Project 3: Ludobots

James Gaskell, Jonathan Fischman

I. Methods

In this project, we followed Josh Bongard's Ludobots tutorial on Reddit, through Part K: Random Search. We began by installing various Python libraries, including pybullet, in Part A. In Part B, we set up the pybullet simulation and created an empty, 3D world. In Part C, we learned how to create a 3D object and have it appear in the world. This cube we created will act as one body segment (referred to as a link) of the robot. In Part D, we added gravity to our pybullet simulation. We also added a plane, to act as a floor and stop the cube from from continuing to fall. We also created more cubes, and learned how to manipulate the position in 3D space they would generate in. In Part E, we added joints to our Ludobot, creating connections between adjacent links. In Part F, we added touch sensors to the Ludobot's legs, allowing us to determine if the leg is touching the floor at a given moment. In Part G, we added motors to the joints of the Ludobot, finally allowing it to move. The Ludobot's movement was based on a sine function. In Part H, we refactored our code, following object-oriented programming practices. In Part I, we added neurons, allowing our robot to "think" and interpret the touch sensor data, using a neural network. In Part J, we added synapses, finally connecting the neurons to the motors and allowing the Ludobot to traverse the plane on its own. Finally, in Part K, we implemented a random search program, allowing us to generate and simulate robots with different sets of synaptic weights, resulting in differing behaviors.

II. Results

The discussion of results is separated into two sections. The first section discusses the Ludobot's behavior when it is controlled by a sine wave. The second section describes the Ludobot's behavior when it is controlled by the neural network.

A. Results after Part H

After completing the refactoring in Part H, we ran the simulation, collected data about the Ludobot's motor target angles and sensor values, and graphed the results. At this point, the Ludobot's movement was being controlled by a sine function. This can be seen in Figure 1, where the motor target angle values for both the front and back legs follow sinusoidal curves. It is notable that the sine wave for the back legs does not align perfectly with the sine wave for the front legs. It seems that this is because the Ludobot's front leg begins to move before the back leg.

In Figure 2, we see the values of the touch sensors on both the front and back legs, which seem to differ greatly. This indicates that the front and back legs were usually not touching the ground at the same time while the Ludobot was walking.

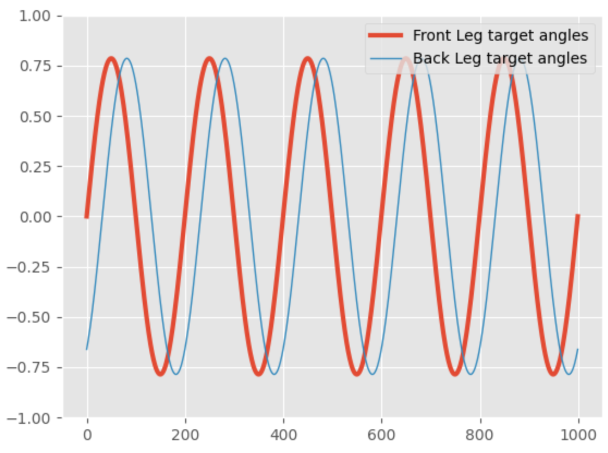

Fig. 1: Motor Target Angles for Sinusoidal Motion

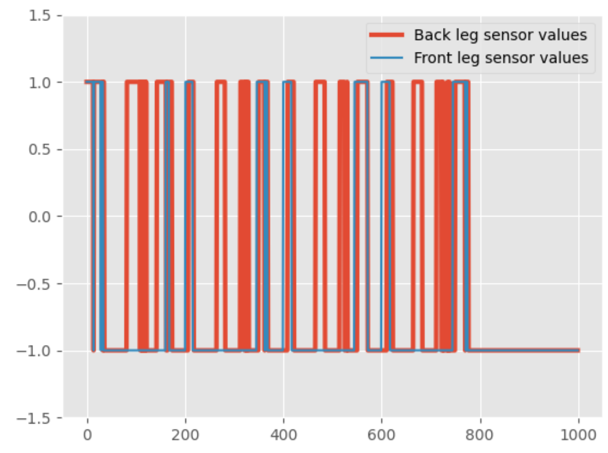

Fig. 2: Sensor Values for Sinusoidal Motion

B. Results after Part K

After completing Part K, we ran the simulation, collected data, and graphed the results. The Ludobot's movement was no longer controlled by the sine function. Instead, the movement was controlled entirely by the robot's neurons and synapses. This resulted in very different behavior compared to the trial in Part H. The resulting graph does not resemble the sine wave in Figure 1. In Figure 3, the motor target angle values match almost perfectly for the front and back legs. This indicates that the neurons and synapses controlling were giving the same instructions to both legs.

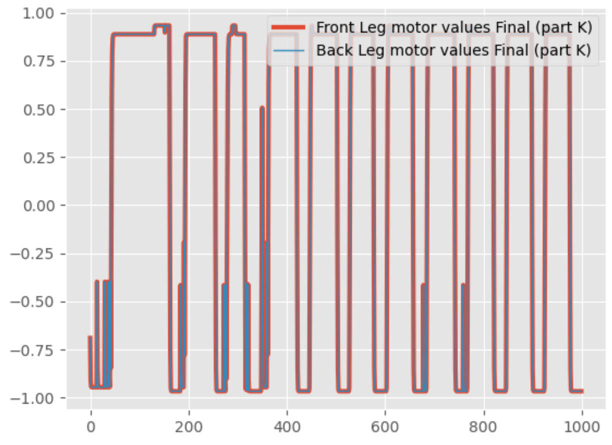

Fig. 3: Motor Target Angles for Part K

The data graphed in Figure 4 shows that the touch sensor values for the front and back legs were the. This indicates that the legs were moving in unison, as they both touched the ground at the same time, and stopped touching the ground simultaneously.

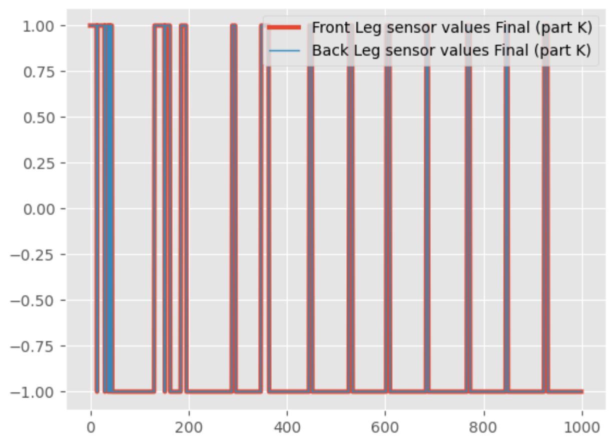

Fig. 4: Sensor Values for Part K