Midterm

Introduction

The goal of this project is to study the motion of a spinning soccer ball in motion as someone takes a free kick. The ball is kicked with a certain amount of spin about each axis allowing the player to kick it with a certain trajectory to both avoid the wall and score the goal. The spin of the ball is very important because it allows the player to more easily avoid the wall and deceive the goalkeeper defending the goal.

To setup this problem, we will treat the field as a 3-Dimensional plane. The x-direction will be along the width of the field, the y-direction along the length, and the z-direction will be vertical. We will then place the origin at the center of the goal. From there we will place the ball on the field at the initial location (x_0, y_0, z_0). The player will then kick the ball giving it an initial velocity of (v_x^0, v_y^0, v_z^0). Once the ball has begun moving, there are now three forces that influence the trajectory of the ball. These are the gravitational force, the drag force due to air resistance, and the lift force due to the Magnus Effect. According to Newton's second law, from this situation we obtain the following formula

$$m\vec{a} = m\vec{g} - c_{drag}|v|\vec{v} + c_{lift}|v|\vec{s} \times \vec{v} \tag{1}$$

where $\vec{a} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}$ $\vec{v} = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}$ $\vec{s} = \begin{pmatrix} s_x \\ s_y \\ s_z \end{pmatrix}$ are the vectors for acceleration, velocity, and spin in 3-Dimensional space. In order to solve these equations we will need the constants: $m = .437$ kg is the mass of the ball, $g \approx 9.81\frac{m}{s^2}$ is the free-fall acceleration, $c_{drag} = .0057$ is the drag coefficient, and $c_{lift} = .0061$ is the lift (Magnus Effect) coefficient.

In order to make the problem clearer, we will now break this equation down into a system of second-order ODEs:

$$\begin{cases} m\dfrac{d^2x}{dt^2} = -c_{drag}|v|v_x + c_{lift}|v|(s_y v_z - s_z v_y) \\ m\dfrac{d^2y}{dt^2} = -c_{drag}|v|v_y + c_{lift}|v|(s_z v_x - s_x v_z) \\ m\dfrac{d^2z}{dt^2} = -mg - c_{drag}|v|v_z + c_{lift}|v|(s_x v_y - s_y v_x) \end{cases} \tag{2}$$

where $|v| = \sqrt{v_x^2 + v_y^2 + v_z^2}$ and $v_x = \frac{dx}{dt}, v_y = \frac{dy}{dt}, v_z = \frac{dz}{dt}$.

The starting position and velocity of the ball constitute initial conditions for the system above:

$$\begin{cases} x(0) = x^0, \quad y(0) = y^0, \quad z(0) = z^0 \\ \dfrac{dx}{dt}(0) = v_x^0, \quad \dfrac{dy}{dt}(0) = v_y^0, \quad \dfrac{dz}{dt}(0) = v_z^0 \end{cases} \tag{3}$$

Part 1

Task: Write the system of equations (2) as a system of first-order ODEs.

Given what we know about velocity and position and their relationship we can rewrite our system of second-order ODEs as a system of first-order ODEs. We know that $v_x = \frac{dx}{dt}, v_y = \frac{dy}{dt}, v_z = \frac{dz}{dt}$. So using that the system of equations becomes:

$$\begin{cases} \dfrac{dx}{dt} = v_x \\ \dfrac{dy}{dt} = v_y \\ \dfrac{dz}{dt} = v_z \\ \dfrac{dv_x}{dt} = \dfrac{-c_{drag}|v|v_x + c_{lift}|v|(s_y v_z - s_z v_y)}{m} \\ \dfrac{dv_y}{dt} = \dfrac{-c_{drag}|v|v_y + c_{lift}|v|(s_z v_x - s_x v_z)}{m} \\ \dfrac{dv_z}{dt} = \dfrac{-mg - c_{drag}|v|v_z + c_{lift}|v|(s_x v_y - s_y v_x)}{m} \end{cases} \tag{4}$$

And the initial conditions for that system of equations becomes:

$$\begin{cases} x(0) = x^0, \quad y(0) = y^0, \quad z(0) = z^0 \\ v_x(0) = v_x^0, \quad v_y(0) = v_y^0, \quad v_z(0) = v_z^0 \end{cases} \tag{5}$$

And so we have completed the task of rewriting the system of equations (2) as a system of first-order ODEs.

Part 2

Task: Write a MATLAB function for solving a general system of first-order ODEs of size m using the fourth-order Runge-Kutta Method (RK4). The function should take in as input parameters a vector-valued function $\vec{f} = \begin{pmatrix} f^{(1)} \\ ... \\ f^{(m)} \end{pmatrix}$, a vector of initial conditions $\vec{y_0} = \begin{pmatrix} y_0^{(1)} \\ ... \\ y_0^{(m)} \end{pmatrix}$, initial and final times t_{start}, and t_{final} and time-step Δt. The output of the function should be an array **t** containing all instants of time $t_0 = t_{start}, t_1 = t_0 + \Delta t, ..., t_{n_{total}} = t_{final}$ and a matrix **y** of size $m \times n_{total}$ containing numerical solutions at corresponding instants of time as columns.

The function is of the form:
function [t,y] = name_of_your_function(f, y0, t_start, t_final, dt)
...
end

The following steps were taken in the function to find the solution.

1) Find the size that the arrays y and t will be and preallocate them.
2) Put initial values into these preallocated arrays.
3) Loop through for all times t_{start} to t_{final} using the value dt. In this loop I will calculate the next values of y using the RK4 Method and the next value of t using dt. These values are then put into the return arrays y and t.

We can write our system of m first-order differential equations as:

$$\begin{cases} \dfrac{dy^{(1)}}{dt} = f^{(1)}(t, y^{(1)}, ..., y^{(m)}) \\ ... \\ \dfrac{dy^{(m)}}{dt} = f^{(m)}(t, y^{(1)}, ..., y^{(m)}) \end{cases} \tag{6}$$

and the initial conditions as:

$$\begin{cases} y^{(1)}(t_0) = y_0^{(1)} \\ ... \\ y^{(m)}(t_0) = y_0^{(1=m)} \end{cases} \tag{7}$$

where $\vec{y} = \begin{pmatrix} y^{(1)}(t) \\ \vdots \\ y^{(m)}(t) \end{pmatrix}$ are the solutions.

$\vec{f} = \begin{pmatrix} f^{(1)}(t, y^{(1)}, ..., y^{(m)}) \\ \vdots \\ f^{(m)}(t, y^{(1)}, ..., y^{(m)}) \end{pmatrix}$ are the given functions that take in m+1 variables.

$\vec{y_0} = \begin{pmatrix} y_0^{(1)} \\ \vdots \\ y_0^{(m)} \end{pmatrix}$ are the initial values of the solution given to the functions.

We use these vectors in the Runge-Kutta Method (RK4), which is used to solve systems of ODE's. This method of solving is fourth-order accurate. It is as follows:

$$\vec{y_{n+1}} = \vec{y_n} + \Delta t \left(\frac{\vec{K_1} + 2\vec{K_2} + 2\vec{K_3} + \vec{K_4}}{6}) \right) \tag{8}$$

We use the following equations to solve for the slopes $\vec{K_1}, \vec{K_2}, \vec{K_3}$, and $\vec{K_4}$.

$$\begin{aligned} \vec{K_1} &= \vec{f}(t_n, \vec{y_n}) \\ \vec{K_2} &= \vec{f}(t_n + .5\Delta t, \vec{y_n} + .5\Delta t \vec{K_1}) \\ \vec{K_3} &= \vec{f}(t_n + .5\Delta t, \vec{y_n} + .5\Delta t \vec{K_2}) \\ \vec{K_4} &= \vec{f}(t_n + \Delta t, \vec{y_n} + \Delta t \vec{K_3}) \end{aligned} \tag{9}$$

Part 3

Task: Test your MATLAB function using the system of equations

$$\begin{cases} \dfrac{dy^{(1)}}{dt} = y^{(2)} \\ \dfrac{dy^{(2)}}{dt} = sin(1 - exp(y^{(3)}) \\ \dfrac{dy^{(3)}}{dt} = \dfrac{1}{1 - ln(y^{(5)} - 1)} \\ \dfrac{dy^{(4)}}{dt} = \left(\dfrac{1}{2}t^2 + t\right) y^{(2)} + exp(y^{(3)})y(1) \\ \dfrac{dy^{(5)}}{dt} = 1 - y^{(5)} \\ \dfrac{dy^{(6)}}{dt} = -3\left(\dfrac{exp(y^{(3)}) - 1}{1 + t^3}\right)^2 \end{cases}$$

with the initial conditions

$$y^{(1)}(0) = 0, \quad y^{(2)}(0) = 1, \quad y^{(3)}(0) = 0, \quad y^{(4)}(0) = 0, \quad y^{(5)}(0) = 2, \quad y^{(6)}(0) = 1$$

The exact solution to this test problem is

$$\begin{cases} y^{(1)}_{exact}(t) = sin(t) \\ y^{(1)}_{exact}(t) = cos(t) \\ y^{(1)}_{exact}(t) = ln(1 + t) \\ y^{(1)}_{exact}(t) = \left(\dfrac{1}{2}t^2 + t\right) sin(t) \\ y^{(1)}_{exact}(t) = 1 + exp(-t) \\ y^{(1)}_{exact}(t) = \dfrac{1}{1 + t^3} \end{cases}$$

Take $t_{start} = 0, t_{final} = 1$. Compute the order of accuracy using time-steps $\Delta t = 0.1$, 0.05, 0.025, 0.0125 and 0.00625 and make sure that your ODE solver produces the expected order of accuracy. Use the following formula to calculate the error of a numerical solution

$$E = max\left(\sqrt{(y^{(1)}_{exact}(t_n) - y^{(1)}_n)^2 + ... + (y^{(6)}_{exact}(t_n) - y^{(6)}_n)^2}\right) \quad for\ 1 \le n \le n_{total}$$

Δt	Max-Error	Order of Accuracy
.01	5.55667e-06	4.046657297219
.05	3.36240e-07	4.023664714592
.025	2.06731e-08	4.011691756693
.0125	1.28164e-09	4.005932267986
.00625	7.97738e-11	4.004899168490

Table 1: Errors and Orders of Accuracy

Part 4

Task: Use your function to solve the system of equations describing the motion of a spinning ball and obtain trajectories of the ball for each of the cases presented in Table 1 (in all cases $c_{drag} = 0.0057, c_{lift} = 0.0061, h_{wall} = 2$). Take $t_{start} = 0, t_{final} = 2$ and $\Delta t = 0.02$.

	Location			Velocity			Spin			Wall			
case no.	x^0	y^0	z^0	v_x^0	v_y^0	v_z^0	s_x	s_y	s_z	x_{start}^{wall}	y_{start}^{wall}	x_{end}^{wall}	y_{end}^{wall}
1	-15	23	0	18	-23	8.5	0.10	0.10	-0.99	-11.2	14.7	-8.6	16.0
2	25.5	15	0	-28	-11	7.5	0	0	1	18.1	9.7	17	11.3
3	-4	35	0	-8	-36	5	-0.32	0	0.95	-4.3	25.9	-0.8	25.9
4	16	28	0	-25	-20	8	0.10	0.15	0.98	12.6	19.6	9.9	20.8

Table 2: Initial values and constants

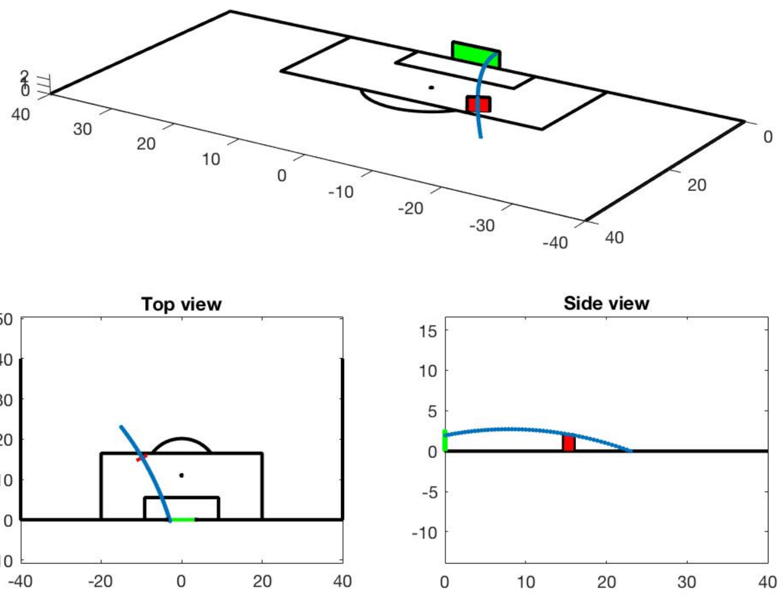

Figure 1: Case 1

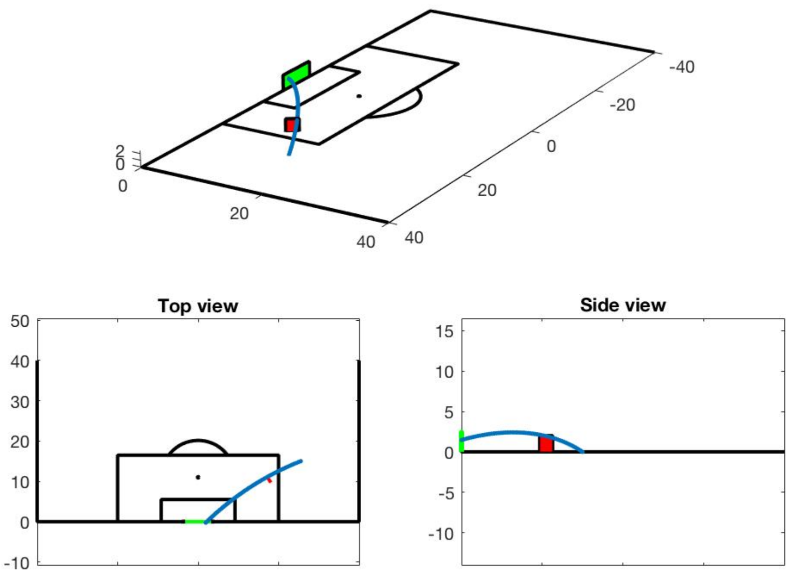

Figure 2: Case 2

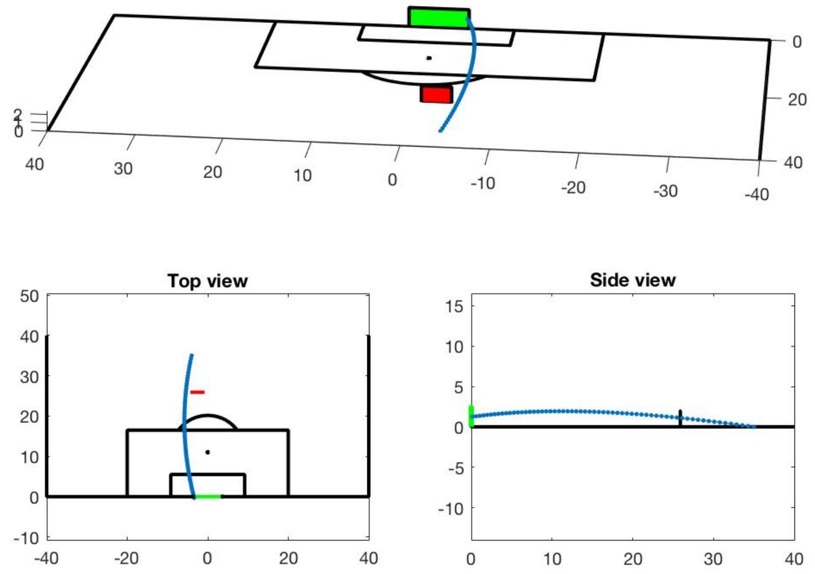

Figure 3: Case 3

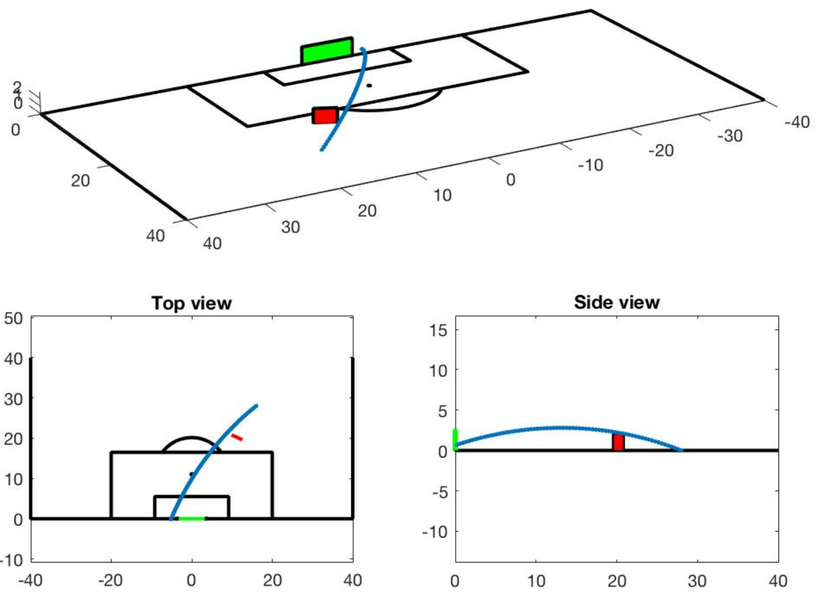

Figure 4: Case 4

Part 5

Task: Write a code that analyzes the trajectories of the ball and determines whether the ball hits the defensive wall and whether the ball goes into the goal. Determine the outcome of the free kick for each of the cases in Table 1.

To do this I first calculated the trajectory of the ball which gave me the position from time t_{start} to t_{final}. I then used a set of equations to check whether it hit the wall and another set to check if it went in the goal.

In order to check if it hit the wall, I needed to check for 2 points where the ball passes by the wall. In order to do this I had to use the equations:

$$\begin{aligned} y_n &> y_s^w + (x_n - x_s^w)\frac{y_e^w - y_s^w}{x_e^w - x_s^w} \\ y_{n+1} &< y_s^w + (x_{n+1} - x_s^w)\frac{y_e^w - y_s^w}{x_e^w - x_s^w} \end{aligned} \tag{10}$$

where x_s^w = start of wall x-coordinate, x_e^w = end of wall x-coordinate, y_s^w = start of wall y-coordinate, and y_e^w = end of wall y-coordinate.

For a point n, it must satisfy both of these equations in order to tell us that the ball has passed the wall. When we find the correct value of n, we can then use this to solve for the x, y, and z coordinates at the moment it passes the wall. These will be called x^*, y^*, and z^*. In order to begin to solve for these we must set the two equations we have for y^* equal to each other to find an equation for x^*.

$$y_n + (x^* - x_n)\frac{y_{n+1} - y_n}{x_{n+1} - x_n} = y_s^w + (x^* - x_s^w)\frac{y_e^w - y_s^w}{x_e^w - x_s^w} \tag{11}$$

When we solve for x^* we get the equation:

$$x^* = \frac{y_s^w - y_n + x_n\frac{y_{n+1} - yn}{x_{n+1} - x_n} - x_s^w\frac{y_e^w - y_s^w}{x_e^w - x_s^w}}{\frac{y_{n+1} - y_n}{x_{n+1} - x_n} - \frac{y_e^w - y_s^w}{x_e^w - x_s^w}} \tag{12}$$

We now solve for x^*, which allows us to solve for y^* and z^* using the following equations:

$$\begin{aligned} y^* &= y_n + (x^* - x_n)\frac{y_{n+1} - y_n}{x_{n+1} - x_n} \\ z^* &= z_n + (x^* - x_n)\frac{z_{n+1} - z_n}{x_{n+1} - x_n} \end{aligned} \tag{13}$$

Now that we have the coordinates at the exact time that the ball passes by the wall, we check if these coordinates are within the bounds of the wall. If they are then the ball has hit the wall and if they aren't then it hasn't.

Next we need to check if the ball goes in the goal. This time rather than check if the ball has passed by the wall, we need to check if the ball is passing the goal line. So we need to find the point n where $y(n) > 0$ and $y(n+1) < 0$. We are checking for a point n where $y(n)$ is just before the goal line and $y(n+1)$ is just after the goal line. Now from here we use an equation to solve for the exact time t^*. In this case y^* (the y-coordinate when passing the goal line) = 0.

$$\begin{aligned} y^* &= y_n + (t^* - t_n)\frac{y_{n+1} - y_n}{t_{n+1} - t_n} \\ t^* &= -y_n\frac{t_{n+1} - t_n}{y_{n+1} - y_n} + t_n \end{aligned} \tag{14}$$

Now that we have the exact time where the ball crosses the goal line, we can solve for the x-coordinate and z-coordinate at this time, x^* and z^*. We will use the following equations:

$$\begin{aligned} x^* &= x_n + (t^* - t_n)\frac{x_{n+1} - x_n}{t_{n+1} - t_n} \\ x^* &= z_n + (t^* - t_n)\frac{z_{n+1} - z_n}{t_{n+1} - t_n} \end{aligned} \tag{15}$$

We have now found the x and z coordinates for the ball as it passes the goal line. Now all we have to do is check weather these coordinates are within the bounds of the goal and if they are, then we have scored.

Case 1	Missed the wall and went in the goal.
Case 2	Hit the wall but would have gone in if the wall wasn't there.
Case 3	Missed the wall and went in the goal.
Case 4	Missed the wall and missed the goal.

Table 3: Test case outcomes

References

1 Daniil Bochkov, CS 111 - Introduction to Computational Science - Midterm, Fall 2017

2 Daniil Bochkov, CS 111 - Introduction to Computational Science - Lecture 4 - High-Order Methods, Fall 2017

3 Daniil Bochkov, CS 111 - Introduction to Computational Science - Lecture 6 - Systems of ODEs, Fall 2017

4 Daniil Bochkov, CS 111 - Introduction to Computational Science - Lecture 7 - High-Order ODEs, Fall 2017