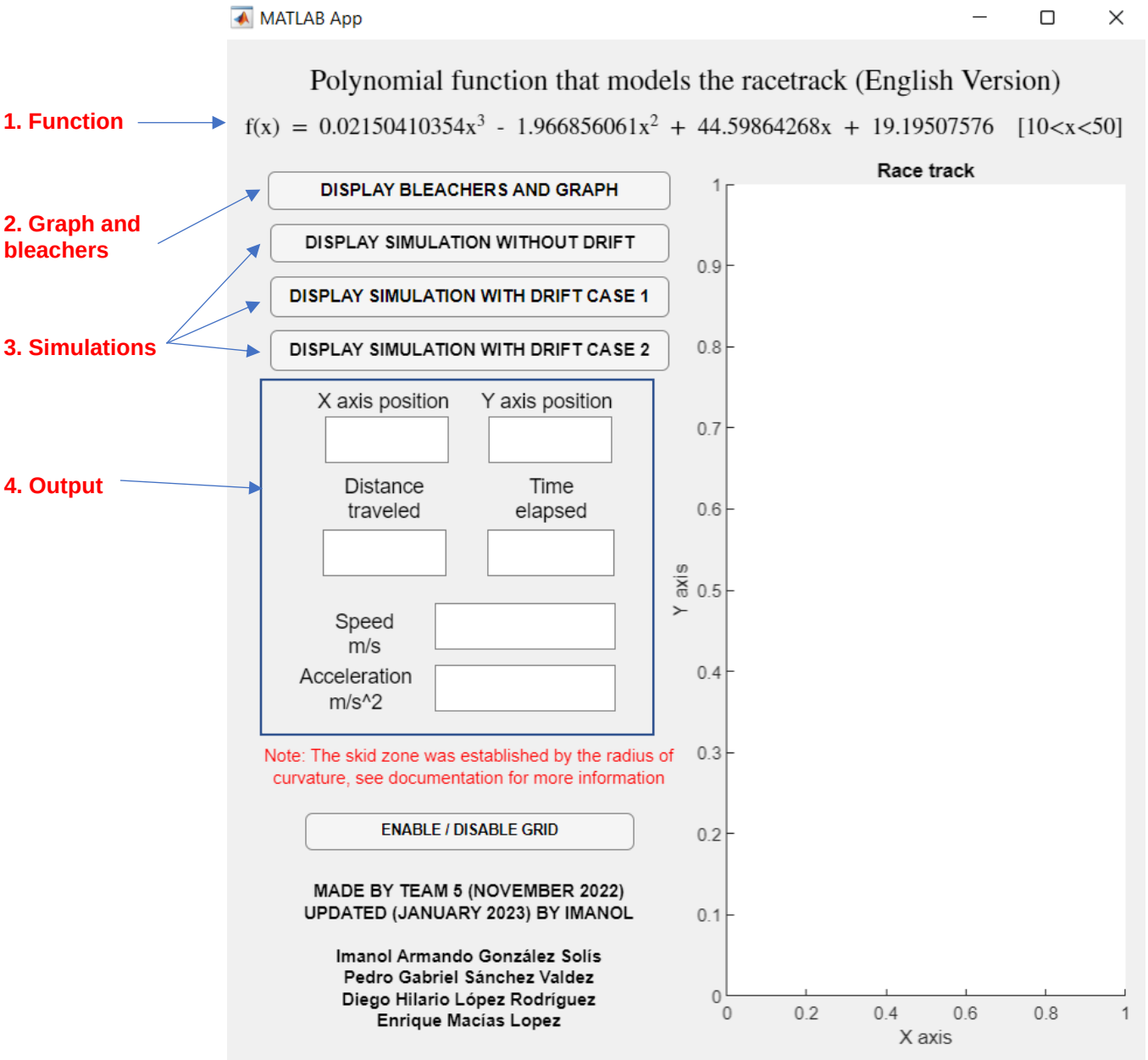
RACETRACK SIMULATOR DOCUMENTATION

By Imanol González – January 20th 2023

The racetrack simulator was a project part of the subject F1005B Computational Modeling Applying Conservation Laws and it was made by the following people:

- Imanol Armando González Solís (me)
- Pedro Gabriel Sánchez Valdez
- Diego Hilario López Rodriguez
- Enrique Macias López

We developed an UI using MATLAB App Designer with the following components



1. FUNCTION

To generate the function that models the race track that can be seen in the program, we were given certain conditions that had to be met which are the following ones:

- The curve must start at the point (10,290) and must end at the point (50,20).
- The length of the curve had to be greater than 300 and lesser than 500.
- The polynomial created had to be of third e in order to have two zones of curvature.

In order to meet the above conditions, we set ourselves the task of creating a program in MATLAB that would help us generate random coefficients, for that, we use equation systems in matrix form to obtain curves that will pass through certain points:

- The point (10,290)
- The point (50,20)
- A random point between x = 31 and x = 49
- A random point between x = 11 and x = 30

```
ax1 = randi(20)+10;
ay1 = randi(450)-50;
ax2 = randi(19)+30;
ay2 = randi(450)-50;
```

After using that code to generate two random points, we use the following algorithm to solve the system equation for that points which return an array of four numbers which are the coefficients of the polynomial function.

```
% Matriz (x) to solve
EqSystem = [
    1000 100 10 1; % Point 1 (10,290)
    125000 2500 50 1; % Point 2 (50, 20)
    ax1^3 ax1^2 ax1 1;% Random point 3
    ax2^3 ax2^2 ax2 1]; % Random point 4

% Matriz (y) to solve
y = [290 ; 20; ay1 ; ay2];

x = inv(EqSystem)*y; coef = x; % This line solve de matrix
```

After having created the random polynomial, we had to calculate the length condition using the following formula:

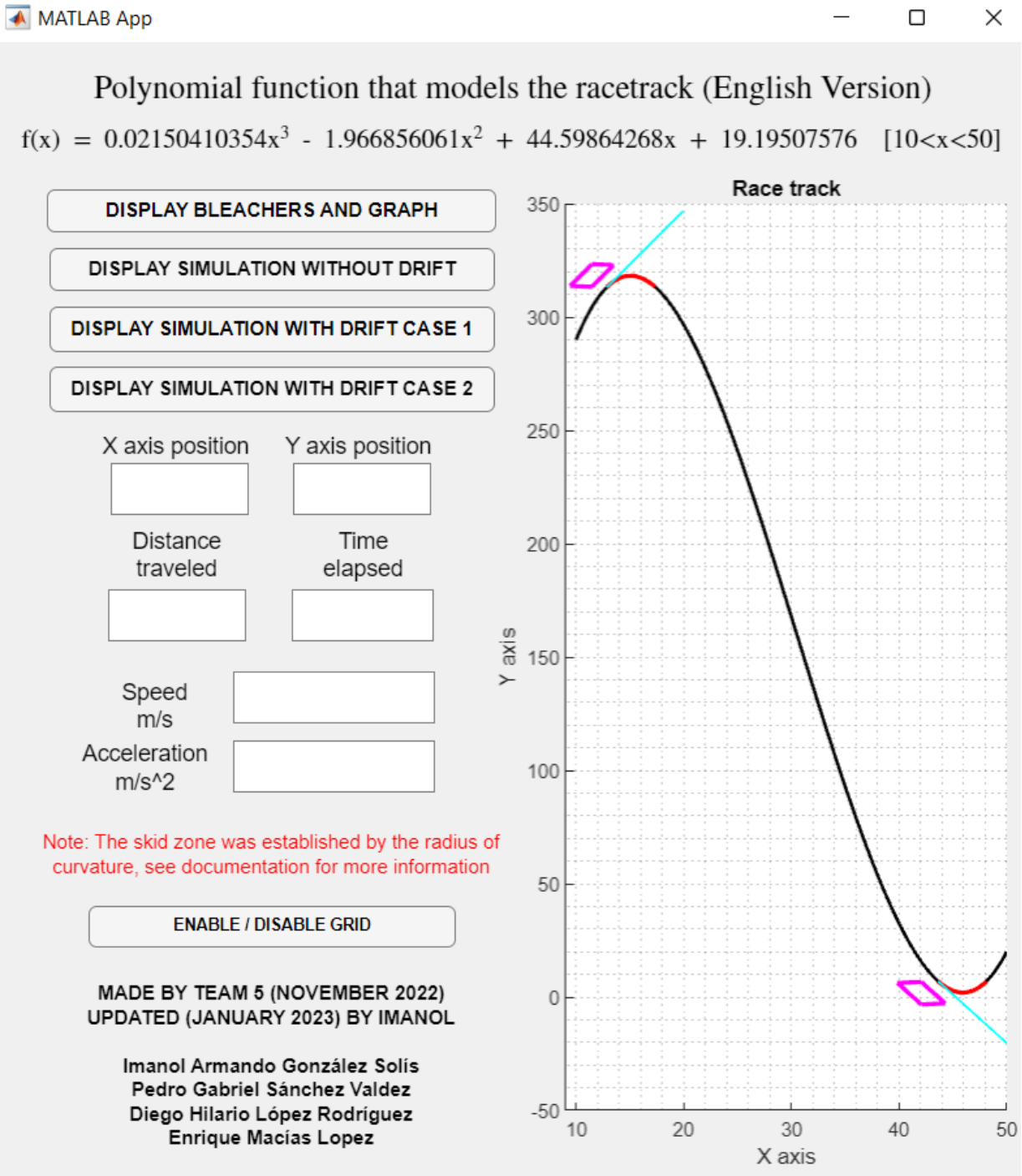
$$S = \int_a^b \sqrt{1 + [f'(x)]^2}\, dx$$

After calculating the length of the curve, it enters a loop where if the condition is not met, the program starts again and generates other random points, this is done several times until the program finds the points that create the polynomial that meets the initial condition.

The result of having done that brought us the following function that is the one we used for the program:

$$f(x) = 0.02150410354x^3 - 1.966856061x^2 + 44.59864268x + 19.19507576$$

2. GRAPH AND BLEACHERS



Once we had the polynomial graph, our task was to find a safe zone for the bleachers, for that we used the formula for the radius of curvature to find from where this is less than or equal to 50 because in that area there is the possibility of skidding.

$$50 = \frac{(1 + f'(x)^2)^{\frac{3}{2}}}{f''(x)}$$

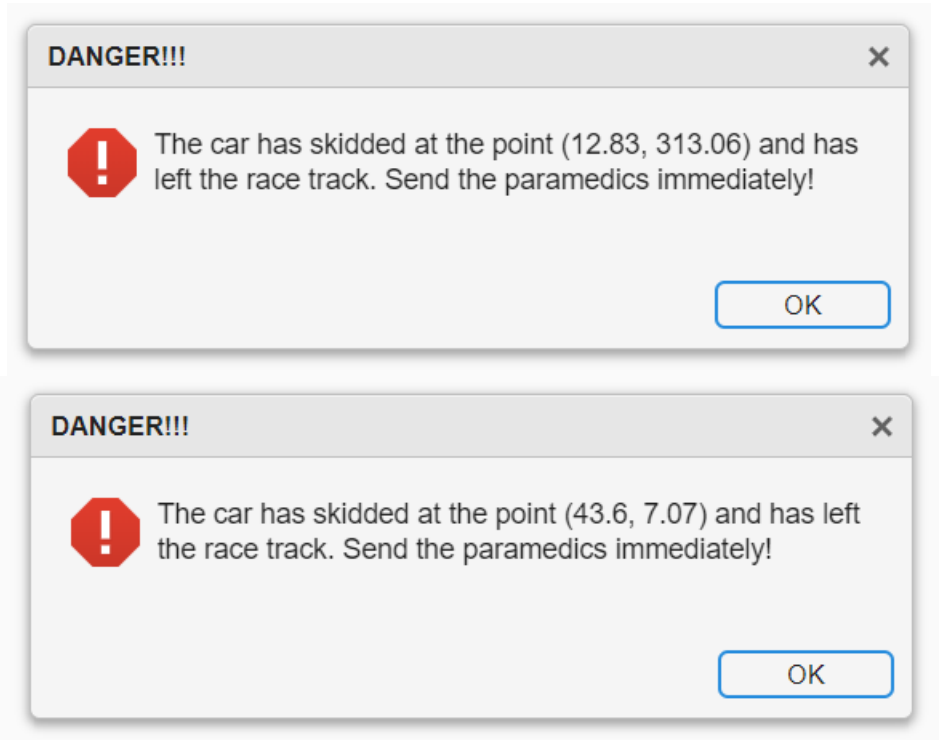
Solving that differential equation, we realized that the range in which the car can skid is as follows: [12.83 < x < 17.37] [43.6 < x < 48.14]. in that range is the skid zone that is represented in the graph as the red segment.

Later what we did was to draw a tangent line to the point where the car can start skidding, to see the trajectory that the car would take (Represented with cyan color). Based on that, we placed the stands parallel to these trajectories to make sure that the car would not collide with them (Represented with magenta color).

3. SIMULATIONS

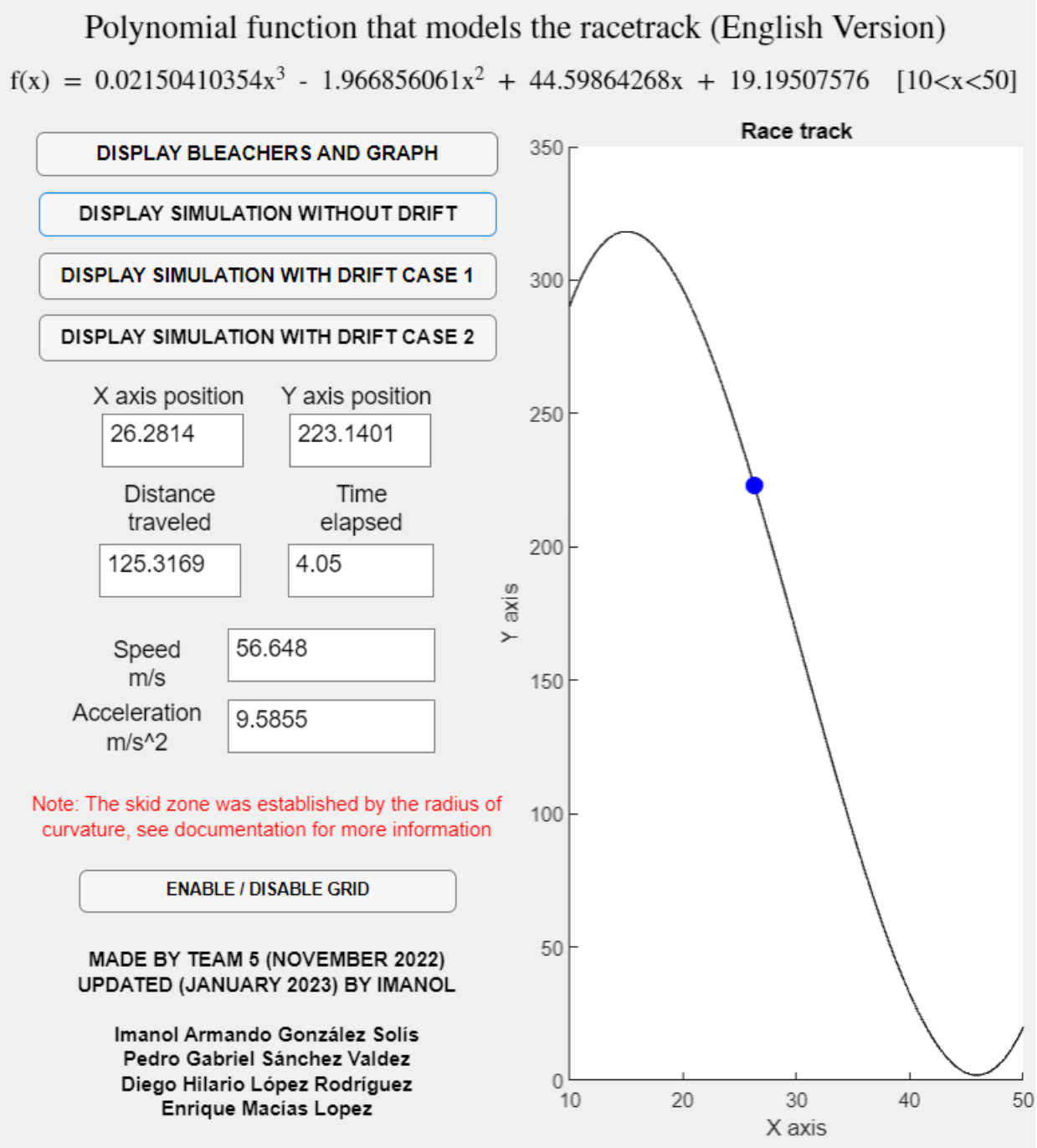
For the simulations, we animate a graph using MATLAB's pause function which makes the program to wait a certain amount of time between each frame.

In the case of skidding simulations, the program shows the trajectory that the car would follow if it were to skid in the range where the curvature is less than 50, at the end it shows an alert message on the screen



4. OUTPUT

In the case of the output parameters, the position and elapsed time do not need much explanation, but in the case of the distance traveled we use the formula for the length of the curve in the range traveled to be able to calculate it exactly.



To calculate the speed in real time, we use the distance change in the previous frame and the time interval between each frame. To calculate the acceleration we did the same process only instead of using the change of distance, we used the change of velocity.

Find the complete project at
https://github.com/imanolgzz/Project-Portfolio/tree/main/MATLAB/racetrackSimulator