ASEN 5051

Mini-Project 2

Lucas Calvert and Duncan McGough

Due soon in 2018

Contents

List of Figures

1 Introduction

In this project, the dynamics of the interaction between two ideal vortex rings were examined.

IN theory, a vortex ring is formed when thw two ends of a finite length *vortex filament* are connected to eachother. This connection forms a ring of radius R, with a strength Γ. While these rings are infintesimally small and have no mass, they do induce a velocity at a distance, due to the circulation about the rings. This velocity can be characterized relatively simply by applying the Biot-Savart Law:

$$u_{induced} = \frac{\Gamma}{4\pi} \int_c \frac{\hat{e}_\omega \times (\vec{x} - \vec{x}')}{|(\vec{x} - \vec{x}')|} dl$$

Where $\hat{e}_\omega$ is the direction about which the vorticity acts about, $\vec{x}$ is the point where the velocity id being induced, $\vec{x}'$ is the "current" point on the vortex ring that is being integrated around and c is the curve corresponding to one vortex ring. If we wanted to make a more general code to examine the interaction of any arbitrary (closed) vortex filament loops, it would involve using this integral to detemine the effects at every point, from every point on a filament. Luckily, however, in this project a special symmetry may be exploited, simplifying the problem and drastically reducing the coputational time required to run a simulation.

The simplifying assumption used is as follows: since we know (by our setup) that both vortex rings are initially perfectly circular, we can expect that they will remain circular for all time. Just by noticing this simple fact, we can now determine the motion of an entire vortex ring by analyzing the velocity induced at a **single point** along it. The application of this idea will be discussed further in the following sections.

It should also be noted that, in order to simplify the computations, this integral was converted from cartesian to polar coordinates. Knowing that:

$$dl = Rd\theta$$

$$x = Rcos(\theta)$$

$$y = Rsin(\theta)$$

We can tranform the terms in the Biot-Savart integral as follws:

$$\vec{x} = [,,]^T$$

$$\vec{x}' = [,,]^T$$

$$e_\omega = [,,]^T$$

2 Task 1

For this section the project, the interaction between "two vortex rings of identical strength and a similar sense of rotation" were analyed. It is expected that the rings will exhibit a "leap-frogging" frogging behavior where they move across the domain, moving inside and outside of eachother, which is exactly what happened.

With a time step 0f 0.2s, 100 discrete points in each ring, an initial radius of 1m and an initial distance apart 1m, the simulation was run for 50s. The resulting radius vs. time and horizontal distance between rings vs. time output plots are shown below:

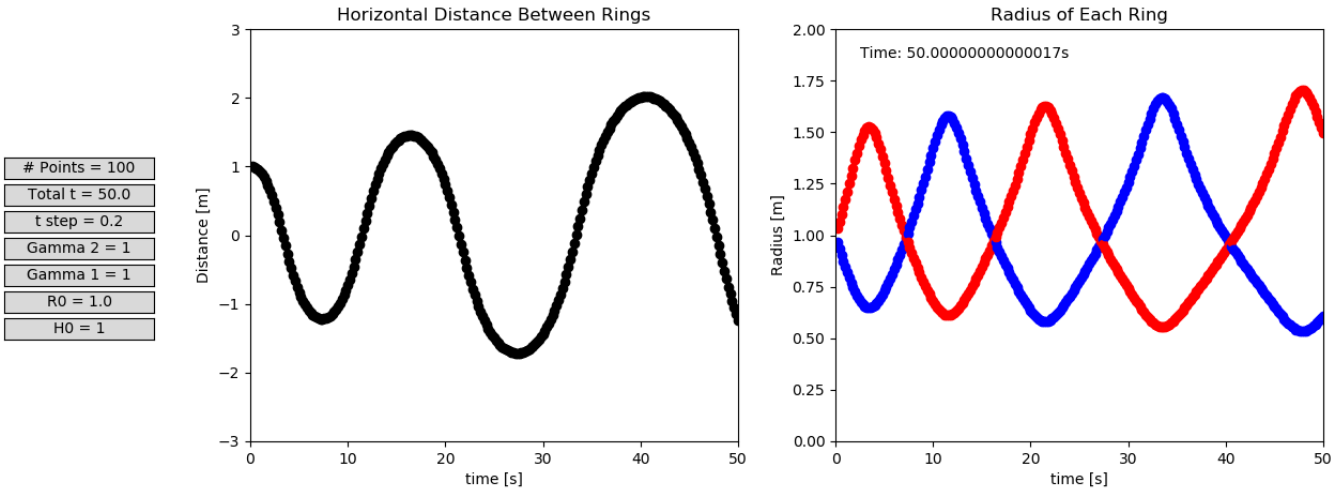

Figure 1: Low Resolution, Low discretization Results

With these resolution settings (time step and discretization level), an unexpected behavior may be seen - that is: the rings begin to spread out and increase in maximum radius with every leapfrogging cycle. This is due to the nature of this numerical foreward-euler simulation. Errors in numerically evaluating the Biot-Savart integral complile, causing the simulation to diverge.

In an attempt to correct this behavior, the discretization level was increased by an order of magnitude (from 100 points along each ring to 1000). The results are as follows:

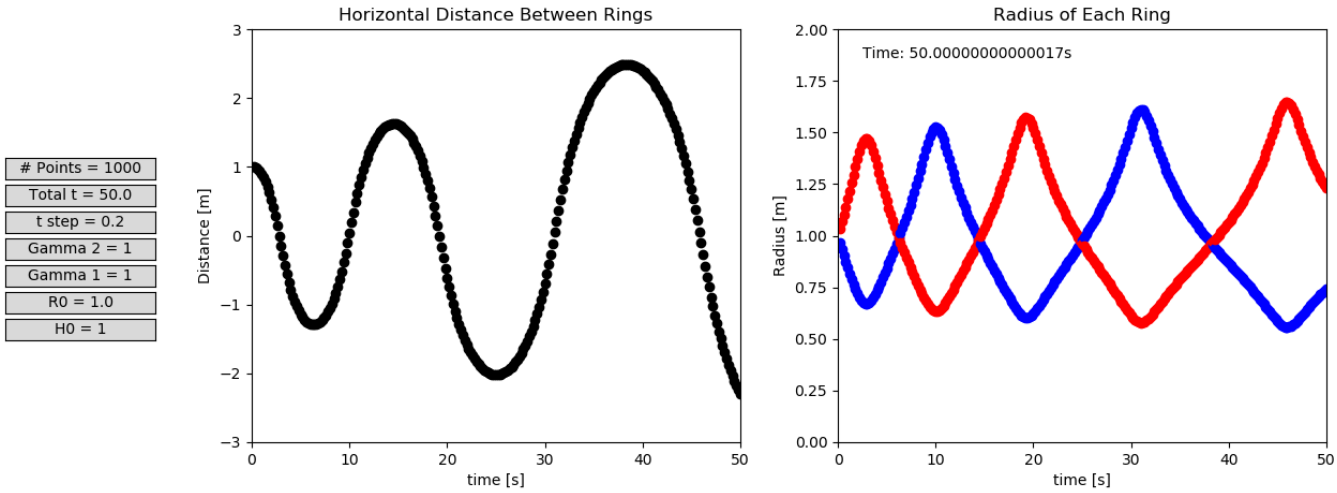

Figure 2: Low Resolution, High discretization Results

Increasing the discretization level did not correct the result - it just made the simulation run for a longer time. Next, the time step was reduced drastically. Running the simulation again produced the following output:

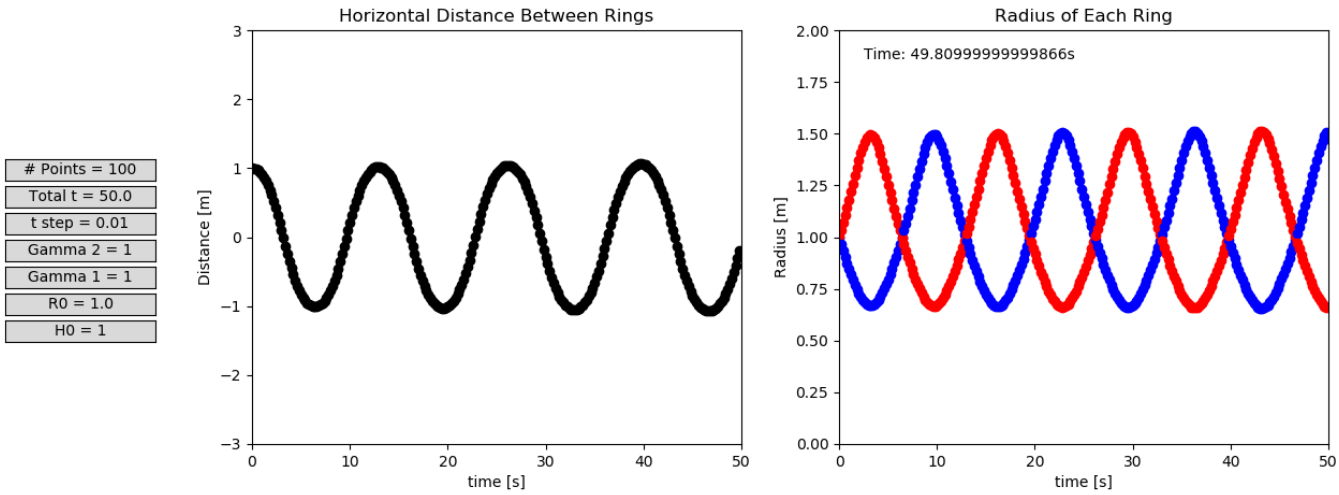

Figure 3: High Resolution (small time step) Results

This adjustment (while it also caused the simulation to run noticeably slower) produced much better and consistent looking results. Figure 3 shows a consistent period of "leap-frogging" and consistent radii for both rings.

Two videos (animations) of the transient states obsevred during this simulation are included with the submission of this report - one for the low resolution simulation with low discretization and on for the high resolution simulation. The videos are titled "task1_lowRes.mp4" and "task1_highRes.mp4", respectively.

3 Task 2

For this section of the project, the interaction between "two vortex rings of identical strength and an opposite sense of rotation". In much the same was as with the first task, the simulation was first run with ethe "low resolution" settings from before. The initial setup was such that the initial distance between the rings was 10m, their radii start at 1m, Γ was set to 1 and the timestep was set to 0.2 seconds. The simulation was run for ****s to produce the following results:

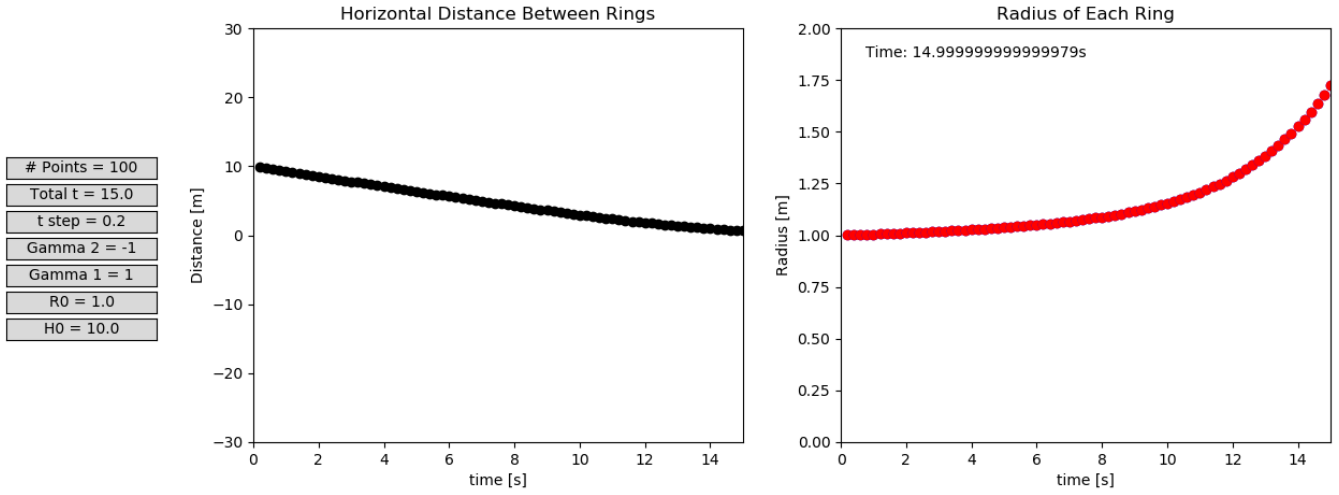

Figure 4: Low Resolution Results

So that a comparison between fine and course time steps could also be made for this task, the time step value was again reduced to 0.01s. This produced the following results:

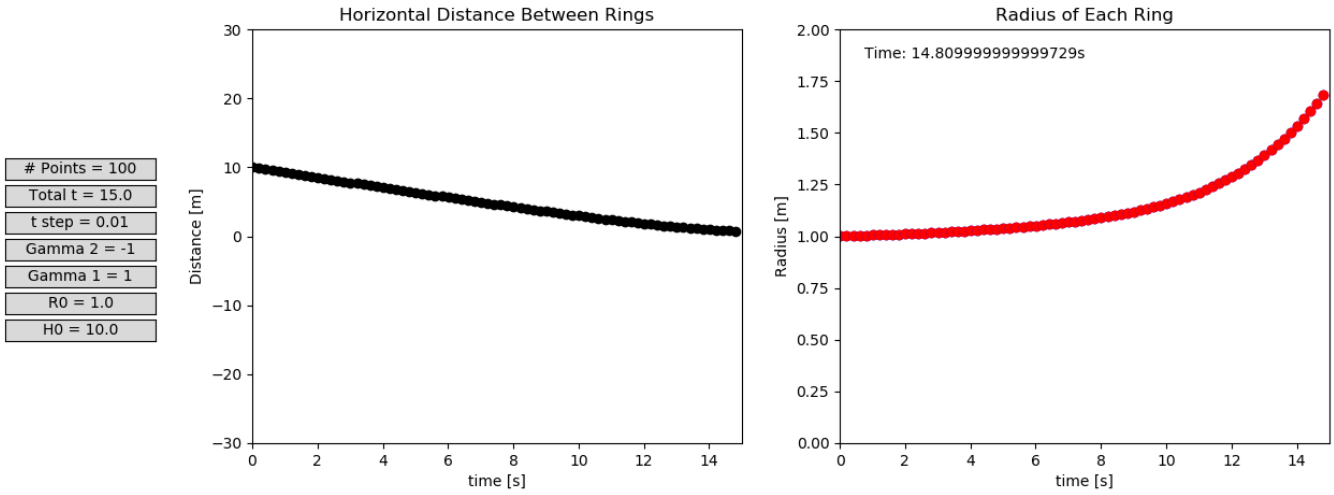

Figure 5: High Resolution Results

In this case, refining the time step caused a much less noticeable change. This is likely due to the non-cyclic behavior of this setup. The results with the refined time step are likely more accurate, even if they are visually similar to the unrefined time step results.

Videos of both of these simulations have also been included with this submission.

4 Task 3

For this task, we want to determine an analytical formula for the time T of a leapfrogging cycle in terms of R, H, and Γ. Let's start out by formulating the dimensional matrix:

	T	R	H	Γ
M	0	0	0	0
L	0	1	1	2
T	1	0	0	-1

We can then take this matrix and use Gauss-Jordan to obtain the rank of the dimensional matrix:

$$\text{rref}\left(\begin{bmatrix}0 & 0 & 0 & 0\\ 0 & 1 & 1 & 2\\ 1 & 0 & 0 & -1\end{bmatrix}\right) = \begin{bmatrix}1 & 0 & 0 & -1\\ 0 & 1 & 1 & 2\\ 0 & 0 & 0 & 0\end{bmatrix} \rightarrow r = 2$$

Now, we determine the number of dimensionless groups $n - r$ where n is the number of dimensional variables and r is the rank of the dimensional matrix. $n - r = 4 - 2 = 2$ dimensional groups.

Let:

$$\Pi_1 = \frac{\Gamma T}{RH} \rightarrow \frac{L^2}{T}\frac{T}{1}\frac{1}{L}\frac{1}{L} \rightarrow \text{ dimensionless}$$

Let's choose our next Pi group:

$$\Pi_2 = \frac{R}{H} \rightarrow \frac{L}{1}\frac{1}{L} \rightarrow \text{ dimensionless}$$

We can then formulate that:

$$\Pi_1 = \psi(\Pi_2)$$

$$\frac{\Gamma T}{RH} = \psi\left(\frac{R}{H}\right)$$

We can select values for Γ, R, and H and use the Python code developed in Tasks 1 and 2 to determine the complete leapfrog period T. This was done several times, and is tabulated in Table 1.

Γ [m^2/s]	R [m]	H [m]	T [s]
1	1	1	12.81
1	2	1	17.62
2	1	1	6.61
1.5	1.5	1	10.81
1	1	0.5	4.61
0.5	1	0.5	8.82
0.8	2	0.2	1.21
1	0.8	0.4	3.01
1.2	0.7	0.6	4.41
5	2	1	3.62
5	2	2	10.41
2	4	1	10.01
2	5	1	10.02
2	6	1	10.21
2	7	1	10.21
2.0	7	1.2	14.4
2	7.2	1.4	19.62
2	6.8	1.4	19.62
1.6	6.5	1.5	27.6
1	3	0.2	0.841
1	3	1	19.01
1	1.6	0.2	0.821
1	2.5	1	18.82
0.5	0.8	1	21.62

Table 1: Collected Data from Python Script for Complete Leapfrog

Once the data is collected Π_1 is plotted versus Π_2. Then, an exponential curve can be fit to the data to obtain ψ. The plot and fitted curve can be seen in Figure 6.

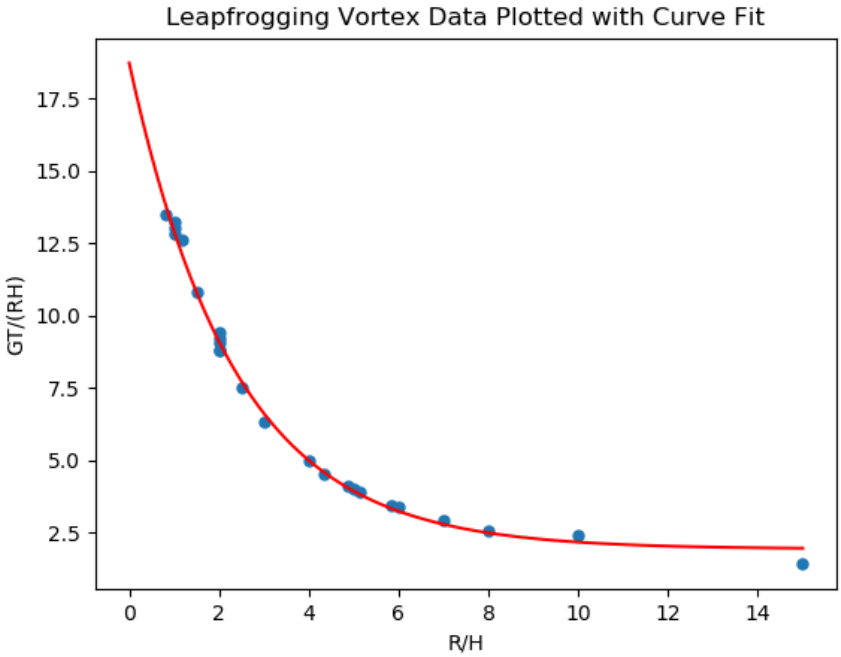

Figure 6: Collected Data Points for Leapfrogging Vortices

This exponential curve takes the form:

$$\psi(x) = Ae^{-Bx} + C$$

The curve fitting in our Python script returns values of: $A = 16.81$ and $B = 0.425$ $C = 1.926$.

The function can then be solved for T, leaving us with:

$$\boxed{T = \frac{RH}{\Gamma}\left[16.81e^{-0.425\frac{R}{H}} + 1.926\right]}$$

To validate this empirical formula, the leap-frogging period for 3 test cases were calculated using both the formula and the Python code. The input parameters, results (from both methods) and percent error (considering the simulation as "truth") have been tabulated below:

Γ [m^2/s]	R [m]	H [m]	$T_{simulation}$ [s]	$T_{formula}$ [s]	% Difference
1	1	1	12.81	12.92	0.86 %
1	2	0.5	5.02	5.000	0.40 %
3	3	1	6.61	6.62	0.15 %

Table 2: Collected Data from Python Script for Complete Leapfrog

From table 4, we can see the the formula derived above matched very closely to the values caluclated using the simulations, with an average error over the three test cases of **0.47**%

5 Task 4

Next we are interested when the vortex rings are separated by half their original distance, with their vortices having equal strength but opposite direction. The dimensional analysis used will be the same as in Task 3:

$$\frac{\Gamma T}{RH} = \psi\left(\frac{R}{H}\right)$$

Using our leapfrogging vortex Python script again, we can enter in data points and plot the results. The data points used are in Table 3.

Γ [m^2/s]	R [m]	H [m]	T [s]
1	1	1	1.01
1	2	1	2.81
2	1	1	0.51
1.5	1.5	1	1.21
1	1	0.5	0.75
0.5	1	0.5	1.41
0.8	2	0.2	2.41
1	0.8	0.4	0.45
1.2	0.7	0.6	0.405
5	2	1	0.561
5	2	2	0.841
2	4	1	4.4
2	5	1	7.01
2	6	1	10.4
2	7	1	14.3
2.0	7	1.2	13.8
2	7.2	1.4	13.8
2	6.8	1.4	12.7
1.6	6.5	1.5	14.3
1	3	1	5.4
1	1.6	0.2	1.21
1	2.5	1	4.0
0.5	0.8	1	1.61

Table 3: Collected Data from Python Script for Half Distance to Leapfrog

These data points were then input into another plotting and curve fitting Pyhton script. A linear fit was found for the data, and the results were plotted in figure 7 below.

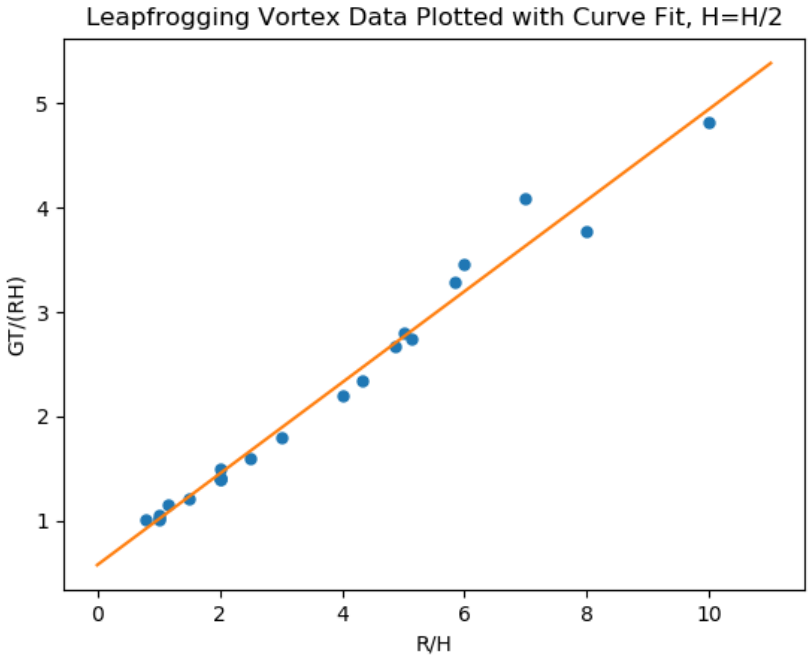

Figure 7: Collected Data for Expanding Vortices

From out fit line, we can see that:

$$\Pi 1 = 0.437\Pi 2 + 0.576$$

Subsituting the parameters that make up the Pi groups and solving for T results in the following expression:

$$\boxed{T = \frac{RH}{\Gamma}(0.437(\frac{R}{H}) + 0.576)]}$$

To validate this empirical formula, the time the rings take to reach half of their starting separation distance for 3 test cases were calculated using both the formula and the Python code. The input parameters, results (from both methods) and percent error (considering the simulation as "truth") have been tabulated below:

Γ [m^2/s]	R [m]	H [m]	$T_{simulation}$ [s]	$T_{formula}$ [s]	% Difference
1	1	10	6.91	6.197	10.3 %
2	4	6	11.05	10.41	5.8 %
6	1	6	0.71	0.65	8.4%

Table 4: Collected Data from Python Script for Complete Leapfrog

From table 4, we can see that the average error over the three test cases was **8.2**%. While this is not as low as the leap-frogging model, the error is still sufficiently low to validate the formula.

6 Running the Code

The simulation and plotting code for this project is fairly simple. First, make sure that you have Python 3 installed properly. Next, adjust the input parameters on lines 30-40 of "rings.py" (this is the script that performs the simulation). Save the file. To run, open a terminal window, navigate to the folder containing this submission on type "python rings.py" - the simulations should now run be visible on your screen. If the simulation runs slowly, try adjustinng the

"tStep" and "displayValue" variables, as these will change the time step of the simulation and how often the plots are updated.