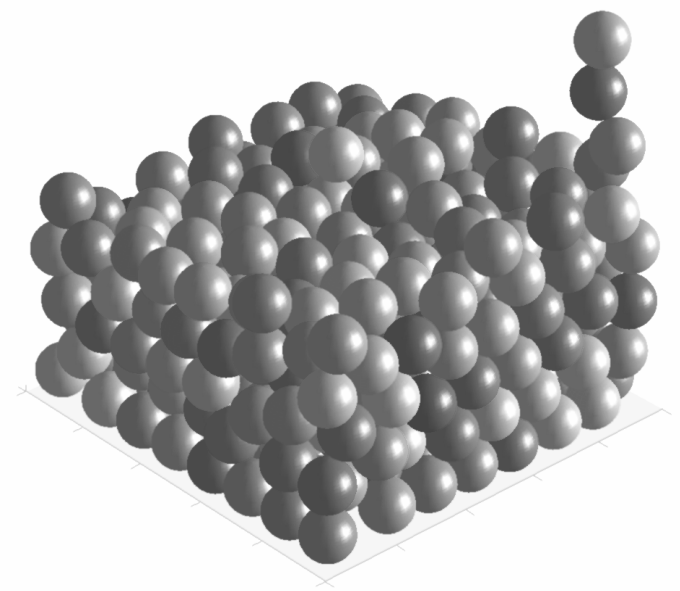
Computer Lab 2
A Pile of Particles

Mikica Kocic
miko0008@student.umu.se

2012-03-14



Mark what exercises you have done in this lab		*Points (by supervisor)*
Basic level (mandatory)	×	
Extra exercise #1: linear-spring model and preservation of the total energy and momentum	×	
Extra exercise #2: stability / piling of particles in impulse collision model	×	
Extra exercise #3: stability / piling of particles in linear-spring collision model	×	
Extra exercise #4: simulation in 2- and 3-D using impulse collision model	×	
Extra exercise #5: simulation in 2- and 3-D using linear-spring model	×	
The report is brief and to the point but still coherent and well structured.	☺	
Late report		
Total points		

Tutors: **Martin Servin** and **John Nordberg**

1 Background

The purpose of this computer lab was to simulate a number of spherical particles bouncing on top of each other and to study the basics of modeling and simulation of collisions. Two contact models were implemented and tested: the impulse method (instantaneous transfer of momenta) and the penalty method (linear springs). The preservation of the total energy, the total momentum and the stability of the method were analyzed for the both models.

2 Implementation and Simulation

2.1 Theory

All implemented models in this lab follow overall algorithm for a simulation of a mechanical system as presented *Notes on Discrete Mechanics*. There is a slight departure though, which is described in the following text.

Orientation of a plane and plane impenetrability

Inequality (5.8) for contact and expression (5.9) for overlap in section 5.2.2 in *Notes* can produce bugs. Namely, since $\left|(\mathbf{x}_{(a)}+\mathbf{r})^{\mathrm{T}}\hat{\mathbf{n}}\right|$ is an *absolute* (always positive) distance between a particle and a plane, the inequality (5.9) does not distinguish sides of the plane.

As one of the sides of the plane is impenetrable (the side opposite to the normal unit vector $\hat{\mathbf{n}}$), the correct expression for the overlap (actually the *penetration*) is:

$$\Delta x_{(a)} = r_{(a)} - (\mathbf{x}_{(a)}+\mathbf{r})^{\mathrm{T}}\hat{\mathbf{n}} \qquad (5.9)$$

and the condition for contact is/when:

$$\Delta x_{(a)} \geq 0 \qquad (5.8)$$

Conditions for the application of the impulse method

1) The hardcore interaction method (section 5.2.4 in *Notes*) is applicable to impacting contacts *only*, i.e. only when both conditions (5.3) and (5.5) are met. This is not explicitly stated in section 5.2.4 in *Notes* – it is rather implicitly assumed in a phrase “incoming velocities”. (Note that using the impulse method in case of a separating contact will make velocities of diverging particles to flip and converge, which is wrong.)

2) The second problem is when it occurs that the *relative* velocity $\mathbf{v}_{(ab)}$ is zero during the contact. This means also that the impulse $\mathbf{j}_{(ab)}$ is also zero, so the overlapping particles *may* stay indefinitely in the same state according to Eqs. (5.11) and (5.12) thus causing a bug. This case can resolved by using a position projections with a very small “projection factor” k_{p}.

Position projections factor k_{p}

A position projections factor k_{p} is introduced into equations (5.27) and (5.28) found in *Projections* subsection in *Notes*.

The modified expressions for the position projections (with the factor k_{p}) are then:

$$\mathbf{x}'_{(a)} = \mathbf{x}_{(a)} + k_{\mathrm{p}} \cdot \delta_{(a)}\hat{\mathbf{n}}_{(ab)} \qquad (5.27)$$

$$\left\{\ \mathbf{x}'_{(b)} = \mathbf{x}_{(b)} + k_{\mathrm{p}} \cdot \delta_{(b)}\hat{\mathbf{n}}_{(ba)},\quad \hat{\mathbf{n}}_{(ba)} = -\hat{\mathbf{n}}_{(ab)}\ \right\} \qquad (5.28)$$

The projection factor k_{p} values are in range from 0 to 1, where the factor $k_{\mathrm{p}} = 0$ disables position projections and $k_{\mathrm{p}} = 1$ enables projections in full extent.

Other values $0 < k_{\mathrm{p}} < 1$ make position projections to act like a spring (softening displacements).

On position projection factor seen as spring stiffness

Substituting $\delta_{(a)}$ from Eq. (5.29) in *Notes* into earlier modified Eq. (5.27) (with k_p), we get:

$$\mathbf{x}'_{(a)} - \mathbf{x}_{(a)} = k_p \cdot \frac{m_{(a)}^{-1}}{m_{(a)}^{-1} + m_{(b)}^{-1}} \Delta x_{(ab)} \cdot \hat{\mathbf{n}}_{(ab)} \qquad \text{(a)}$$

On the other side, from the integrator algorithm perspective, the position projection can be seen as there is an ad-hoc spring force $\mathbf{f}_{(a)} = k_s \Delta x_{(ab)} \cdot \hat{\mathbf{n}}_{(ab)}$ giving the same position displacement during the integrator time-step (but not influencing the velocity, however):

$$\mathbf{x}'_{(a)} - \mathbf{x}_{(a)} = \mathbf{f}_{(a)} m_{(a)}^{-1} \cdot h^2 \quad \text{i.e.}$$

$$\mathbf{x}'_{(a)} - \mathbf{x}_{(a)} = \left(k_s \Delta x_{(ab)} \cdot \hat{\mathbf{n}}_{(ab)} \right) \cdot m_{(a)}^{-1} \cdot h^2 \qquad \text{(b)}$$

Combining equations (a) and (b) yields:

$$k_p \cdot \frac{m_{(a)}^{-1}}{m_{(a)}^{-1} + m_{(b)}^{-1}} \Delta x_{(ab)} \cdot \hat{\mathbf{n}}_{(ab)} = \left(k_s \Delta x_{(ab)} \cdot \hat{\mathbf{n}}_{(ab)} \right) \cdot m_{(a)}^{-1} \cdot h^2 \qquad \text{(c)}$$

Finally, canceling $\Delta x_{(ab)} \hat{\mathbf{n}}_{(ab)} \cdot m_{(a)}^{-1}$ and substituting $m_{(ab)}^{-1} = m_{(a)}^{-1} + m_{(b)}^{-1}$ yields the relation:

$$k_p = k_s m_{(ab)}^{-1} \cdot h^2 \qquad \text{(d)}$$

Since the integrator method is assumed stable for $(k_s / m) \cdot h^2 = (\omega h)^2 < 1$, then, for $m_{(a)} = m_{(b)} = m$, it follows from (d), that the method is stable for the position projection factors $k_p < 0.5$.

2.2 Implementation

The model is implemented using MATLAB and the simulation framework is organized in the following modules:

- **`lab2_defs.m`:** implements `lab2_defs` function that constructs the `params` structure which holds definitions, parameters and constants used throughout the simulation.
- **`lab2_main.m`:** implements `lab2_main` function that simulates and visualizes a system of colliding particles and plots the total energy, the linear-momentum and the barycenter of the system. The function is intended to be called with the `params` structure as the argument. However, when called without arguments, `lab2_main` runs a demo simulation. The function returns `params` structure as the result, which can be used as the input to further simulations.
- **`lab2.m`:** contains a front-end GUI application that allows user to configure the `params` structure interactively and run a series of simulations. The user can stop the current simulation and change some parameters before continuing. Alternatively the simulation can be reset to the initial conditions and restarted.

The recommended simulation algorithm for using `lab2_defs` and `lab2_main` is:

```
(1)  params = lab2_defs;                          % Construct the params structure
     params.N_dim = ...; params.X_0 = ...         % Setup the simulation parameters
     params.X_0 = ...; params.V_0 = ...           % Modify some simulation parameters, optionally
(2)  params = lab2_main( params );                % Start the simulation
     ... = params.X; ... = params.V;              % Use the results, optionally
     params.X_0 = ...; params.V_0 = ...           % Modify some simulation parameters, optionally
(2)  params = lab2_main( params );                % Start the simulation
     ... = params.X; ... = params.V;              % Use the results, optionally
                                                  % and so on...
```

This algorithm is implemented in `lab2` using two event handlers: `combo_Configuration` and `button_Start` callbacks, which implements steps (1) and (2) of the algorithm respectively. The user may change parameters interactively between the steps and save the simulation data to MAT-files for later analysis (see `lab2_sim23.m` script for example). Saved MAT-files can be also used as carry-over to further simulations.

Space Dimensions

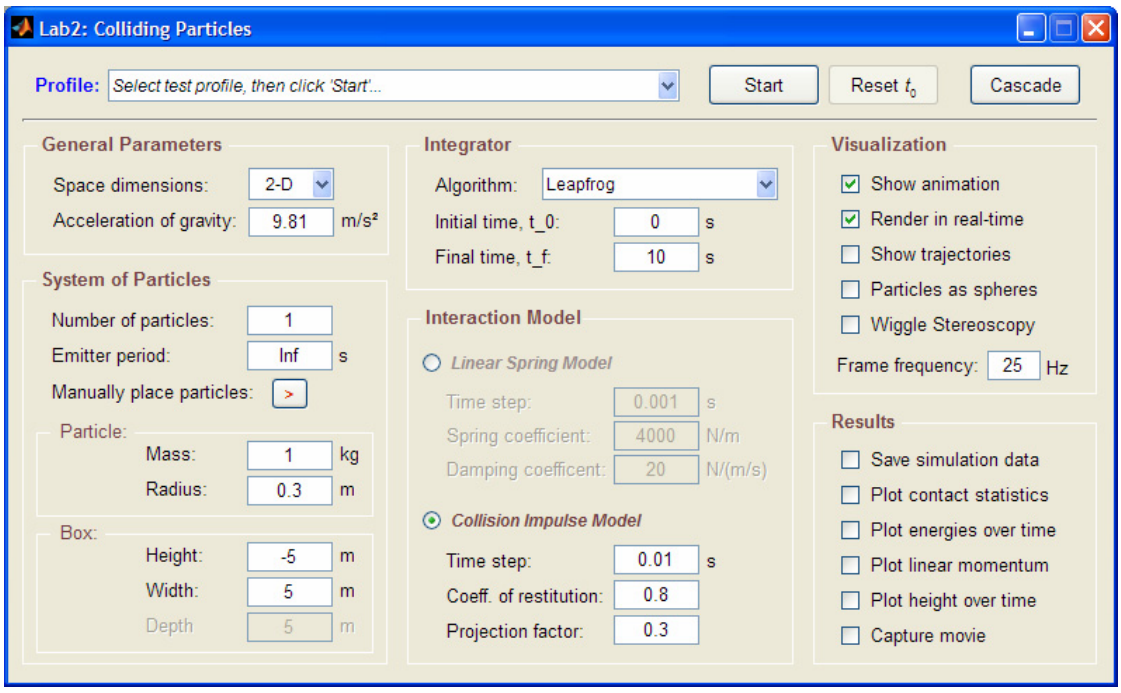
Number of space dimensions N_{dim} is a system parameter. The last dimension is always denoted as 'height' and labeled as z-coordinate; consequently, vectors in 2-D have x- and z-components and vectors in 1-D have a single z-component.

Integrator Algorithms

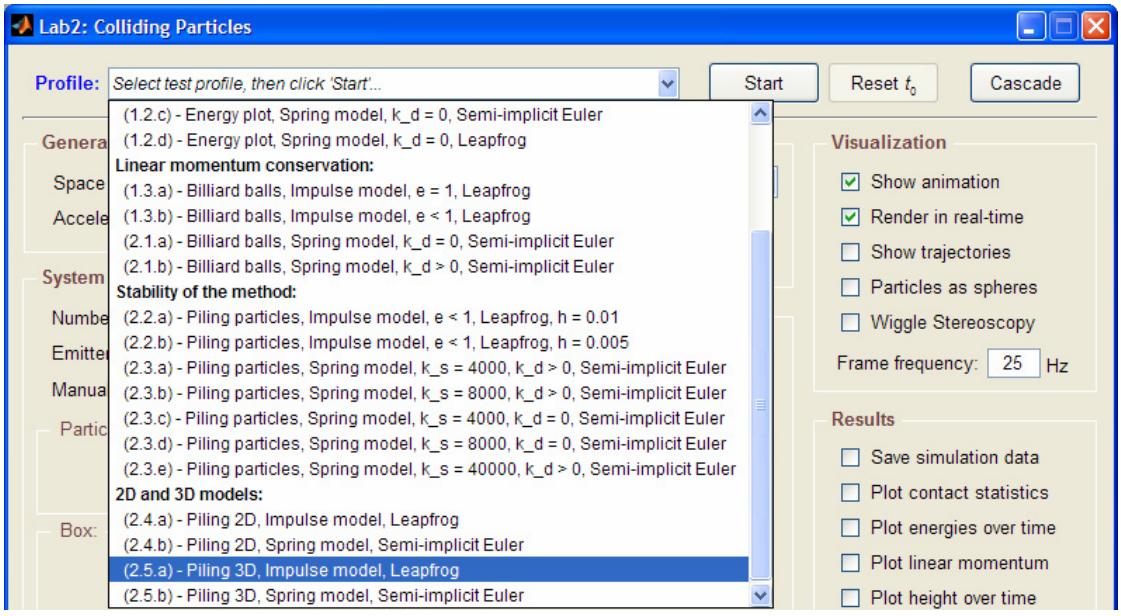
Beside the semi-implicit Euler algorithm, the leapfrog integrator was also used for solving differential equations of motions. The leapfrog integrator algorithm is taken from (Hut & Makino, 2004) and incorporated together with the semi-implicit Euler algorithm in the same main simulation loop; for more details see the comments in *Main Simulation Loop* and *Step Forward and Update State Variables* sections in `lab2_main.m`.

2.3 Running Simulation

1) Execute **`lab2`** from the MATLAB command prompt. The gui should appear:



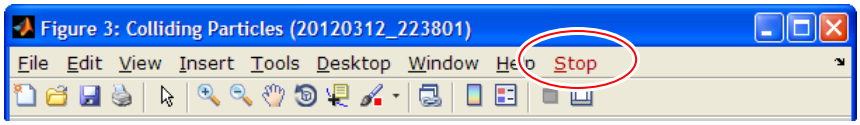
2) Select the configuration profile:



3) Change parameters (optionally). All parameters have a tool-tip with short usage info.

4) Select '**Start**' button.

Note that you can always stop the current simulation and change some parameters before continuing or alternatively, restart the simulation from the initial conditions (with button Reset t_0).



3 Results

3.1 Basic Level Tasks

Sample snapshots of the various simulations with different configurations are shown on Figure 3.1–1. Images in the top row show simulations of piling objects in 1D and several bouncing balls in 2D, while the graphics in the last two rows show the trajectory of the single bouncing ball in 2D simulated using two different collisions models under the same initial conditions.

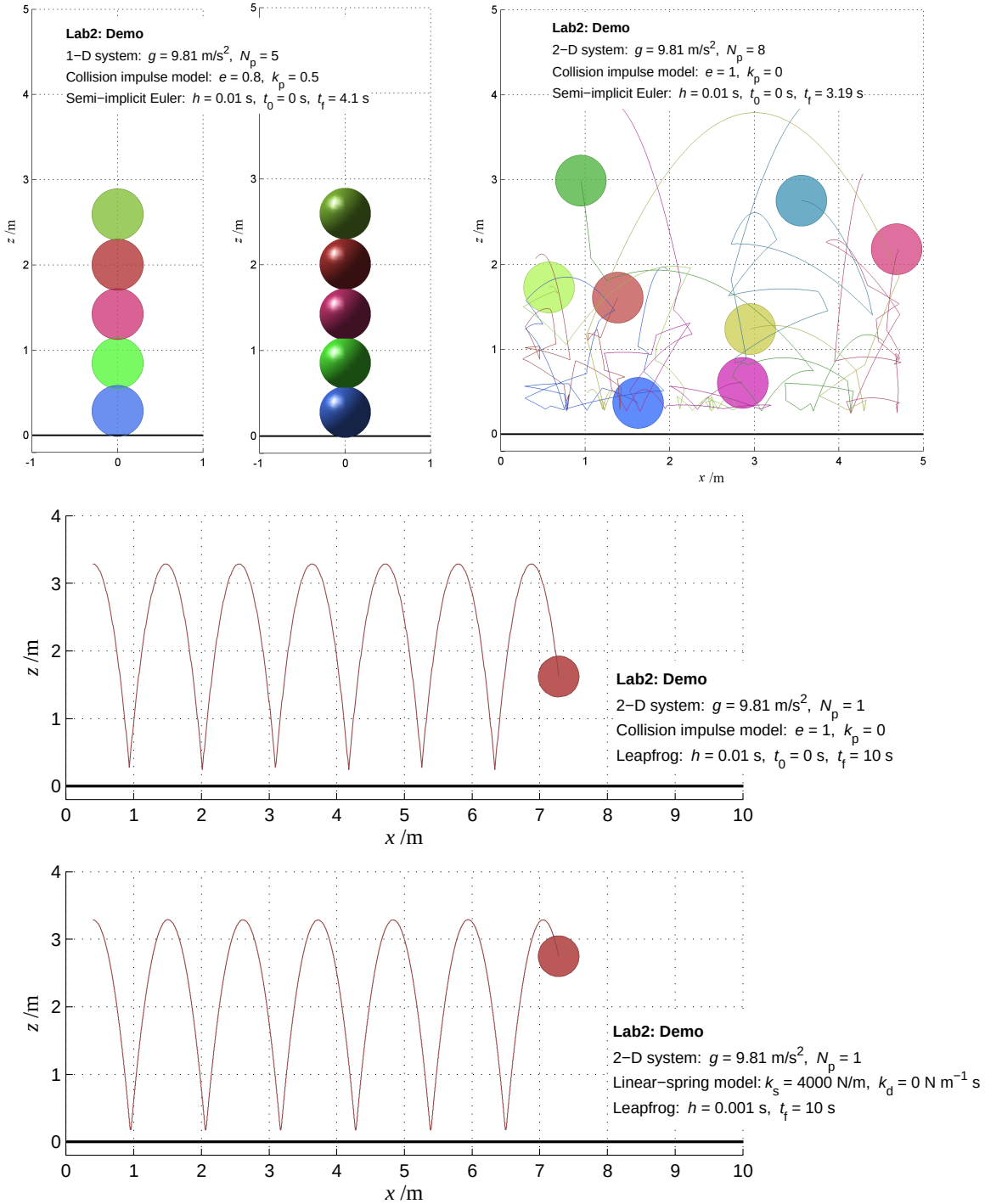


Figure 3.1–1: Snapshots of the various demo simulations

Conservation of the Total Energy in the Impulse Collision Model

The system energy levels of single particle system simulations with the impulse collision model without dissipation are shown on Figure 3.1–3 when using the semi-implicit Euler integrator and on Figure 3.1–4 when using the leapfrog integrator. The lower plot displays the height of the center of mass of the system (equal to the height of the particle in a single particle system) as function of time.

It can be observed from the figures, that the semi-implicit Euler algorithm loses energy in the gravity field, while the total energy is conserved when using leapfrog integrator.

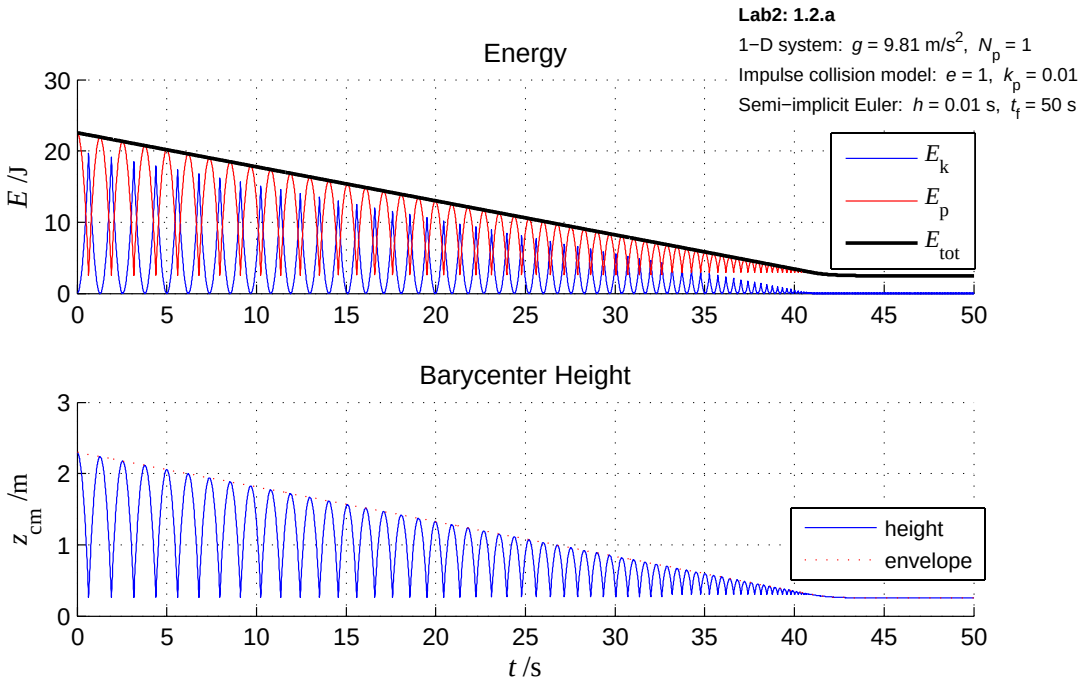


Figure 3.1–2: Impulse collision model, no dissipation, semi-implicit Euler integrator.

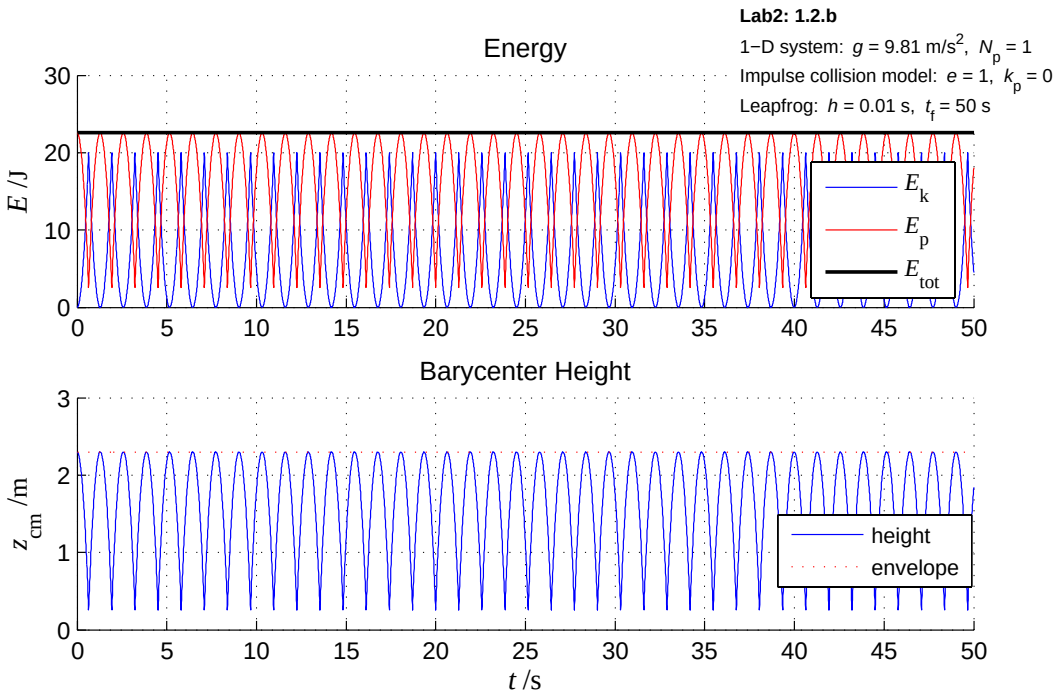


Figure 3.1–3: Impulse collision model, no dissipation, leapfrog integrator.

Conservation of the Total Energy in the Linear-Spring Model

The system energy levels of single particle system simulations with the linear-spring model without dissipation are shown on Figure 3.1–4 when using the semi-implicit Euler integrator and on Figure 3.1–5 when using the leapfrog integrator.

It can be observed from the figures, that the total energy of the system is conserved in both cases and that there is no significant difference between integrator algorithms.

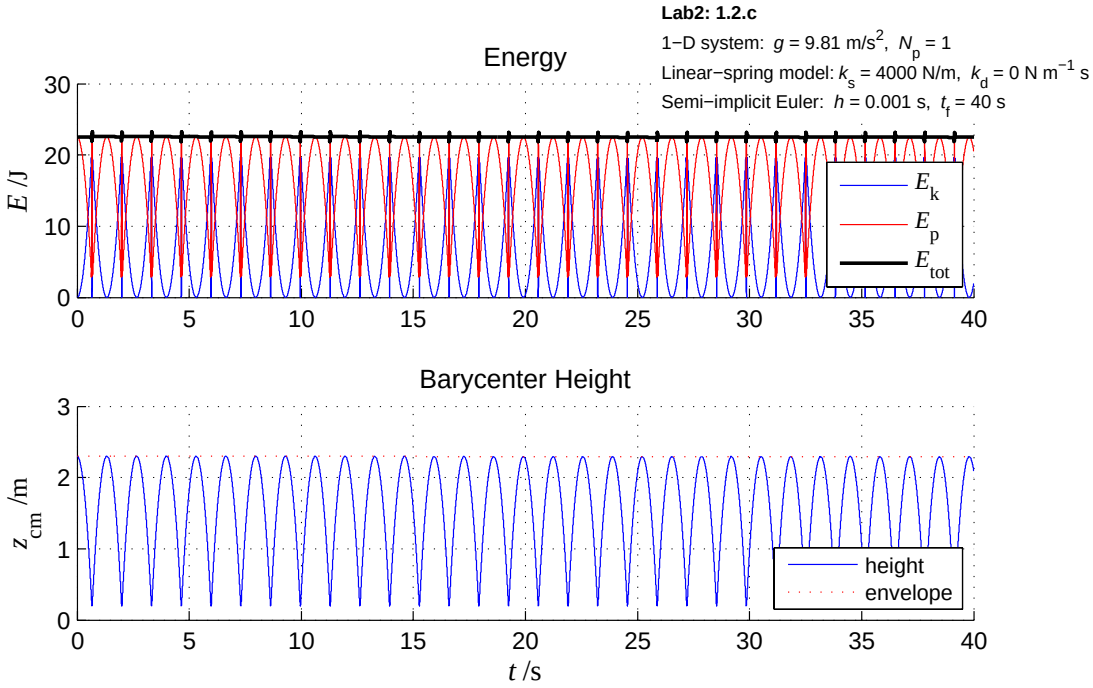


Figure 3.1–4: Linear-spring model, no dissipation, semi-implicit Euler integrator.

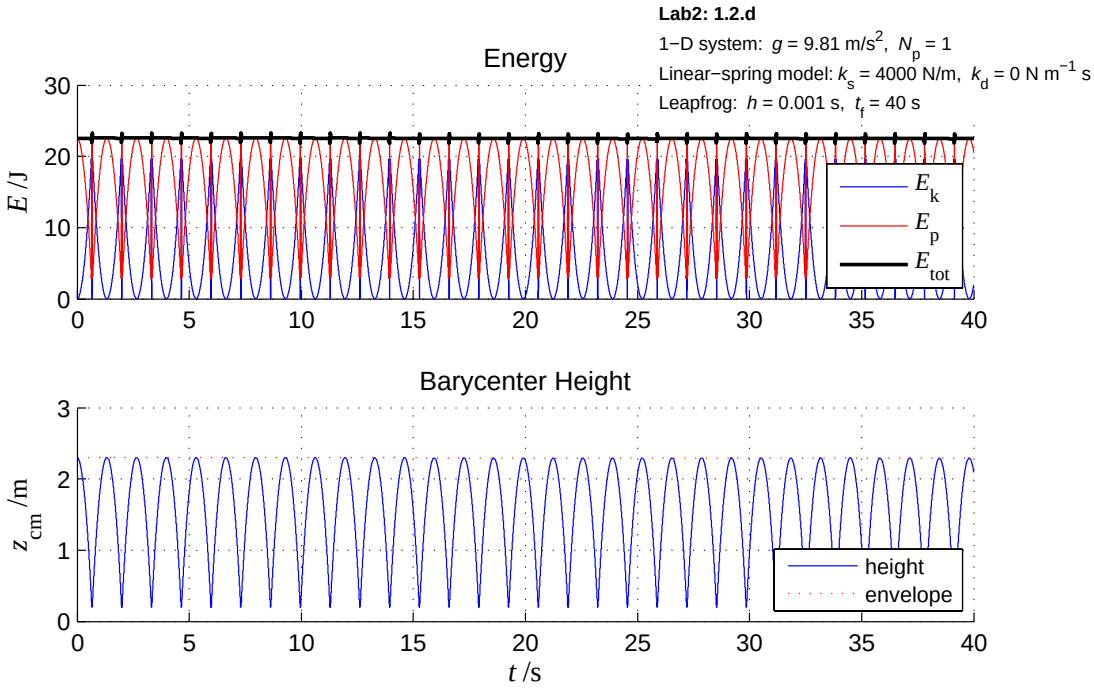


Figure 3.1–5: Linear-spring model, no dissipation, leapfrog integrator.

Conservation of the Linear Momentum – Test Setup

The conservation of the linear momentum was tested using a billiard table like system configuration with several balls with different masses and velocities colliding with each other and with the box walls, as shown on Figure 3.1–6.

The same initial conditions were used both for the simulation with the impulse collision model (in this section) and the linear spring model (in the advanced level section).

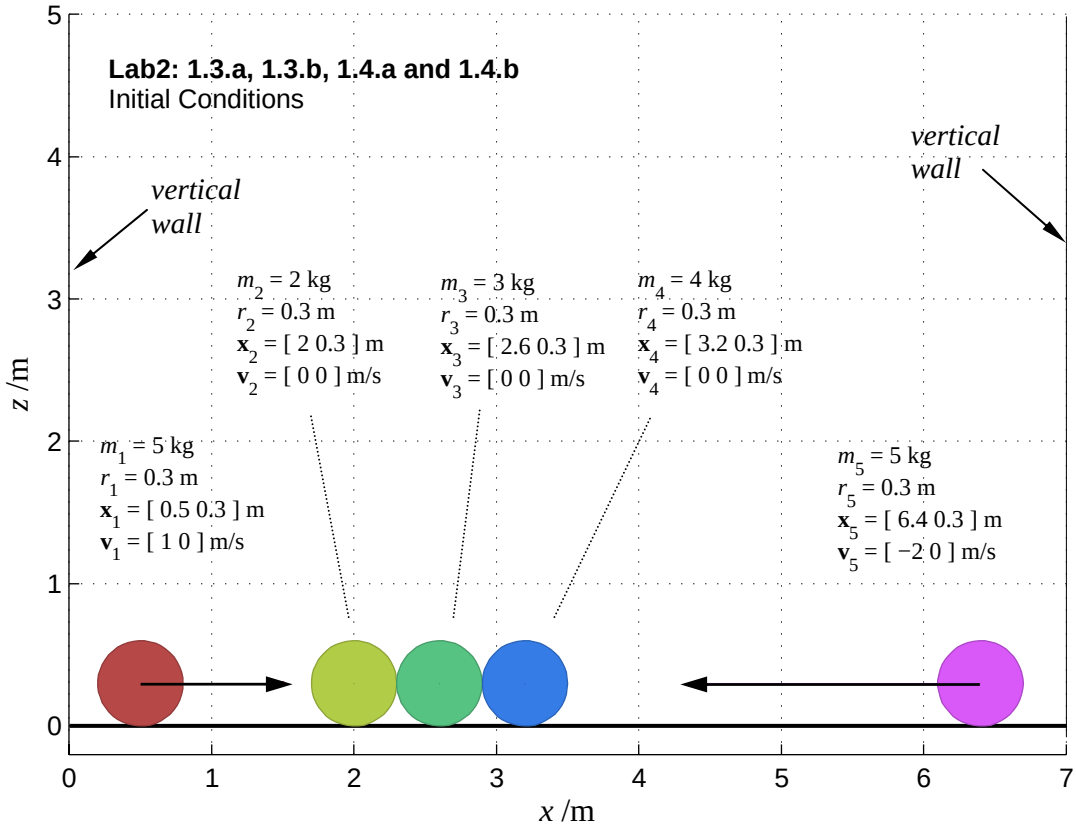


Figure 3.1–6: Conservation of the linear momentum: The billiard ball test setup. The system is confined with impenetrable surfaces (ground and side walls).

The initial total energy of the system is $E_{\text{tot}} = 12.5\,\text{J}$.

The initial linear momentum of the system is $p_{x,\text{tot}} = -5\,\text{kg·m/s}$, $p_{z,\text{tot}} = 0\,\text{kg·m/s}$.

Conservation of the Linear Momentum in the Impulse Collision Model

The simulation of the system described on Figure 3.1–6 using the impulse collision model without dissipation is shown on Figure 3.1–7, where particle-particle collisions are highlighted with brackets.

It can be observed from the figure, that particle-to-particle collisions did not change the linear momentum of the system, while collisions between particles and surfaces changed the linear momentum of the system.

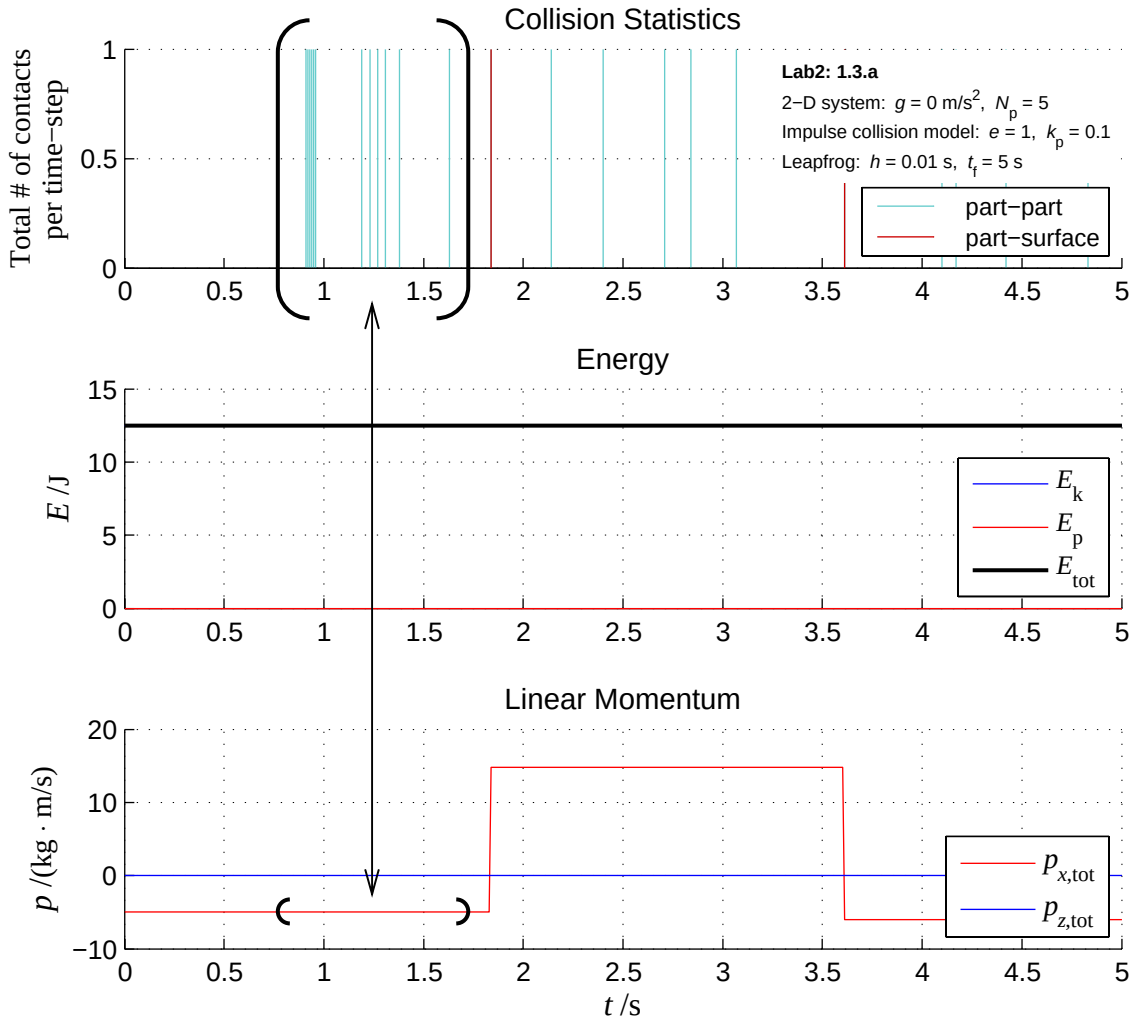


Figure 3.1–7: Conservation of the linear momentum: Impulse collision model, without dissipation, leapfrog integrator. Collision statistics shows discrete collision events between particles (light blue) and particles to surfaces (red).

Conservation of the Linear Momentum in the Impulse Collision Model (contd.)

The simulation of the system described on Figure 3.1–6 using the impulse collision model with dissipation is shown on Figure 3.1–8, where particle-particle collisions are highlighted with brackets.

It can be observed from the figure, that particle-to-particle collisions did not change the linear momentum of the system, while collisions between particles and surfaces changed the linear momentum of the system.

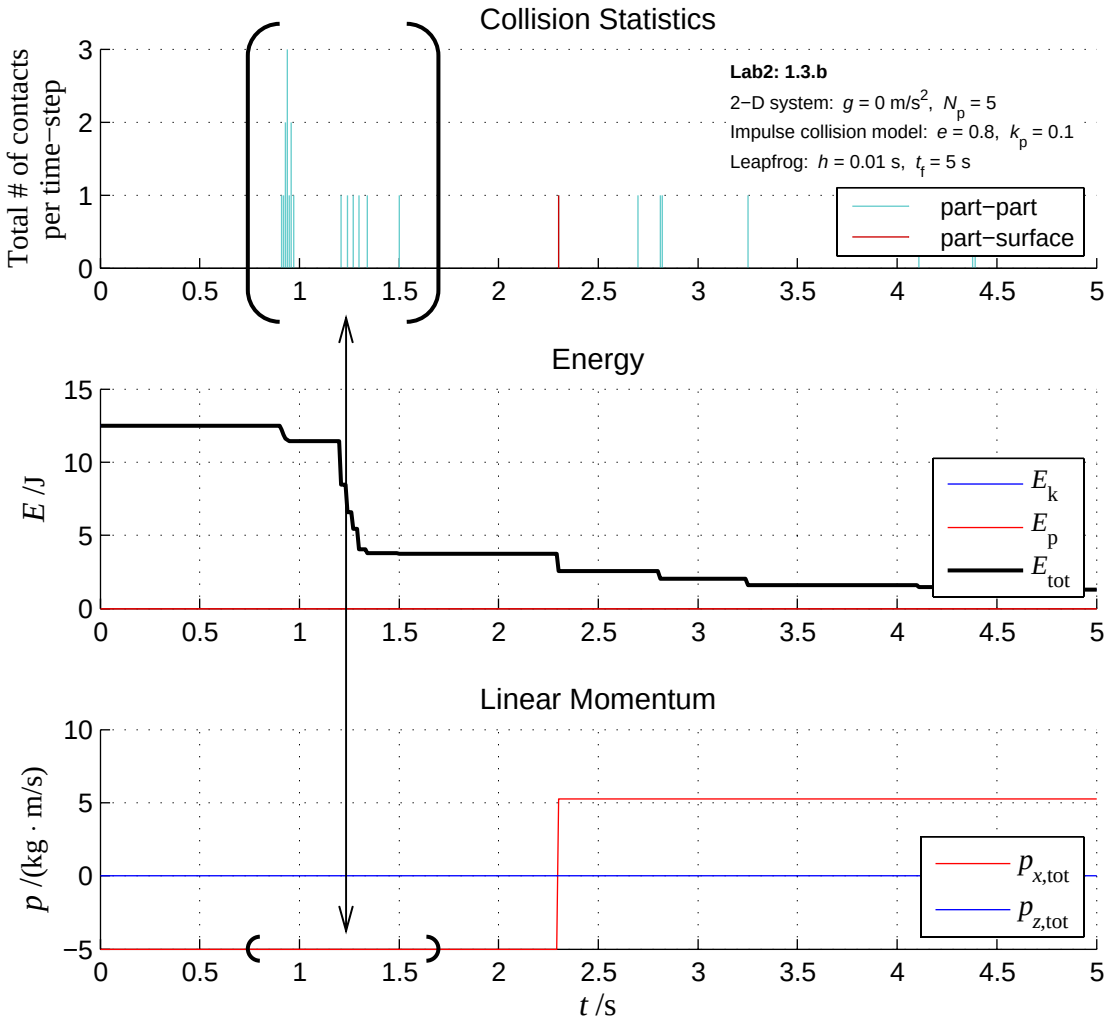


Figure 3.1–8: Conservation of the linear momentum: Impulse collision model, with dissipation, leapfrog integrator. Collision statistics shows discrete collision events between particles (light blue) and particles to surfaces (red).

3.2 Advanced Level Tasks

Conservation of the Linear Momentum in the Linear-Spring Model

The simulation of the system described on Figure 3.1–6 using the linear-spring model without dissipation is shown on Figure 3.2–1, where particle-particle collisions are highlighted with brackets.

It can be observed from the figure, that particle-to-particle collisions did not change the linear momentum of the system, while collisions between particles and surfaces changed the linear momentum of the system.

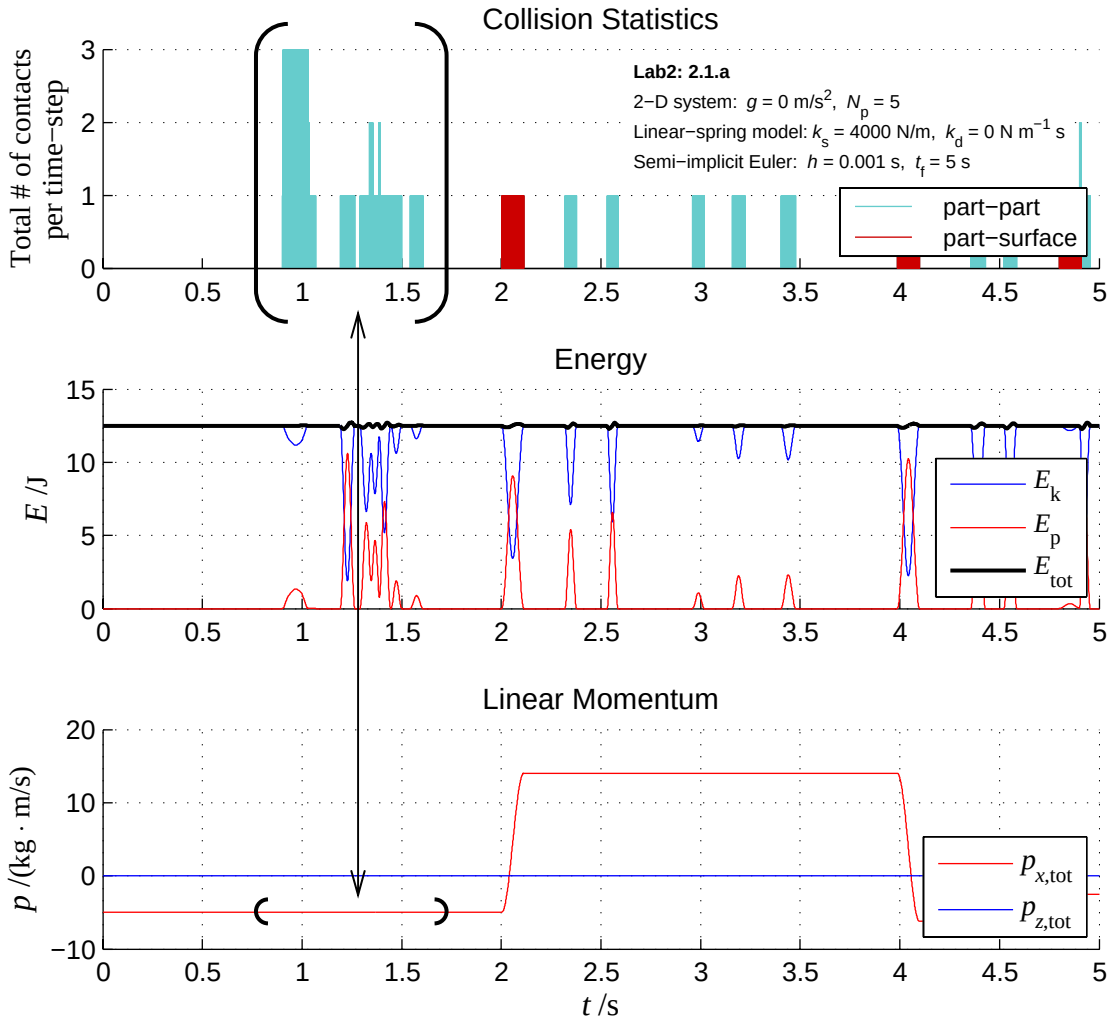


Figure 3.2–1: Conservation of the linear momentum: Linear-spring model, without dissipation, semi-implicit Euler integrator. Collision statistics shows discrete collision events between particles (light blue) and particles to surfaces (red).

Conservation of the Linear Momentum in the Linear-Spring Model (contd.)

The simulation of the system described on Figure 3.1–6 using the linear-spring model with dissipation is shown on Figure 3.2–2, where particle-particle collisions are highlighted with brackets.

It can be observed from the figure, that particle-to-particle collisions did not change the linear momentum of the system, while collisions between particles and surfaces changed the linear momentum of the system.

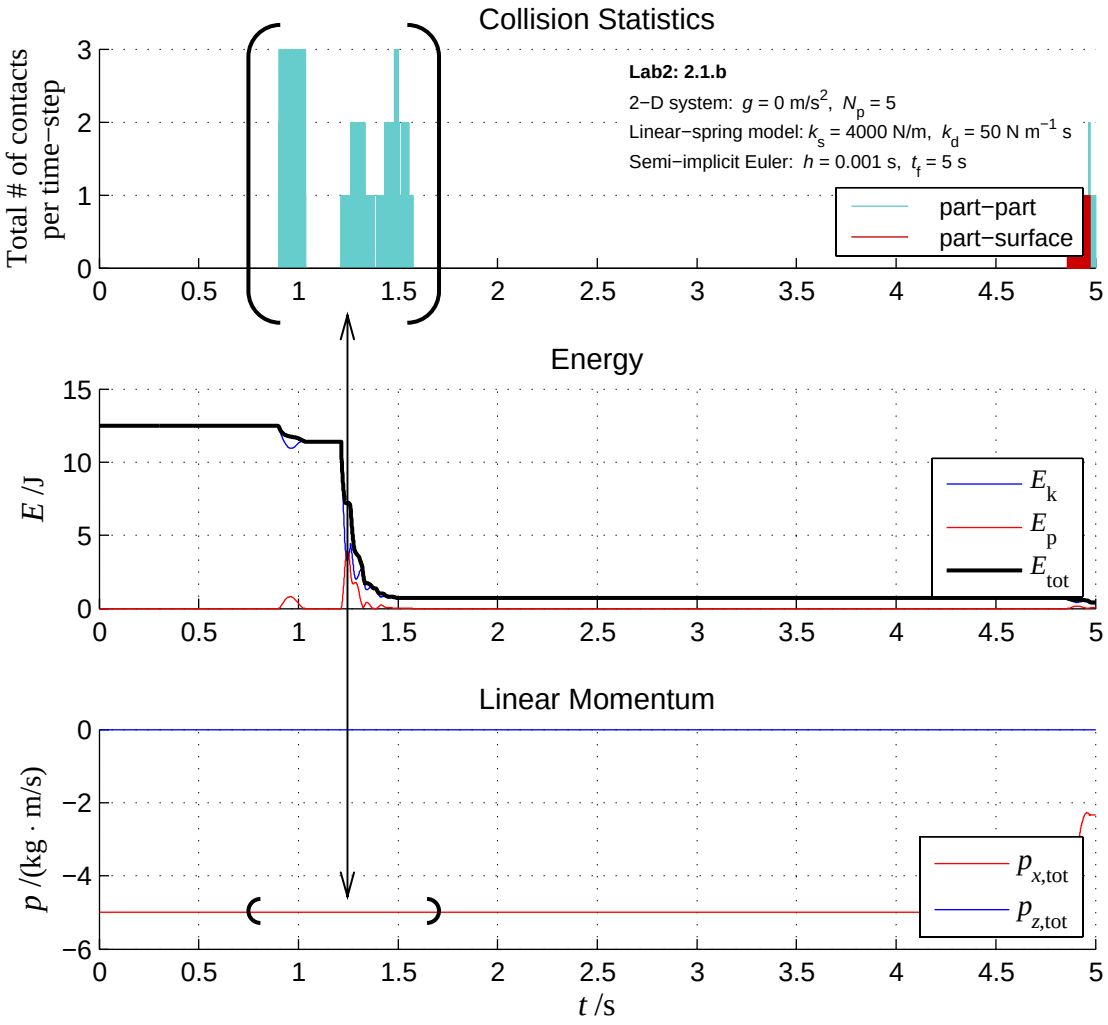


Figure 3.2–2: Conservation of the linear momentum: Linear-spring collision model, with dissipation, semi-implicit Euler integrator. Collision statistics shows discrete collision events between particles (light blue) and particles to surfaces (red).

Stability of the Impulse Collision Model

The snapshots of the simulation of a piling particles system, with the impulse collision model, are shown on Figure 3.2–3 with the total energy of the system shown on Figure 3.2–4. A single particle is emitted to the system each second. Over the time, the system starts exhibiting jittery behavior, finally resulting in the total instability ('explosion') after the number of particles reaches $N_p = 13$.

The simulation of the same system, solved with the integrator time-step decreased from $h = 0.01$ s to $h = 0.005$ s, is shown on Figure 3.2–5. In this case the system gets unstable after reaching $N_p = 21$.

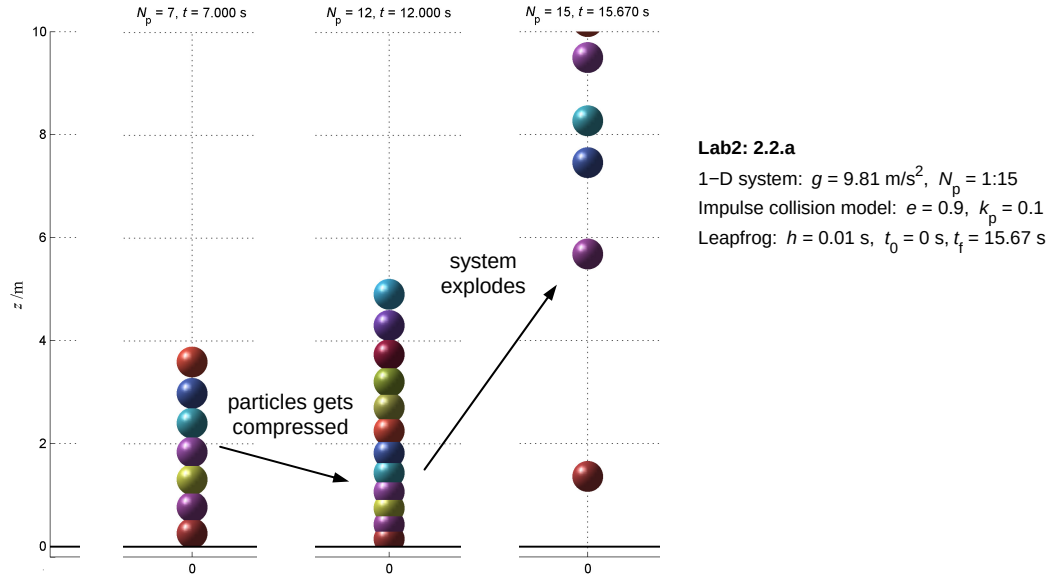


Figure 3.2–3: Piling particles on the top of each other: Impulse collision model, leapfrog integrator, $h = 0.01$ s. Simulation snapshots at different times.

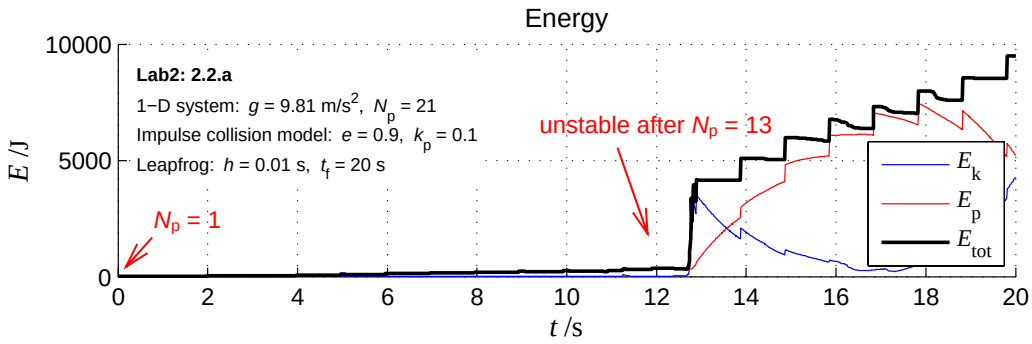


Figure 3.2–4: Energy of the system, as shown on Figure 3.2–3, with $N_p(t_0) = 1$ and $h = 0.01$ s.

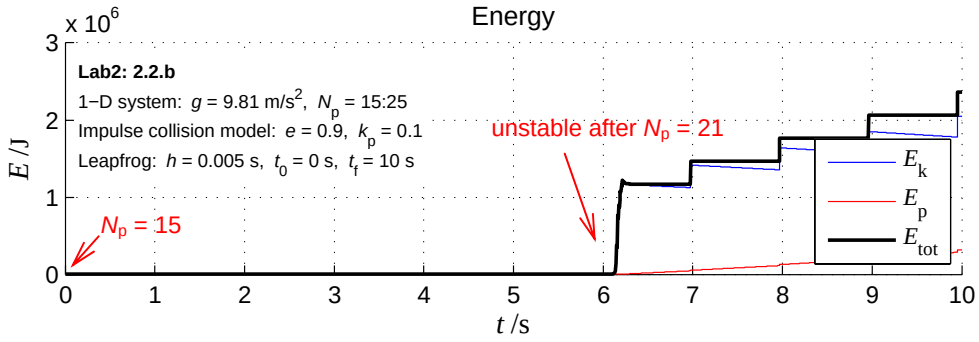
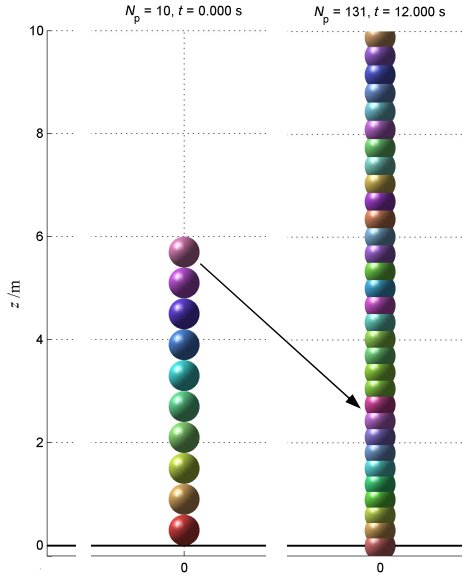


Figure 3.2–5: Energy of the system, with $N_p(t_0) = 15$ and reduced time-step to $h = 0.005$ s.

Stability of the Linear Spring Model

The snapshots of the simulation of a piling particles system using the penalty method and spring coefficient k_s = 4000 N/m, are shown on Figure 3.2–6 with the bottom-particle diameters shown on Figure 3.2–7. One particle is emitted to the system every 100 ms and the system is simulated for 12 s resulting in N_p = 131 at the end. The bottom-particle gets compressed to 50 % of the original size when N_p = 117 at t_f = 11.654 s.



Lab2: 2.3.a
1−D system: g = 9.81 m/s^2, N_p = 131
Linear−spring model: k_s = 4000 N/m, k_d = 40 N m^{-1} s
Semi−implicit Euler: h = 0.001 s, t_f = 12 s

Final pile height: 58.56 m (of max $N_p \cdot 2r_p$ = 78.6 m)
Pile compression: 25.5 %
Bottom particle diameter: 0.298 m (of max 0.6 m)
Bottom particle compression: 50.4 %

Figure 3.2–6: Piling particles on the top of each other: Linear-spring model, k_s = 4000 N/m, semi-implicit Euler integrator; snapshots at initial and final time.

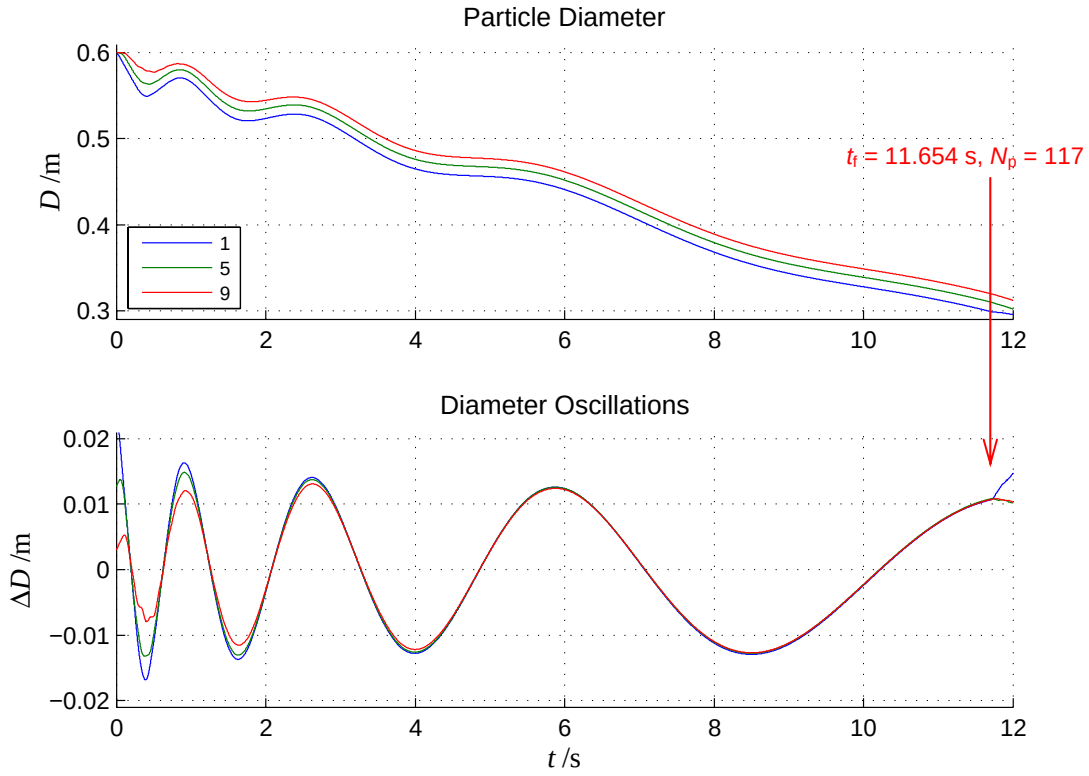


Figure 3.2–7: Particle diameters and diameter oscillations graph for the lowest particles in pile; simulation parameters are given on Figure 3.2–6.

The snapshots of the simulation of a piling particles system using the penalty method after increasing the spring stiffness to k_s = 8000 N/m, are shown on Figure 3.2–8 and the bottom-particle diameters are shown on Figure 3.2–9. One particle is emitted to the system every 100 ms and the system is simulated for 12 s resulting in N_p = 131 at the end whilst the bottom-particle gets compressed to 73 % of the original size.

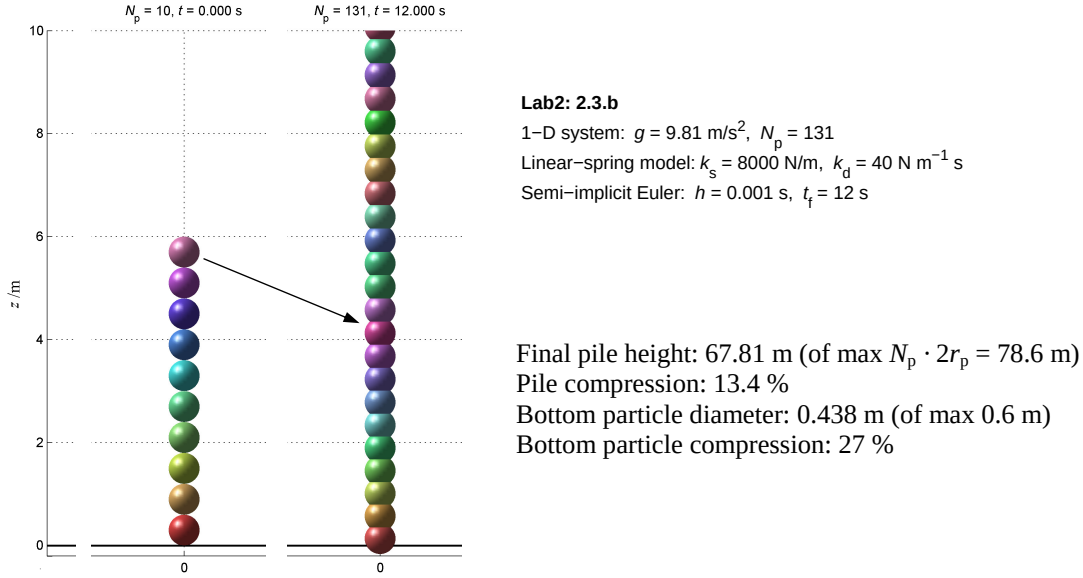


Figure 3.2–8: Piling particles on the top of each other: Linear-spring model, k_s = 8000 N/m, semi-implicit Euler integrator; snapshots at initial and final time.

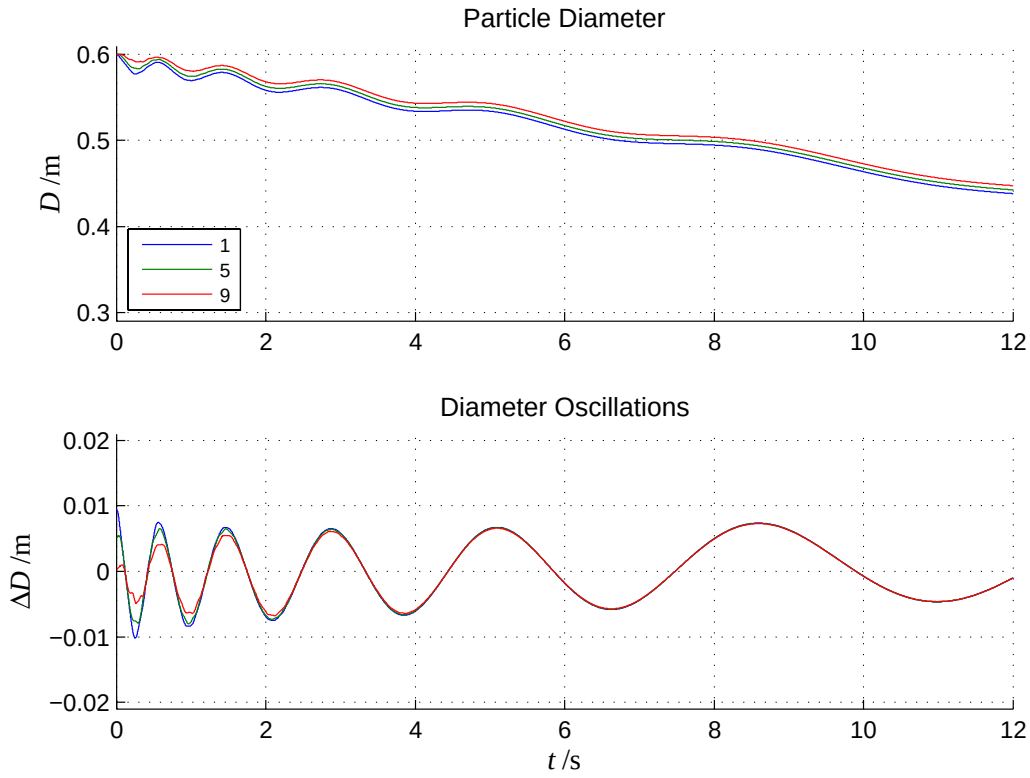


Figure 3.2–9: Particle diameters and diameter oscillations graph for the lowest particles in the pile; simulation parameters are given on Figure 3.2–8.

After increasing the spring stiffness further to $k_s = 4 \times 10^4$ N/m, the bottom-particle diameters during the simulation are shown on Figure 3.2–10.

Final pile height: 76.5 m (of max $N_p \cdot 2r_p$ = 78.6 m)
Pile compression: 2.7 %
Bottom particle diameter: 0.568 m (of max 0.6 m)
Bottom particle compression: 5.3 %

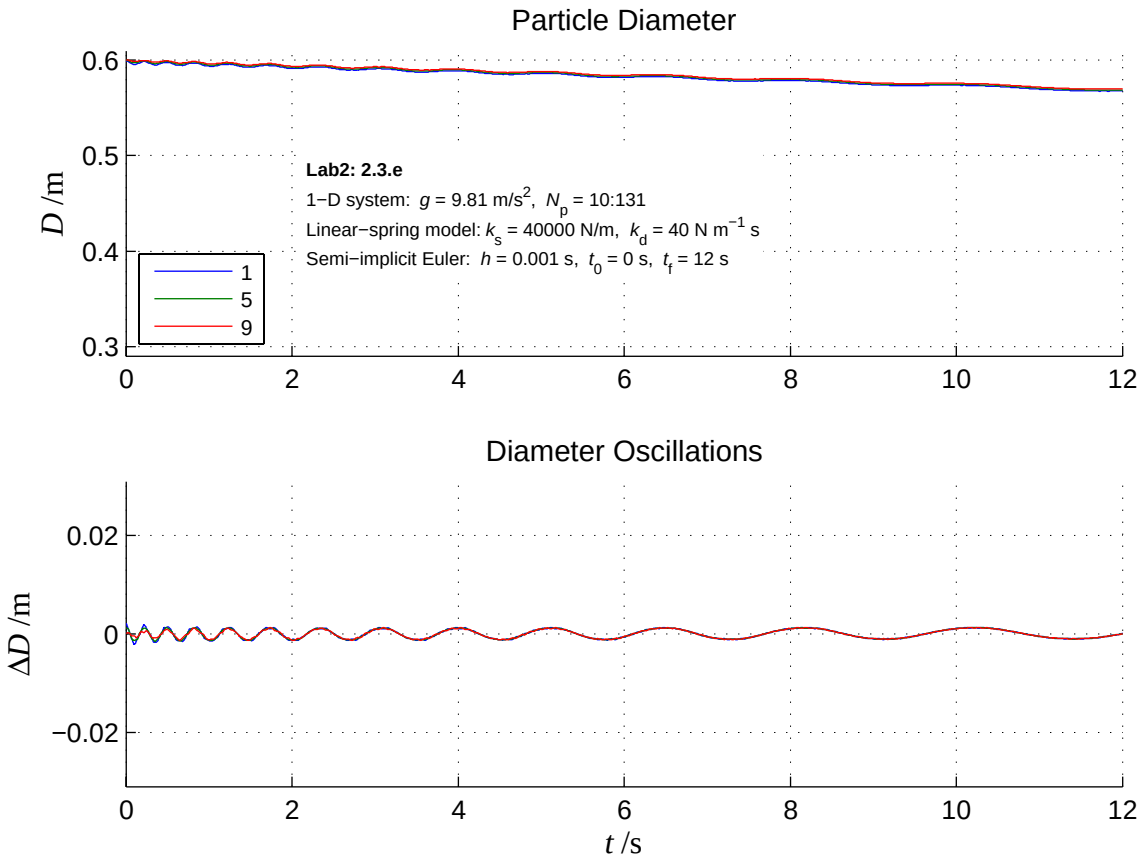


Figure 3.2–10: Particle diameters and diameter oscillations graph for the lowest particles in pile; simulation parameters are as on Figure 3.2–8 but with $k_s = 4 \times 10^4$ N/m.

The previous simulation results are summarized in the following table:

Table 3.2-1: The compression of the system as a function of the spring coefficient in the linear spring model

Spring coeff. k_s / (N/m)	Pile height z / m	Pile compression / %	Bottom particle D / m	Bottom particle compression / %
4 000	58.6	25.5	0.298	50.4
8 000	67.8	13.4	0.438	27.0
40 000	76.5	2.7	0.568	5.3

The results of the simulations similar to Figure 3.2–6 and Figure 3.2–8 but *without dissipation* are shown for comparison on Figure 3.2–11 and Figure 3.2–12, respectively:

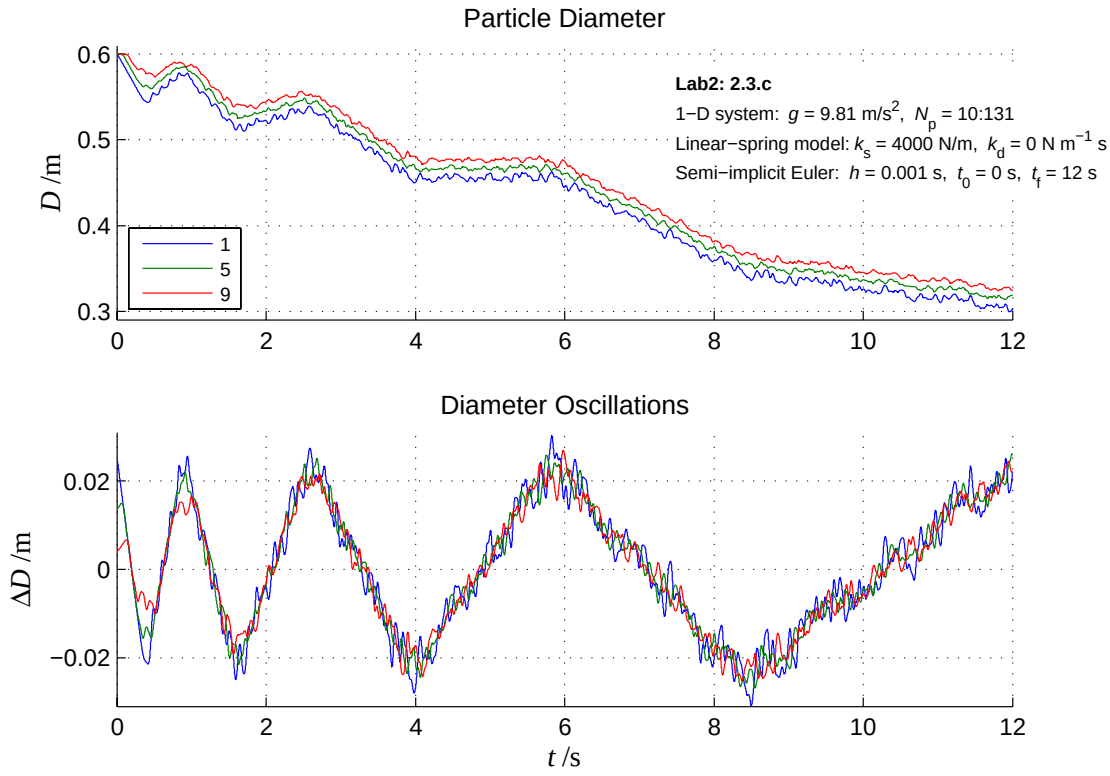


Figure 3.2–11: Particle diameters and diameter oscillations graph for the lowest particles in pile; simulation parameters are as on Figure 3.2–6 but *without dissipation*.

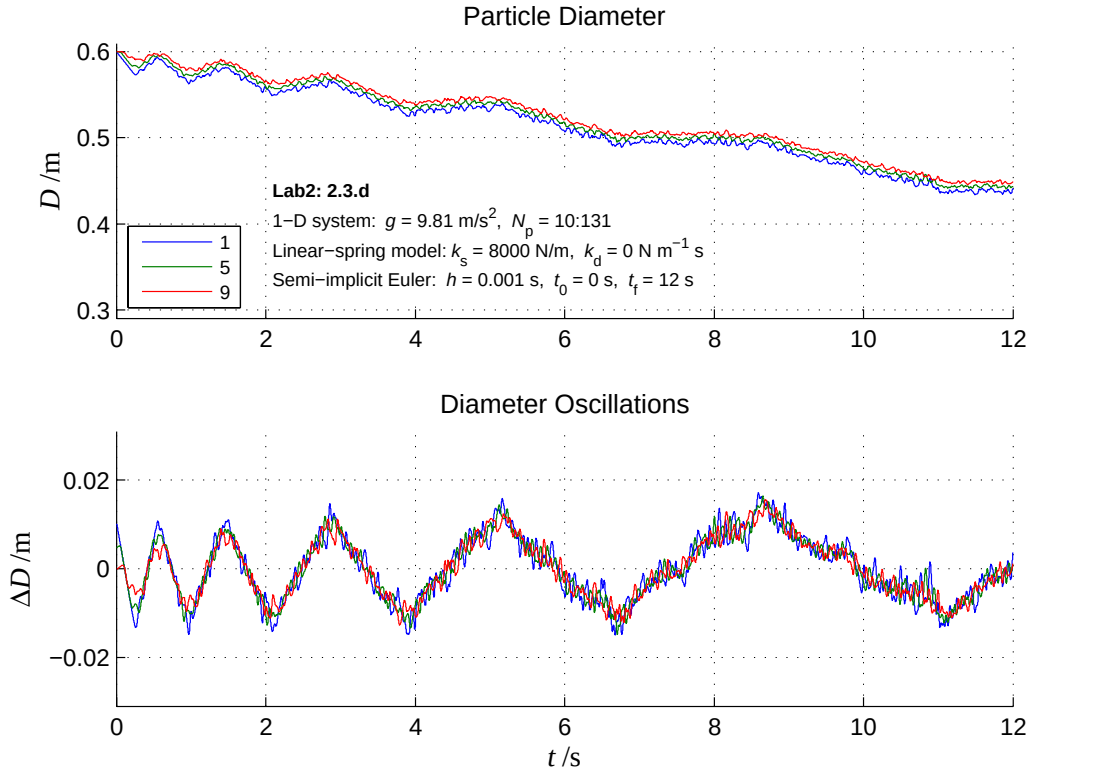


Figure 3.2–12: Particle diameters and diameter oscillations graph for the lowest particles in pile; simulation parameters are as on Figure 3.2–8 but *without dissipation*.

2- and 3-D Implementation of the Impulse Collision Method

Snapshots of a piling of particles in a box, simulated using the impulse collision model, are shown on Figure 3.2–13.

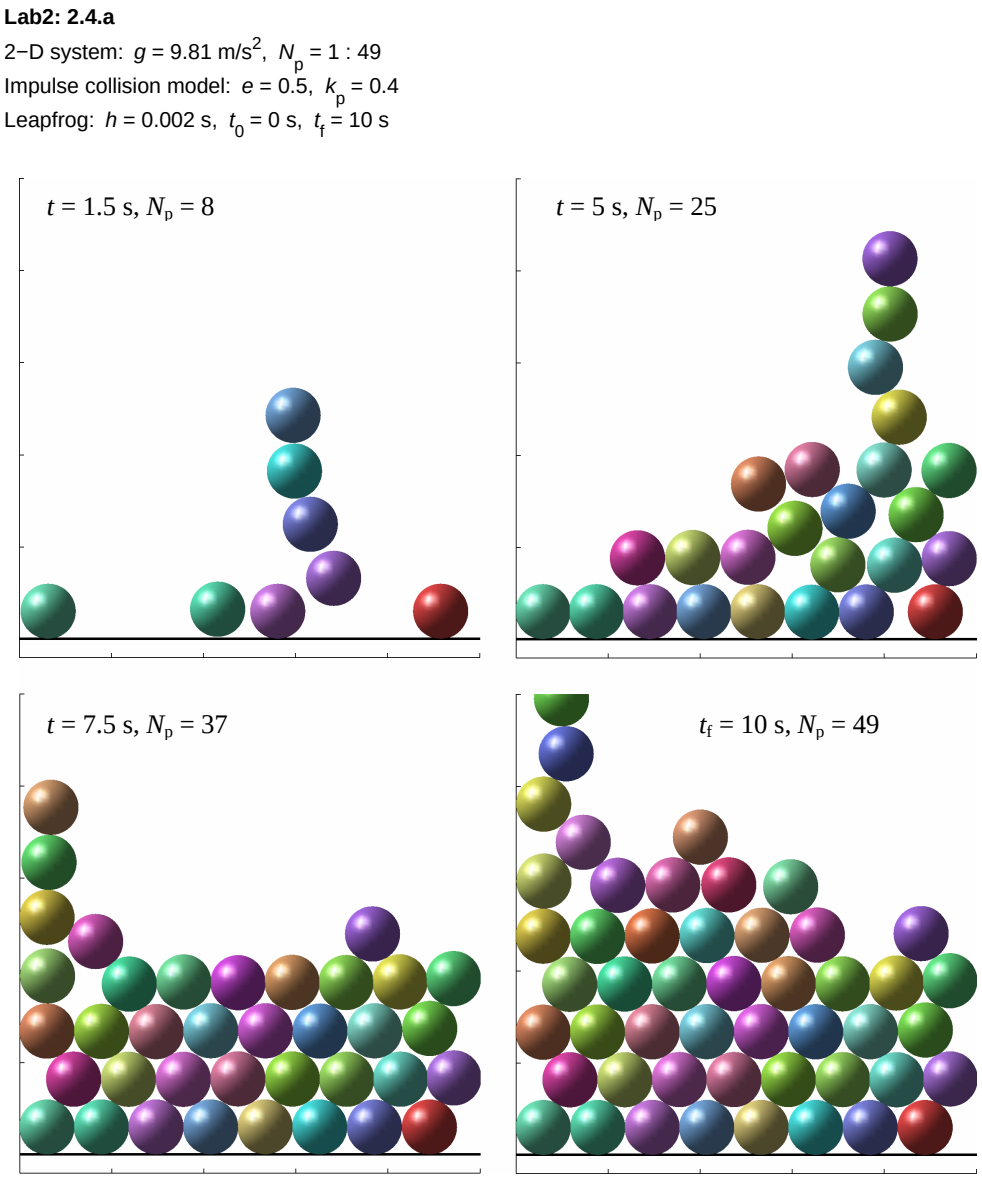


Figure 3.2–13: Snapshots of a piling of particles in 2D at different times simulated using the impulse collision model.

2- and 3-D Implementation of the Linear Spring Method

Snapshots of a piling of particles in a box, simulated using the linear-spring model, are shown on Figure 3.2–14.

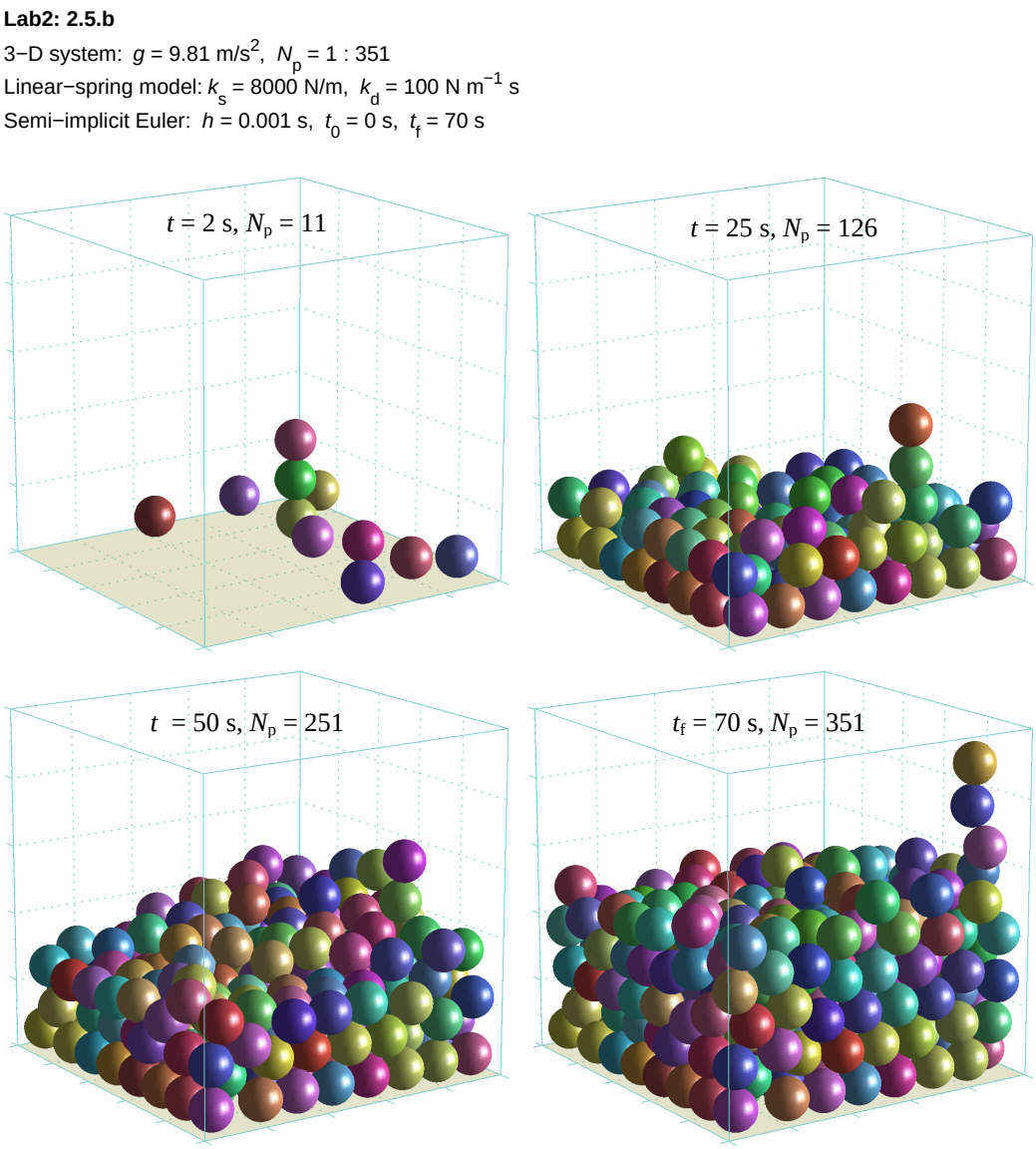


Figure 3.2–14: Snapshots of a piling of particles in 3D at different times simulated using the linear-spring model.

4 Discussion

Conservation of the total energy

The results show that the both contact models preserve the total energy of the system when used with the leapfrog integrator. However, the impulse model loses energy when used with the semi-implicit Euler integrator due to uneven external force computation.

Conservation of the linear momentum

The results show that, in particle-to-particle collisions, the linear momentum of the system is preserved in the both contact models. However, in particle-to-surface contacts with static surfaces, the linear momentum is not preserved in any contact model. The linear momentum would be preserved though, if surfaces had a finite mass and were allowed to move.

Stability of the method

The impulse collision model exhibits jittery behavior in multiple particle collisions, especially if position projections are used with $k_p = 1$. However, this jittery behavior can be reduced by choosing a position projection factor in range $0 < k_p < 0.5$ (as values above or equal 0.5 have tendency to add too much energy the system), and the stability of the many-particle system can be further improved by decreasing integrator time-step size (which increases number of minute projection iterations over contact pairs).

The particle compression problem in the linear-spring contact model can be reduced by increasing spring stiffness. Since the linear-spring model with semi-implicit Euler integrator performs well for time-steps $h < \sqrt{m / k_s}$ and given that in our case we have $m > 0.1$ kg and $h_{soft} = 0.001$ s, the upper limit for the spring stiffness is $k_s < 10^5$ N/m for the method to be stable.

5 References

1. Hut, P. and Makino, J., *The Art of Computational Science, Volume I*, 2004-01-25
Section 4.1: *Two Ways to Write the Leapfrog*
Retrieved 2012-02-28 from: http://www.artcompsci.org/vol_1/v1_web/node34.html