Directions for Drag & Drop Profile Generator

This software was designed to be as user friendly as possible by using on-screen buttons and popups to allow for the creation of custom, complex, perturbation profiles.

This was built in MATLAB R2021a. Newer version will work, older versions may or may not.

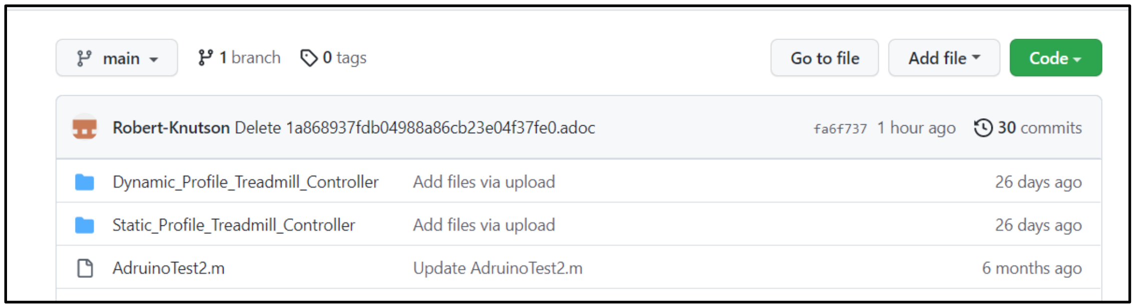
Quick Start Guide:

1. Run the MATLAB program and enter the number of points and total perturbation time in the first two popups.
2. Drag the points on the plot wherever you want.
3. Click the calculate button to create verification plots and a CSV file that contains motor speed and timing values.
4. Paste these values into the Arduino code and you're done!

Instillation:

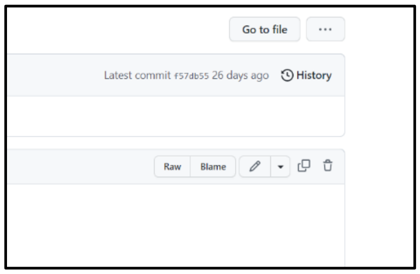
Download the source code for the entire project by clicking on the green code button in the main project folder. This will download all files used in this project.

Alternatively, go to a specific file you are interested in and click the copy raw contents button which can then be pasted into a blank MATLAB sketch.



Robert-Knutson Delete 1a868937fdb04988a86cb23e04f37fe0.adoc fa6f737 1 hour ago 30 commits

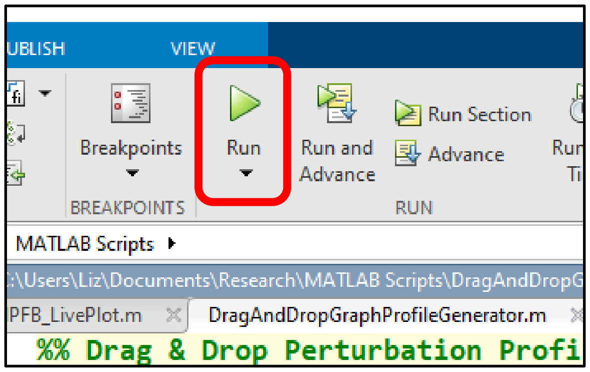
Dynamic_Profile_Treadmill_Controller	Add files via upload	26 days ago
Static_Profile_Treadmill_Controller	Add files via upload	26 days ago
AdruinoTest2.m	Update AdruinoTest2.m	6 months ago



Latest commit f57db55 26 days ago

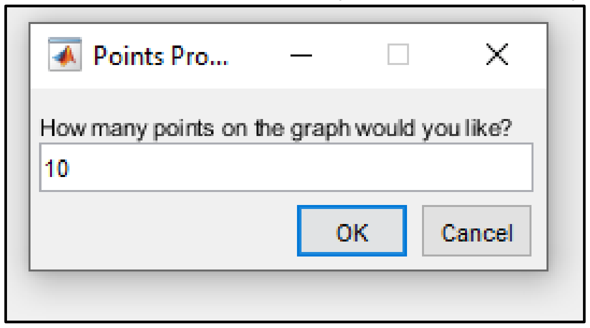
Use:

1. Open the DragAndDropProfileGenerator.m file in MATLAB if you decided to download the entire directory.
2. Click the run button to start the program. If you don't see it click somewhere in the code and it should appear.



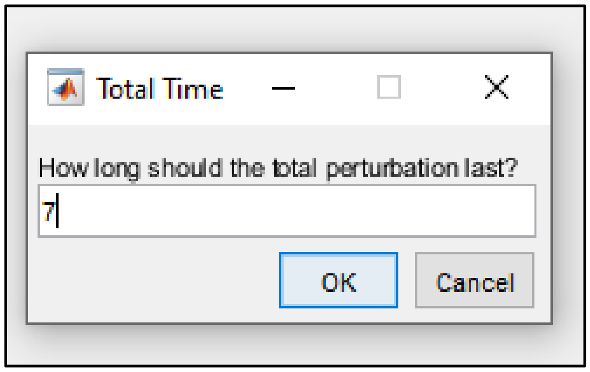
```
%% Drag & Drop Perturbation Profi
```

3. The first popup will ask you to enter the number of points you need. Each point is a moveable item, the minimum recommended is three, but you can use as many as you want.



How many points on the graph would you like?
10

4. The second popup is going to ask you how long the total perturbation should last in seconds. The is no minimum or maximum, so choose whatever you like, but be as close to your desired total time as possible as it makes the following graphs easier to read.



How long should the total perturbation last?
7

5. A graph will be displayed based on the number of points you entered and the total duration. On this graph each point, indicated by the blue circle, is a draggable item that can be moved anywhere desired. The limits on the y-axis are based on the maximum belt speeds for the motor input voltage (220 volt in the images below).

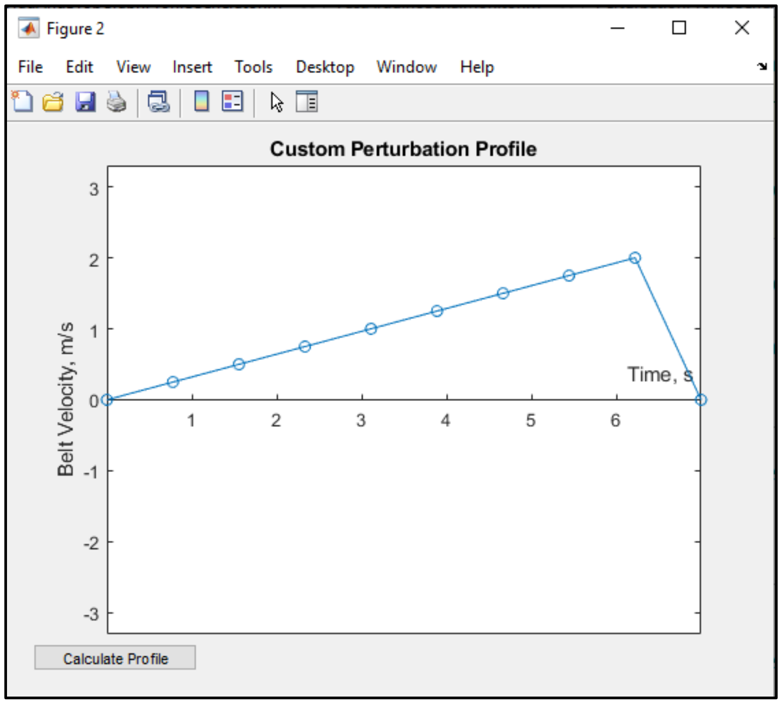


Figure 2
File Edit View Insert Tools Desktop Window Help
Custom Perturbation Profile
Belt Velocity, m/s
Time, s
Calculate Profile

6. Once you have dragged the points to wherever you want them, click the Calculate Profile button in the lower left corner of the screen. Clicking this button will result in the function you generated being converted into speed and timing values that can be used to control the motors.

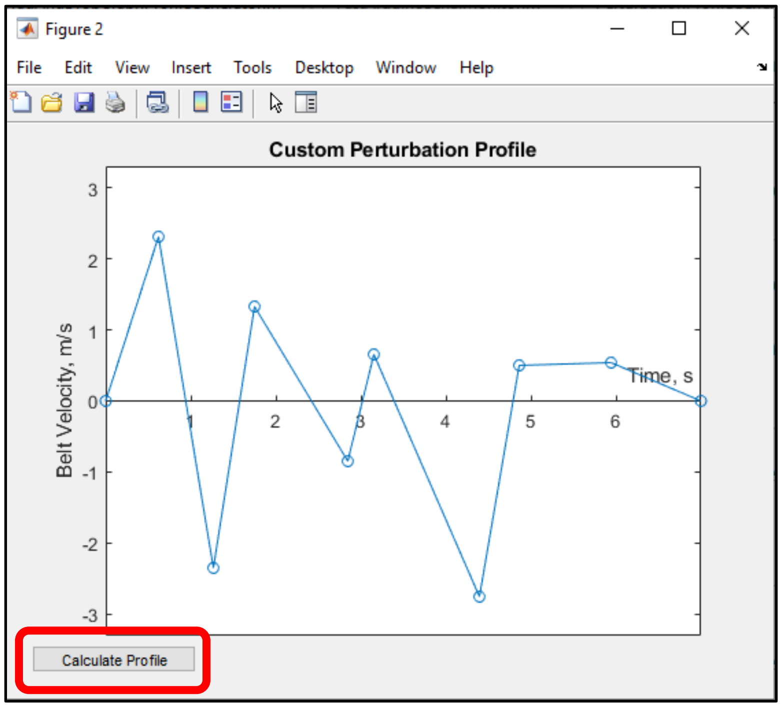
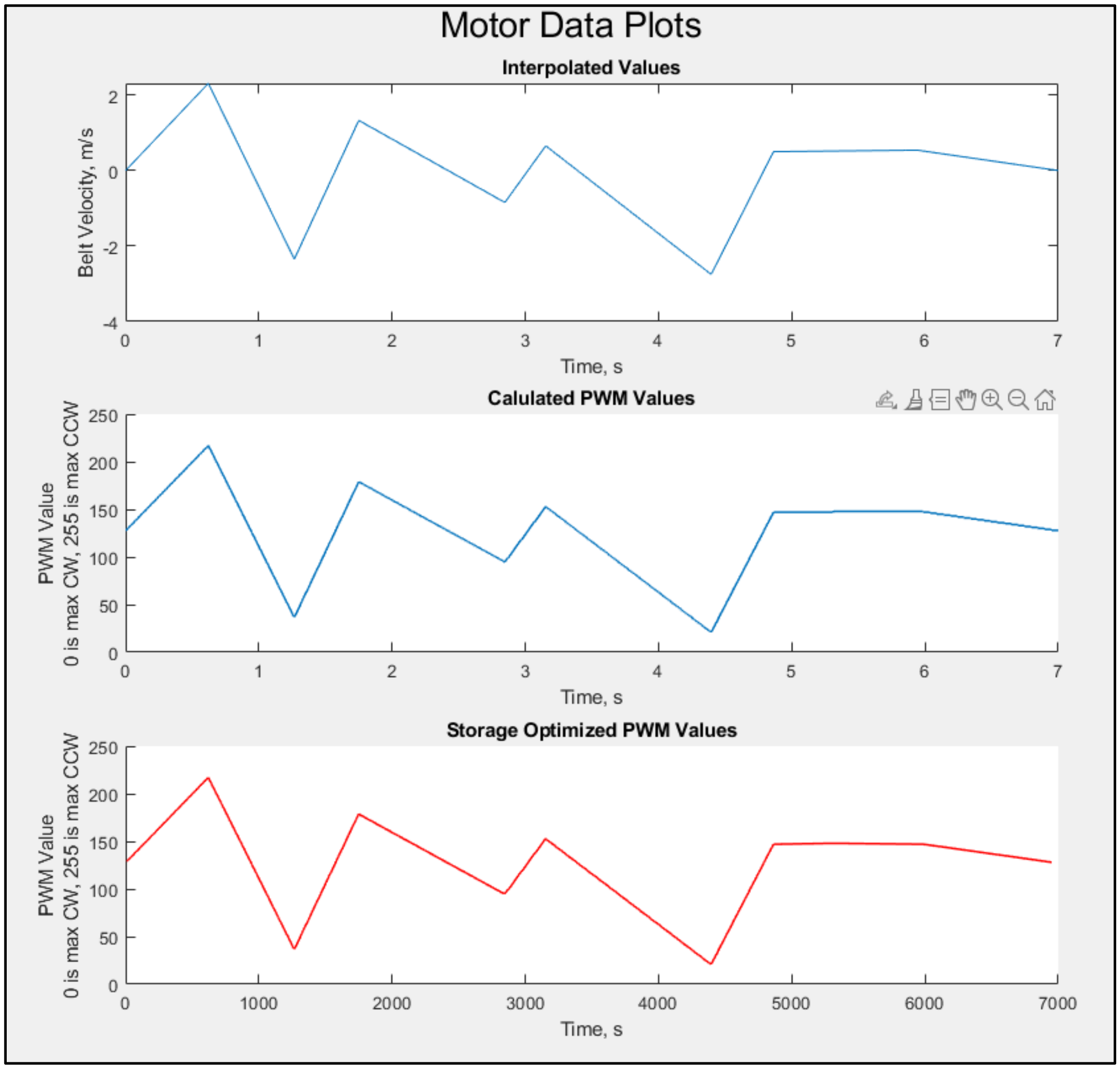
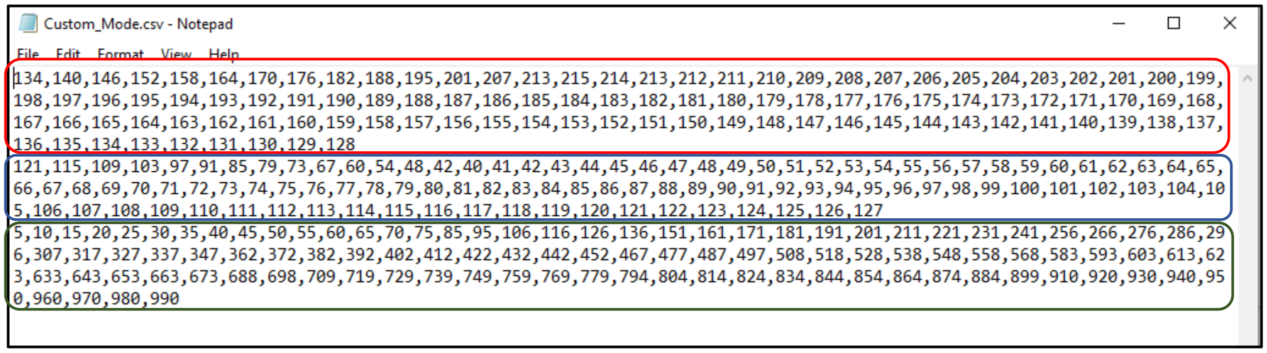


Figure 2
File Edit View Insert Tools Desktop Window Help
Custom Perturbation Profile
Belt Velocity, m/s
Time, s
Calculate Profile

7. A figure containing three graphs will be generated when the calculate profile button is clicked.
    a. The top graph displays result of interpolating the function you created. This can be used to verify that the generated points match up with the original function.
    b. The second graph is the result of calculating the PWM values that the treadmill software uses to control the motors. This can be used to make sure the resulting graph matches the interpolated result. Some data fidelity is lost in this step because each data point is converted to a number between 0 and 255.
    c. The third graph displays the optimized PWM array. This should match the calculated PWM values in the graph above it, but with duplicate points removed. This step significantly reduces the amount of storage the profile will use on the Arduino, with no loss in data fidelity.
    d. It is important to note that the original drag and drop profile can still be modified, and the graphs will update when the calculated profile button is clicked.

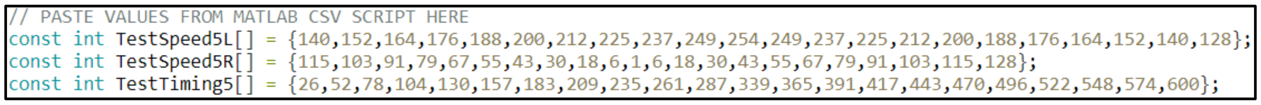


8. The final step is to copy the resulting speed and timing values into the Arduino code.
    a. Clicking the calculate profile button also resulted a file named Custom_Mode.csv being created in the folder the MATLAB script is currently running in.
    b. Open this file with Notepad or similar, but do not use Excel. Using Excel will result in all the data being put into rows and columns and make it much more difficult to paste into the Arduino code.
    c. Copy the PWM and timing values into the Static Profile Arduino software, using the following format. Different sections can be identified by the large space following the last entry.
        i. The first section contains the speeds for the left motor. This should be copied and pasted into the TestSpeed_L variable, where "_" denotes the operating mode you want to save this in, it can be mode 1-5.
        ii. The second section contains speeds for the right motor. Paste these values into the TestSpeed_R variable.
        iii. The third section is the timing array, paste these values into the TestTiming_ variable.



```
134,140,146,152,158,164,170,176,182,188,195,201,207,213,215,214,213,212,211,210,209,208,207,206,205,204,203,202,201,200,199,
198,197,196,195,194,193,192,191,190,189,188,187,186,185,184,183,182,181,180,179,178,177,176,175,174,173,172,171,170,169,168,
167,166,165,164,163,162,161,160,159,158,157,156,155,154,153,152,151,150,149,148,147,146,145,144,143,142,141,140,139,138,137,
136,135,134,133,132,131,130,129,128
121,115,109,103,97,91,85,79,73,67,60,54,48,42,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,
66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100,101,102,103,104,10
5,106,107,108,109,110,111,112,113,114,115,116,117,118,119,120,121,122,123,124,125,126,127
5,10,15,20,25,30,35,40,45,50,55,60,65,70,75,85,95,106,116,126,136,151,161,171,181,191,201,211,221,231,241,256,266,276,286,29
6,307,317,327,337,347,362,372,382,392,402,412,422,432,442,452,467,477,487,497,508,518,528,538,548,558,568,583,593,603,613,62
3,633,643,653,663,673,688,698,709,719,729,739,749,759,769,779,794,804,814,824,834,844,854,864,874,884,899,910,920,930,940,95
0,960,970,980,990
```

9. Once everything is pasted into the Arduino code, make sure the correct data structure is used. As in curly brackets should encapsulate the data with a semicolon at the end.



```
// PASTE VALUES FROM MATLAB CSV SCRIPT HERE
const int TestSpeed5L[] = {140,152,164,176,188,200,212,225,237,249,254,249,237,225,212,200,188,176,164,152,140,128};
const int TestSpeed5R[] = {115,103,91,79,67,55,43,30,18,6,1,6,18,30,43,55,67,79,91,103,115,128};
const int TestTiming5[] = {26,52,78,104,130,157,183,209,235,261,287,339,365,391,417,443,470,496,522,548,574,600};
```

10. Upload the code to the board and that's it. Congratulations! Great Job!

Variables You Can Modify in the Code:

There are only two variables you can change in the code; they are the data points per second (DPS) and the motor voltage.

- DPS controls how many data points per seconds are created as a result of interpolating the function you generated. More data points means a closer approximation but significantly impacts the storage required on the Arduino. 40 DPS is a good starting point, that equates to 0.025 seconds per point, which is imperceptible to most people.
- Voltage dictates how to scale the PWM values to the minimum and maximum motor speeds. This value should match whatever the motor is going to be connected to and can be either 110 or 220.