Detailed Homework 4

Jason Press

September 15, 2024

1 Checks

For getting Δx when the sphere starts and stops at the same height, I use the following equation:

$$\Delta x = \frac{v_0^2}{g}\sin(2\theta) \tag{1}$$

All of the checks are correct, within a couple fractions of a second. The reason this simulation is not perfectly accurate is the `time_step` variable: our simulation is an *approximation* of reality, using a discrete procedural approach to determine the answer. This approach would give a perfect answer if `time_step = 0`, but that is not possible. Instead, I approximate it with `0.01` so the simulation runs in real time. To achieve better agreement with the equations, lower `time_step` to even closer to `0`, such as `0.00001`. However, this is more costly to run, since the computer has to run more calculations.

The check answers are rounded to the amount of significant figures in the program.

1.1 $v_0 = 10\frac{\mathbf{m}}{\mathbf{s}} \wedge \theta = 45^\circ$

$$\Delta x = \frac{10^2}{9.81}\sin(2\cdot 45^\circ)\text{m} = 10.19\text{m}$$

The simulation gives 10.32m. While this is not perfectly accurate, it is very close.

1.2 $v_0 = 10\frac{\mathbf{m}}{\mathbf{s}} \wedge \theta = 90^\circ$ (straight up)

$$\Delta x = \frac{10^2}{9.81}\sin(2\cdot 90^\circ)\text{m} = 0\text{m}$$

The simulation gives 0.00m, which is the correct answer.

1.3 $v_0 = 15\frac{\mathbf{m}}{\mathbf{s}} \wedge \theta = 60^\circ$

$$\Delta x = \frac{15^2}{9.81}\sin(2 \cdot 60^\circ)\text{m} = 19.86\text{m}$$

The simulation gives 19.95m. While this is not perfectly accurate, it is very close.

1.4 $v_0 = 10\frac{\mathbf{m}}{\mathbf{s}} \wedge \theta = 45^\circ \wedge = \frac{9.81}{2}$

$$\Delta x = \frac{10^2}{\frac{9.81}{2}}\sin(2 \cdot 45^\circ)\text{m} = 20.39\text{m}$$

The simulation gives 20.51m. While this is not perfectly accurate, it is very close.

2 Extra Credit: Starting Higher Up

The initial position of the ball in the original program is `projectile.pos = vector(0, 0, 0)`. By simply taking in a new input `initial_height` and making sure it is a non-negative number with a while loop, I can change the initial position to be `projectile.pos = vector(0, initial_height, 0)`.

To calculate the new total distance traveled, I can rearrange the kinematic equation in the y direction for time to give the times when $y = 0$:

$$t = \frac{-v_0 \sin\theta \pm \sqrt{v_0^2 \sin^2\theta + 4gy_0}}{2y_0} \tag{2}$$

Then, I can plug the two answers from Equation 2 into the kinematic equation in the x direction to get the following:

$$x = v_0 \cos\theta t = v_0 \cos\theta \frac{-v_0 \sin\theta \pm \sqrt{v_0^2 \sin^2\theta + 2gy_0}}{-g} \tag{3}$$

This will give two numbers: one less than or equal to zero, and one greater than or equal to zero. Only caring about the positive displacement, since the ball is traveling forwards in time, I can rewrite Equation 3 to the following:

$$x = v_0 \cos\theta t = v_0 \cos\theta \frac{v_0 \sin\theta + \sqrt{v_0^2 \sin^2\theta + 2gy_0}}{g} \tag{4}$$

The program still suffers the small inaccuracy described in checks.

2.1 $v_0 = 10\frac{\mathbf{m}}{\mathbf{s}} \wedge \theta = 45^\circ \wedge h_0 = 10\mathbf{m}$

The output of that equation gives 16.41m. The simulation returns 16.48m.

2.2 $v_0 = 15\frac{\mathbf{m}}{\mathbf{s}} \wedge \theta = 60^\circ \wedge h_0 = 20\mathbf{m}$

The output of that equation gives 128.04m. The simulation returns 28.12m.

From these couple examples, it is apparent that the simulation is as consistent with the equations as a discrete approximation can be.

3 Extra Credit: Maximum Height Coordinates

To achieve this, I added four variables to indicate the maximum y, and only updated them with the current values if the current y is higher than the maximum recorded y:

```
max_x = projectile.pos.x
max_y = projectile.pos.y
min_x_velo = projectile_velocity.x
min_y_velo = projectile_velocity.y

# In the simulation loop

        if max_y < projectile.pos.y:
            max_y = projectile.pos.y
            max_x = projectile.pos.x
            min_x_velo = projectile_velocity.x
            min_y_velo = projectile_velocity.y

        # Display the four variables
```

To find the minimum acceleration, I just used g, since the only component of acceleration in the simulation is g.

The following equation returns the maximum height when the initial height is equal to the final height:

$$y_{max} = \frac{v_0^2 \sin^2 \theta}{2g} \tag{5}$$

3.1 $v_0 = 10\frac{\mathbf{m}}{\mathbf{s}} \wedge \theta = 45^\circ$

The equation yields a maximum y of 2.5484. The simulation returns 2.5838.

3.2 $v_0 = 15\frac{\mathbf{m}}{\mathbf{s}} \wedge \theta = 60^\circ$

The equation yields a maximum y of 8.600. The simulation returns 8.666.

Additionally, for both of these simulations, the maximum x is half of the horizontal displacement, which is correct for when the initial height is equal to the final height.

4 Additional Resources

Link to ChatGPT conversation.

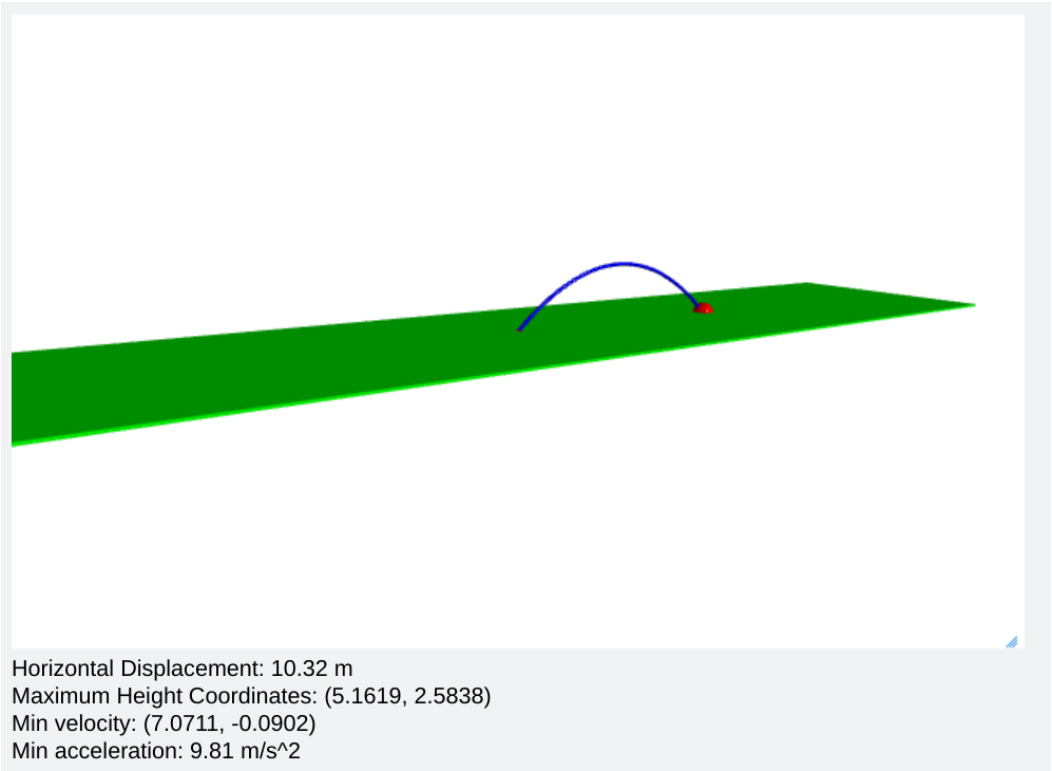
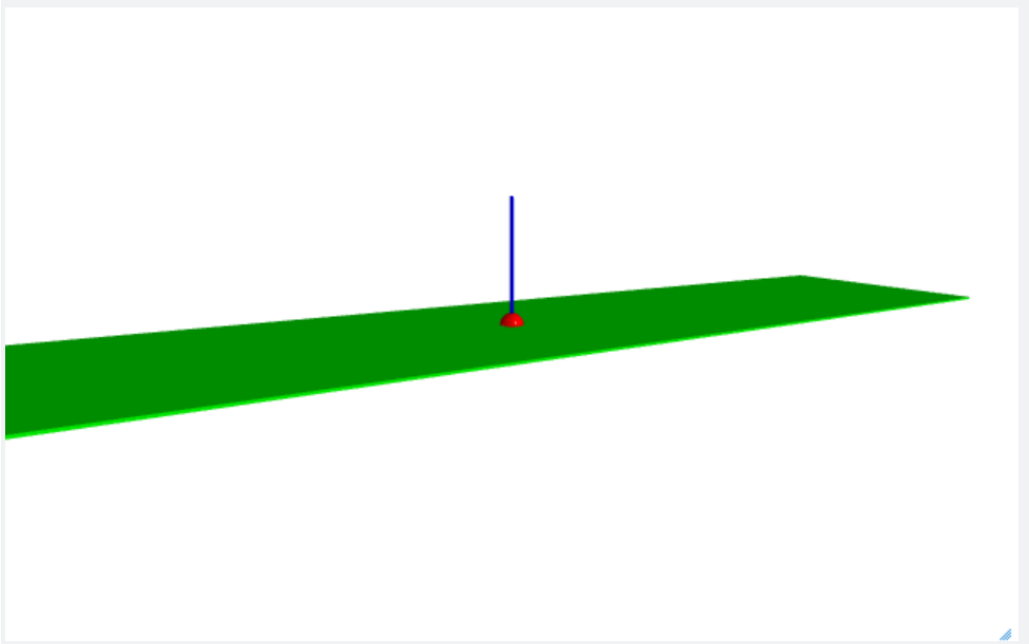
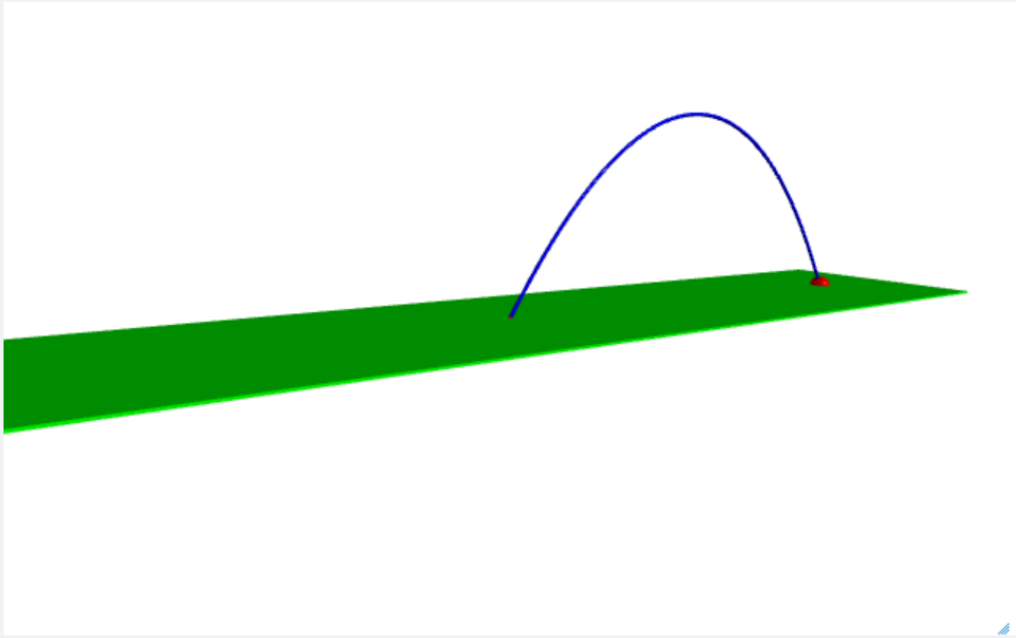


Figure 1: 10 meters per second, 45 degrees



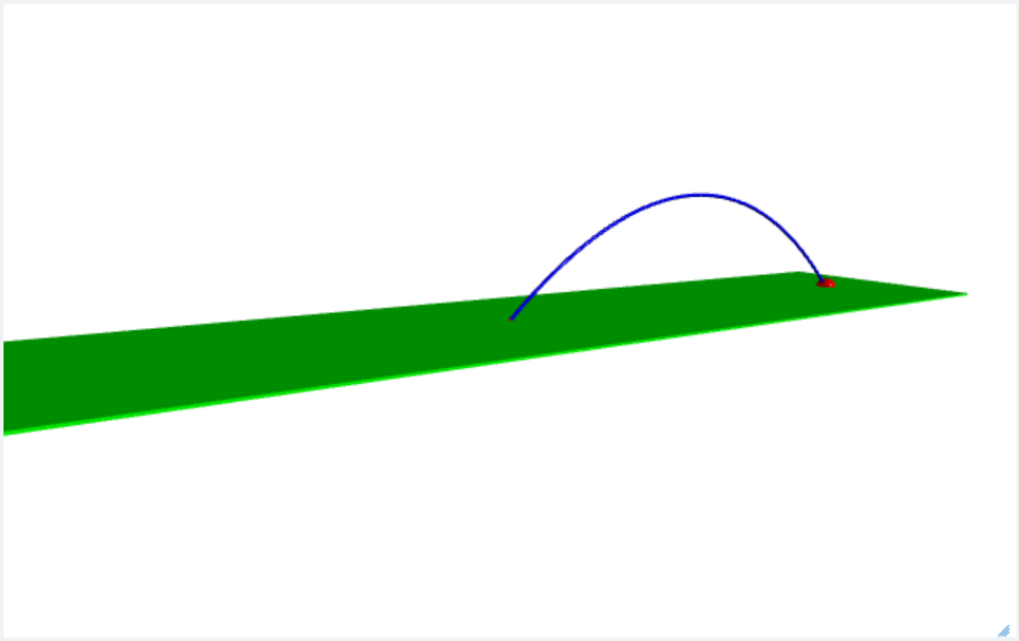
Horizontal Displacement: 0.00 m
Maximum Height Coordinates: (0, 5.1469)
Min velocity: (0, -0.0062)
Min acceleration: 9.81 m/s^2

Figure 2: 10 meters per second, 90 degrees



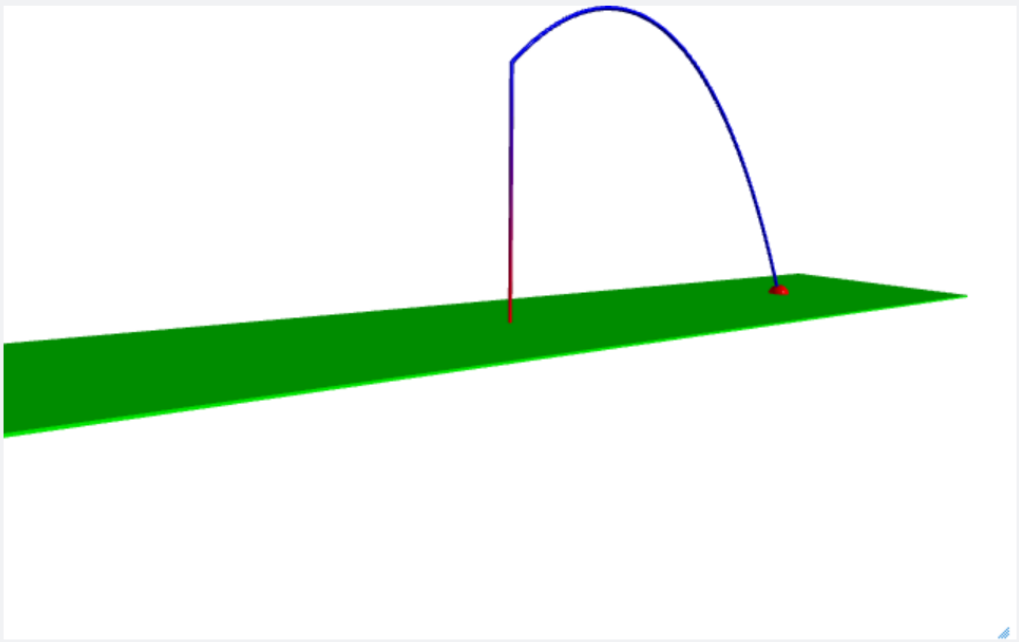
Horizontal Displacement: 19.95 m
Maximum Height Coordinates: (9.975, 8.666)
Min velocity: (7.5, -0.0569)
Min acceleration: 9.81 m/s^2

Figure 3: 15 meters per second, 60 degrees



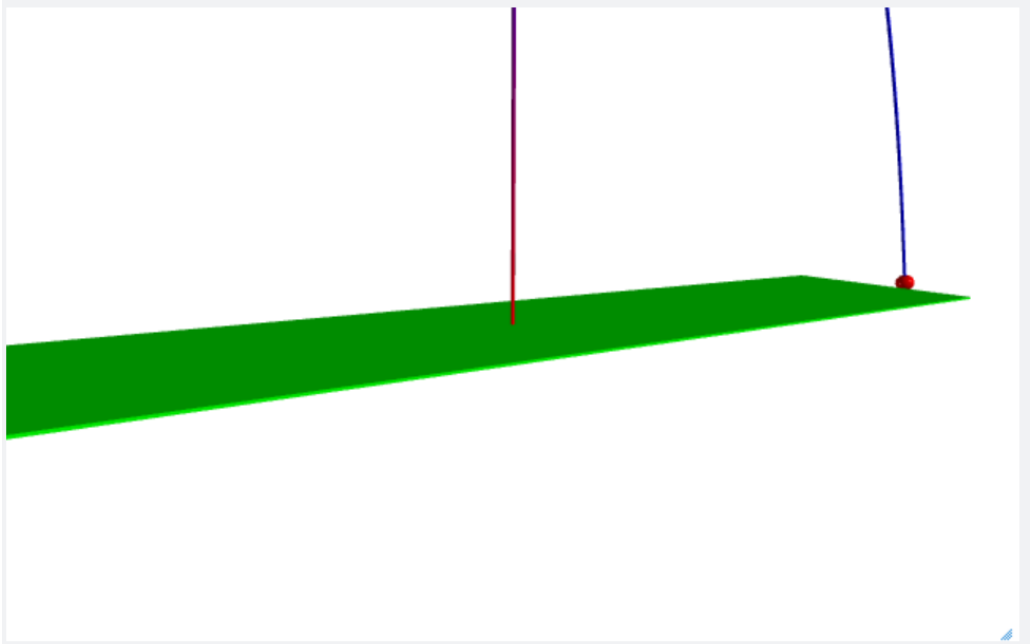
Horizontal Displacement: 20.51 m
Maximum Height Coordinates: (10.253, 5.1322)
Min velocity: (7.0711, -0.0412)
Min acceleration: 4.905 m/s^2

Figure 4: 10 meters per second, 45 degrees, half gravity



Horizontal Displacement: 16.48 m
Maximum Height Coordinates: (5.1619, 12.5838)
Min velocity: (7.0711, -0.0902)
Min acceleration: 9.81 m/s^2

Figure 5: 10 meters per second, 45 degrees, 10m up



Horizontal Displacement: 28.12 m
Maximum Height Coordinates: (9.975, 28.666)
Min velocity: (7.5, -0.0569)
Min acceleration: 9.81 m/s^2

Figure 6: 15 meters per second, 60 degrees, 20m up