Physics Simulation - Samhitha Bodangi

Description: The simulation models a damped simple harmonic motion. Specifically, it models a vertical spring and its position over time. Damped motion is where the amplitudes (or heights) of periodic motions gradually decrease over time due to external force. The external force can be due to air resistance and/or energy transfer.

The user will input the mass of the block (*m*), the spring constant (*k*), and the initial stretch distance (x_0) of the spring. The output is a CSV table with the position of the mass over time. It is assumed that the spring itself has a negligible mass.

With the user's inputs being *m, k,* and x_0, below is the final equation that models position vs. time in a vertical spring.

$$x = x_0 \cdot \left(e^{\left(\frac{-t}{\left(\frac{-20\pi\sqrt{\frac{m}{k}}}{ln(0.37)}\right)}\right)}\right) \cdot sin\left(\sqrt{\frac{k}{m}} \cdot t\right)$$

The equation was made by combining and simplifying the equations for the amplitude for damped simple harmonic motion and the simple harmonic motion position v. time equation.

Because an external force is acting on the vertical spring, the amplitude of each oscillation will be slightly less than the amplitude of the oscillation before. The amplitude for the displacement of damped oscillations can be written as

$$x_{max} = x_{0\,max} \cdot \left(e^{\left(\frac{-t}{\mathcal{T}}\right)}\right)$$

Where *t* is the time in seconds, and $\mathcal{T}$ is the time constant, which represents the time to lose 63% of the original displacement. This equation represents the amplitude of each oscillation, which decreases over time. The equation can be substituted in place of *A* in the equation for displacement versus time:

$$x = x_{0\,max} \cdot \left(e^{\left(\frac{-t}{\mathcal{T}}\right)}\right) \cdot sin\left(\sqrt{\frac{k}{m}} \cdot t\right)$$

In order to rewrite the time constant in terms of the user's inputs, we can find an expression for the time constant that ensures that 63% of the initial stretch distance is maintained for 10 cycles or more. The work for the time constant expression is below:

$$e^{\frac{-t}{\mathcal{T}}} = 0.37 \rightarrow \frac{-t}{\mathcal{T}} = ln(0.37) \rightarrow \mathcal{T} = \frac{-t}{ln(0.37)}$$

If we want 63% of the initial stretch distance to be maintained for at least 10 cycles, then 63% of the initial amplitude must be maintained for 10 periods. The period (*T*) of a spring-mass system can be written as

$$T = 2\pi\sqrt{\frac{m}{k}} \quad \rightarrow \quad 10\,periods = 20\pi\sqrt{\frac{m}{k}}$$

Substituting in for *t* gives the time constant expression as:

$$\mathcal{T} = \frac{-20\pi\sqrt{\frac{m}{k}}}{ln(0.37)}$$

Substituting this into our displacement v. time equation gives the first equation and can be used in the Java program to calculate the position of the mass in a vertical spring-mass system.

Coding: The "Simulation" class is the program where the user inputs the values for *m, k,* and x_0, and they get a CSV output which can then be pasted into Excel to make a graph.

However, there must be conditionals put in place to check whether the user's input values are within the domain of reasonable inputs for the three inputs. Below is the criteria for the domains of each input value:

Mass: Must be a positive, nonzero value. A negative value would make the time constant value imaginary (negative inside sq. rt.) and a mass of 0 kg is not realistic, and would make the equation undefined (dividing by 0).

Spring Constant: Similar to the mass domain, the spring constant must be a positive, nonzero value for the same reasons.

Stretch Distance: The stretch distance can be negative as it would represent a compression.

As this is purely a simulation, there are no upper limits on any of the values. However, if the user were to input unrealistic values, the output graph would reflect that. The full code is below:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Scanner;
public class Simulation {
    public static void main(String[] args) throws FileNotFoundException {

    Scanner scanner = new Scanner(System.in);

    double m;
    double k;
    double x;

    System.out.print("Enter the value of the mass of the block (m) in kg: ");
    m = scanner.nextDouble();
    while (m <= 0) {
       System.out.println("ERROR: The mass must be positive and non-zero. Please try again.");
       System.out.print("Enter the value of the mass of the block (m) in kg: ");
       m = scanner.nextDouble();
    }

    System.out.print("Enter the value of the spring constant (k) in N/m: ");
    k = scanner.nextDouble();

    while (k <= 0) {
       System.out.println("ERROR: The spring constant must be positive and non-zero. Please try again.");
       System.out.print("Enter the value of the spring constant (k) in N/m ");
       k = scanner.nextDouble();
    }
    System.out.print("Enter the value of the initial stretch distance (x) in m: ");
    x = scanner.nextDouble();

    scanner.close();
    double eExpDen = (-20 * Math.PI * Math.sqrt(m / k)) / (Math.log(0.35));

    File csvFile = new File("simulation.csv");
    PrintWriter out = null;
    try {
       out = new PrintWriter(csvFile);
```

```
        for (double t = 0.0; t <= 100.0; t += 0.1) {
            double y = x * (Math.exp(-t / eExpDen)) * Math.sin((Math.sqrt(k / m) * t));
            out.printf("%f, %f\n", t, y);
        }

        out.close();
        System.out.println("CSV file created successfully.");
    } catch (Exception e) {
        e.printStackTrace();
    }
  }
}
```

Testing: To test the program, two sets of inputs were tested and graphed:

Test 1:

The sample inputs are: mass = 1kg spring constant = 200 N/m initial displacement = 0.8m
The times for output were chosen as 1.2 seconds, 4.5 seconds, and 9.9 seconds

Predicted Outputs:
Using the developed equation, *m* was substituted as 1kg, *k* was substituted as 200 N/m, and x_0 was substituted as 0.8m. The equation was solved three times to find the exact position at *t*=1.2, *t*=4.5, and *t*=9.9 seconds:

$$0.8 \cdot \left(e^{\left(\frac{-1.2}{\left(\frac{-20\pi\sqrt{\frac{1}{200}}}{ln(0.37)}\right)}\right)}\right) \cdot sin\left(\sqrt{\frac{200}{1}} \cdot 1.2\right) = -0.5741$$

$$0.8 \cdot \left(e^{\left(\frac{-4.5}{\left(\frac{-20\pi\sqrt{\frac{1}{200}}}{ln(0.37)}\right)}\right)}\right) \cdot sin\left(\sqrt{\frac{200}{1}} \cdot 4.5\right) = 0.1997$$

$$0.8 \cdot \left(e^{\left(\frac{-9.9}{\left(\frac{-20\pi\sqrt{\frac{1}{200}}}{ln(0.37)}\right)}\right)}\right) \cdot sin\left(\sqrt{\frac{200}{1}} \cdot 9.9\right) = 0.0755$$

The calculated outcomes for a vertical spring-mass system with a mass of 1kg, a spring with a spring constant of 200 N/m, and an initial stretch of 0.8m are below:

Time (sec)	Displacement from Equilibrium (m)
1.2 seconds	-0.5741 meters
4.5 seconds	0.1997 meters
9.9 seconds	0.0755 meters

For the simulation, the code below was used to find the displacements for the specific times:

```
double ans1 = x * (Math.exp(-1.2 / eExpDen)) * Math.sin((Math.sqrt(k / m) * 1.2));
double ans2 = x * (Math.exp(-4.5 / eExpDen)) * Math.sin((Math.sqrt(k / m) * 4.5));
double ans3 = x * (Math.exp(-9.9 / eExpDen)) * Math.sin((Math.sqrt(k / m) * 9.9));

System.out.println("Displacement after 1.2 seconnds: " + ans1);
System.out.println("Displacement after 4.5 seconnds: " + ans2);
System.out.println("Displacement after 9.9 seconnds: " + ans3);
```

Below are the predicted outcomes from the simulation:

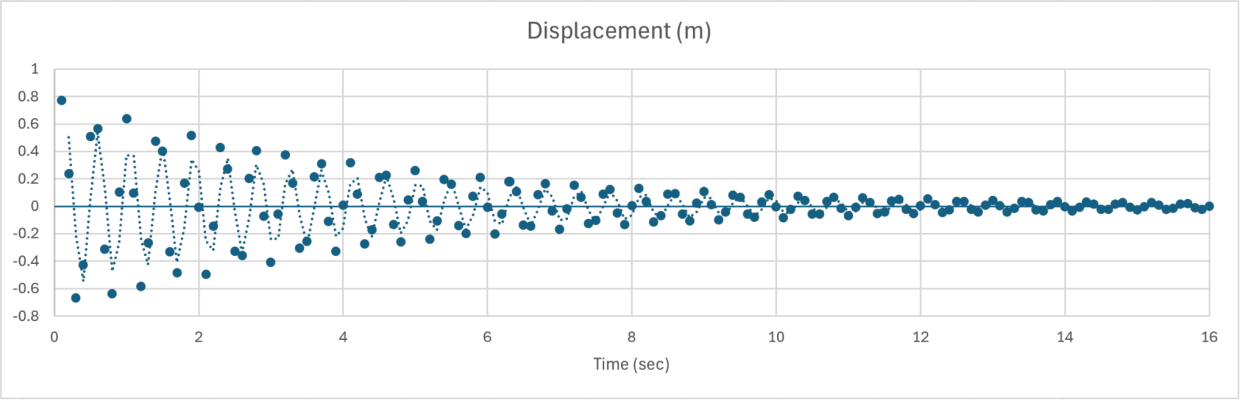
```
<terminated> Simulation [Java Application] C:\Users\samhi\.p2\pool\plugins\org.eclipse.
Enter the value of the mass of the block (m) in kg: 1
Enter the value of the spring constant (k) in N/m: 200
Enter the value of the initial stretch distance (x) in m: 0.8
Displacement after 1.2 seconnds: -0.5827773184672018
Displacement after 4.5 seconnds: 0.21121368218292672
Displacement after 9.9 seconnds: 0.08543110359712383
CSV file created successfully.
```

The table below compares the true and predicted displacements along with their respective percent errors:

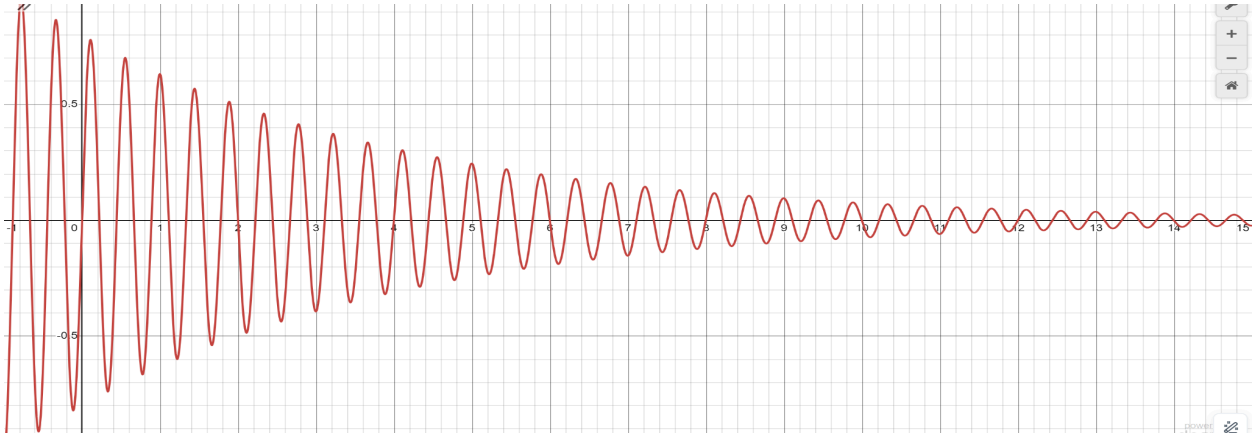
Time (sec)	True Displacement from Equilibrium (m)	Predicted Displacement from Equilibrium (m)	Percent Error
1.2 seconds	-0.5741 meters	-0.5828 meters	1.5154%
4.5 seconds	0.1997 meters	0.2112 meters	5.7586%
9.9 seconds	0.0755 meters	0.0854 meters	13.1126%

While relatively small, the percent error shows a slight difference between the predicted and true displacements. This could be because of intermediate roundings, or the choice to store the values as doubles. Because the computer has to convert the numbers and calculations into binary digits and back, there could be a loss in precision which lead to different predictions.

Using the CSV file, we can use Excel to plot a graph that depicts displacement v. time. Below is the graph made from the CSV file that was output from the program.



Below is the graph made by Desmos by entering the equation.



Test 2:

A similar process was completed as in Test 1 for another sample test case. The sample inputs were chosen as: mass = 0.5kg spring constant = 350 N/m initial displacement = -0.2m
The times for output were chosen as 2.6 seconds, 4.7 seconds, and 5.2 seconds

Predicted Outputs:
Using the developed equation, *m* was substituted as 0.5kg, *k* was substituted as 350 N/m, and x_0 was substituted as -0.2m. The equation was solved three times to find the exact position at *t*=2.6, *t*=4.7, and *t*=5.2 seconds:

$$-0.2 \cdot \left(e^{\left(\frac{-2.6}{\left(\frac{-20\pi\sqrt{\frac{0.5}{350}}}{ln(0.37)}\right)}\right)}\right) \cdot sin\left(\sqrt{\frac{350}{0.5}} \cdot 2.6\right) = 0.0203$$

$$-0.2 \cdot \left(e^{\left(\frac{-4.7}{\left(\frac{-20\pi\sqrt{\frac{0.5}{350}}}{ln(0.37)}\right)}\right)}\right) \cdot sin\left(\sqrt{\frac{350}{0.5}} \cdot 4.7\right) = 0.0242$$

$$-0.2 \cdot \left(e^{\left(\frac{-5.2}{\left(\frac{-20\pi\sqrt{\frac{0.5}{350}}}{ln(0.37)}\right)}\right)}\right) \cdot sin\left(\sqrt{\frac{350}{0.5}} \cdot 5.2\right) = 0.0121$$

The calculated outcomes for a vertical spring-mass system with a mass of 0.5kg, a spring with a spring constant of 350N/m, and an initial stretch of -0.2m are below:

Time (sec)	Displacement from Equilibrium (m)
2.6 seconds	0.0203 meters
4.7 seconds	0.0242 meters
5.2 seconds	0.0121 meters

For the simulation, the code below was used to find the displacements for the specific times:

```
double ans1 = x * (Math.exp(-2.6 / eExpDen)) * Math.sin((Math.sqrt(k / m) * 2.6));
double ans3 = x * (Math.exp(-4.7 / eExpDen)) * Math.sin((Math.sqrt(k / m) * 4.7));
double ans2 = x * (Math.exp(-5.2 / eExpDen)) * Math.sin((Math.sqrt(k / m) * 5.2));

System.out.println("Displacement after 2.6 seconnds: " + ans1);
System.out.println("Displacement after 4.7 seconnds: " + ans2);
System.out.println("Displacement after 5.2 seconnds: " + ans3);
```

Below are the predicted outcomes from the simulation:

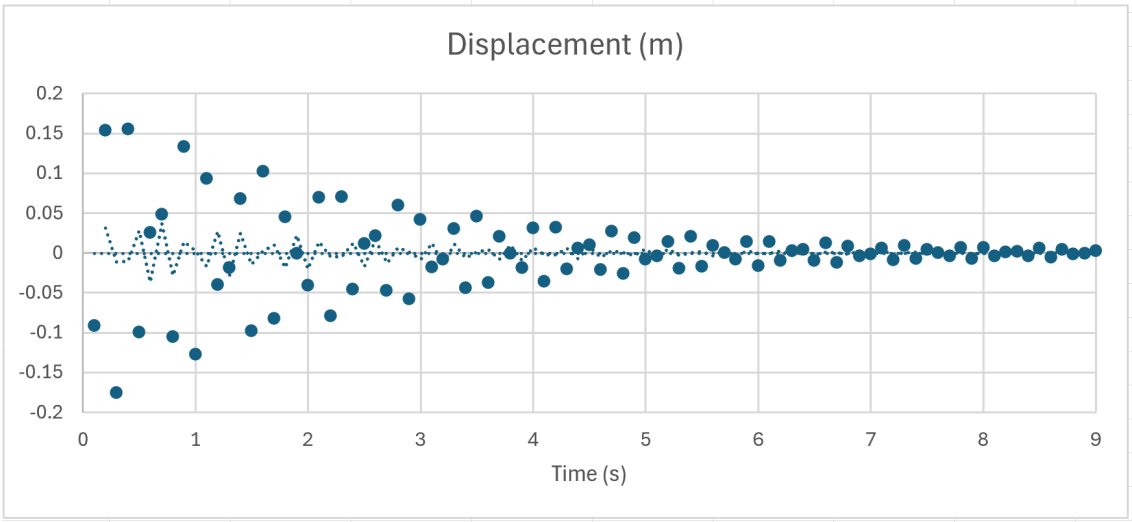
```
<terminated> Simulation [Java Application] C:\Users\samhi\.p2\pool\plugins\org.eclipse.ju
Enter the value of the mass of the block (m) in kg: 0.5
Enter the value of the spring constant (k) in N/m: 350
Enter the value of the initial stretch distance (x) in m: -0.2
Displacement after 2.6 seconnds: 0.021535212517831588
Displacement after 4.7 seconnds: 0.02703401094624835
Displacement after 5.2 seconnds: 0.013740813805534761
CSV file created successfully.
```

The table below compares the true and predicted displacements along with their respective percent errors:

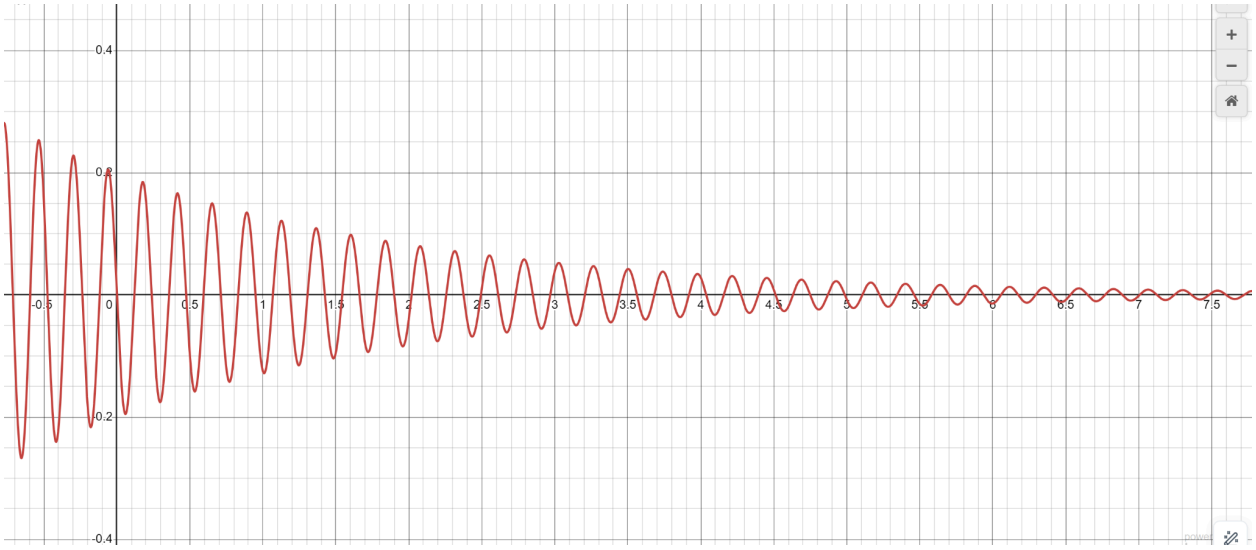
Time (sec)	True Displacement from Equilibrium (m)	Predicted Displacement from Equilibrium (m)	Percent Error
2.6 seconds	0.0203 meters	0.0215 meters	5.9113%
4.7 seconds	0.0242 meters	0.0270 meters	11.5702%
5.2 seconds	0.0121 meters	0.0137 meters	13.2231%

Similar to Test 1, there is still a slight difference between the true values and predicted values. Again, it could partly be because of the multiple conversions the computer has to compute, resulting in less precision and less accurate displacements.

Using the CSV file, we can use Excel to plot a graph that depicts displacement v. time. Below is the graph made from the CSV file that was output from the program.



Below is the graph made by Desmos by entering the equation.



In conclusion, even though the model is accurate in modeling the dampening of displacement over time in a spring mass system, there is still a slight difference between the model's values and the predicted values. Nonetheless, it serves as a great tool to understand the relationships between specific variables such as mass, the spring constant, and stretch distance on displacement over time.