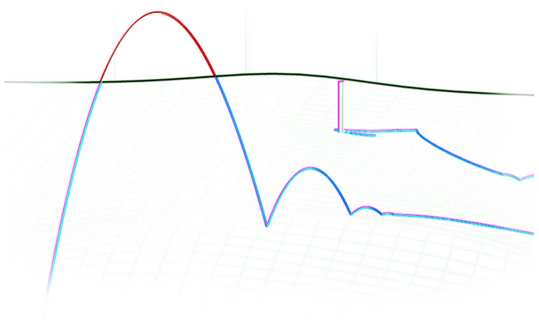
Computer Lab 3
The Golf Game

Mikica Kocic
miko0008@student.umu.se

2012-04-04 (v2)



Mark what exercises you have done in this lab		*Points (by supervisor)*
Basic level (mandatory)	×	
Extra exercise #1: collision with a non-planar terrain height field	×	
Extra exercise #2: decaying of angular velocity by dissipation from the air resistance	×	
Extra exercise #3: inclusion of a constant fluid velocity in viscous forces	×	
Extra exercise #4: GUI that allows menus/mouse interaction	×	
Extra exercise #5: friction in impulse collision model between particles, surfaces and height fld.	×	
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 implement a simple golf simulator that would include most of the relevant physics such as viscous damping forces and Magnus force, with different contact detection models used for both collision response and simulation termination. Advanced level tasks were to include friction and effects of spin on motion, effects of a moving fluid velocity, collisions with a non-planar terrain height field and development of a model for dissipative for angular velocity decay.

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.

Equation (3.11)

Descriptive subscripts in symbols for the viscous damping coefficients were changed from k_L and C_L to k_D and C_D, respectively.

Equation (3.12)

Magnus force coefficients are made physically homogenous using Π-theorem by introducing a reference angular velocity ω_M into k_M term

$$\mathbf{f}^\mathrm{M} = k_\mathrm{M} \, |\mathbf{v}| (\boldsymbol{\omega} \times \mathbf{v}) \tag{L.1}$$

$$k_\mathrm{M} = \frac{1}{2} \rho A \frac{C_\mathrm{M}}{\omega_\mathrm{M}} \tag{L.2}$$

Also, using C_M from literature gives unreasonable values for Magnus force with original set of equations, e.g. $|\mathbf{f}^\mathrm{M}| = 3\,|\mathbf{f}^\mathrm{gravity}|$ at angular velocity as low as $\omega = 10\ \mathrm{s}^{-1}$. This gets even worse in real case scenarios where, for example, a typical angular velocity is around 10 800 rpm ($\omega \simeq 10^3\ \mathrm{s}^{-1}$) for a golf ball with initial velocity of 150 km/h when hit with a 9 iron. Introducing ω_M this way allows also experimentation with a model without touching a given referent Magnus force coefficient C_M.

Equations (5.8) and (5.9)

As described in the earlier lab 2 report, equations are changed to

$$\Delta x_{(a)} = r_{(a)} - (\mathbf{x}_{(a)} - \mathbf{r})^\mathrm{T} \hat{\mathbf{n}} \tag{L.3}$$

$$\Delta x_{(a)} \geq 0 \tag{L.4}$$

Further motivation is given in Appendix B.

Equation (5.25)

Since the expression for unit tangent vector $\hat{\mathbf{t}}_{(ab)}$ in the equation (5.25) actually defines opposite of the tangential component of the velocity $\mathbf{v}$ (i.e. $-\mathbf{v}^\mathbf{t}_{(ab)}$), the equation was corrected into

$$\mathbf{v}^\mathbf{t}_{(ab)} = \hat{\mathbf{n}}_{(ab)} \times (\mathbf{v}_{(ab)} \times \hat{\mathbf{n}}_{(ab)}) \tag{L.5}$$

$$\hat{\mathbf{t}}_{(ab)} = \mathbf{v}^{\mathbf{t}}_{(ab)} / \left|\mathbf{v}^{\mathbf{t}}_{(ab)}\right| \tag{L.6}$$

This also fixes the problem with the invalid sign before the friction coefficient in equation (5.26) and is motivated further in Appendix B.

Note that his implementation uses rather the following equivalent equation of the expression (L.5)

$$\mathbf{v}^{\mathbf{t}}_{(ab)} = \mathbf{v}_{(ab)} - \hat{\mathbf{n}}_{(ab)} \left\langle \hat{\mathbf{n}}_{(ab)}, \mathbf{v}_{(ab)} \right\rangle \tag{L.7}$$

Equations (5.27) and (5.28)

As described in the earlier lab2 report, a position projections factor k_{p} is introduced into the equations. Further motivation is given in Appendix B.

Model for Decaying Angular Velocity

One of the requirements was to extend the model to include decaying angular velocity caused by a drag from the air resistance, which will be addressed here.

While rotating, there is a boundary layer near the surface of the ball where viscous forces are dominant, where the passing fluid causes a drag on the surface causing decay of the angular velocity.

Let us begin by taking a look at the velocities on the surface of the ball, assuming that the velocity $\mathbf{v}$ is normal to the angular velocity $\boldsymbol{\omega}$, as given on the following figure

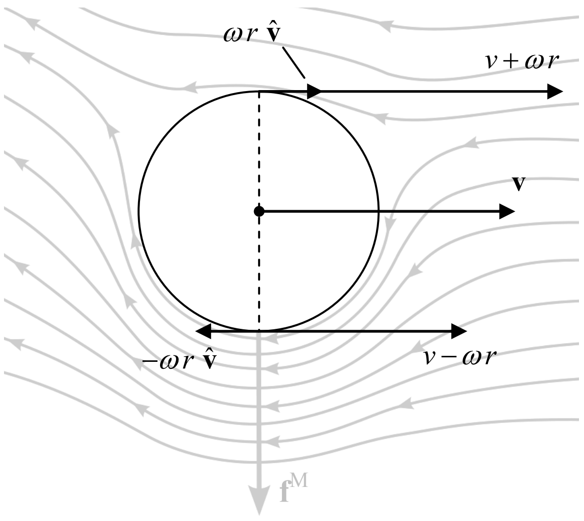

$\omega r\,\hat{\mathbf{v}}$
$v+\omega r$
$\mathbf{v}$
$-\omega r\,\hat{\mathbf{v}}$
$v-\omega r$
$\mathbf{f}^{\mathrm{M}}$

Figure 2.1–1: Velocity and angular velocity of the rotating spherical object.

Let us also assume that the drag force at surface takes a similar form like in equation (3.11) in *Notes*

$$\mathbf{f}^{\mathbf{visc}} = -k_{\mathrm{D}}\,|\mathbf{v}|^{\gamma}\,\hat{\mathbf{v}} = -k_{\mathrm{D}}\,|\mathbf{v}|^{\gamma-1}\,\mathbf{v} \quad \text{where} \quad k_{\mathrm{D}} = \frac{1}{2}\rho A C_{\mathrm{D}} \qquad \textit{Notes}\text{, Eq. (3.11)}$$

Then, approximating case for just two points on the opposite sides of a ball, the net torque is

$$\boldsymbol{\tau} = \mathbf{r}_{\mathrm{top}} \times \mathbf{f}^{\mathrm{visc}}_{\mathrm{top}} + \mathbf{r}_{\mathrm{bot}} \times \mathbf{f}^{\mathrm{visc}}_{\mathrm{bot}} \tag{L.8}$$

from which, assuming turbulent flow i.e. having $\gamma = 2$, we get the net torque

$$\boldsymbol{\tau} = -k_{\mathrm{D}}(v+\omega r)^2 r\,\hat{\boldsymbol{\omega}} + k_{\mathrm{D}}(v-\omega r)^2 r\,\hat{\boldsymbol{\omega}} \tag{L.9}$$

$$\begin{aligned} \tau &= -k_{\mathrm{D}} \cdot \left((v+\omega r)^2 - (v-\omega r)^2\right) \cdot r \\ &= -4k_{\mathrm{D}}\, v\,\omega r^2 \end{aligned} \tag{L.10}$$

On the other side

$$\boldsymbol{\tau} = I\,\dot{\boldsymbol{\omega}} \tag{L.11}$$

where *I* is the moment of inertia of an object. Since, in our case for a sphere, we have

$$I = \frac{2}{5}mr^2 \tag{L.12}$$

combining the last three equations gives

$$\tau = I\dot{\omega} = \left(\frac{2}{5}mr^2\right)\dot{\omega} = -4k_{\mathrm{D}}\,v\,\omega\,r^2 \tag{L.13}$$

or

$$\dot{\omega} = -\frac{10k_{\mathrm{D}}}{m}v\,\omega \tag{L.14}$$

Further substituting k_{D} from Eq. (3.11) gives

$$\dot{\omega} = -5\frac{\rho A C_{\mathrm{D}}}{m}v\,\omega \tag{L.15}$$

which may be finally homogenized with the notation used throughout the *Notes* as

$$\dot{\omega} = -k_{\omega}v\,\omega \tag{L.16}$$

$$k_{\omega} = \frac{1}{2}\frac{\rho A}{m}C_{\omega} \tag{L.17}$$

where C_{ω} is an angular velocity damping coefficient that depends on geometrical shape (including moment of inertia!) and type of surface of an object. Variables has the following physical dimensions: $[C_{\omega}] = 1$, $[k_{\omega}] = \mathsf{L}^{-1}$ and $[\dot{\omega}] = \mathsf{T}^{-2}$.

In a similar manner, we can derive equations for $\gamma = 1$ that produces (L.16) but without the term $4 \cdot v$. The compound equation in scalar form is then

$$\dot{\omega} = -k_{\omega}v^{\gamma-1}\,\omega \tag{L.18}$$

limited only to $\gamma = 1$ and $\gamma = 2$ values and with an assumption that C_{ω} is also dependent on γ.

The last equation is derived for case when **v** is normal to **ω**. In more general case, we should use a normal projection of **v** to **ω** instead of **v**, i.e. $v = |\mathbf{v} \times \hat{\boldsymbol{\omega}}|$, in which case the last equation becomes

$$\dot{\boldsymbol{\omega}} = -k_{\omega}\left|\mathbf{v} \times \hat{\boldsymbol{\omega}}\right|^{\gamma-1}\boldsymbol{\omega}\,. \tag{L.19}$$

However, more thorough analysis (presented in Appendix C) yields the following expression for angular velocity decay

$$\dot{\boldsymbol{\omega}} = -k_{\omega}\left(|\mathbf{v}| + R|\boldsymbol{\omega}|\right)^{\gamma-1}\boldsymbol{\omega}\,. \tag{L.20}$$

2.2 Implementation

The implemented model is continuation of the earlier simulation framework developed for spherical particle collisions in computer lab 2. The major differences are, first, that the new implementation is object oriented with the physical model encapsulated in a 'world of particles' (WoP) class, and second, that the most critical part of the main loop was moved to a MEX function and implemented in C++. Individual files of the implementation are summarized in the following table with a short description, where more important files are highlighted.

File Name	***Description***
`@WoP`	Folder with the implementation of the WoP class.
`@WoP/WoP.m`	Declaration of the WoP class. Contains also definitions of the WoP constructor and destructor.
`@WoP/Simulate.m`	Implements the simulation engine that simulates and visualizes a system of colliding particles. Calls MEX function `WoP_Solver` for the most critical part.
`@WoP/ResetSimulation.m`	Resets initial time to 0 and removes final state variables.
`@WoP/GetConfigIDs.m`	Returns a list of recognized configuration profiles.
`@WoP/loadConfiguration.m`	Configures a simulation according to selected profile.
`@WoP/GenerateHeightField.m`	Generates a random discrete height field grid.
`@WoP/Envelope.m`	Finds upper and lower envelopes of a given signal.
`@WoP/ExportAnnotation.m`	Saves a tagged annotation from a figure to an EPS file and removes annotation from the figure.
`@WoP/ExportFigure.m`	Saves a figure to a file like 'print' function but fixing object's transparencies and used PS fonts.
`@WoP/diminishDimensions.m`	Diminishes dimensions of a matrix of spatial vectors.
`@WoP/fixPsFonts.m`	Fixes matlab's usage of 'base 16' PS fonts in EPS file.
`@WoP/randomFaceColor.m`	Generates a random particle face color and alpha.
`WoP_Solver.cpp` `WoP_Solver.mex*`	The critical part of the `@WoP/Simulate` method (in the main loop) which is implemented as MEX 'nested' function. ***Simulation algorithms are implemented here!***
`WoP_Matrix.h`	The `Matrix` class used by the `WoP_Solver` function.
`WoP_Eval.cpp` `WoP_Eval.mex*`	The `Matrix` class method evaluator to be used with a regression tester.
`WoP_RegressionTests.m`	Regression tests for the WoP `Matrix` class
`make.m`	Rebuilds MEX binaries
`lab3.m` `lab3.fig`	GUI application that allows user to configure the properties of WoP objects interactively and run a series of simulations.
`playGolfC.m`	Command window application for the Golf Game (basic level task).
`playGolf.m` `playGolf.fig`	Simple GUI application for the Golf Game (advanced level task)

The recommended simulation algorithm for using the `WoP` class (also implemented in `lab3`, `playGolfC` and `playGolf` modules) is:

```
(1)  obj = WoP( 'profile' );               % Instantiate the WoP object
     obj.N_dim = ...; obj.X_0 = ...        % Setup the simulation parameters
     obj.X_0 = ...; obj.V_0 = ...          % Modify some simulation parameters, optionally
(2)  obj.Simulate;                         % Start the simulation
     ... = obj.X; ... = obj.V;             % Use the results, optionally
     obj.X = ...; obj.V = ...              % Modify some simulation parameters, optionally
(2)  obj.Simulate;                         % Start the simulation
     ... = obj.X; ... = obj.V;             % Use the results, optionally
                                           % and so on...
```

2.3 Compiling the Code

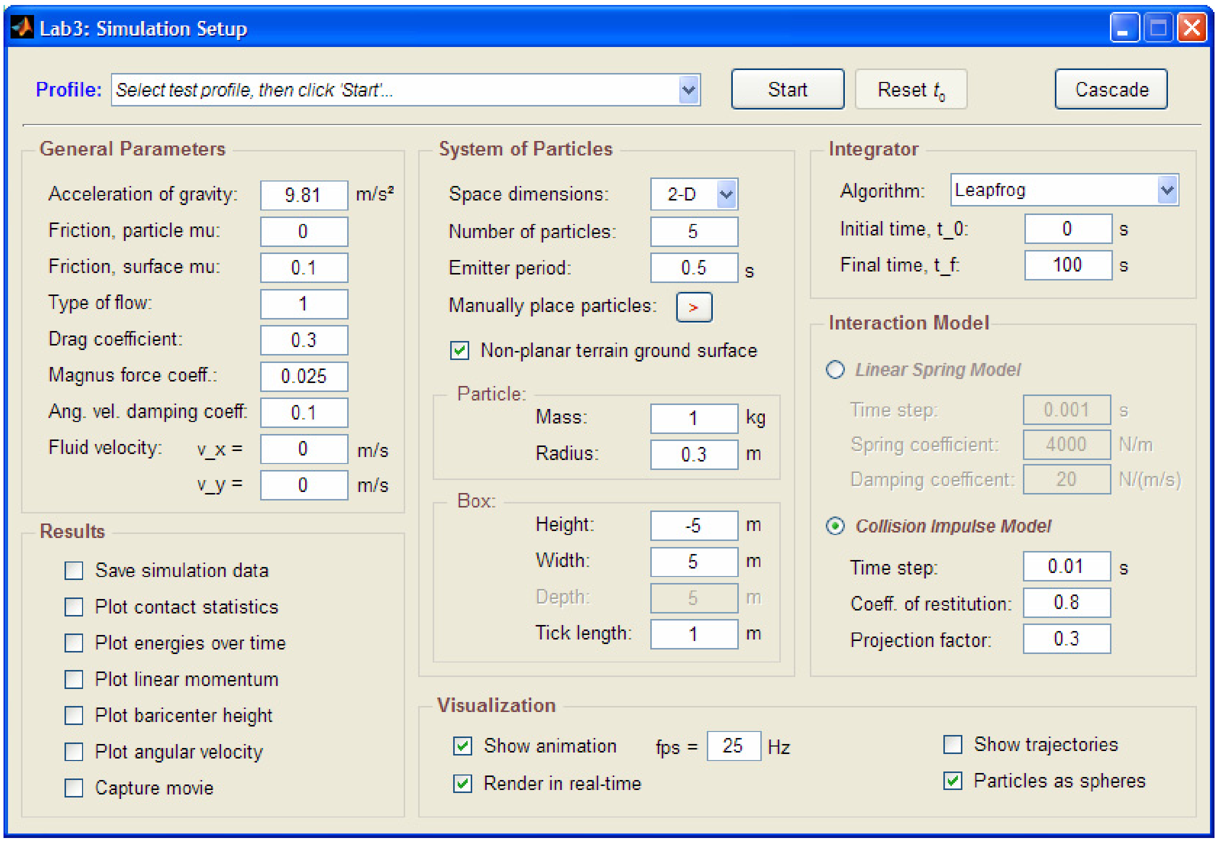
MEX source files need to be compiled before executing simulations.

The compilation is tested with MS VS 2010 (32/64-bit), 2008 (32-bit) and VC6.0 compilers.
The default distribution contains `mexw32` and `mexw64` binaries compiled with MS VS 2010 compiler.

To rebuild binaries, just execute **`make`** m-file.

2.4 Running Test Simulations

1) Execute `lab3` from the MATLAB command prompt. The gui should appear:


Lab3: Simulation Setup

Profile: Select test profile, then click 'Start'... | Start | Reset t_0 | Cascade

General Parameters

Acceleration of gravity:	9.81	m/s²
Friction, particle mu:	0	
Friction, surface mu:	0.1	
Type of flow:	1	
Drag coefficient:	0.3	
Magnus force coeff.:	0.025	
Ang. vel. damping coeff:	0.1	
Fluid velocity: v_x =	0	m/s
v_y =	0	m/s

Results

☐ Save simulation data
☐ Plot contact statistics
☐ Plot energies over time
☐ Plot linear momentum
☐ Plot baricenter height
☐ Plot angular velocity
☐ Capture movie

System of Particles

Space dimensions:	2-D	
Number of particles:	5	
Emitter period:	0.5	s
Manually place particles:	>	

☑ Non-planar terrain ground surface

Particle:

Mass:	1	kg
Radius:	0.3	m

Box:

Height:	-5	m
Width:	5	m
Depth:	5	m
Tick length:	1	m

Integrator

Algorithm:	Leapfrog	
Initial time, t_0:	0	s
Final time, t_f:	100	s

Interaction Model

○ *Linear Spring Model*

Time step:	0.001	s
Spring coefficient:	4000	N/m
Damping coefficent:	20	N/(m/s)

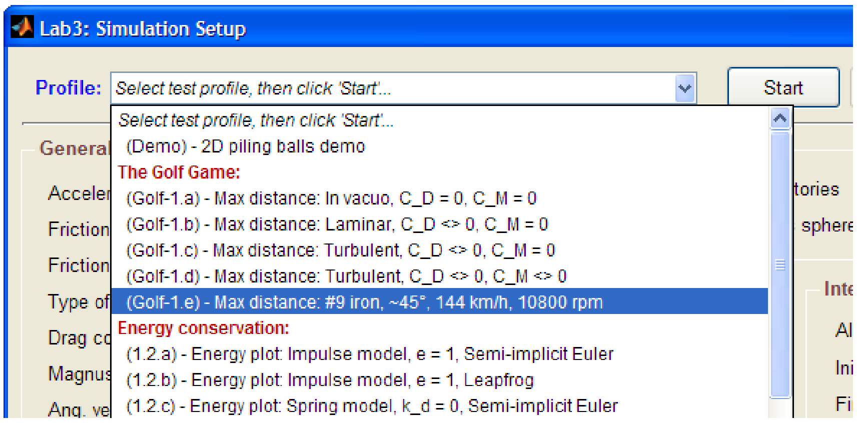
● *Collision Impulse Model*

Time step:	0.01	s
Coeff. of restitution:	0.8	
Projection factor:	0.3	

Visualization

☑ Show animation fps = 25 Hz
☑ Render in real-time
☐ Show trajectories
☑ Particles as spheres

2) Select the configuration profile:


Lab3: Simulation Setup

Profile: Select test profile, then click 'Start'... | Start

Select test profile, then click 'Start'...
(Demo) - 2D piling balls demo
The Golf Game:
(Golf-1.a) - Max distance: In vacuo, C_D = 0, C_M = 0
(Golf-1.b) - Max distance: Laminar, C_D <> 0, C_M = 0
(Golf-1.c) - Max distance: Turbulent, C_D <> 0, C_M = 0
(Golf-1.d) - Max distance: Turbulent, C_D <> 0, C_M <> 0
(Golf-1.e) - Max distance: #9 iron, ~45°, 144 km/h, 10800 rpm
Energy conservation:
(1.2.a) - Energy plot: Impulse model, e = 1, Semi-implicit Euler
(1.2.b) - Energy plot: Impulse model, e = 1, Leapfrog
(1.2.c) - Energy plot: Spring model, k_d = 0, Semi-implicit Euler

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).

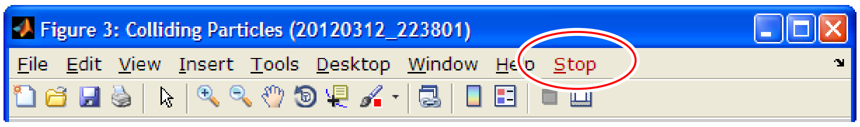


Figure 3: Colliding Particles (20120312_223801)
File Edit View Insert Tools Desktop Window Help Stop

2.5 Running the Golf Game

Console Mode Application

Execute **`playGolfC`** at the MATLAB command prompt. The simulation should start and place a golf ball on a random position.

Before each shot, the user will be informed about the current wind and prompted to enter horizontal direction, initial velocity, club loft and ball spin, for example (user input is highlighted in blue):

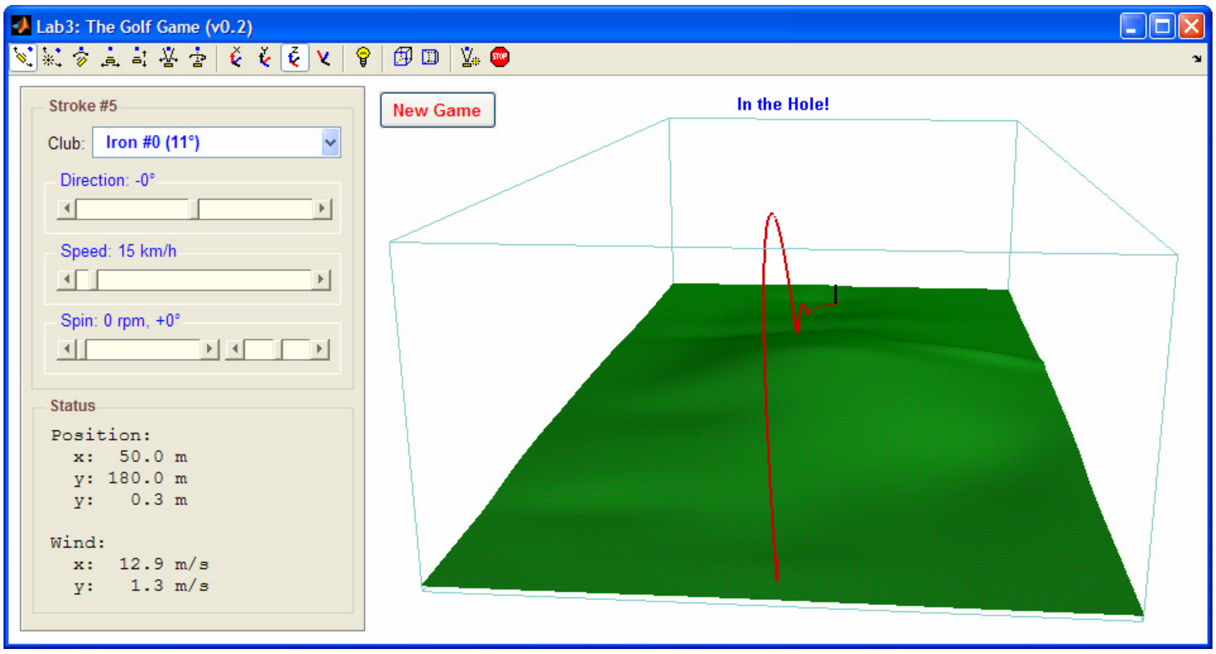
```
...
----------------------------------------------------------
Wind =     5.53471   -1.36099         0
X    =     50.0001    3.50375   0.413924
----------------------------------------------------------
Note: 0 degrees is along depth (Y-axis direction)
----------------------------------------------------------
Direction (degrees) ? -10
Velocity  (km/h)    ? 54
Loft      (degrees) ? 23
Spin      (rpm)     ? 3600
```

Application will cycle until the ball is put in hole or goes out of bounds.

GUI Application

Execute **`playGolf`** at the MATLAB command prompt. The simulation should start and place a golf ball on a random position.

The player has opportunity, besides to select a club from a combo box and modify other parameters before hitting the ball, to change also the camera view of the golf field (even during simulation while the ball is flying).



Stroke #5

Club: Iron #0 (11°)

Direction: -0°

Speed: 15 km/h

Spin: 0 rpm, +0°

Status

```
Position:
  x:  50.0 m
  y: 180.0 m
  y:   0.3 m

Wind:
  x:  12.9 m/s
  y:   1.3 m/s
```

Figure 2.5–1: The snapshot of the GUI application.

3 Results

3.1 Basic Level Tasks

Maximum Distance

Maximum distance dependence on the initial velocity angle (club loft) under different simulation models (with or without dissipation or Magnus force) was analyzed.

The simulated trajectories (with the separation angle between different trajectories of 0.5°**)** under different conditions are shown on Figures 3.1-1 to 3.1-4 and the results are summarized in Table 3.1-1. The results are also verified using Wolfram Mathematica and presented in Appendix A.

Table 3.1-1: Dependence of the maximum distance on the initial conditions.

Conditions	v_0 /(m/s)	***angle*** /°	ω_0 /(rad/s)	γ	C_D	C_M	***distance*** /m
Reference solution (in vacuo)	80	45.0	0	-	0	0	652
Laminar flow, with drag but without Magnus force	80	44.5	0	1	0.3	0	626
Turbulent flow, with drag but without Magnus force	80	36.5	0	2	0.3	0	212
Turbulent flow, with drag and with Magnus force	80	11.5	10	2	0.3	0.025	244

The reference simulation shows that maximum distance is reached at 45 degrees angle, as expected.

Simulation for the laminar flow shows negligent differences in change of the angle and maximum distance when compared to the reference solution.

Simulations for turbulent flow without Magnus forces, when compared to the both laminar flow and the reference solution, shows a significant drop in maximum distance for 58 % that is reached at angle decreased for 20 %. This shows significant impact of the drag force on the maximum distance.

Simulations for turbulent flow with Magnus forces shows increase of 20 % in maximum distance at further reduced angle at 33 %. This shows how the Magnus effect extends maximum distance as increasing the lift force.

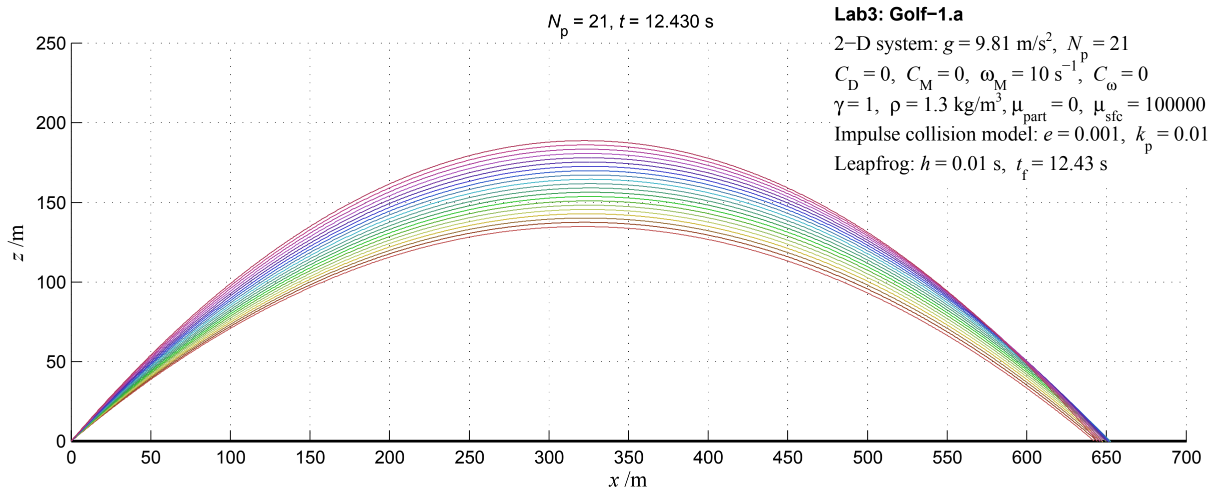


Figure 3.1–1: Reference trajectories for v_0 = 80 m/s and different initial angles 40°-50°, *in vacuo*

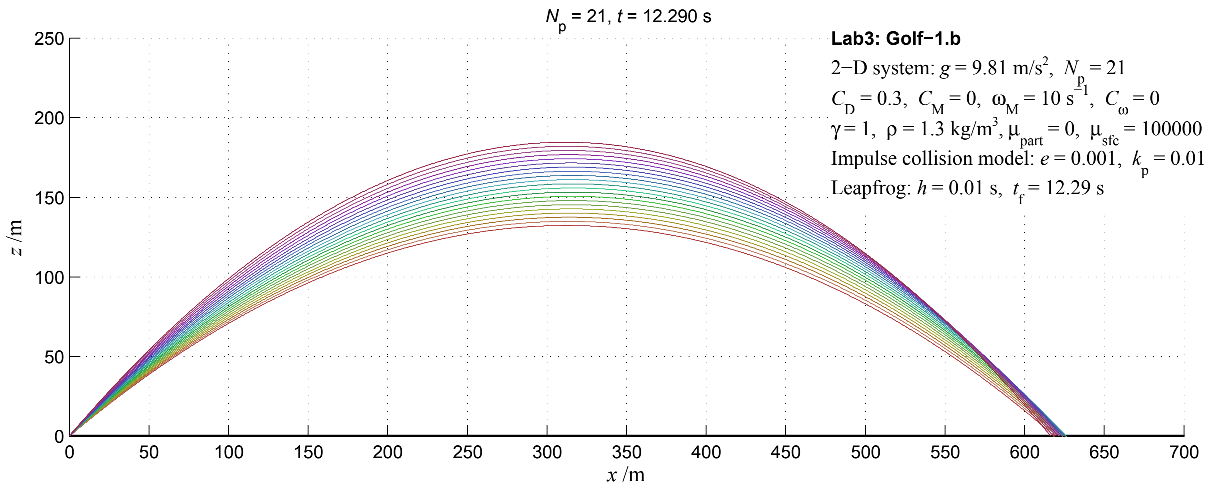


Figure 3.1–2: Trajectories for v_0 = 80 m/s and different initial angles 40°-50°, for laminar flow with air drag, without Magnus force

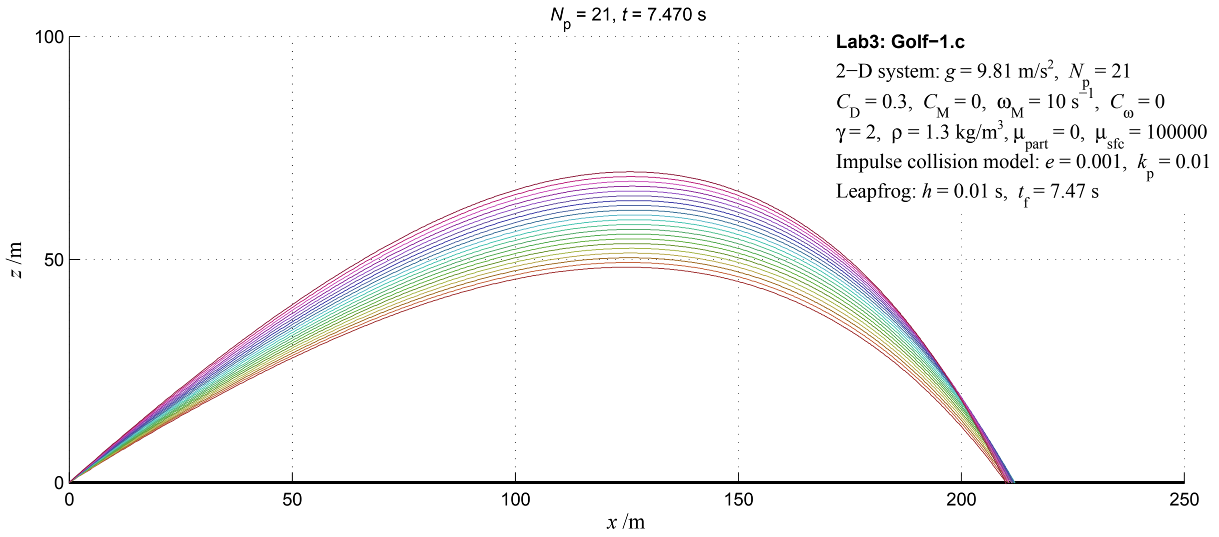


Figure 3.1–3: Trajectories for v_0 = 80 m/s and different initial angles 32°-42°, for turbulent flow with air drag, without Magnus force.

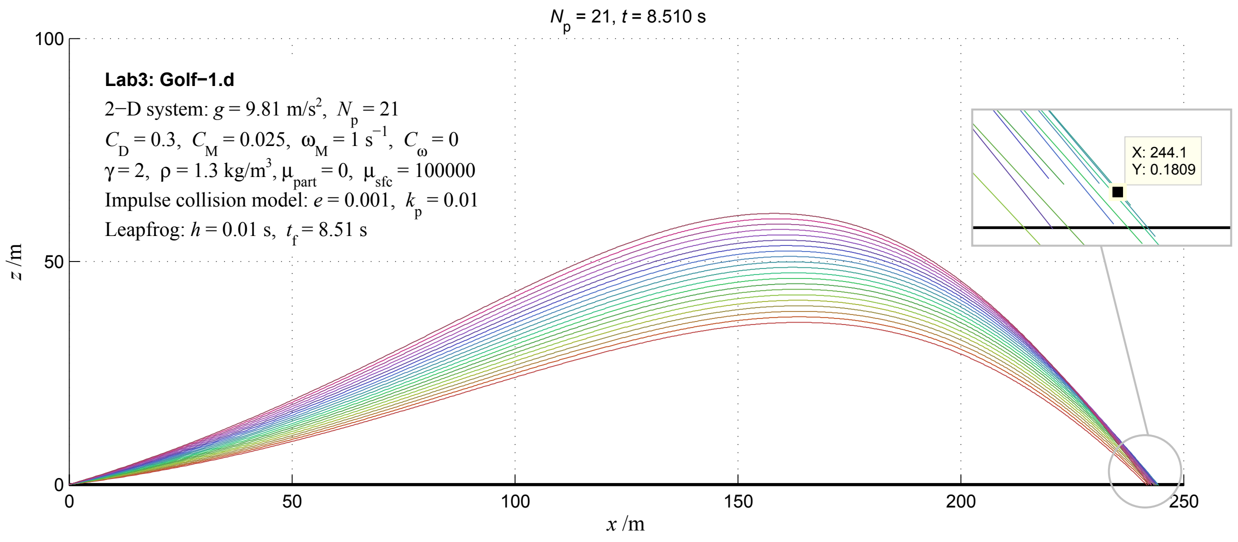


Figure 3.1–4: Trajectories for v_0 = 80 m/s and different initial angles 7°-17°, for turbulent flow with air drag, with Magnus force, initial angular velocity ω_0 = 10 rad/s and without angular velocity decay.

Example Simulation

Example simulation producing a trajectory where the ball stops in the hole is given on Figure 3.1–5.

The ball was hit 5 times with the following initial conditions:

- direction = 0°, v = 40 km/h, loft = 45°, spin = 0 rpm
- direction = 0°, v = 120 km/h, loft = 40°, spin = 600 rpm
- direction = −90°, v = 40 km/h, loft = 10°, spin = 0 rpm
- direction = −90°, v = 30 km/h, loft = 10°, spin = 0 rpm
- direction = 130°, v = 5 km/h, loft = 0°, spin = 0 rpm

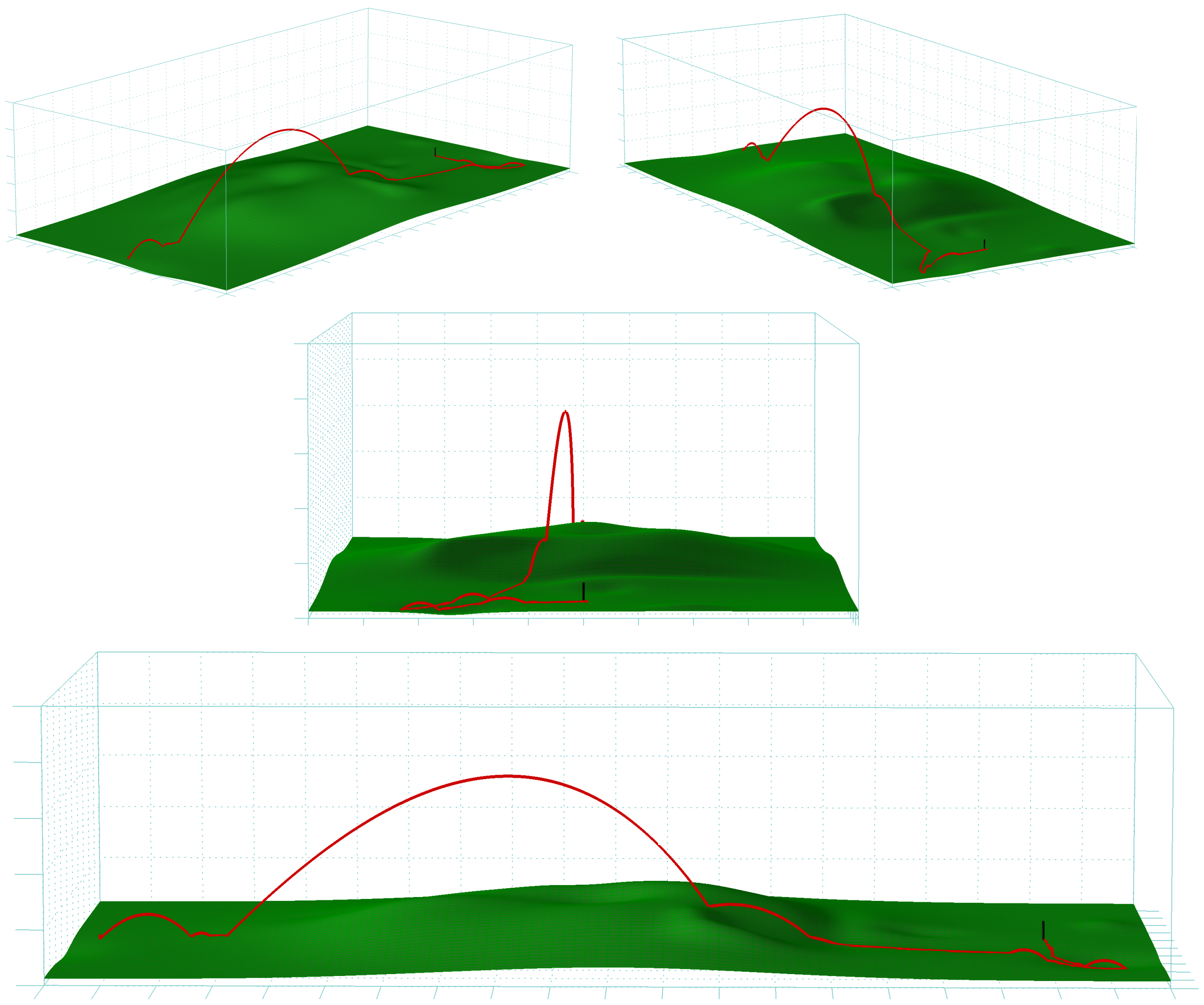


Figure 3.1–5: Snapshots of the simulation producing a trajectory where the ball stops in the hole

3.2 Advanced Level Tasks

Non-planar Terrain Height Field

Simulations demonstrating collisions with a non-planar terrain height, beside the previous simulation shown on Figure 3.1-6, are shown on the following two figures:

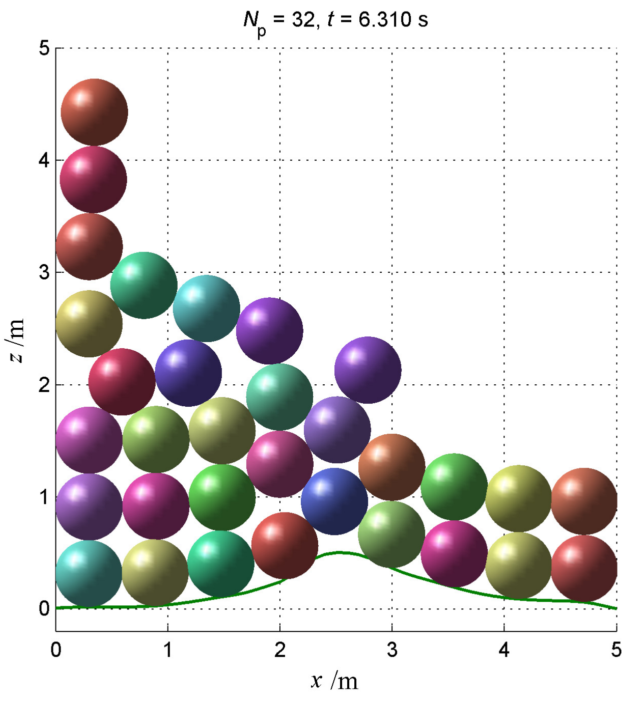


Figure 3.2–1: Simulation snapshot showing collisions with non-planar terrain height field in 2D.

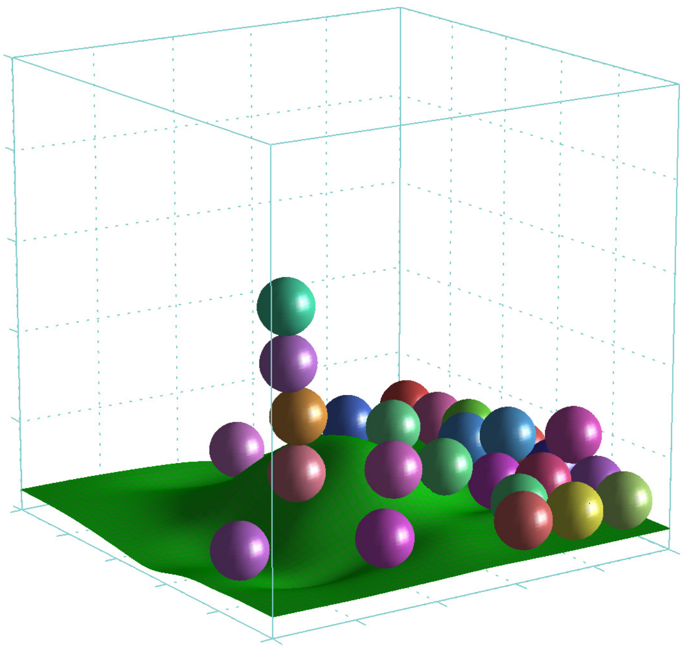


Figure 3.2–2: Simulation snapshot showing collisions with non-planar terrain height field in 3D.

Decaying Angular Velocity

The model for decaying angular velocity is presented in the *Theory* section. The graph displaying decaying angular velocity *of the first golf ball* is shown on the top of the following Figure 3.2–3. Figure which shows the snapshot of simulation where the ball was hit with iron #9 at initial velocity 144 km/h and initial spin 10800 rpm.

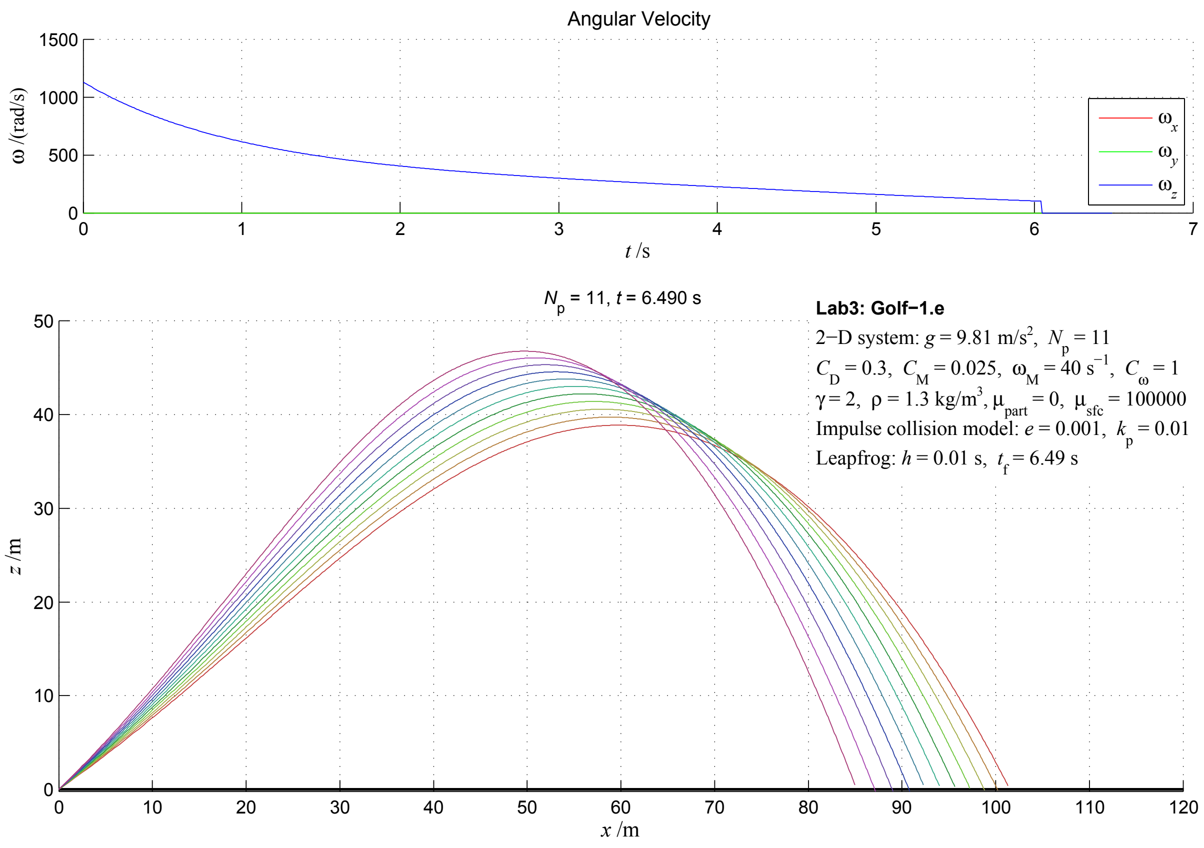


Figure 3.2–3: Decaying angular velocity of the ball example for $C_\omega = 1$.
Trajectories for simulation with iron #9 hit at varied initial angles with initial ball velocity v_0 = 144 km and spin 10 800 rpm, turbulent flow with air drag, with Magnus force and with angular velocity decay included.

Impact of Fluid Velocity on Dissipative Forces

All dissipative forces in a fluid depend on the *relative* velocity of the object to the fluid. Equations for viscous drag force, Magnus force and angular velocity decay should be changed accordingly by replacing velocities with relative velocities to the fluid, where the relative velocity is calculated as $\mathbf{v}_{\text{rel}} = \mathbf{v} - \mathbf{v}_{\text{fluid}}$. This means that the relative velocity is zero when object and fluid are traveling in the same direction at the same speed. The resulting equations are then:

$$\mathbf{f}^{\text{M}} = k_{\text{M}}\,|\mathbf{v}_{\text{rel}}|\left(\boldsymbol{\omega} \times \mathbf{v}_{\text{rel}}\right) \tag{L.21}$$

$$\mathbf{f}^{\text{visc}} = -k_{\text{D}}\,|\mathbf{v}_{\text{rel}}|^{\gamma-1}\,\mathbf{v}_{\text{rel}} \tag{L.22}$$

$$\dot{\boldsymbol{\omega}} = -k_{\omega}\left(|\mathbf{v}_{\text{rel}}| + R|\boldsymbol{\omega}|\right)^{\gamma-1}\boldsymbol{\omega} \tag{L.23}$$

The snapshots of the simulations showing an impact of the wind speed are shown on the following two figures:

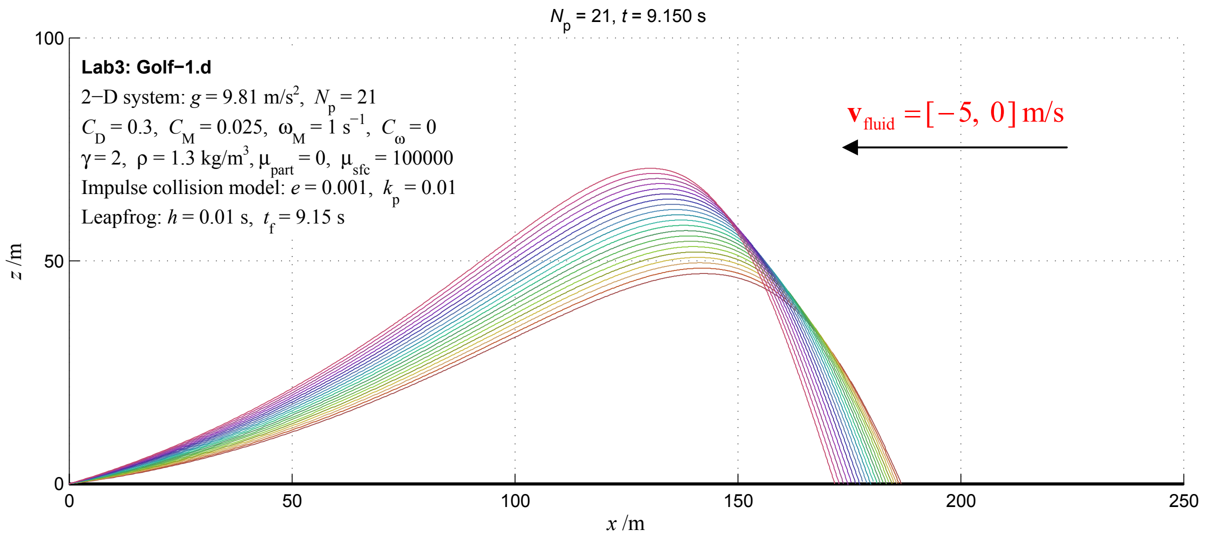


Figure 3.2–4: Trajectories for balls facing a headwind 5 m/s. Shots are with initial velocity v_0 = 80 m/s and varied different initial angles 7°-17°, for turbulent flow with air drag and Magnus force but without angular velocity decay.

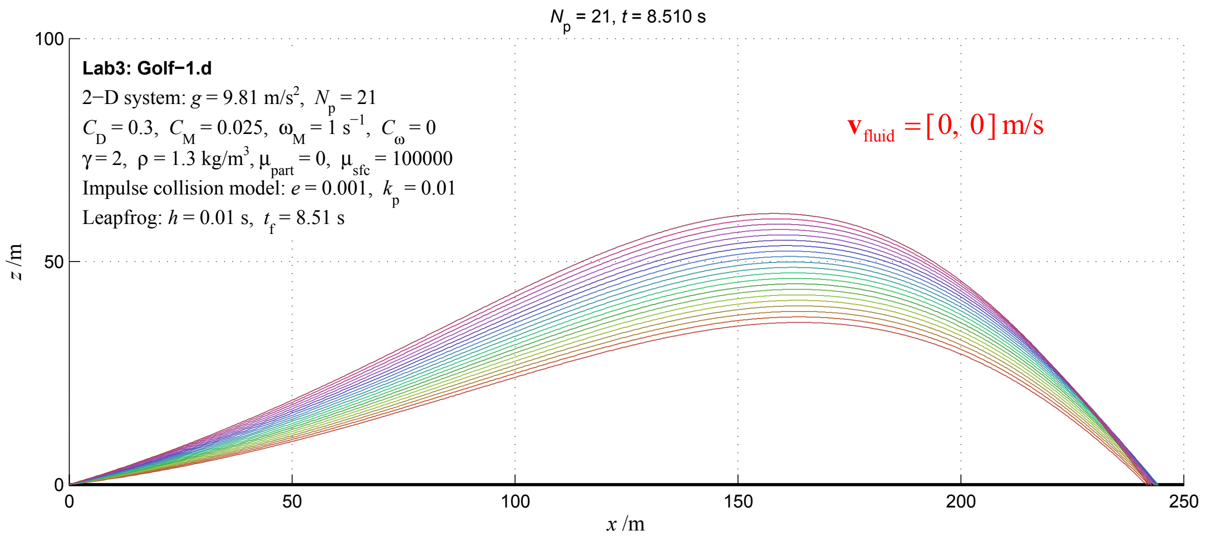


Figure 3.2–5: Trajectories for balls facing no wind. Shots are with initial velocity v_0 = 80 m/s and varied different initial angles 7°-17°, for turbulent flow with air drag and Magnus force but without angular velocity decay.

Effects of Friction on Collision

The snapshot of the simulation showing a ball on a non-planar terrain without friction is shown on Figure 3.2–6. The snapshot of the simulation with the same initial conditions but with friction between particle and the surface (for $\mu = 0.4$) is shown on Figure 3.2–7. Note how particle stops on an inclined part of the terrain on the later figure.

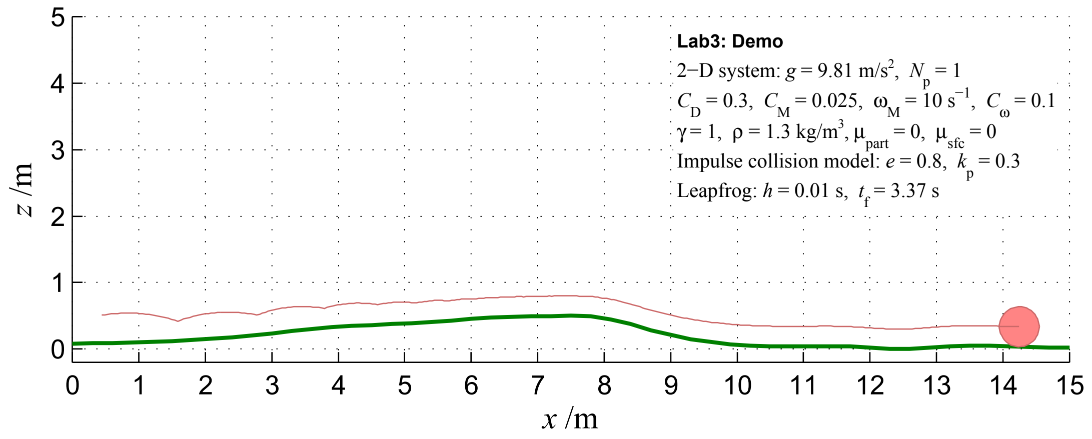


Figure 3.2–6: A ball jumping on a non-planar surface without a friction.

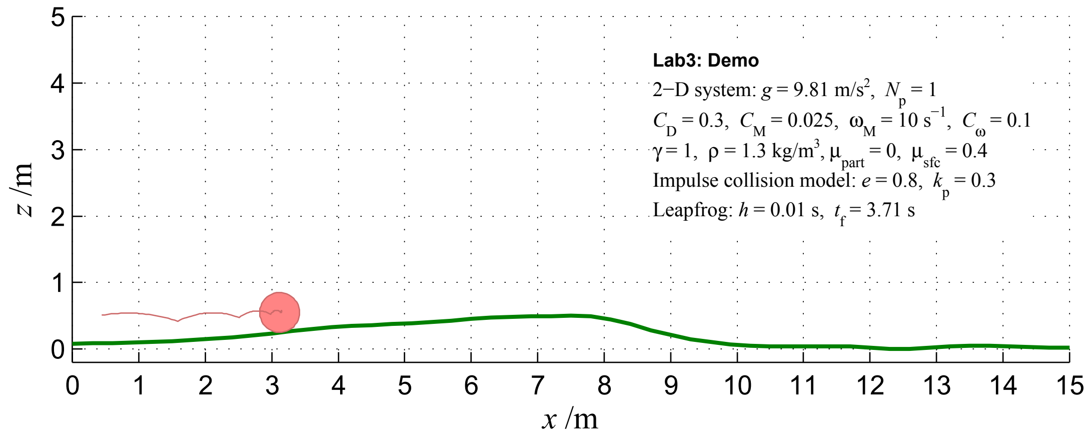


Figure 3.2–7: A boll jumping on a non-planar surface with a friction.

GUI Control Application

The snapshot of the GUI control application is shown on Figure 2.5–1 in *Running the Golf Game* section.

4 Discussion

The results emphasize an importance of verification of physical models and a high sensitivity of a physical model on values, and accepted ranges of values, of physical constants and properties emerged during a modeling of the physical system.

Lab 3 Appendix A: Max Distance

Parameters

```
Remove["`*"]; (* Remove all global symbols *)
```

Physical Constants

```
g := {0, -9.81}; (* Acceleration of gravity, m/s^2 *)
```

```
γ := 1; (* Type of flow, 1 = laminar *)
```

```
ρ := 1.3; (* Fluid mass density, kg/m^3 *)
```

```
CD := 0.3; (* Drag coefficient *)
```

Object Properties

```
m := 0.045; (* Particle mass, kg *)
```

```
r := 0.02; (* Particle radius, m *)
```

Initial Conditions

```
t0 = 0; (* Initial time *)
```

```
tf = 12; (* Final time *)
```

```
x0 := {0, 0}; (* Initial position, m *)
```

```
v0 := 80; (* Initial velocity, m/s *)
```

Forces and Energy

Total Force

$$F[x\_?VectorQ, v\_?VectorQ] := m\,g - kD\,Norm[v]^{\gamma}\,\frac{v}{Norm[v]};$$

where

$$kD := \frac{1}{2}\,CD\,\rho\,A; \text{ (* Viscous damping coefficient *)}$$

$$A := r^2\,\pi; \text{ (* Cross section area, } m^2 \text{ *)}$$

Potential and Kinetic Energy

```
Ep[x_?VectorQ] := -m g.x;
```

$$Ek[v\_?VectorQ] := \frac{m\,v.v}{2};$$

Equations of Motion

```
SolveODE[
   t0_, tf_,
   x0_?VectorQ, v0_?VectorQ,
   method_: Automatic, h_: Automatic
  ] := Module[
   {X, V, sol},
   sol = NDSolve[
     {
      (* Differential Equations: *)
          X'[t] == V[t],
          m V'[t] == F[X[t], V[t]],
      (* Initial conditions: *)
          X[t0] == x0,
          V[t0] == v0
     },
     {X, V}, (* Dependent variables *)
     {t, t0, tf} (* Range of the independent variable *)
     , Method → method
     , StartingStepSize → h
     , MaxStepSize → h
     , MaxSteps → Infinity
    ];
   ({X, V} /. sol)[[1]]
  ];
```

Solve Equations

Subroutine that solves ode using Runge-Kutta 4th order with $h = 0.01$ for a given angle

```
FindMax[θ_] := Module[
   {X, V, tmax},
   {X, V} = SolveODE[t0, tf, x0,
     {v0 Cos[θ π / 180], v0 Sin[θ π / 180]}, "ExplicitRungeKutta", 0.01];
   tmax = t /. FindRoot[X[t][[2]] == 0, {t, 10, tf}] ;
   X[tmax][[1]]
  ];
```

Find max distance without viscous damping (reference solution)

```
CD = 0.0;
MaxXref = Quiet[Table[{θ, FindMax[θ]}, {θ, 40, 50, 0.2}]];
```

Find max distance with viscous damping enabled, laminar flow

```
CD = 0.3; γ := 1;
MaxXvisc = Quiet[Table[{θ, FindMax[θ]}, {θ, 40, 50, 0.2}]];
```

Find max distance with viscous damping enabled, turbulent flow

```
CD = 0.3; γ := 2;
MaxXvisc2 = Quiet[Table[{θ, FindMax[θ]}, {θ, 32, 42, 0.2}]];
```

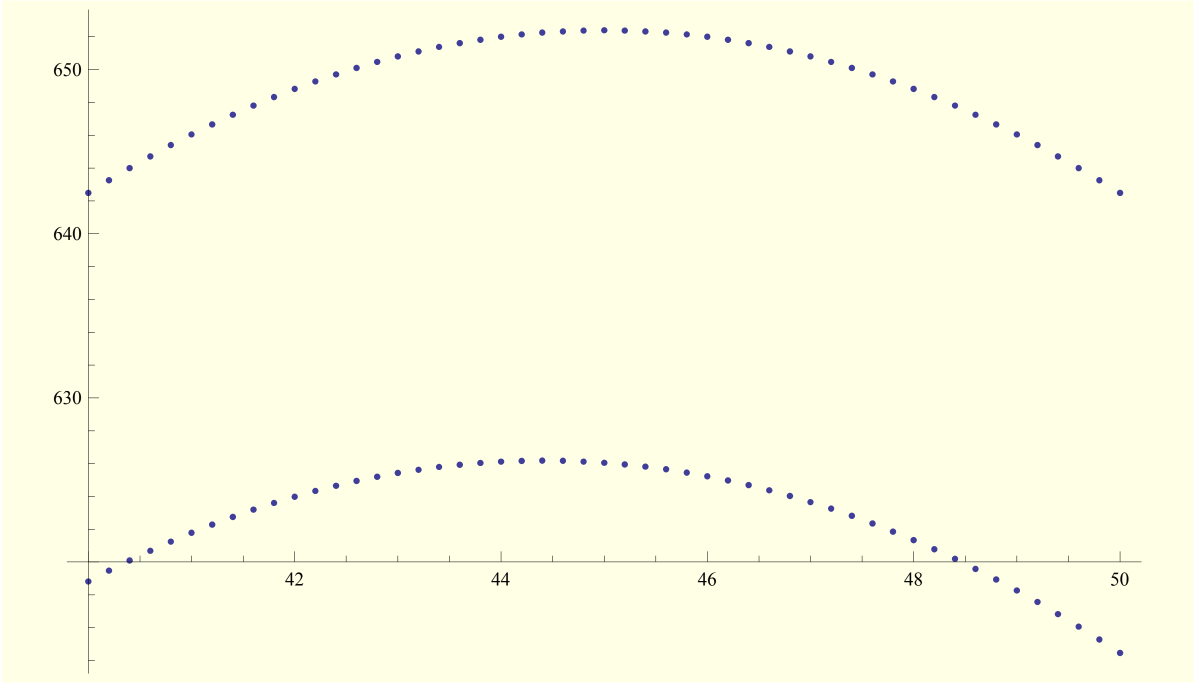
The Results

```
MaxX = Transpose[{
    MaxXvisc[[All, 1]], MaxXvisc[[All, 2]], MaxXref[[All, 2]],
    Table["    ", {i, 0, 10, 0.2}],
    MaxXvisc2[[All, 1]], MaxXvisc2[[All, 2]]
   }] // TableForm
```

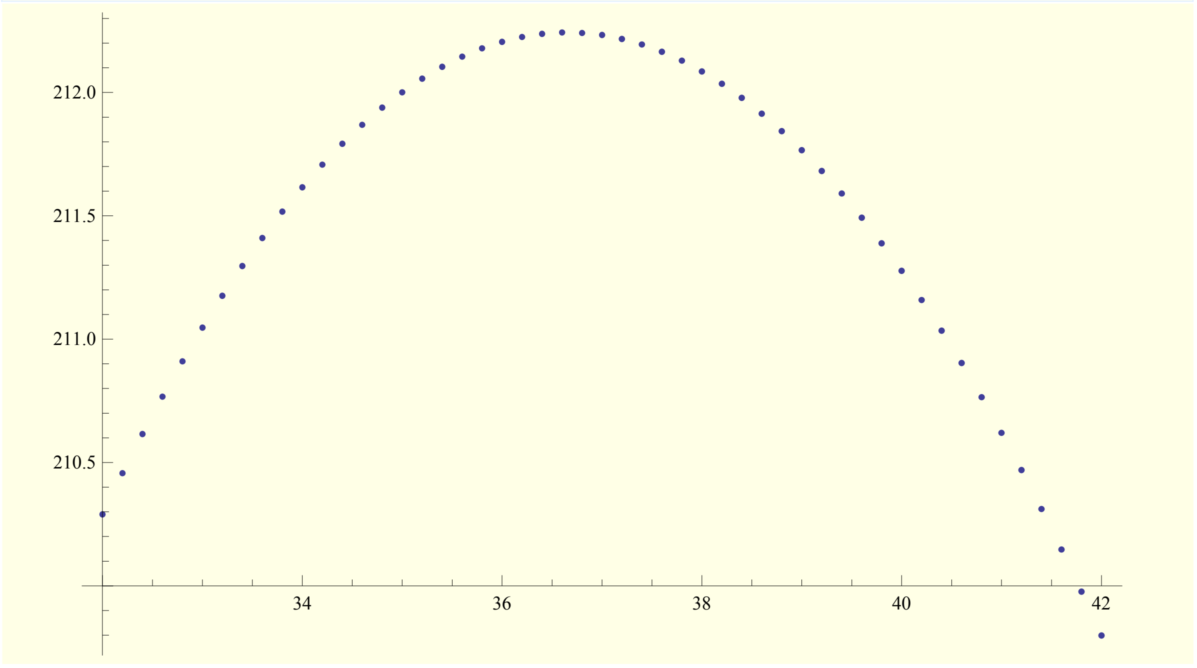
```
40.   618.821  642.484       32.   210.29
40.2  619.473  643.259       32.2  210.457
40.4  620.094  644.003       32.4  210.616
40.6  620.685  644.716       32.6  210.767
40.8  621.246  645.397       32.8  210.911
41.   621.777  646.046       33.   211.047
41.2  622.278  646.665       33.2  211.176
41.4  622.748  647.251       33.4  211.297
41.6  623.189  647.806       33.6  211.41
41.8  623.599  648.33        33.8  211.517
42.   623.979  648.822       34.   211.616
42.2  624.328  649.282       34.2  211.707
42.4  624.648  649.711       34.4  211.792
42.6  624.937  650.107       34.6  211.869
42.8  625.196  650.473       34.8  211.938
43.   625.425  650.806       35.   212.001
43.2  625.623  651.108       35.2  212.056
43.4  625.791  651.378       35.4  212.104
43.6  625.929  651.617       35.6  212.145
43.8  626.037  651.823       35.8  212.179
44.   626.114  651.998       36.   212.206
44.2  626.162  652.141       36.2  212.225
44.4  626.179  652.252       36.4  212.238
44.6  626.165  652.332       36.6  212.243
44.8  626.122  652.38        36.8  212.241
45.   626.048  652.396       37.   212.233
45.2  625.945  652.38        37.2  212.217
45.4  625.811  652.332       37.4  212.195
45.6  625.647  652.252       37.6  212.165
45.8  625.452  652.141       37.8  212.129
46.   625.228  651.998       38.   212.085
46.2  624.973  651.823       38.2  212.035
46.4  624.689  651.617       38.4  211.978
46.6  624.374  651.378       38.6  211.914
46.8  624.03   651.108       38.8  211.844
47.   623.655  650.806       39.   211.766
47.2  623.251  650.473       39.2  211.682
47.4  622.816  650.107       39.4  211.591
47.6  622.352  649.711       39.6  211.493
47.8  621.857  649.282       39.8  211.388
48.   621.333  648.822       40.   211.277
48.2  620.779  648.33        40.2  211.159
48.4  620.195  647.806       40.4  211.035
48.6  619.582  647.251       40.6  210.903
48.8  618.939  646.665       40.8  210.765
49.   618.266  646.046       41.   210.621
49.2  617.563  645.397       41.2  210.47
49.4  616.831  644.716       41.4  210.312
49.6  616.069  644.003       41.6  210.147
49.8  615.278  643.259       41.8  209.976
50.   614.458  642.484       42.   209.799
```

Maximimum length (in meters) as a function of initial velocity angle (in degrees):

```
Show[ListPlot[Table[ MaxXref[[i, All]], {i, 1, 51}]],
 ListPlot[Table[ MaxXvisc[[i, All]], {i, 1, 51}]]]
```



```
ListPlot[Table[MaxXvisc2[[i, All]], {i, 1, 51}]]
```



Reference solution without viscous forces:

```
Max[MaxXref[[All, 2]]]
```

```
652.396
```

```
MaxXref[[Position[ MaxXref[[All, 2]], Max[MaxXref[[All, 2]]]][[1]][[1]], 1]]
```

```
45.
```

With viscous damping, laminar flow:

```
Max[MaxXvisc[[All, 2]]]
```

```
626.179
```

```
MaxXvisc[[Position[ MaxXvisc[[All, 2]], Max[MaxXvisc[[All, 2]]]][[1]][[1]], 1]]
```

```
44.4
```

With viscous damping, turbulent flow:

```
Max[MaxXvisc2[[All, 2]]]
```

```
212.243
```

```
MaxXvisc2[[Position[ MaxXvisc2[[All, 2]], Max[MaxXvisc2[[All, 2]]]][[1]][[1]], 1]]
```

```
36.6
```

Appendix B: The Impulse Collision Method Algorithm

a.1) For a particle-particle interaction, calculate the relative position vector $\mathbf{x}_{(ab)}$

$$\mathbf{x}_{(ab)} = \mathbf{x}_{(a)} - \mathbf{x}_{(b)} \quad \text{and its norm} \quad x_{(ab)} = \left|\mathbf{x}_{(ab)}\right|$$

a.2) Compute the overlap (penetration depth)

a) $\Delta x_{(ab)} = r_{(a)} + r_{(b)} - x_{(ab)}$ – in case of particle-particle interaction

b) $\Delta x_{(as)} = r_{(a)} - \left\langle (\mathbf{x}_{(a)} - \mathbf{r}_{(s)}), \hat{\mathbf{n}}_{(s)} \right\rangle$ – in case of particle-surface interaction

The two objects interact (have a contact), if the overlap is *positive*, i.e. if

$$\Delta x_{(ab)} > 0 \quad \text{alt.} \quad \Delta x_{(as)} > 0$$

Proceed with steps (b) and (c) only for objects that overlaps.

b.1) For particle-particle interaction, calculate the unit collision direction from *b* to *a*, i.e.

$$\hat{\mathbf{n}}_{(ab)} = \mathbf{x}_{(ab)} / x_{(ab)}$$ (for particle-surface interaction we already have $\hat{\mathbf{n}}_{(as)} = \hat{\mathbf{n}}_{(s)}$)

b.2) Calculate the velocity projection $v^{\mathbf{n}}_{(ab)}$ on the unit collision direction $\hat{\mathbf{n}}_{(ab)}$ as scalar product

$$v^{\mathbf{n}}_{(ab)} = \left\langle \hat{\mathbf{n}}_{(ab)}, \mathbf{v}_{(ab)} \right\rangle$$

We can distinguish three cases:

$v^{\mathbf{n}}_{(ab)} < 0$ – impacting contact

$v^{\mathbf{n}}_{(ab)} = 0$ – resting contact

$v^{\mathbf{n}}_{(ab)} > 0$ – separating contact

Proceed with (c) ***only*** *in case of* ***impacting contact*** *when using the* ***impulse interaction*** *method.*

c.1) Calculate the reduced mass to particle mass ratio $m^{\text{red}}_{(a)} = m_{(ab)} m^{-1}_{(a)}$

a) $m^{\text{red}}_{(a)} = \dfrac{m_{(b)}}{m_{(a)} + m_{(b)}}$ – in case of ***particle-particle*** interaction, when $m_{(a)} \approx m_{(b)}$

b) $m^{\text{red}}_{(a)} = 1$ – in case of ***particle-surface*** interaction, when $m_{(a)} \ll m_{(s)}$

c.2) Calculate the scalar component of the velocity jolt $v^{\text{jolt}}_{(ab)}$

$$v^{\text{jolt}}_{(a)} = -(1+e)\, m^{\text{red}}_{(a)}\, v^{\mathbf{n}}_{(ab)}$$ (note that $v^{\mathbf{n}}_{(ab)} < 0$ for impacting contact, so $v^{\text{jolt}}_{(a)} > 0$)

c.3) Calculate the tangential unit vector of contact

$$\hat{\mathbf{t}}_{(ab)} = \mathbf{v}^{\mathbf{t}}_{(ab)} / \left|\mathbf{v}^{\mathbf{t}}_{(ab)}\right| \quad \text{where} \quad \mathbf{v}^{\mathbf{t}}_{(ab)} = \mathbf{v}_{(ab)} - \hat{\mathbf{n}}_{(ab)} \left\langle \hat{\mathbf{n}}_{(ab)}, \mathbf{v}_{(ab)} \right\rangle$$

c.4) Calculate the velocity jolt (where $\mu > 0$ in case of friction)

$$\mathbf{v}^{\text{jolt}}_{(a)} = v^{\text{jolt}}_{(a)} (\hat{\mathbf{n}}_{(ab)} - \mu\, \hat{\mathbf{t}}_{(ab)})$$ and **clip** $\mu v^{\text{jolt}}_{(a)}$ term not to exceed $v^{\mathbf{t}}_{(ab)}$ component **(!)**

c.5) Calculate the position projection

$$\mathbf{x}^{\text{jolt}}_{(a)} = k_{\text{p}}\, m^{\text{red}}_{(a)}\, \Delta x_{(ab)}\, \hat{\mathbf{n}}_{(ab)}$$ (note that $\Delta x_{(ab)} < 0$ since objects are in contact)

c.6) Apply the velocity jolt and the position projection when appropriate (e.g. in the odd half-step of the leapfrog integrator to keep external forces balanced)

$$\mathbf{v}'_{(a)} = \mathbf{v}_{(a)} + \mathbf{v}^{\text{jolt}}_{(a)}$$

$$\mathbf{x}'_{(a)} = \mathbf{x}_{(a)} + \mathbf{x}^{\text{jolt}}_{(a)}$$

Notes on Two-body Collision Impulses

Let us introduce a velocity jolt $\mathbf{v}_{(a)}^{\text{jolt}}$ as the change of the velocity of a particle *a* during the hardcore interaction, i.e.

$$\mathbf{v}_{(a)}^{\text{jolt}} = \mathbf{v}'_{(a)} - \mathbf{v}_{(a)} \tag{B.1}$$

Then, from Eq. (5.11) $\mathbf{v}'_{(a)} = \mathbf{v}_{(a)} + m_{(a)}^{-1}\,\mathbf{j}_{(ab)}$ we have

$$\mathbf{v}_{(a)}^{\text{jolt}} = m_{(a)}^{-1}\,\mathbf{j}_{(ab)} \tag{B.2}$$

and further, from Eq. (5.14) $\mathbf{j}_{(ab)} = j\,\hat{\mathbf{n}}_{(ab)}$, we get expression for the velocity jolt

$$\mathbf{v}_{(a)}^{\text{jolt}} = j\,m_{(a)}^{-1}\,\hat{\mathbf{n}}_{(ab)} \tag{B.3}$$

On the other hand, substituting impulse *j* from Eq. (5.16)

$$j = -(1+e)\frac{1}{m_{(a)}^{-1} + m_{(b)}^{-1}}\left\langle \hat{\mathbf{n}}_{(ab)}, \mathbf{v}_{(ab)} \right\rangle \tag{5.16}$$

into Eq. (B.3) yields

$$\mathbf{v}_{(a)}^{\text{jolt}} = -(1+e)\frac{1}{m_{(a)}^{-1} + m_{(b)}^{-1}}\left\langle \hat{\mathbf{n}}_{(ab)}, \mathbf{v}_{(ab)} \right\rangle m_{(a)}^{-1}\,\hat{\mathbf{n}}_{(ab)} \tag{B.4}$$

Denoting as $v_{(ab)}^{\mathbf{n}} = \left\langle \hat{\mathbf{n}}_{(ab)}, \mathbf{v}_{(ab)} \right\rangle$ the projection of the velocity $\mathbf{v}_{(ab)}$ on the unit collision direction $\hat{\mathbf{n}}_{(ab)} = \mathbf{x}_{(ab)} / x_{(ab)}$ (from particle *b* to particle *a*), and substituting into Eq. (B.4), we get

$$\mathbf{v}_{(a)}^{\text{jolt}} = -(1+e)\frac{m_{(b)}}{m_{(a)} + m_{(b)}} v_{(ab)}^{\mathbf{n}}\hat{\mathbf{n}}_{(ab)}\,. \tag{B.5}$$

At this point, it is convenient to introduce a *reduced mass* $m_{(ab)}$ of a two-body system as $m_{(ab)}^{-1} = m_{(a)}^{-1} + m_{(b)}^{-1}$ and the reduced mass to particle *a* ratio $m_{(a)}^{\text{red}} = m_{(ab)} m_{(a)}^{-1}$ i.e.

$$m_{(a)}^{\text{red}} = m_{(ab)} m_{(a)}^{-1} = \frac{m_{(b)}}{m_{(a)} + m_{(b)}} \tag{B.6}$$

Substituting the last reduced mass to particle ratio into (B.5) gives

$$\mathbf{v}_{(a)}^{\text{jolt}} = -(1+e)\,m_{(a)}^{\text{red}}\,v_{(ab)}^{\mathbf{n}}\,\hat{\mathbf{n}}_{(ab)}\,. \tag{B.7}$$

If we further denote as $v_{(a)}^{\text{jolt}}$ the scalar component of the velocity jolt $v_{(a)}^{\text{jolt}} = \mathbf{v}_{(a)}^{\text{jolt}} / \hat{\mathbf{n}}_{(ab)}$, then

$$\mathbf{v}_{(a)}^{\text{jolt}} = v_{(a)}^{\text{jolt}}\,\hat{\mathbf{n}}_{(ab)} \tag{B.8}$$

and from Eq. (B.5)

$$v_{(a)}^{\text{jolt}} = -(1+e)\,m_{(a)}^{\text{red}}\,v_{(ab)}^{\mathbf{n}} \tag{B.9}$$

Combining Eq. (B.8) with Eq. (B.3), we get also relationships

$$v_{(a)}^{\text{jolt}} = j\,m_{(a)}^{-1} \quad \text{i.e.} \quad j = m_{(a)} v_{(a)}^{\text{jolt}}\,. \tag{B.10}$$

Notes on Collision Impulses with the Static Surface

We can assume that the mass of the surface is $m_{(b)} \to \infty$. In this case the reduced mass to particle ratio $m^{\text{red}}_{(a)} \to 1$ and Eq. (B.9) becomes

$$v^{\text{jolt}}_{(a)} = (1+e)\, v^{\mathbf{n}}_{(ab)} \tag{B.11}$$

Notes on Simplified Frictional Collision Impulses

Applying the triple vector product rule $\mathbf{a}\times(\mathbf{b}\times\mathbf{c}) = \mathbf{b}\langle\mathbf{a},\mathbf{c}\rangle - \mathbf{c}\langle\mathbf{a},\mathbf{b}\rangle$ to modified Eq. (5.25)

$$\hat{\mathbf{t}}_{(ab)} = \mathbf{v}^{\mathbf{t}}_{(ab)} / \left|\mathbf{v}^{\mathbf{t}}_{(ab)}\right| \quad \text{where} \quad \mathbf{v}^{\mathbf{t}}_{(ab)} = \hat{\mathbf{n}}_{(ab)} \times (\mathbf{v}_{(ab)} \times \hat{\mathbf{n}}_{(ab)}) \tag{5.25}$$

which gives

$$\mathbf{v}^{\mathbf{t}}_{(ab)} = \mathbf{v}_{(ab)} \left\langle \hat{\mathbf{n}}_{(ab)}, \hat{\mathbf{n}}_{(ab)} \right\rangle - \hat{\mathbf{n}}_{(ab)} \left\langle \hat{\mathbf{n}}_{(ab)}, \mathbf{v}_{(ab)} \right\rangle \tag{B.12}$$

Recalling from earlier that $v^{\mathbf{n}}_{(ab)} = \left\langle \hat{\mathbf{n}}_{(ab)}, \mathbf{v}_{(ab)} \right\rangle$ and since $\left\langle \hat{\mathbf{n}}_{(ab)}, \hat{\mathbf{n}}_{(ab)} \right\rangle = 1$, the last expression becomes

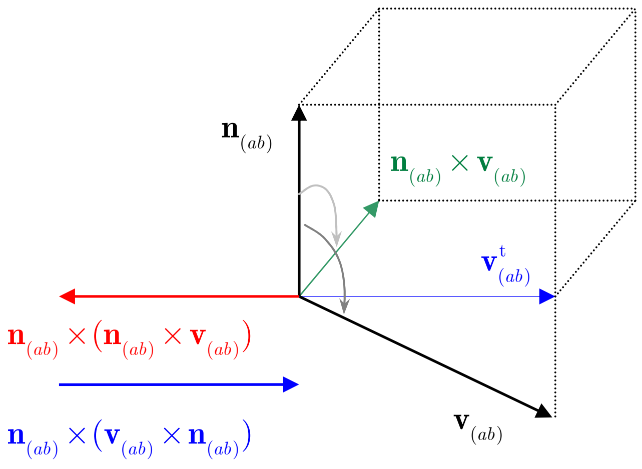
$$\mathbf{v}^{\mathbf{t}}_{(ab)} = \mathbf{v}_{(ab)} - \hat{\mathbf{n}}_{(ab)} \left\langle \hat{\mathbf{n}}_{(ab)}, \mathbf{v}_{(ab)} \right\rangle \tag{B.13}$$

and

$$\hat{\mathbf{t}}_{(ab)} = \mathbf{v}^{\mathbf{t}}_{(ab)} / \left|\mathbf{v}^{\mathbf{t}}_{(ab)}\right|$$

Note that Eq. (B.13) can be calculated for *any* number of spatial dimensions (1, 2 or 3), while Eq. (5.25) can be calculated only for three-dimensional vectors (cross product operation requires arrays of length 3).

From performance view, Eq. (B.13) consists of one scalar multiplication (with velocity projection scalar product calculated earlier) and vector subtraction. This is much faster than two cross products from Eq. (5.25) and also easier to "vectorize" for many-particle system (e.g. in MATLAB).


$\mathbf{n}_{(ab)}$
$\mathbf{n}_{(ab)} \times \mathbf{v}_{(ab)}$
$\mathbf{v}^{\mathbf{t}}_{(ab)}$
$\mathbf{n}_{(ab)} \times (\mathbf{n}_{(ab)} \times \mathbf{v}_{(ab)})$
$\mathbf{n}_{(ab)} \times (\mathbf{v}_{(ab)} \times \mathbf{n}_{(ab)})$
$\mathbf{v}_{(ab)}$

Finally, since

$$\mathbf{j}_{(ab)} = j\, \hat{\mathbf{n}}_{(ab)} - \mu j\, \hat{\mathbf{t}}_{(ab)} \tag{B.14}$$

it follows

$$\mathbf{v}^{\text{jolt}}_{(a)} = v^{\text{jolt}}_{(a)}\, \hat{\mathbf{n}}_{(ab)} - \mu\, v^{\text{jolt}}_{(a)}\, \hat{\mathbf{t}}_{(ab)} \tag{B.15}$$

Notes on Position Projections

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

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

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

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

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

Let us denote with $\mathbf{x}^\text{jolt}_{(a)}$ a position projection, i.e. as the change of the position of a particle *a* during the hardcore interaction

$$\mathbf{x}^\text{jolt}_{(a)} = \mathbf{x}'_{(a)} - \mathbf{x}_{(a)} \tag{B.16}$$

Then, from Eq. (5.27) $\mathbf{x}'_{(a)} = \mathbf{x}_{(a)} + k_\text{p} \cdot \delta_{(a)} \hat{\mathbf{n}}_{(ab)}$ we have

$$\mathbf{x}^\text{jolt}_{(a)} = \delta_{(a)}\ \hat{\mathbf{n}}_{(ab)}$$

Since, from Eq. (5.29)

$$\delta_{(a)} = \frac{m_{(b)}}{m_{(a)} + m_{(b)}} \Delta x_{(ab)} = m^\text{red}_{(a)}\ \Delta x_{(ab)} \tag{B.17}$$

the position projection is

$$\mathbf{x}^\text{jolt}_{(a)} = k_\text{p}\ m^\text{red}_{(a)}\ \Delta x_{(ab)}\ \hat{\mathbf{n}}_{(ab)} \tag{B.18}$$

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

$$\mathbf{x}'_{(a)} - \mathbf{x}_{(a)} = k_\text{p} \cdot \frac{m^{-1}_{(a)}}{m^{-1}_{(a)} + m^{-1}_{(b)}} \Delta x_{(ab)} \cdot \hat{\mathbf{n}}_{(ab)} \tag{B.19}$$

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

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

$$\mathbf{x}'_{(a)} - \mathbf{x}_{(a)} = \left(k_\text{s} \Delta x_{(ab)} \cdot \hat{\mathbf{n}}_{(ab)}\right) \cdot m^{-1}_{(a)} \cdot h^2 \tag{B.20}$$

Combining equations (B.19) and (B.20) yields

$$k_\text{p} \cdot \frac{m^{-1}_{(a)}}{m^{-1}_{(a)} + m^{-1}_{(b)}} \Delta x_{(ab)} \cdot \hat{\mathbf{n}}_{(ab)} = \left(k_\text{s} \Delta x_{(ab)} \cdot \hat{\mathbf{n}}_{(ab)}\right) \cdot m^{-1}_{(a)} \cdot h^2 \tag{B.21}$$

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

$$k_\text{p} = k_\text{s} m^{-1}_{(ab)} \cdot h^2 \tag{B.22}$$

Since the semi-implicit Euler integrator method is assumed stable for $(k_\text{s} / m) \cdot h^2 = (\omega h)^2 < 1$, then, for $m_{(a)} = m_{(b)} = m$, it follows from (B.22), that the same method is stable for the position projection factors $k_\text{p} < 0.5$.

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:

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

and the condition for contact is/when:

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

Motivation 1

The equations (5.8) and (5.9) are mentioned to be used for detecting contacts in *collisions* with surfaces. However, the equations fail with a finite time-step integrator in case when:

- a particle has a high velocity, or
- a particle has a very small size, or
- time-step is too large.

In such cases the integrator can "tunnel" a particle through the surface without noticing that there was a contact (left case on Figure A.1)! In particular, tunneling might happen if $r < h\,v$, where r is particle radius, h is time-step size and v is particle velocity. However, **a box wall** or **ground surfaces** should not be 'transparent' for particles in classical mechanics (right case on Figure A.1).

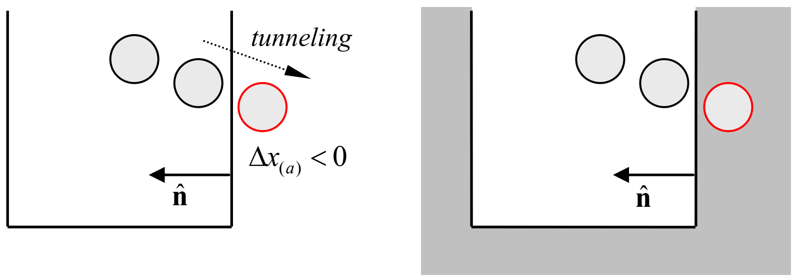


Figure A.1: Tunneling of a particle with unmodified (5.9) giving $\Delta x_{(a)} < 0$ (left case)

So, the collision detection algorithm based on Eqs. (5.8-9) has *a bug*, which makes usage of the equations erroneous. Otherwise, one has to *ensure* that particle velocities in a system *does not become larger* than r/h, which seems as an unreasonable requirement.

Motivation 2

Another way to look on this problem is to perceive an Euclidian plain as a limit of the sphere with center $\mathbf{R}_{(s)}$ at infinity and having infinite radius $R_{(s)} \to \infty$. In this case, $\Delta x_{(a)} = r_{(a)} - (\mathbf{x}_{(a)} - \mathbf{r})^{\mathrm{T}}\hat{\mathbf{n}}$ follows from Eq. (5.4), i.e. from $\Delta x_{(a)} = r_{(a)} + R_{(s)} - \left|\mathbf{x}_{(a)} - \mathbf{R}_{(s)}\right|$ (given here without proof).

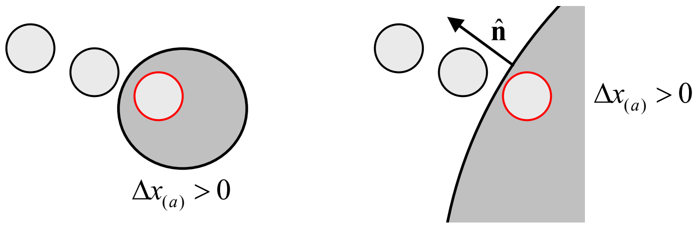


Figure A.2: Collision with a plain seen as a collision with sphere of an infinite radius

Lab3 Appendix C: Angular Velocity Decay

Initialization

General assumptions when solving equations

```
In[1]:= $Assumptions = R > 0 ⋀ 0 ≤ ϕ ≤ 2 π ⋀ 0 ≤ θ ≤ π ⋀
           ρ > 0 ⋀ m > 0 ⋀ C_D ≥ 0 ⋀
           vx ∈ Reals ⋀ vy ∈ Reals ⋀ vz ∈ Reals ⋀ ωx ∈ Reals ⋀ ωy ∈ Reals ⋀ ωz ∈ Reals;
```

Definition of the vector norm symbol

In[2]:= $|v\_| := \sqrt{v_{[[1]]}^2 + v_{[[2]]}^2 + v_{[[3]]}^2}$

Equations

Velocity of the surface element δA at position **r** relative to the fluid is

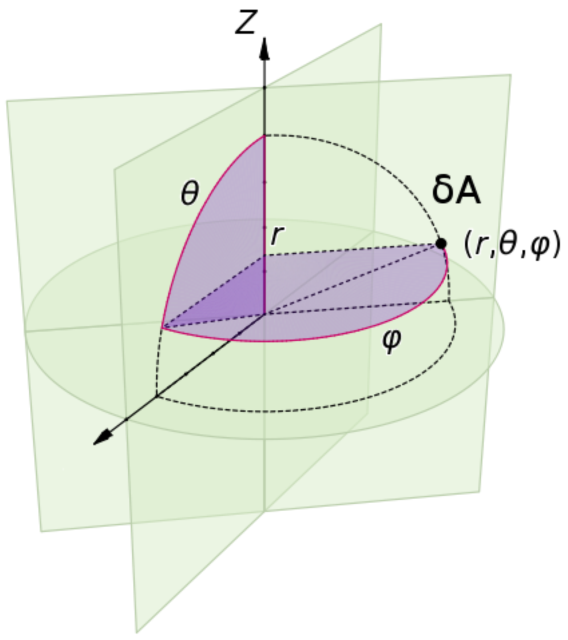
In[3]:= $\mathcal{U} := v + \omega \times r$

where **v** is velocity of the ball throught the fluid and $\boldsymbol{\omega}$ is angular velocity of the ball.

Let us assume that an infinitesimal drag force at the surface element δA is

In[4]:= $\delta f := -\frac{1}{2} \rho\, C_D\, \delta A\, |\mathcal{U}|^{\gamma-1}\, \mathcal{U}$

where ρ is mass density of the fluid, C_D is a constant that depends on the surface material, and γ characterizes type of the flow ($\gamma = 1$ for laminar flow and $\gamma = 2$ for turbulent flow).



An infinitesimal torque caused by the drag force is then

In[5]:= $\delta\tau := r \times \delta f$

where the net torque is $\tau = \int \delta\tau$ integrated across the surface.

Angular acceleration of the ball can be expressed from the net torque as

In[6]:= $\dot{\omega} := \mathcal{I}^{-1}\, \tau$

where $\mathcal{I}$ is the moment of inertia of the ball (i.e. sphere)

In[7]:= $\mathcal{I} := \frac{2}{5} m R^2$

Parameters

Taking as a parameter ball cross section area A

```
In[8]:=  A := π R^2
```

together with a constant C_w that depends on type of surface and moment of inertia

```
In[9]:=  C_w := 20/3 C_D
```

and introducing constant of proportionality k_w

```
In[10]:= k_w := 1/2 (ρ A C_w)/m
```

Coordinate System

Spherical Coordinates

Position of the surface element δA in in spherical coordinates is

```
In[11]:= r := {
           R Cos[ϕ] Sin[θ],
           R Sin[ϕ] Sin[θ],
           R Cos[θ]
          }
```

The area of the surface element is

```
In[12]:= δA := R^2 δΩ
```

where $\delta\Omega$ is an infinitesimal solid angle subtended by the surface element, in spherical coordinates expressed as

```
In[13]:= δΩ := Sin[θ] δθ δϕ
```

Velocity of the ball throught the fluid **v** in cartesian coordinates

```
In[14]:= v := {vx, vy, vz}
```

Angular velocity ω of the ball in cartesian coordinates

```
In[15]:= ω := {ωx, ωy, ωz}
```

Velocity of the surface element relative to the fluid in cartesian coordinates

```
In[16]:= {{"𝓊x =", "𝓊y =", "𝓊z ="}, 𝓊} // Transpose // TableForm
Out[16]//TableForm=
    𝓊x  =  vx + R ωy Cos[θ] − R ωz Sin[θ] Sin[ϕ]
    𝓊y  =  vy − R ωx Cos[θ] + R ωz Cos[ϕ] Sin[θ]
    𝓊z  =  vz − R ωy Cos[ϕ] Sin[θ] + R ωx Sin[θ] Sin[ϕ]
```

Sanity-Check

Total area of the ball should be $\int \delta \mathrm{A} = 4\,\pi\,R^2$

$$\text{In[17]:=}\quad \int_0^{\pi}\int_0^{2\pi}\left(\frac{\delta A}{\delta\theta\;\delta\phi}\right) d\phi\, d\theta$$

$$\text{Out[17]=}\quad 4\,\pi\,R^2$$

Solving Equations

Let's find angular acceleration $\alpha = \frac{\tau}{k_w I}$ by integrationg the net torque $\tau = \int \delta\tau$ across the whole surface.

Laminar Flow

```
In[18]:= γ = 1; θ =.; ϕ =.;
         v = {vx, vy, vz};
         ω = {ωx, ωy, ωz};
```

In[21]:= $\delta\alpha[\gamma] = \frac{\delta\tau}{I\ \delta\theta\ \delta\phi}$ // FullSimplify;

In[22]:= $\alpha[\gamma] = \int_0^{\pi}\int_0^{2\pi} \delta\alpha[\gamma]\, d\phi\, d\theta$;

In[23]:= {{"$\dot{\omega}_x$ =", "$\dot{\omega}_y$ =", "$\dot{\omega}_z$ ="}, α[γ]} // Transpose // TableForm

Out[23]//TableForm=

$\dot{\omega}_x = -\frac{10\,\pi\,R^2\,\rho\,\omega x\,C_D}{3\,m}$

$\dot{\omega}_y = -\frac{10\,\pi\,R^2\,\rho\,\omega y\,C_D}{3\,m}$

$\dot{\omega}_z = -\frac{10\,\pi\,R^2\,\rho\,\omega z\,C_D}{3\,m}$

In[24]:= Solve[{αx, αy, αz} == α[γ] && $k_w == \frac{1}{2}\frac{\rho\,A\,C_w}{m}$, {αx, αy, αz}][[1]] // TableForm

Out[24]//TableForm=

$\alpha x \to -\omega x\,k_w$

$\alpha y \to -\omega y\,k_w$

$\alpha z \to -\omega z\,k_w$

Conclusion:

$\boldsymbol{\alpha} == -k_w\ \boldsymbol{\omega}$

▼ Turbulent Flow, $\omega = 0$

```
In[25]:=  γ = 2; θ =.; ϕ =.;
          v = {vx, vy, vz};
          ω = {0, 0, 0};
```

In[28]:= $\delta\alpha[\gamma] = \dfrac{\delta\tau}{I\,\delta\theta\,\delta\phi}$ // FullSimplify;

In[29]:= $\alpha[\gamma] = \int_0^{\pi}\int_0^{2\pi} \delta\alpha[\gamma]\, d\phi\, d\theta;$

In[30]:= $\{\{"\dot{\omega}_x\ =", "\dot{\omega}_y\ =", "\dot{\omega}_z\ ="\}, \alpha[\gamma]\}$ // Transpose // TableForm

Out[30]//TableForm=

$\dot{\omega}_x\ =\ 0$

$\dot{\omega}_y\ =\ 0$

$\dot{\omega}_z\ =\ 0$

In[31]:= Solve$\left[\{\alpha x, \alpha y, \alpha z\} == \alpha[\gamma]\ \&\&\ k_w == \frac{1}{2}\frac{\rho A C_w}{m}, \{\alpha x, \alpha y, \alpha z\}\right]$[[1]] // TableForm

Out[31]//TableForm=

$\alpha x \to 0$

$\alpha y \to 0$

$\alpha z \to 0$

Conclusion:

$\boldsymbol{\alpha} == 0$ for $\boldsymbol{\omega} == 0$

▼ Turbulent Flow, v = 0

In[32]:=
```
γ = 2; θ =.; ϕ =.;
v = {0, 0, 0};
ω = {0, 0, ωz};
```

In[35]:=
```
δα[γ] = δτ/(I δθ δϕ) // FullSimplify;
```

In[36]:=
```
α[γ] = ∫_0^π ∫_0^(2π) δα[γ] dϕ dθ;
```

In[37]:=
```
{{"ω̇x =", "ω̇y =", "ω̇z ="}, α[γ]} // Transpose // TableForm
```

Out[37]//TableForm=

$\dot\omega_x = 0$

$\dot\omega_y = 0$

$\dot\omega_z = -\frac{15\,\pi^2\,R^3\,\rho\,\omega z\,\mathrm{Abs}[\omega z]\,C_D}{16\,m}$

In[38]:=
```
Solve[{αx, αy, αz} == α[γ] && k_w == 1/2 (ρ A C_w)/m, {αx, αy, αz}][[1]] // TableForm
```

Out[38]//TableForm=

$\alpha x \to 0$

$\alpha y \to 0$

$\alpha z \to -\frac{9}{32}\,\pi\,R\,\omega z\,\mathrm{Abs}[\omega z]\,k_w$

Conclusion:

$\alpha_z == -k_w\,\omega_z\,|R\,\omega_z|$ for $\mathbf{v} == 0$.

▼ Turbulent Flow, vx ≠ 0, ωz ≠ 0, θ == π/2

```
γ = 2; θ = π / 2; ϕ =.; R =.;
v = {vx, 0, 0};
ω = {0, 0, ωz};
```

$$\delta\alpha[\gamma] = \frac{\delta\tau}{k_w\, \mathcal{I}\, \delta\theta\, \delta\phi} \text{ // FullSimplify}$$

$$\left\{0,\ 0,\ \frac{3\,(-R\,\omega z + vx\,\mathrm{Sin}[\phi])\,\sqrt{vx^2 + R^2\,\omega z^2 - 2\,R\,vx\,\omega z\,\mathrm{Sin}[\phi]}}{8\,\pi\,R}\right\}$$

$$\alpha 3 = \int \delta\alpha[\gamma]_{[[3]]}\, d\phi \text{ // FullSimplify}$$

$$\Bigg(-2\,R\,vx\,\omega z\,\mathrm{Cos}[\phi]\,\left(vx^2 + R^2\,\omega z^2 - 2\,R\,vx\,\omega z\,\mathrm{Sin}[\phi]\right) + (vx - R\,\omega z)^2\,\Bigg(\left(vx^2 + 7\,R^2\,\omega z^2\right)\,\mathrm{EllipticE}\left[\frac{1}{4}\,(\pi - 2\,\phi),\ -\frac{4\,R\,vx\,\omega z}{(vx - R\,\omega z)^2}\right] - (vx + R\,\omega z)^2\,\mathrm{EllipticF}\left[\frac{1}{4}\,(\pi - 2\,\phi),\ -\frac{4\,R\,vx\,\omega z}{(vx - R\,\omega z)^2}\right]\Bigg)\,\sqrt{\frac{vx^2 + R^2\,\omega z^2 - 2\,R\,vx\,\omega z\,\mathrm{Sin}[\phi]}{(vx - R\,\omega z)^2}}\Bigg) \Bigg/ \left(8\,\pi\,R^2\,\omega z\,\sqrt{vx^2 + R^2\,\omega z^2 - 2\,R\,vx\,\omega z\,\mathrm{Sin}[\phi]}\right)$$

```
(α3 /. ϕ → 2 π) - (α3 /. ϕ → 0) // FullSimplify
```

$$-\frac{1}{8\,\pi\,R^2\,\omega z}\,\mathrm{Abs}[vx - R\,\omega z]\,\Bigg(\left(vx^2 + 7\,R^2\,\omega z^2\right)\,\mathrm{EllipticE}\left[\frac{\pi}{4},\ -\frac{4\,R\,vx\,\omega z}{(vx - R\,\omega z)^2}\right] + \left(vx^2 + 7\,R^2\,\omega z^2\right)\,\mathrm{EllipticE}\left[\frac{3\,\pi}{4},\ -\frac{4\,R\,vx\,\omega z}{(vx - R\,\omega z)^2}\right] - (vx + R\,\omega z)^2\,\left(\mathrm{EllipticF}\left[\frac{\pi}{4},\ -\frac{4\,R\,vx\,\omega z}{(vx - R\,\omega z)^2}\right] + \mathrm{EllipticF}\left[\frac{3\,\pi}{4},\ -\frac{4\,R\,vx\,\omega z}{(vx - R\,\omega z)^2}\right]\right)\Bigg)$$

Continued with manual simplification...

```
In[39]:= γ = 2; θ = π / 2; ϕ =.; R =.;
         v = {vx, 0, 0};
         ω = {0, 0, ωz};
         vx =.; ωz =.;
```

In[43]:= $\texttt{E1E := EllipticE}\left[\frac{\pi}{4}, -\frac{4\,R\,vx\,\omega z}{(vx - R\,\omega z)^2}\right] + \texttt{EllipticE}\left[\frac{3\pi}{4}, -\frac{4\,R\,vx\,\omega z}{(vx - R\,\omega z)^2}\right];$

In[44]:= $\texttt{E1F := EllipticF}\left[\frac{\pi}{4}, -\frac{4\,R\,vx\,\omega z}{(vx - R\,\omega z)^2}\right] + \texttt{EllipticF}\left[\frac{3\pi}{4}, -\frac{4\,R\,vx\,\omega z}{(vx - R\,\omega z)^2}\right];$

In[45]:= $\texttt{E3} := -\frac{\texttt{Abs}[vx - R\,\omega z]}{8\,\pi\,R^2\,\omega z}\left(\left(vx^2 + 7\,R^2\,\omega z^2\right)\texttt{E1E} - (vx + R\,\omega z)^2\,\texttt{E1F}\right)$

In[46]:= $\texttt{E3} := -\frac{\texttt{Abs}[vx - R\,\omega z]}{8\,\pi}\,\omega z\left(\left(\left(\frac{vx}{R\,\omega z}\right)^2 + 7\right)\texttt{E1E} - \left(\frac{vx}{R\,\omega z} + 1\right)^2\texttt{E1F}\right)$

```
In[47]:= Limit[ E3, vx → 0, Assumptions → ωz ∈ Reals ⋀ R > 0]
```

Out[47]= $-\frac{3}{4}\,R\,\omega z\,\texttt{Abs}[\omega z]$

```
In[48]:= Limit[ E3, ωz → 0]

Out[48]= 0

In[49]:= ωz = 1; vx = -1.001; R = 1; E3 // N

Out[49]= -1.27419
```

In[50]:= ωz = 1; vx = -10^6; R = 1; E3 // N

Out[50]= -1.12498×10^6

In[51]:= ωz = 1; vx = -10^7; R = 1; E3 // N

Out[51]= -1.12652×10^7

In[52]:= ωz = 10^4; vx = -1; R = 1; E3 // N

Out[52]= -7.5×10^7

In[53]:= ωz = 10^5; vx = -1; R = 1; E3 // N

Out[53]= -7.5×10^9

In[54]:= ωz = 1; vx = -1.001; R = 10^8; E3 // N

Out[54]= -7.5×10^7

In[55]:= ωz = 1; vx = -1.001; R = 10^9; E3 // N

Out[55]= -7.5×10^8

There exists:

1) square dependence of angular velocity,
2) linear dependence of linear velocity, and
3) linear dependence on radius.

Conclusion:

$$\alpha_z \propto -k_w\,\omega_z\,(|v_x| + |R\,\omega_z|)$$

▼ Turbulent Flow, golf ball, numerical analysis

```
In[56]:= γ = 2; θ =.; ϕ =.;
         ρ = 1.3; m = 0.045; R = 0.02; C_D = 1; C_w = 10;
         ωz =.; vx =.;
         v := {vx, 0, 0}
         ω := {0, 0, ωz}
```

Numerical solution

$$\text{In[61]:=}\quad \delta\alpha[\gamma] = \frac{\delta\tau}{I\,\delta\theta\,\delta\phi}\ \text{// FullSimplify;}$$

```
In[62]:= foo[vx_, ωz_] = δα[γ][[3]];

In[63]:= sol[vx_, ωz_] := NIntegrate[foo[vx, ωz], {ϕ, 0, 2 π}, {θ, 0, π}]
```

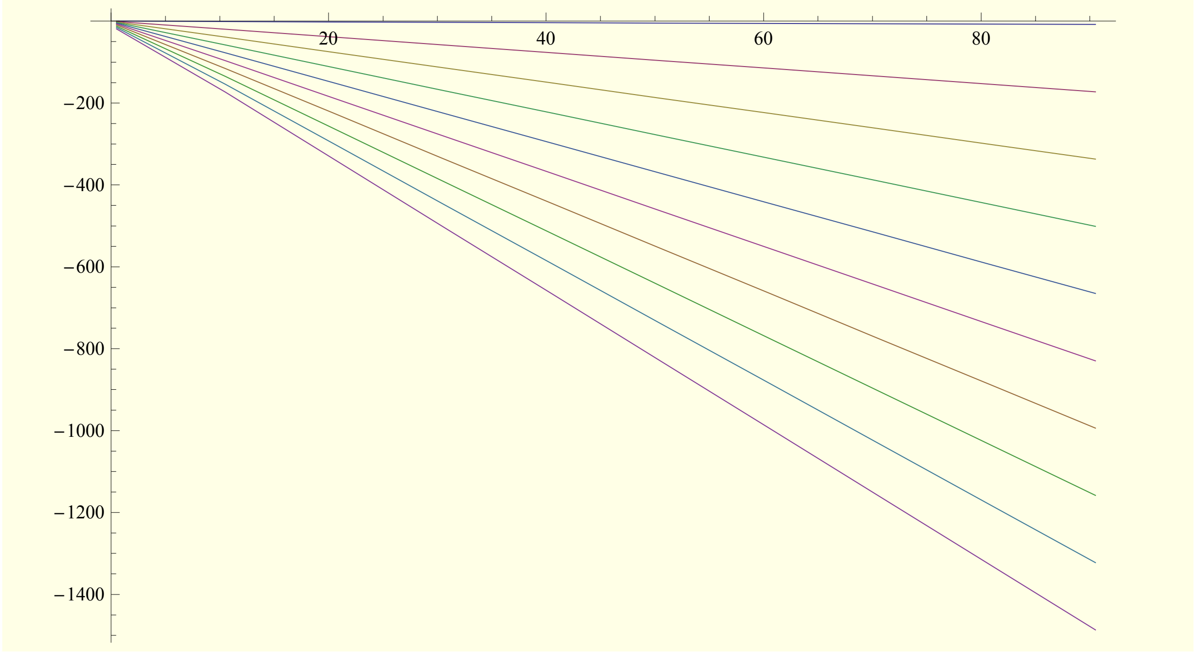
Guessed solution

$$\text{In[64]:=}\quad \text{guessed[vx\_, ωz\_]} := -\frac{1}{2}\,\frac{\rho\,A\,C_w}{m}\,\omega z\ (R\,\omega z + vx)$$

Comparison of numerical solution (top diagram) and guessed solution (bottom diagram) where abscissa = v_x and ordinate = $\dot{\omega}$

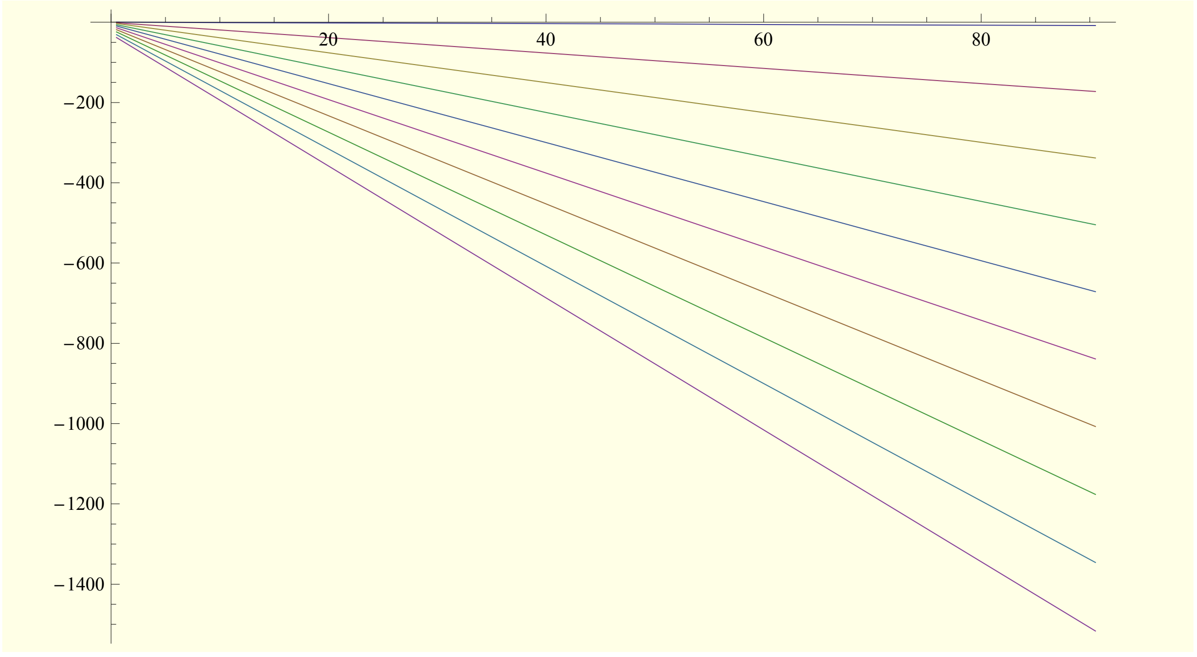
```
In[65]:= ListLinePlot[Table[Table[{vx, sol[vx, ωz]}, {vx, 0.5, 100, 10}], {ωz, 0.5, 100, 10}]]
```

Out[65]=



```
In[66]:= ListLinePlot[
          Table[Table[{vx, guessed[vx, ωz]}, {vx, 0.5, 100, 10}], {ωz, 0.5, 100, 10}]]
```

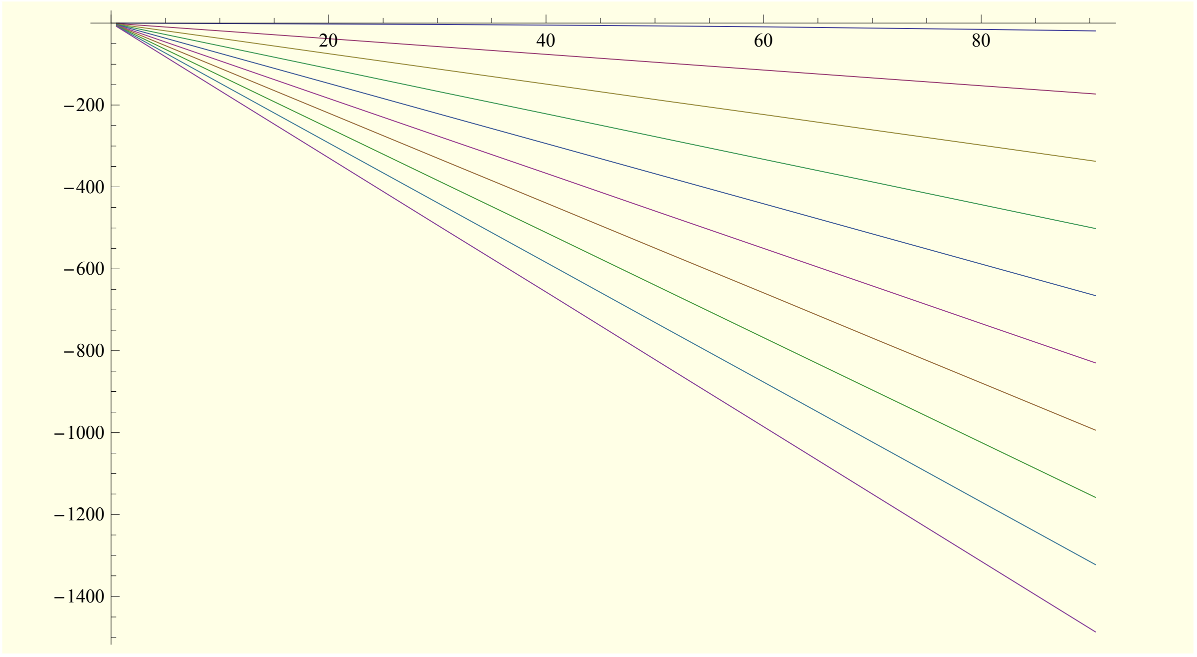
Out[66]=



Comparison of numerical solution (top diagram) and guessed solution (bottom diagram) where abscissa = ω_z and ordinate = $\dot{\omega}$

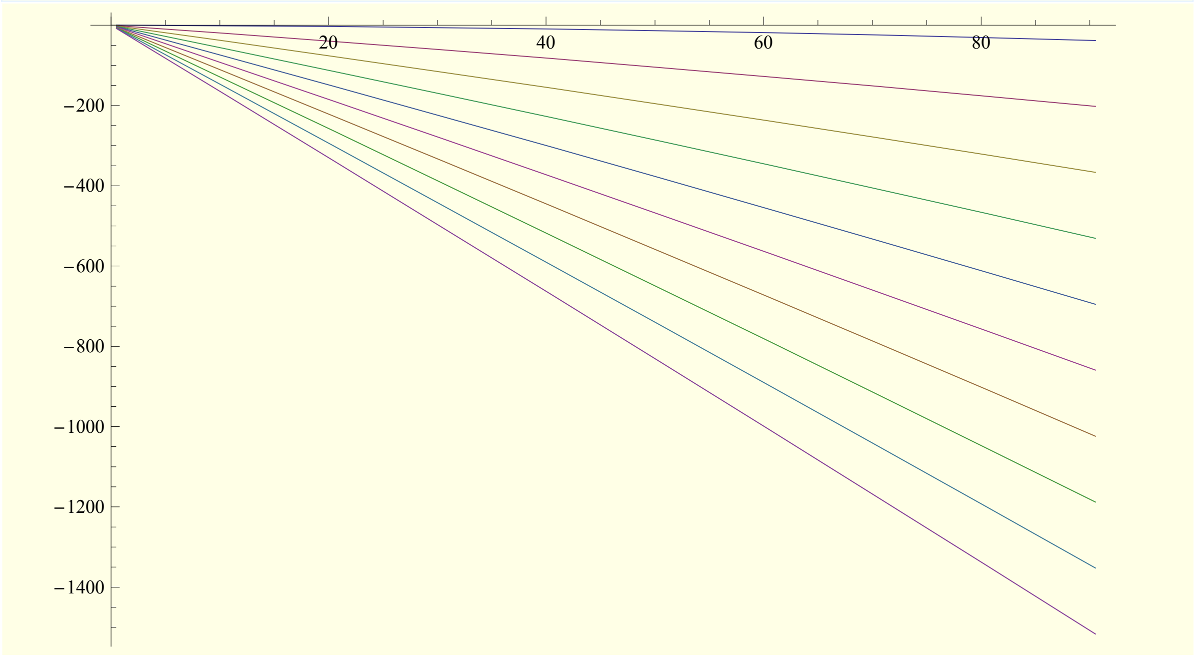
In[67]:=
```
ListLinePlot[Table[Table[{ωz, sol[vx, ωz]}, {ωz, 0.5, 100, 10}], {vx, 0.5, 100, 10}]]
```

Out[67]=



In[68]:=
```
ListLinePlot[
 Table[Table[{ωz, guessed[vx, ωz]}, {ωz, 0.5, 100, 10}], {vx, 0.5, 100, 10}]]
```

Out[68]=



20
40
60
80
−200
−400
−600
−800
−1000
−1200
−1400