ to satisfy:

$$\Delta t < -1/\lambda \tag{18}$$

we can ensure that the numerical calculation is stable.

- Here, if $\lambda = -1$, then $\Delta t < 1$ is stable.
- However, this alone is not sufficient; it is also common to discuss the extent of the error as $\Delta t$ varies.
- The deviation from the converged value in the Euler method systematically changes with an order of $O(\Delta t)$.
- It is necessary to adjust the time step $\Delta t$ depending on how tolerant the phenomena under observation are to errors.

## Contents

## 5.1. Common Commands for Terminal (Unix-based)

- Commonly used commands in a Unix environment:

**Commonly used Unix commands**

| | |
|---|---|
| pwd | Displays the full path of the current directory |
| cd (dir) | Changes the directory |
| cd ~/ | Moves to the home directory |
| mkdir (dir) | Creates a new directory |
| rm -r (dir) | Deletes the specified directory along with its contents |
| cp -r (dirA) (dirB) | Copies a directory |
| mv (file) (dir) | Moves a file to a directory |
| ls | Lists files in the current directory |
| cat (file) | Displays the contents of a file |
| cat (file1) (file2) > (file3) | Concatenates file1 and file2 to create file3 |
| rm (file) | Deletes a file |
| cp (fileA) (fileB) | Copies a file |
| mv (fileA) (fileB) | Renames a file |
| touch (file) | Creates a new empty file (originally used to update the timestamp) |
| find (dir) | Lists files under the specified directory |
| more (file) | Displays file contents (pauses per page) |
| diff (fileA) (fileB) | Shows the differences (changes) between two files |
| history | Displays a list of command history |