222 MATLAB Introduction

The idea with this is to have one main script that runs a series of functions, ideally a function or two for each feature of the track. Each function will output a matrix with positions of the ball (x, y) and other variables (angular acceleration, velocities, etc.) in the columns beside position.

These functions will step through a range of theta, for example, and get approximations of the velocity and other parameters at each value of theta and xy position.

These functions will be called in sequential order from the main script, getting the values we need at various points in each feature. The functions’ output matrices will be added to a big matrix at each stage.

Make sure to keep the columns synced! But feel free to add or re-organize columns wherever.

I’ve been writing the functions as scripts, and then turning them into functions when they are complete, which is why there are a bunch of scripts hanging around. The Obsolete folder is also full of stuff I was experimenting with. I’ve also been putting an “f” before function files to easily distinguish them from scripts.

What’s currently implemented?

Main222 Script initializes the globals and runs the curve acceleration/position function a couple times, and puts their outputs into a big matrix.

SystemInit() sets up a series of global variables like moment of inertia, mass, and number of iterations to do on simulations. Run this before running anything else, or the proper globals won’t be instantiated.

Currently, fLoopXYPos() is a function designed to handle curves. You pass it initial velocity, angular velocity, as well as radius of the curve, and position of the curve’s centre (as if it were a circle).

It steps through a theta range given and gives out a matrix with acceleration due to gravity, time (doesn’t work) and omega (also doesn’t work).

I’m not sure why the omega function gives wonky values but I’ll fix it when I have a few hours to work it out. Time isn’t explicitly required in the project, but if the omega function works, time should as well {(dtheta/dt) = omega function}, so flip omega function, multiply by dtheta, integrate with respect to theta, equals the integral of dt = time. I did have a test version of a time function working, so I’ll keep at it. Here’s a thing with similar math laid out, it’s worth a look: https://ca.answers.yahoo.com/question/index?qid=20101229213823AAjuJJS. Here is another good resource: http://www.myphysicslab.com/RollerSimple.html

initSpring() is something I wrote a while ago, not sure how well it matches the current setup.

What needs to get done?

The main thing is probably the forces - that’s one of the main requirements of the project. So maybe take the base of the floopXYPos and put the appropriate math to calculate the forces in there. The position and theta values are reliable, the force calculation math is what it would need.

Also, functions for the other features need to be done. They should calculate position of the ball, velocities and accelerations.

Richard will get the velocities and accelerations on curves figured out by Tuesday night unless someone wants/needs to do it earlier.