Midterm Report

1 Introduction

In this exercise we consider the scenario of a direct free kick in soccer. We will determine the differential equations of the forces acting on the soccer ball, reduce those equations to a system of first-order differential equations, and solve for the trajectory of ball using an implementation of Runge-Kutta's fourth-order accurate method (RK4). We will also test our implementation of RK4 using a simpler system of ODE with a known exact solution. Finally, we will plot the trajectory of the soccer ball and compute the result of the free kick, whether that be a collision with the defending team's "wall", a goal, or a total miss.

2 Reducing A Higher-Order ODE To A System of First-Order ODE

From Newton's second law (F = ma), we have the following system of second-order differential equations describing the forces acting on a soccer ball which has been kicked:

$$\begin{cases} m\frac{d^2x}{dt^2} = -c_{drag}|v|v_x + c_{lift}|v|(s_yv_z - s_zv_y) \\ m\frac{d^2y}{dt^2} = -c_{drag}|v|v_y + c_{lift}|v|(s_zv_x - s_xv_z) \\ m\frac{d^2z}{dt^2} = -mg - c_{drag}|v|v_x + c_{lift}|v|(s_xv_y - s_yv_x) \end{cases} \tag{1}$$

(1) can be simplified a little further by dividing each equation by mass:

$$\begin{cases} \frac{d^2x}{dt^2} = \frac{-c_{drag}|v|v_x + c_{lift}|v|(s_yv_z - s_zv_y)}{m} \\ \frac{d^2y}{dt^2} = \frac{-c_{drag}|v|v_y + c_{lift}|v|(s_zv_x - s_xv_z)}{m} \\ \frac{d^2z}{dt^2} = -g + \frac{-c_{drag}|v|v_x/m + c_{lift}|v|(s_xv_y - s_yv_x)}{m} \end{cases} \tag{2}$$

From here we want to rewrite (2) as a system of first-order differential equations. To do this we use the following substitutions:

$$\begin{cases} \frac{dx}{dt} = v_x \\ \frac{d^2x}{dt^2} = \frac{dv_x}{dt} \\ \frac{dy}{dt} = v_y \\ \frac{d^2y}{dt^2} = \frac{dv_y}{dt} \\ \frac{dz}{dt} = v_z \\ \frac{d^2z}{dt^2} = \frac{dv_z}{dt} \end{cases} \tag{3}$$

Using (3) we can rewrite (2) as:

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

Solving (4) will require six initial conditions, one for each first-order ODE.

3 Testing RK4

To test the MATLAB function, we use the example system of first-order ODE:

$$\begin{cases} \frac{dy_1}{dt} = y_2 \\ \frac{dy_2}{dt} = \sin\left(1 - \exp(y_3)\right) \\ \frac{dy_3}{dt} = \frac{1}{1 - \ln(y_5 - 1)} \\ \frac{dy_4}{dt} = \left(\frac{1}{2}t^2 + t\right) y_2 + \exp(y_3) y_1 \\ \frac{dy_5}{dt} = 1 - y_5 \\ \frac{dy_6}{dt} = -3\left(\frac{\exp(y_3) - 1}{1 + t^3}\right)^2 \end{cases} \tag{5}$$

With initial conditions: $y_1(0) = 0$, $y_2(0) = 1$, $y_3(0) = 0$, $y_4(0) = 0$, $y_5(0) = 2$, $y_6(0) = 1$. Taking $t_{start} = 0$, $t_{final} = 1$, we test the MATLAB function using $\Delta t = 0.1$, 0.05, 0.025, 0.0125, and 0.00625. Once the approximate solutions for each Δt are obtained, we compare the approximate solutions at each time t with the exact solution to (5):

$$\begin{cases} y_1(t) = \sin(t) \\ y_2(t) = \cos(t) \\ y_3(t) = \ln(1 + t) \\ y_4(t) = \left(\frac{1}{2}t^2 + t\right)\sin(t) \\ y_5(t) = 1 + \exp(-t) \\ y_6(t) = \frac{1}{1 + t^3} \end{cases} \tag{6}$$

The error for each Δt trial will be calculated as:

$$E = \max_{1 \le n \le n_{total}} \left(\sqrt{\left(y_1(t_n) - y_1^{(n)}\right)^2 + ... + \left(y_6(t_n) - y_6^{(n)}\right)^2} \right) \tag{7}$$

Using the results of (7), we calculate the order of accuracy for each Δt:

$$\text{Order k} = \frac{log\left(\frac{E(\Delta t_n)}{E(\Delta t_{n+1})}\right)}{log\left(\frac{\Delta t_n}{\Delta t_{n+1}}\right)} \tag{8}$$

The results of these calculations were:

Δt	Max Error	Order k
0.100000	5.55667e-06	-
0.050000	3.3624e-07	4.046656
0.025000	2.06731e-08	4.023665
0.012500	1.28164e-09	4.011696
0.006250	7.9774e-11	4.005926

Table 1: Accuracy of the RK4 implemented in MATLAB

As expected, the RK4 method implemented in MATLAB is fourth-order accurate.

4 Calculating Trajectory of a Soccer Ball Using RK4

We now apply the function to solve the problem of a free-kick in soccer. Taking as parameters g = -9.81 m/s^2, c_{drag} = 0.0057, c_{lift} = 0.0061, $wall_{height}$ = 2 meters, t_{start} = 0, t_{final} = 2 seconds, Δt = 0.02 seconds, we approximate the trajectory of the ball and plot it using **plot_trajectory.m**

- Case 1:

Table 2: Parameters and initial conditions for Case 1.

Location			Velocity			Spin			Wall			
x_0	y_0	z_0	v_x^0	v_y^0	v_z^0	s_x	s_y	s_z	x_{start}	y_{start}	x_{end}	y_{end}
-15	23	0	18	-23	8.5	0.10	0.10	-0.99	-11.2	14.7	-8.6	16.0

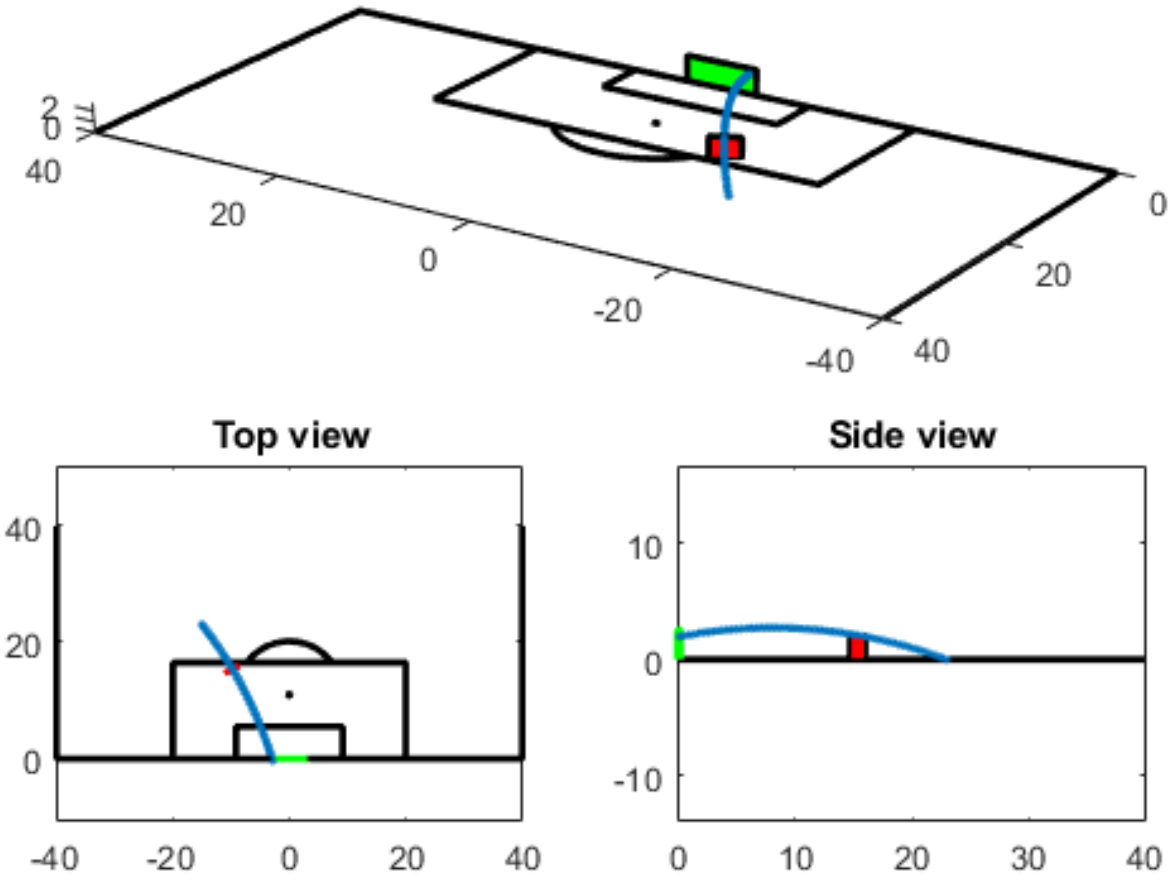

Figure 1: Trajectory of the ball in Case 1.

- Case 2:

Table 3: Parameters and initial conditions for Case 2.

Location			Velocity			Spin			Wall			
x_0	y_0	z_0	v_x^0	v_y^0	v_z^0	s_x	s_y	s_z	x_{start}	y_{start}	x_{end}	y_{end}
25.5	15	0	-28	-11	7.5	0	0	1	18.1	9.7	17	11.3

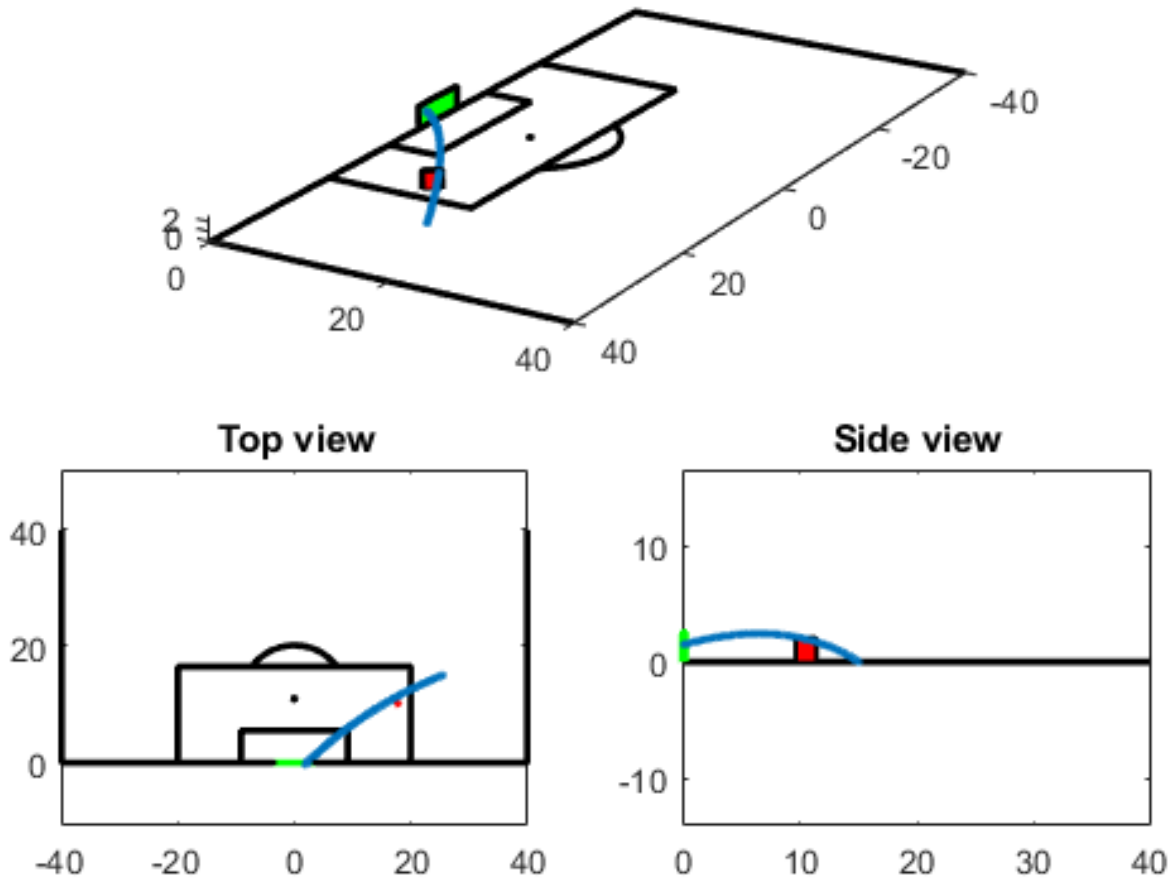

Figure 2: Trajectory of the ball in Case 2.

- Case 3:

Table 4: Parameters and initial conditions for Case 3.

Location			Velocity			Spin			Wall			
x_0	y_0	z_0	v_x^0	v_y^0	v_z^0	s_x	s_y	s_z	x_{start}	y_{start}	x_{end}	y_{end}
-4	35	0	-8	-36	5	-0.32	0	0.95	-4.3	25.9	-0.8	25.9

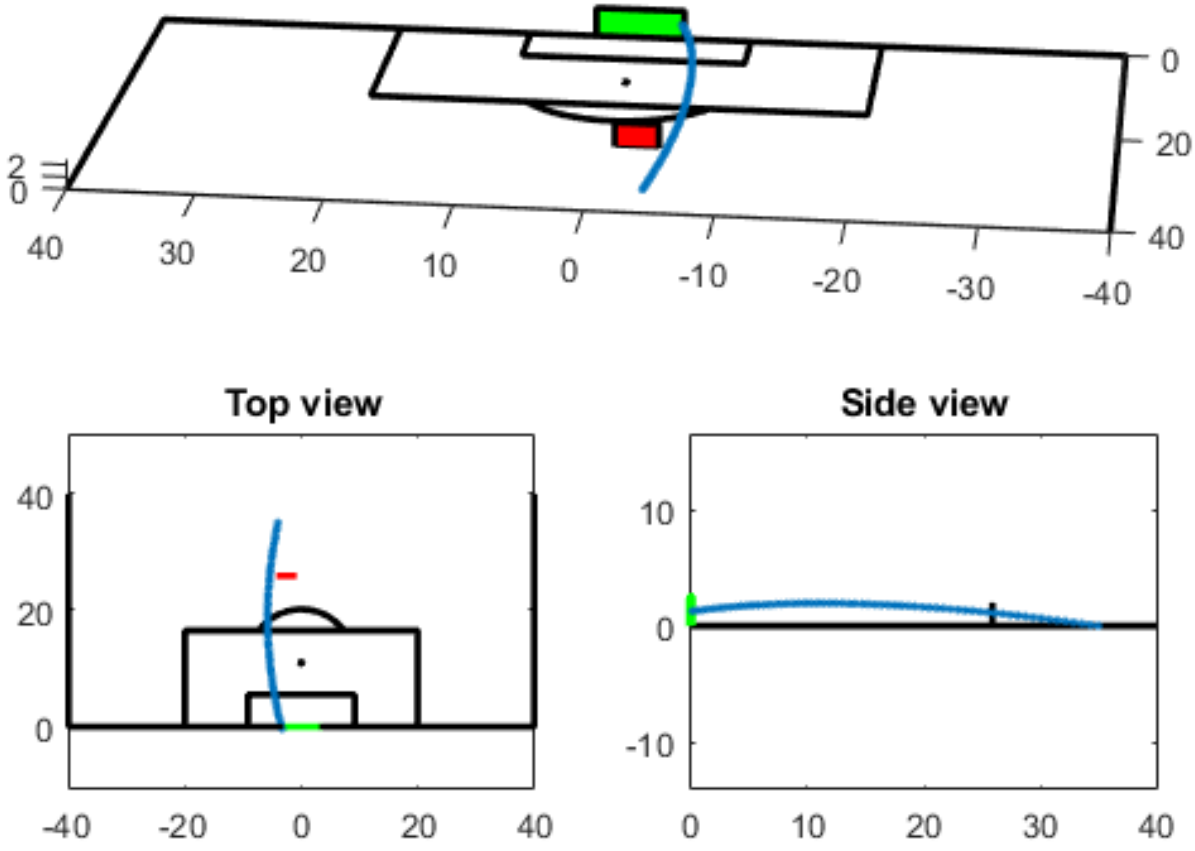

Figure 3: Trajectory of the ball in Case 3.

- Case 4:

Table 5: Parameters and initial conditions for Case 4.

Location			Velocity			Spin			Wall			
x_0	y_0	z_0	v_x^0	v_y^0	v_z^0	s_x	s_y	s_z	x_{start}	y_{start}	x_{end}	y_{end}
16	28	0	-25	-20	8	0.10	0.15	0.98	12.6	19.6	9.9	20.8

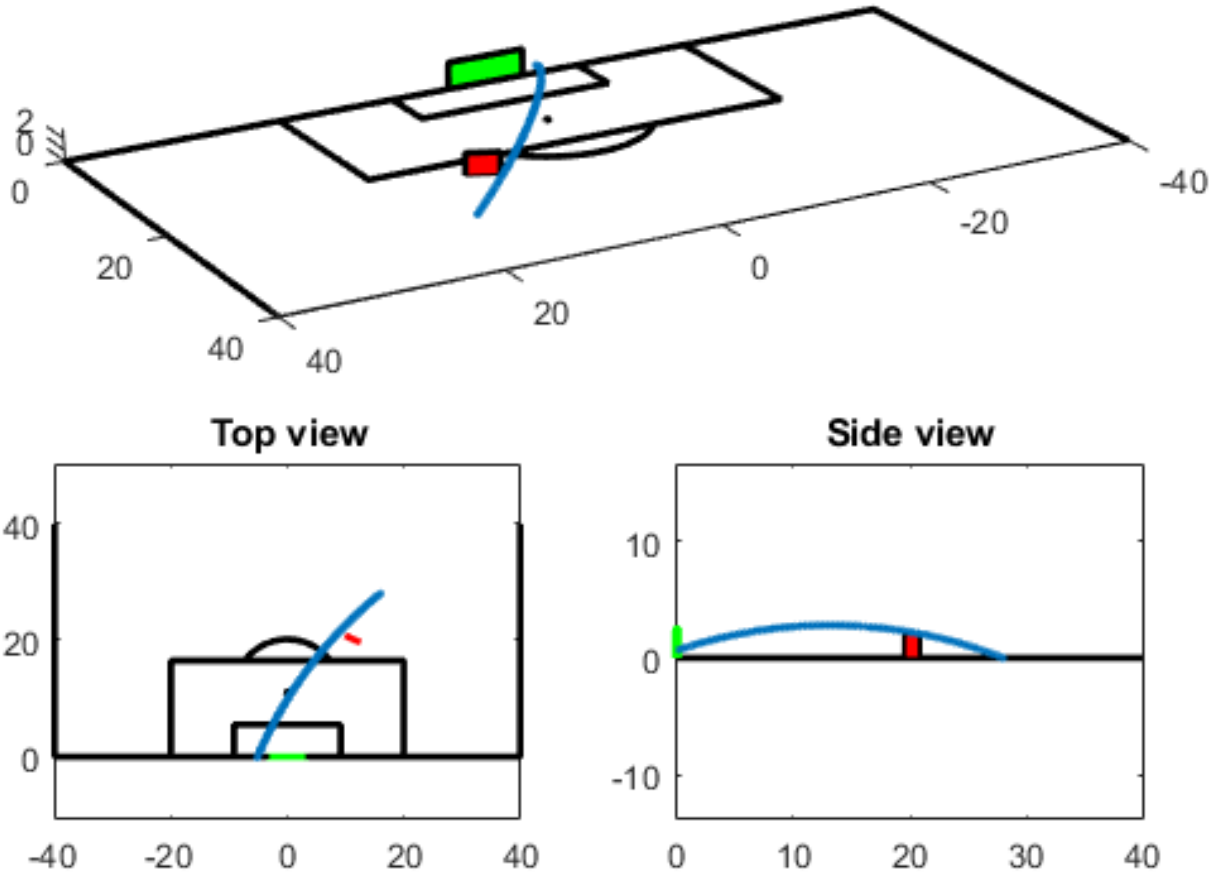

Figure 4: Trajectory of the ball in Case 4.

5 Analyzing Trajectory

To analyze whether or not the ball hit the wall, or went in the goal, or missed both, we do the following:

1. Calculate the trajectory of the ball for each scenario using an RK4 approximation.
2. Determine the equation for the xy hyper-plane that the wall (or goal) lies on.

   This is done by calculating the slope and y-intercept of the wall:

   $$\begin{cases} slope = m = \frac{wall_y^{end} - wall_y^{start}}{wall_x^{end} - wall_x^{start}} \\ y - intercept = b = -m(wall_x^{start}) + wall_y^{start} \end{cases} \tag{9}$$

   Or in the goal's case: This is done by calculating the slope and y-intercept of the wall:

   $$\begin{cases} slope = m = 0 \\ y - intercept = b = 0 \end{cases} \tag{10}$$

3. Iterate through the position solutions checking for the condition that the ball's position along the y-axis at t_n is above the wall (or goal), and the ball's position along the y-axis at t_{n+1} is below the wall (or goal). That is, $y_n \geq m(x_n) + b$ and $y_{n+1} \leq m(x_{n+1}) + b$. In the case of checking for a goal, this simplifies to $y_n \geq 0$ and $y_{n+1} \leq 0$.
4. Create an equation for the xy hyper-plane that the trajectory of the ball from t_n to t_{n+1} lies on.

   In this case we take a different approach than (9). To make it easier to solve for the intersection of the hyper-planes, we will get the equations in the convenient form $Ax + By = C$, where the coefficients A,B,and C are calculated as:

   $$\begin{cases} A = y_{n+1} - y_n \\ B = -(x_{n+1} - x_n) \\ C = A(x_n) + B(y_n) \end{cases} \tag{11}$$

   Likewise for the wall (or goal), using the start and end points of x and y.
5. Find the intersection of these two hyper-planes.

   For this step we put our equations for the ball trajectory and wall (or goal) in the matrices

   M =
   $$\begin{bmatrix} A_1 & B_1 \\ A_2 & B_2 \end{bmatrix}$$
   C =
   $$\begin{bmatrix} C_1 \\ C_2 \end{bmatrix}$$

   and find the intersection in MATLAB with intersection = M\C

6. Check the x value of the intersection point and see that it is between the start and end of the wall (or goal) along the x-axis. (alternatively you can check if the y value at the point of intersection is between the start and end of the wall (or goal) along the y-axis.

7. Calculate a new $\Delta t^* | \Delta t^* \leq \Delta t$, which will be the approximate time of intersection of the xy hyperplanes.

   We calculate $\Delta t^* = \Delta t \frac{|x_{intersection} - x_n|}{|x_{n+1} - x_n|}$

8. Approximate the ball's position at time $t_n + \Delta t^*$ by calling the RK4 method again with $t_{start} = t_n$, $t_{final} = t_n + \Delta t^*$, dt = Δt^*, and initial conditions as the column vector of the ball's position and velocity at time t_n.

9. Check if the ball's position along the z-axis, z_n^*, satisfies $0 \leq z_n^* \leq wall_{height}$ (or $goal_{height}$).

If each condition checked is satisfied for the wall, then the ball collided with the wall. Likewise, if all conditions checked are satisfied for the goal, then the ball made it in the goal, but only if the ball did not previously collide with the wall of course.

The results for Cases 1-4 are:
Case 1: Goal!
Case 2: Ball collided with the wall.
Case 3: Goal!
Case 4: Ball did not collide with the wall and missed the goal.

6 Conclusion

We saw in this exercise that a higher-order ODE can be reduced to a system of first-order ODE and solved numerically using an approximation method (RK4). The RK4 implementation was fourth-order accurate, as expected, and the application of this technique to a direct free kick in soccer allowed us to get a feel for how players take advantage of the Magnus effect to curve the ball around a defensive wall.