Zadania z symulacji komputerowych w fizyce

Zespołowe projekty studenckie 2

Opiekun: prof. dr hab. Jakub Tworzydło

Semestr zimowy 2022/2023

Maria Kalinowska
Daria Matczak
Agnieszka Motyka
Karolina Pachocka

Spis treści

The codes that are the solutions to the tasks are available from the authors and can be made available if reasonably requested.
For this purpose, please contact *prof. dr hab. Jakub Tworzydło* at **Jakub.Tworzydlo@fuw.edu.pl**

Idea projektu

Celem projektu jest stworzenie zestawu ćwiczeń umożliwiających studentom skuteczne przyswojenie i praktyczne opanowanie metod symulacji komputerowych. Uczestnicy projektu samodzielnie, lub w zespole, rozwiązywali zagadnienia oparte na wybranej literaturze. Dla każdego zagadnienia zostały opracowane konieczny wstęp teoretyczny, wskazówki praktyczne oraz przykładowe wyniki symulacji.

Coconuts and Islanders

A) There are N = 300 inhabitants (natives, castaways, etc.) on the island, initially each receives C = 15 coconuts. The inhabitants play a game of paper-scissors-stone, the loser in a pair passes 1 coconut to the other person (if they have one). Pairs are chosen at random and one game corresponds to one playing pair.
Perform a simulation of T = 200000 games and plot a histogram (probability density) as a function of the number of coconuts owned. Mark on the histogram the function (approximation of the distribution for N, C >> 1)
$p(x) = 1/(1+C) * e^{-x/kT}$, where kT = C + 0.5.

B) Plot the number of people with zero coconuts as a function of the number of games (e.g., every 2500 games). Plot with a dashed line the value of p(0) * N.

Exercise prepared on the basis of: *"Coconuts and Islanders: A Statistics-First Guide to the Boltzmann Distribution"* by Brian Zhang; https://doi.org/10.48550/arXiv.1904.04669
Example solution graphs are shown in Figure 1.

Hints on writing code:

```
N = 300; C = 15

x = np.ones(N) * C

I, j = np.random.choice(N, 2)
…
x[i] += -1
…

plt.hist(x, bins='auto', density=True, alpha=0.75)
plt.plot(xarr, yarr)

p = np.random.rand()   # single random value
p_arr = np.random.rand(100)        # array of 100 random values
np.sum(x == 0)        # numer of zeros in array x
fig.set_size_inches(w, h)         # set the figure size in inches (width, height)
```

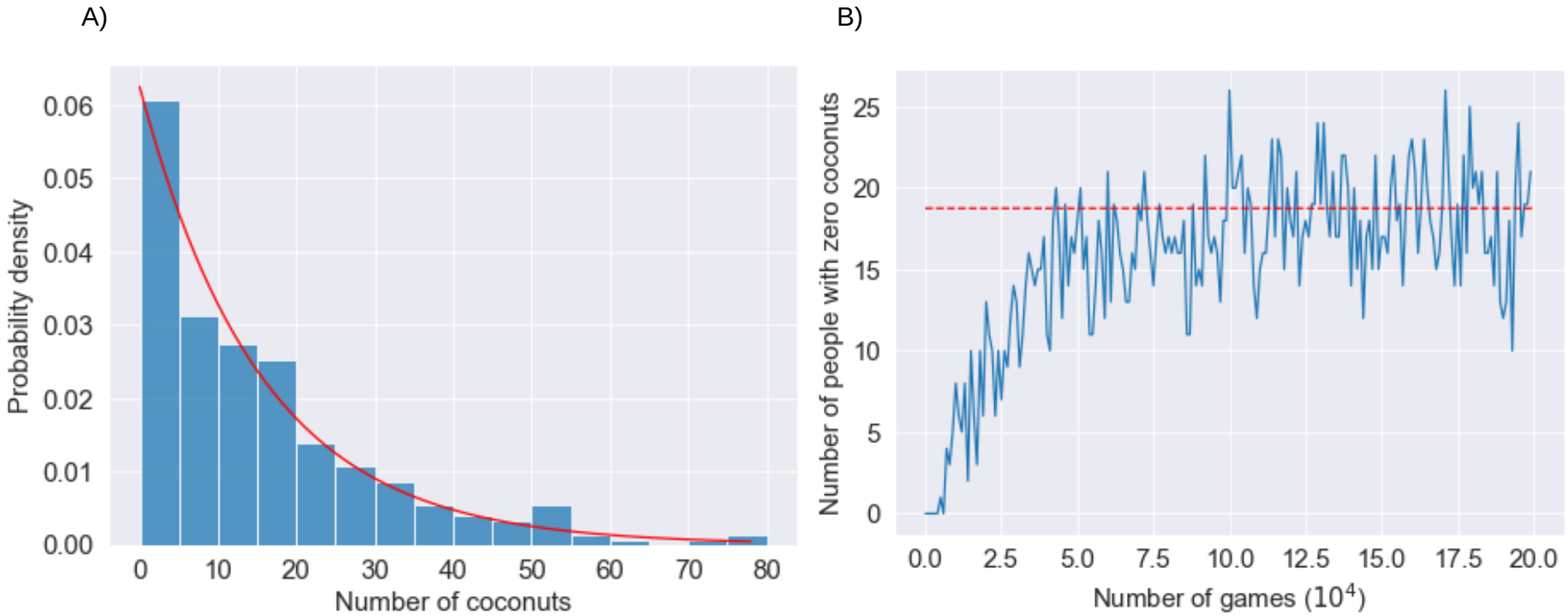


Figure 1. Example solution graphs: A) Histogram of coconut amounts, B) Plot of number of people with zero coconuts. The situation is drawn after each game.

Additional materials on Boltzmann distribution:

- Ideologically the simplest illustration of Boltzman decomposition (according to S.C. Zhang https://arxiv.org/abs/1904.04669)
- Physical model for the energy exchange between the particles of a gas we obtain, when we divide (in each collision) the sum of the possessed energy according to a random ratio
- numerous econo-physical models of wealth accumulation in the population: attempts to reconstruct the power-law distributions "Statistical mechanics, of money" https://arxiv.org/abs/cond-mat/0001432, https://arxiv.org/abs/cond-mat/0004256

Differential equations: the pendulum

Solve ordinary differential equation for a simple harmonic oscillating system using 3 methods.
Equation of a simple pendulum: $d^2\theta/dt^2 = \ddot{\theta} = -g/L\, sin(\theta)$.
Methods for solving equation:

(1) Euler

$\theta(t + \delta) = \theta(t) + \dot{\theta}(t)\,\delta,$
$\dot{\theta}(t + \delta) = \dot{\theta}(t) + \ddot{\theta}(t)\,\delta$

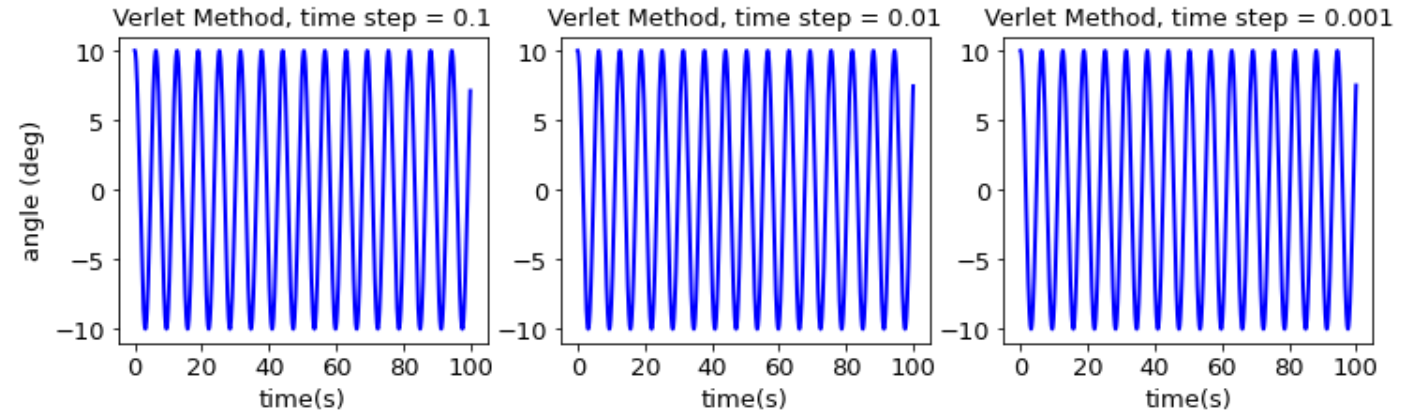
(2) Midpoint

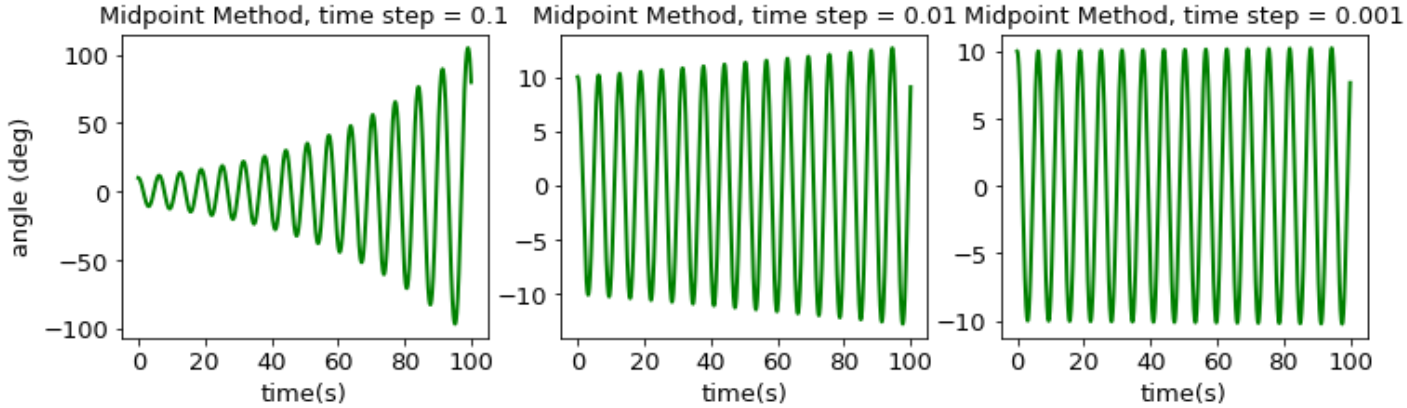
$\dot{\theta}(t + \delta) = \dot{\theta}(t) + \ddot{\theta}(t)\,\delta,$
$\theta(t + \delta) = \theta(t) + \dot{\theta}(t + \delta)\,\delta$

(3) Verlet

$\dot{\theta}(t + \delta/2) = \dot{\theta}(t) + 1/2\ \ddot{\theta}(t)\,\delta,$
$\theta(t + \delta) = \theta(t) + \dot{\theta}(t + \delta/2)\,\delta,$
$\dot{\theta}(t + \delta) = \dot{\theta}(t + \delta/2) + 1/2\ \ddot{\theta}(t + \delta)\delta.$

Plot the pendulum trajectory *θ(t)* for 3 time steps *δ = 0.1*, *0.01*, and *0.001*. Zoom in on the curve at one of the coarse points (say, *t = 1*) and visually compare the values from the three time steps. Initial conditions: *θ_0 = 2π/3*, *θ = 0*, *g = 9.8 m/s2* and *L = 1 m*. Additionally plot equation solution in phase space. Example solution graphs are shown in Figure 2 and Figure 3.





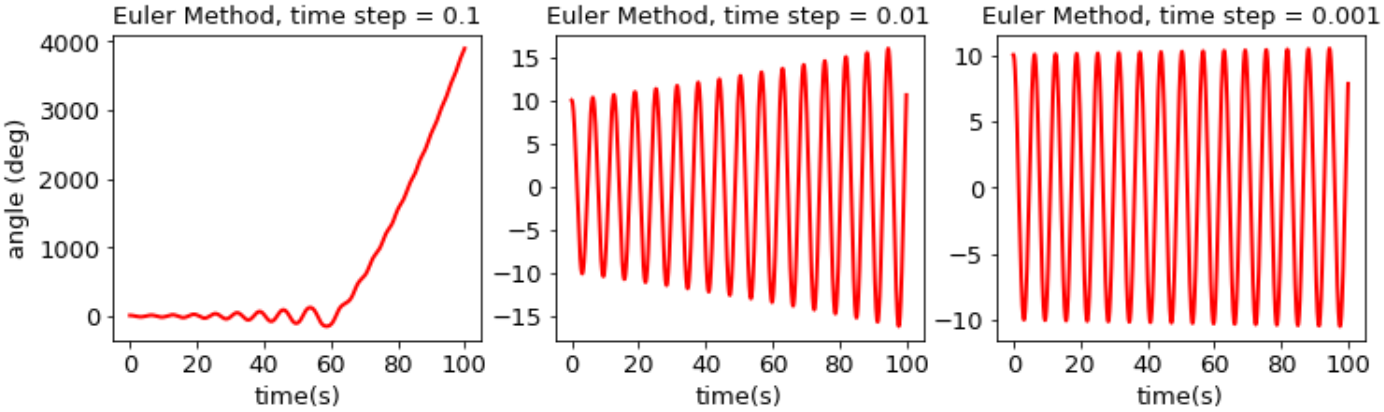
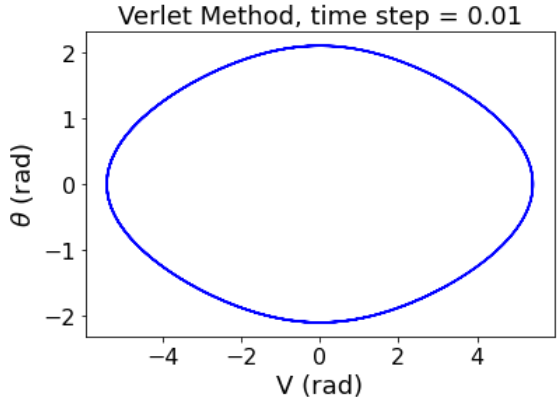
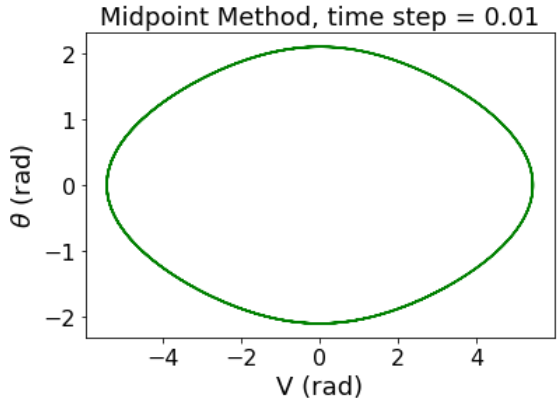


Figure 2. Example solution graphs for 3 methods and 3 time steps.





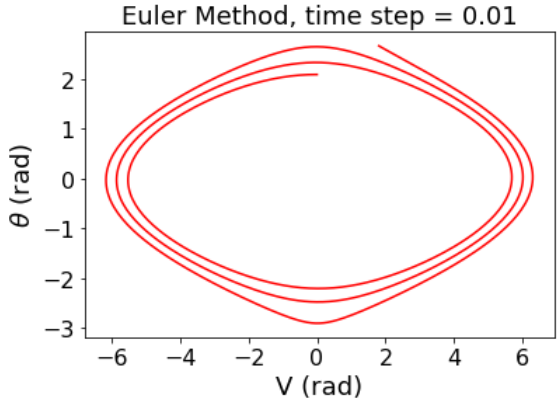


Figure 3. Example solution graphs, phase space.

Colliding disks

Create an event-driven molecular dynamic simulation of the given number of hard disks colliding in a square box of a width equal L. Disks should collide elastically with each other (pair collisions) and with the box walls (wall collisions). Pair collisions conserve momentum and wall collisions just reverse velocity component (normal to the wall). Between collisions disks should behave as free particles - simply move straight ahead. Consider disks as equal both in size and masses with radius value equal σ.

Example with four disks:

The disks should start in a linear arrangement, with three disks in a line and the fourth disk hitting the first disk at a slight angle. The initial positions and velocities of the disks are specified.
For $L = 5$ and $\sigma = 0.3$ the positions of the three standing discs are: (2.5, 0.7), (2.5, 1.3) and (2.5, 1.9).
The position of the fourth disc is set at (1, 4.75) and its velocity is set to (0.2, -0.1).
At each time step, $dt = 0.2$, the positions and velocities of the particles are updated based on their interactions with the walls and each other. Time of the simulation is $t = 20$.

Hints on writing code:

Algorithm event-disks (Event-driven molecular dynamics algorithm for hard disks in a box)

input $\{x_1, \ldots, x_N\}, \{v_1, \ldots, v_N\}, t$
$\{t_{pair}, k, l\} \leftarrow$ next pair collision
$\{t_{wall}, j\} \leftarrow$ next wall collision
$t_{next} \leftarrow \min[t_{wall}, t_{pair}]$
for m = 1, . . . , N do
 $x_m \leftarrow x_m + (t_{next} - t)v_m$
if $(t_{wall} < t_{pair})$ then
 call wall-collision(j)
else
 call pair-collision(k, l)
output $\{x_1, \ldots, x_N\}, \{v_1, \ldots, v_N\}, t_{next}$

#Algorithm pair-collision (Computing the velocities of disks k and l after an elastic collision)

input $\{x_k, x_l\}$ (particles in contact: $|x_k - x_l| = 2\sigma$)
input $\{v_k, v_l\}$
$\Delta_x \leftarrow x_k - x_l$
$\hat{e}\perp \leftarrow \Delta_x/|\Delta_x|$
$\Delta_v \leftarrow v_k - v_l$
$v_k \leftarrow v_k - \hat{e}\perp(\Delta_v \cdot \hat{e}\perp)$
v
$l \leftarrow vl + \hat{e}\perp(\Delta v \cdot \hat{e}\perp)$
output $\{v'_k, v'_l\}$

Algorithm pair-time (computing pair collision time for two particles starting at time t_0 from positions x_k and x_l, and with velocities v_k and v_l.)

input Δ_x ($\equiv x_k(t_0) - xl(t_0)$)
input Δ_v ($\equiv v_k - vl = 0$)
$Y \leftarrow (\Delta_x \cdot \Delta_v)^2 - |\Delta_v|^2(|\Delta x|^2 - 4\sigma^2)$
if (Y > 0 and $(\Delta_x \cdot \Delta_v) < 0$) then
$\{ t_{pair} \leftarrow t_0 - [(\Delta x \cdot \Delta v) + \sqrt{Y}] / \Delta^2$
else
 $t_{pair} \leftarrow \infty$
output t_{pair}

Animation can be achieved by combining multiple pictures with help of programs such as:
https://imagemagick.org/script/index.php
https://www.ffmpeg.org/

Hints on creating series of pictures:

```
def(step)
    plt.clf() # clean picture
    fig = plt.gcf() # define new one

    for p in range(…):
      a = plt.gca() # ‘get current axes’
      cir = Circle((pos[...], pos[…]), radius) # create circle
      a.add_patch(cir) # add another circle
      …
    nStr=str(step) #for saving file numer as 5 digits, starting with 0, e.g. 00324.png
    nStr=nStr.rjust(5,'0')
    …
    plt.savefig('… '+nStr+'.png')
```

Example solution pictures are shown in Figure 4.

Exercise prepared on the basis of: Krauth, Werner. “Statistical Mechanics: Algorithms and Computations.” (2006), chapter 2. Hard Disks and Spheres

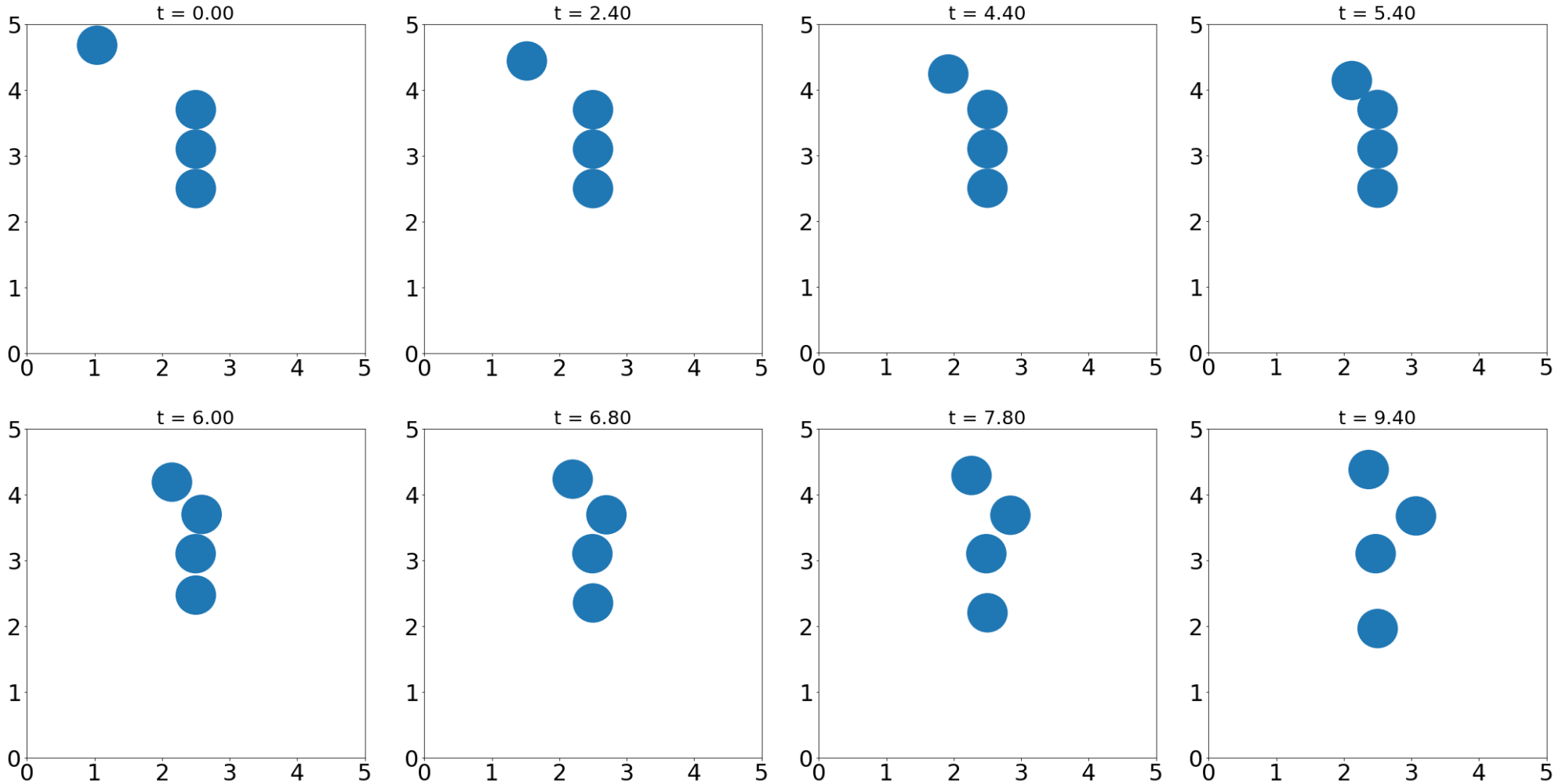


Figure 4. Pictures from different times of simulation with four disks.

Fractals: particle aggregation

Create a simple model of ballistic particle aggregation. Particles should be launched at random from the circumference of a circle of a given size, with equal probability in position and direction of motion.
Each particle in the cluster should be provided with an effective radius of aggregation λ, starting with the seed particle placed in the center. Next particles should be added to aggregate under condition that its trajectory is such that it will intersects boundary of first aggregated particle on its way. It happens when its perpendicular distance to the particles in the cluster is less than λ. In case of more than one aggregated particle being in this range, the closest one along the direction of motion should be determined.
New particle should be added to the cluster from the position of first crossing of the interaction boundary.
The procedure is visualized in Figure 5.

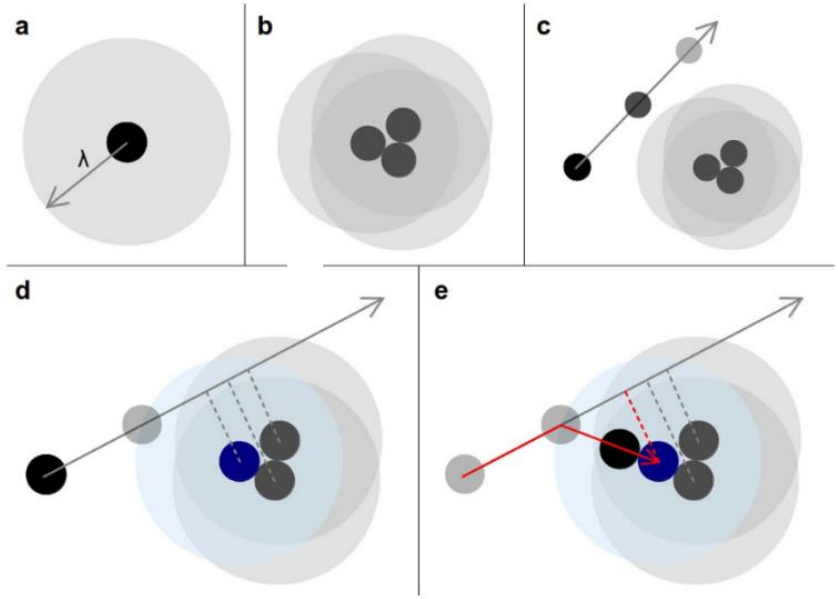


Figure 5. Implementation of the interaction. Figure sourced from: Nicolás-Carlock, J., Carrillo-Estrada, J. & Dossetti, V. Fractality à la carte: a general particle aggregation model. Sci Rep 6, 19505 (2016). https://doi.org/10.1038/srep19505

Example solution aggregate is shown in Figure 6.

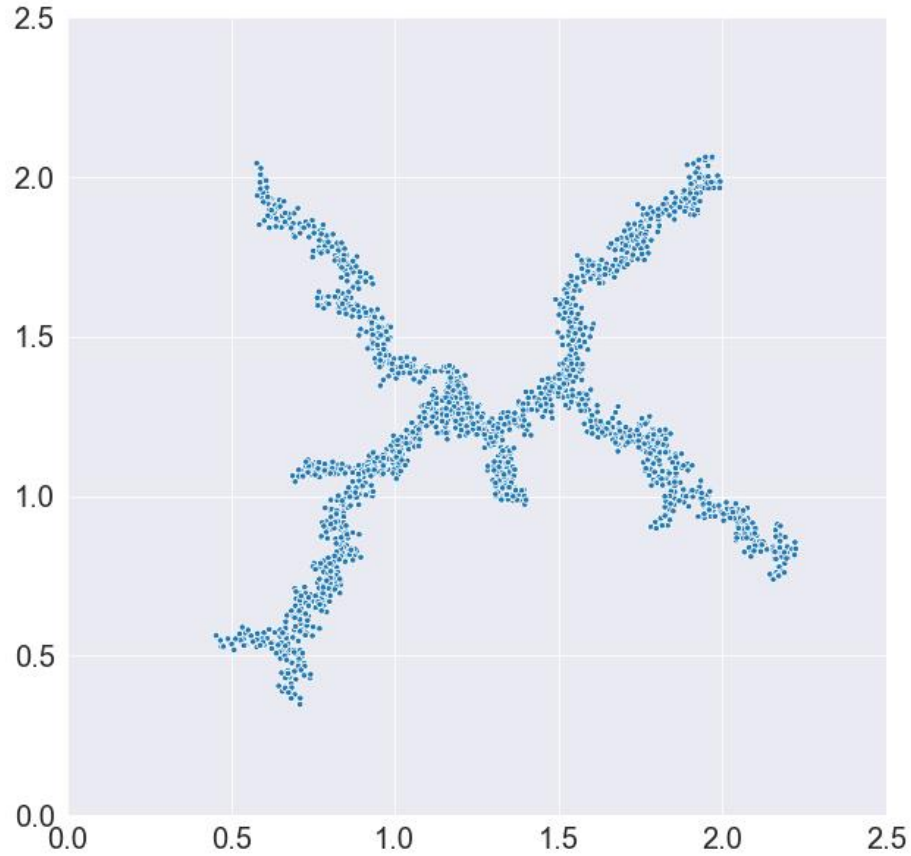


Figure 6. Example solution aggregate for 2000 aggregated particles with radius equal 0.01 and λ = 0.2.

Exercise prepared on the basis of: Nicolás-Carlock, J., Carrillo-Estrada, J. & Dossetti, V. Fractality à la carte: a general particle aggregation model. Sci Rep 6, 19505 (2016). https://doi.org/10.1038/srep19505

Ballistic Curve

The ballistic curve is the curve along which a launched material point without propulsion would move, taking into account the air resistance acting on it.

In practice, the ballistic curve determines the flight path of the bullet's center of mass from the point of exit from the barrel to the point of impact. The shape of this track, as in the case of a material point, depends on the angle of inclination and the initial velocity of the projectile, the drag coefficient of the medium.

The ballistic curve corresponds approximately to a parabola, but in reality it is asymmetric.

$$F_g = -mg\ (weigh\ power)$$

$$F_D = -\frac{1}{2}c\rho A|v|v_{x,y,z}\ \ (resistance\ force\ of\ the\ medium)$$

$$k = \frac{1}{2}c\rho A$$

$$|v| = \sqrt{\dot{x}^2 + \dot{y}^2} = \sqrt{v_x^2 + v_y^2} \qquad v_{0x} = v_0\cos(\theta)\,, \qquad v_{0y} = v_0\sin(\theta)$$

The equations of motion:

$$\begin{cases} m\ddot{x} = -k\sqrt{\dot{x}^2 + \dot{y}^2}\dot{x} \\ m\ddot{y} = -k\sqrt{\dot{x}^2 + \dot{y}^2}\dot{y} - mg \end{cases}$$

$$\begin{cases} \ddot{x} = -\frac{k}{m}\sqrt{\dot{x}^2 + \dot{y}^2}\dot{x} \\ \ddot{y} = -\frac{k}{m}\sqrt{\dot{x}^2 + \dot{y}^2}\dot{y} - g \end{cases}$$

Where:

- $v_0 - \ initial\ velocity\ of\ the\ projectile,$
- $c - resistance\ coefficient\ of\ the\ medium,$
- $\rho - \ air\ density,$
- $g - gravitational\ acceleration,$
- $A - bullet\ cross - sectional\ area,$
- $m - bullet\ weight,$
- $\theta - the\ angle\ at\ which\ the\ projectile\ was\ fired,$
- $v_{0x} - horizontal\ velocity\ of\ the\ projectile\ at\ time\ t = 0$

Exercise: Write interactive code where:

- Using the above equations and conditions, simulate the trajectory of a projectile as it travels parallel to the wind blowing in the opposite direction. Let the parameters that can be modified by moving the sliders be: θ, v_0, c, v_w (wind speed counteracting the horizontal movement). Take as constant values:
  - $\rho - \ air\ density = 1.28\,\frac{\mathrm{kg}}{\mathrm{m}^3}$,
  - $g - gravitational\ acceleration = 9.81\,\frac{\mathrm{m}}{\mathrm{s}^2}$,
  - $A - bullet\ cross - sectional\ area\ (projectile\ radius = 0.05\ \mathrm{m})$,
  - $m - bullet\ weight = 0.2\ \mathrm{kg}$,
- Using the calculated coordinates, calculate and write out:
  - The time it takes for the projectile to hit the ground,
  - The time it takes for the projectile to reach its maximum height,
  - What is the maximum distance the projectile will travel,
  - What maximum height will the projectile rise to.

Example solution is shown in Figure 7.

Extra task: Follow the instructions in the exercise above, but in the equations of motion, include wind with a net velocity v_w opposing the horizontal motion of the projectile.

Example solution is shown in Figure 8.

θ 50

v0 40

c 10.00

```
Time to target = 2.01 s
Time to highest point = 0.73 s
Range to target, xmax = 8.25 m
Maximum height, ymax = 5.10 m
```

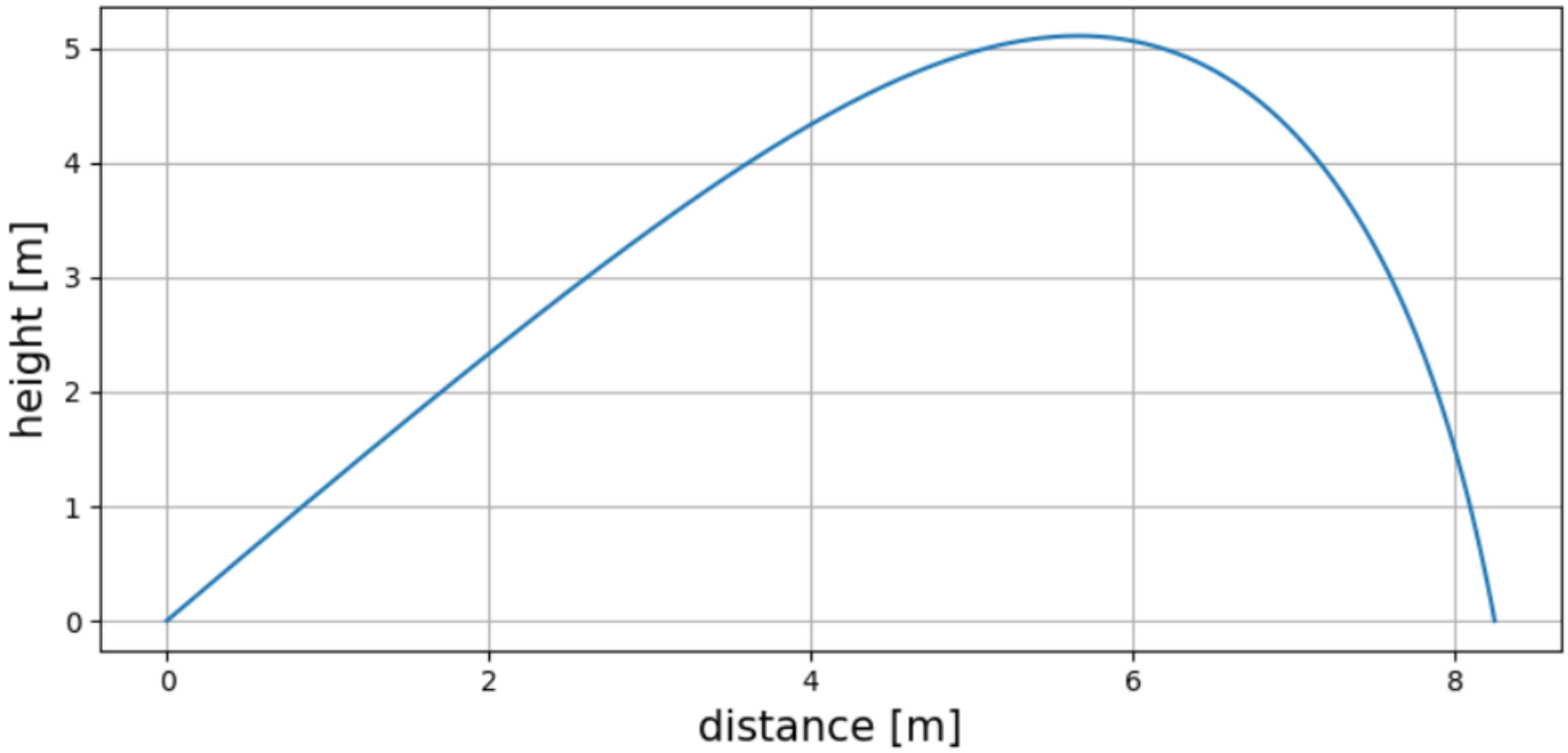


Figure 7. Example solution graph for given parameters.

θ 50

v0 40

vw 4

c 10.00

```
Time to target = 1.95 s
Time to highest point = 0.72 s
Range to target, xmax = 7.93 m
Maximum height, ymax = 4.80 m
```

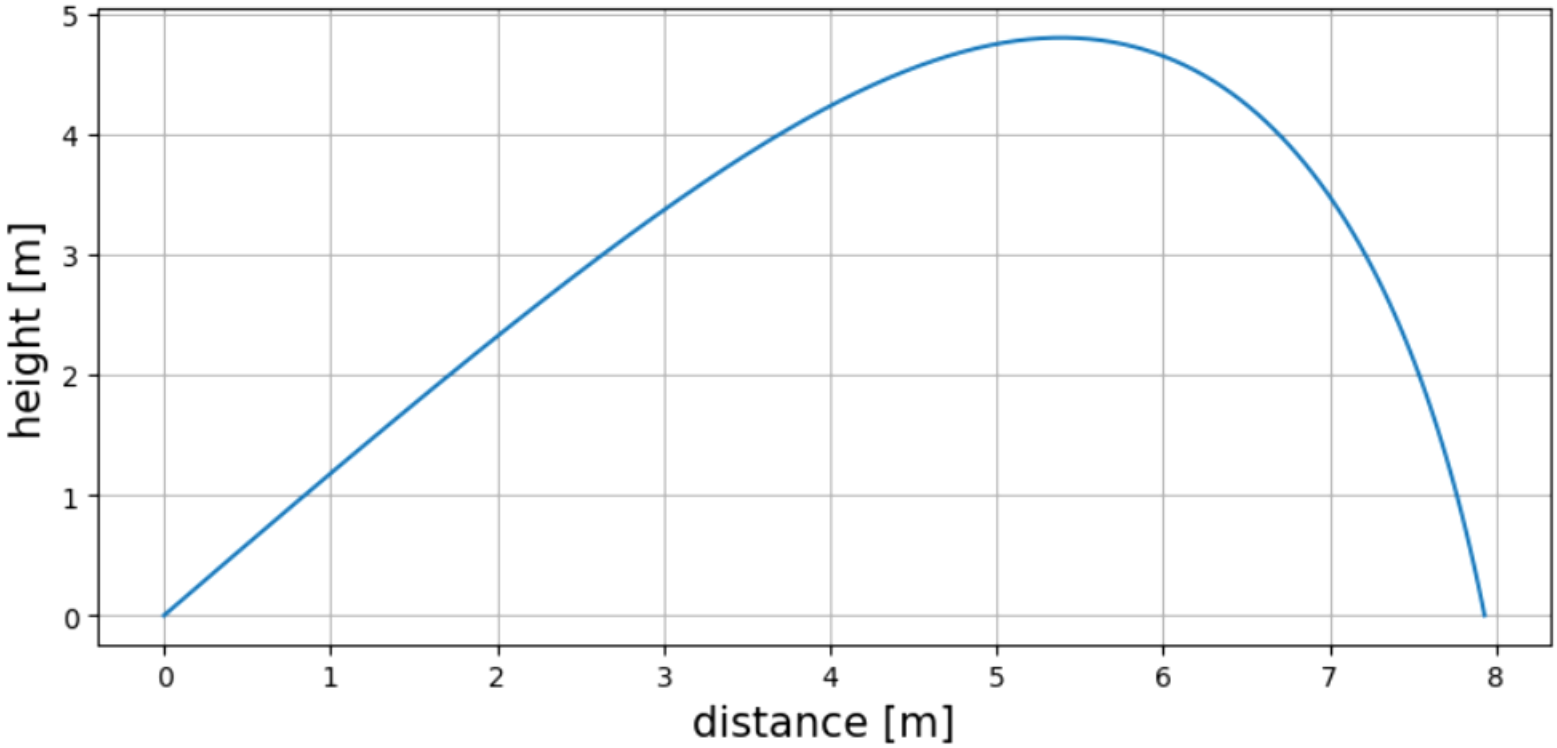


Figure 8. Example solution graph for given parameters.

Hints:

To write interactive code, you'll need:

```
from ipywidgets import interact

⋮

interact(throw,v0 =(10,70),θ=(10,90),vw=(0,9),c=(0.01,20))
```

Function for calculating x and z derivatives:

```
def deriv(t, u):
    x, xdot, z, zdot = u
    speed = np.hypot(xdot, zdot)
    xdotdot = -k/m * speed * xdot
    zdotdot = -k/m * speed * zdot - g
    return xdot, xdotdot, zdot, zdotdot
```

Stochastic Cells

Dimerization: monomer (*M*) joins up with another monomer and becomes a dimer (*D*): $2M \leftrightarrow D$. The forward reaction rate is k_b and the backward reaction rate is k_u. There are two reactions:

- the backward unbinding reaction (r_u) rate per unit volume is $k_u[D]$,
- the forward binding reaction rate (r_b) per unit volume is $k_b[M]^2$.
  Assume that: k_b= 1n$M^{-1}s^{-1}$, k_u = 2s^{-1} , at t = 0 are only N monomers.

A) Continuum dimerization. Find equations for *dM/dt, dD/dt* and write code to simulate continuum dimerization.

B) Stochastic dimerization. Implement algorithm:

(1) Calculate a list of the rates of all reaction in the system for discrete molecules:
$r_b = k_b M(M - 1)$,
$r_u = k_u D$.

(2) Find the total rate (r_t): $r_t = r_b + r_u$.

(3) Pick a random time t_{wait} with probability distribution: $\rho(t) = r_t \exp(-r_t t)$.

(4) If the current time t is bigger than t_{wait}, no further reactions will take place, end.

(5) Otherwise:

- increment t by t_{wait},
- pick a random number r uniformly distributed in the range [0, r_t),
- pick the reaction j:

  if $r < r_b$: $2M \rightarrow D$

  else: $D \rightarrow 2M$,
- execute that reaction by incrementing each chemical involved by its stoichiometry

(6) Repeat.

C) Compare A) and B) for: $N = 2$, $N = 90$ and $N = 10\,100$.

Example solution graphs are shown in Figure 9.

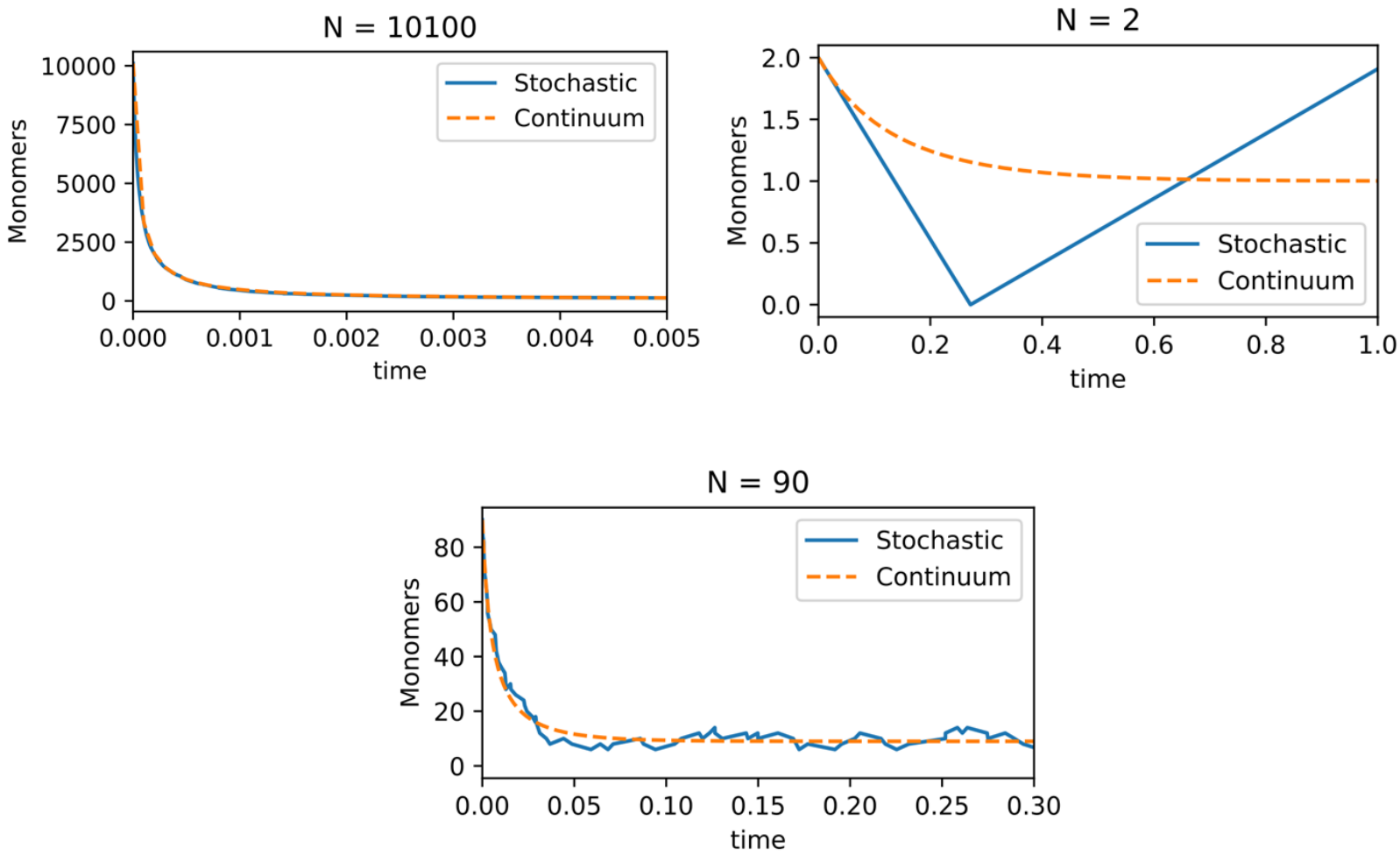


Figure 9. Example solution graphs.

Hints:

A)

find equations for: d*M*/d*t* and *dD/dt*:

$dM/dt = -2k_bM^2 + 2k_uD$

$dD/dt = -k_uD + k_bM^2$

to calculate the integrate you can use: scipy.integrate.odeint()

B)

to find t_{wait} use scipy.stats.expon.rvs(),

remember to set scale,

to choose *r* use random.randrange().

Exercise prepared on the basis of: book “Statistical Mechanics: Entropy, Order, Parameters and Complexity” by James Sethna (p. 178-179).