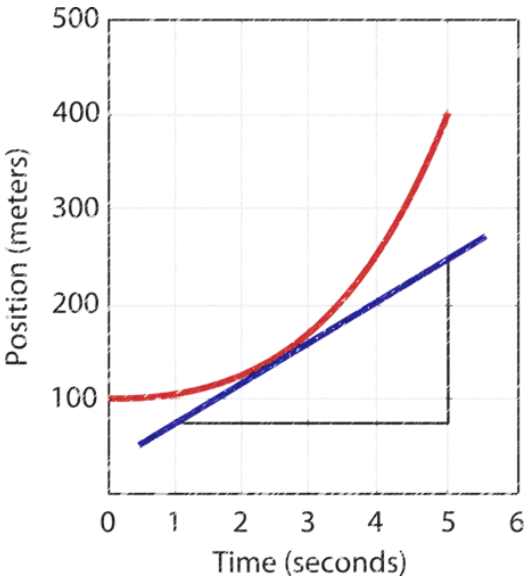
MATLAB Final Project

Engineering Problem #6: Robot Movement and Control

Understanding the Problem

The goal of the project is to reconstruct the trajectory of a robotic vehicle, including deriving the position, velocity, and acceleration of the object at any time during its travel. In a table, I am given the x and y positions of the robot with respect to time elapsed. n a separate scenario, the problem asks to derive the velocity and acceleration values of a robot travelling a 1-minute long circular path of a diameter of 2.4 m. I am aware of the mathematical relationships between position, velocity, and acceleration. Acceleration is the derivative of velocity and velocity is the derivative of position.



Instantaneous velocity can be computed by taking the slope at the point (change in position over change in time) as seen in the picture on the left.

Acceleration, velocity, and position can also be split into individual components in the x and y direction, which can be turned back into original form by using Pythagorean theorem. Velocity is also the vector equivalent of speed, which is equal to the magnitude of the change of position divided by the change in time.

Devising a Plan

#1: Recreating a Piece-Wise Linear Approximation

1. Generate a 3D plot comparing the time (z-axis), x position (x-axis) and the y position (y-axis).
2. Using the equations provided in (a), generate two 2D plots, for an approximation of speed and an approximation of acceleration

#2: Polynomial Curve Fitting

1. Split the 3D plot generated previously into two separate 2D plot functions: (1) time against x position and (2) time against y position.
2. For each 2D plot, using the polyfit and polyval matlab functions, plot an approximated curve of the data points. Use various polynomial degrees (i.e. quadratic, cubic, quartic, etc.) until an appropriate one is found (use a for loop to loop through several degrees). In the end, create a

new graph with the best plot found (best polynomial degree) against time in very small increments (use linspace)

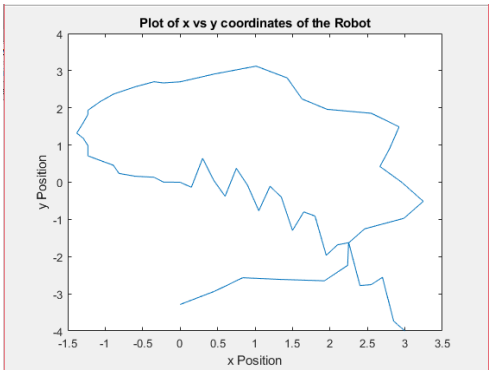
3. To find a curve-fitted plot for velocity, find the components of v_x and v_y, and squareroot the sum of the squares (according to Pythagorean Theorem)
    a. To find v_x, take the change in the x position (from the curve-fitted plot in step 2) divided by the change in time (length of the time increment)
    b. Repeat (a) for v_y: take the change in the y position (from the curve-fitted plot in step 2), divided by the change in time
4. To find a curve-fitted plot for acceleration, find the components of a_x and a_y, and squareroot the sum of the squares (according to Pythagorean Theorem). Use the same process in step 3 to get the components of a_x and a_y by taking the respective change in velocity divided by the time increment.

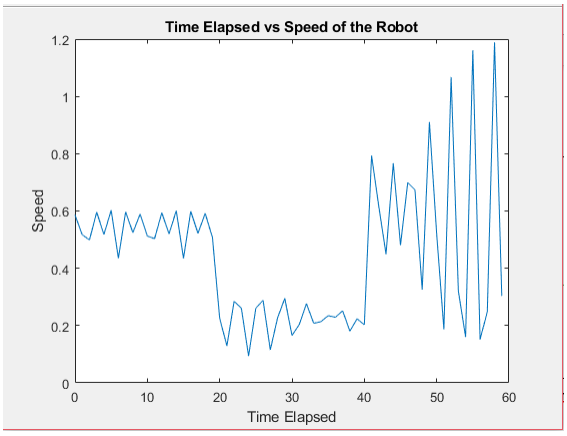
#3: Path Control

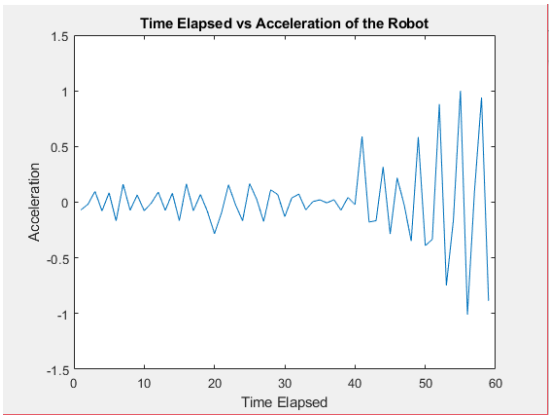
1. Create an equation of the robot's predicted path, relating the x position with the y position. This is a circle including point of the origin (0,0) and has a diameter of 2.4 m.
2. Generate a column of values for each of the x and y positions with respect to time. The revolution of the robot around the path should take 1 minute.
3. Repeat steps 1-4 from *#2: Polynomial Curve Fitting* for the new set of data points.
4. Account for the direction the robot is travelling for each data point as velocity and acceleration by calculating the angle of velocity and acceleration.

Piece-Wise Linear Approximations

The following are the graphs generated from the raw data for x vs y position, time vs magnitude of velocity, and time vs magnitude of acceleration.





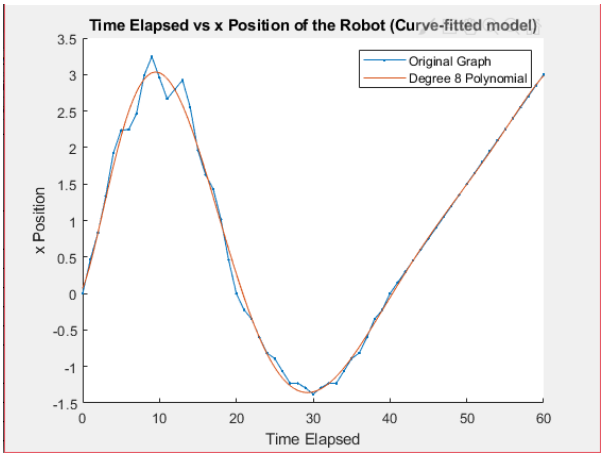


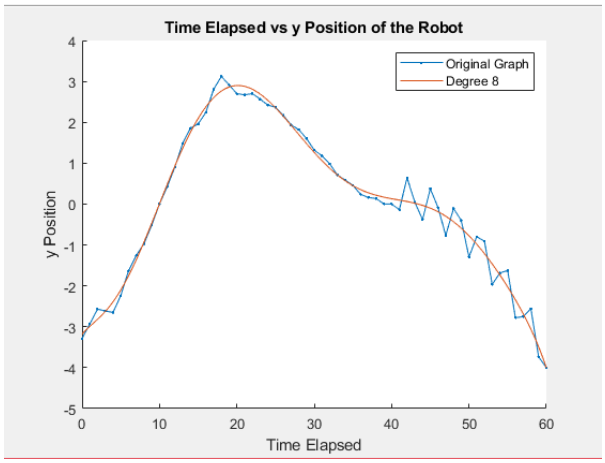
Evaluation of Curve-Fitted Models

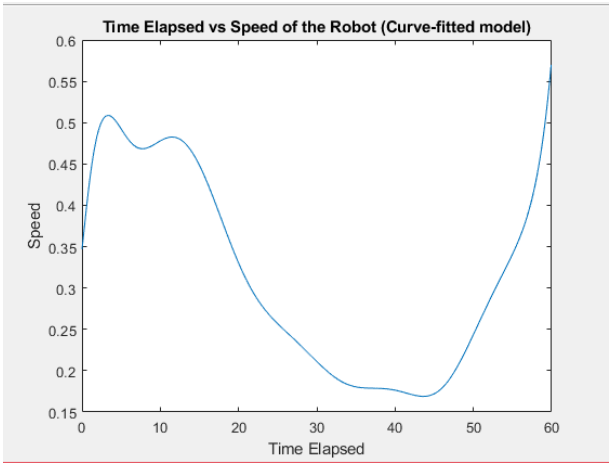
The approximated, curve-fitted graph for *Time Elapsed vs x Position of the Robot (Curve-fitted model)* and *Time Elapsed vs Y Position of the Robot (Curve-fitted model)* accurately represent the data at most times. The curve used is a polynomial of degree 8, which is a high order and appropriately models the raw data. It loses accuracy around areas of maximum or minimum critical points, when the robot's path was not smooth. In the plot *Time vs Y Position*, the model slightly deviates from the raw data in the interval of time = 40 to time = 60 because the robot's movements are more jagged.

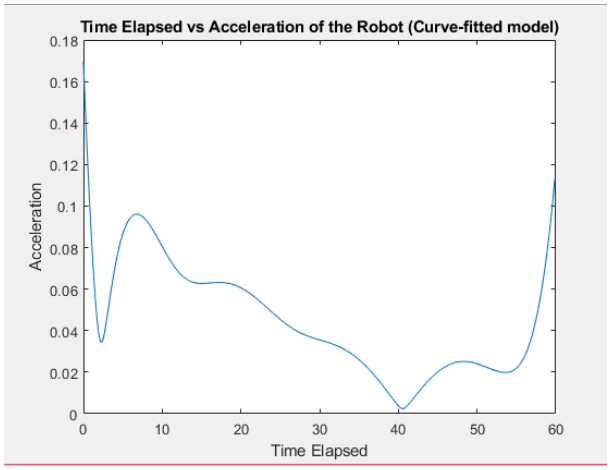
In the last two curve-fitted graphs *Time vs Speed* and *Time vs Acceleration*, the models deviate more significantly from the previous graphs generated because the original graphs have a very jagged-like shape.

Below are the curve-fitted models developed for *Time VS x position*, *Time VS y position, Time VS x speed,* and *Time VS acceleration*.



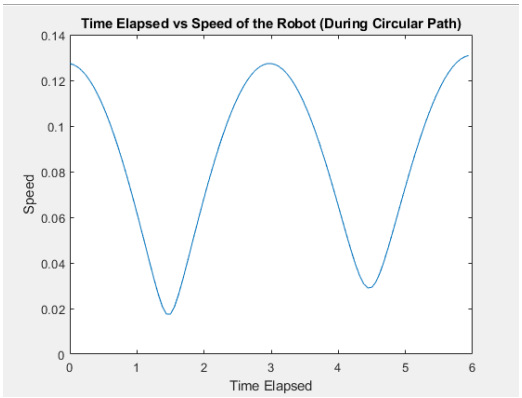


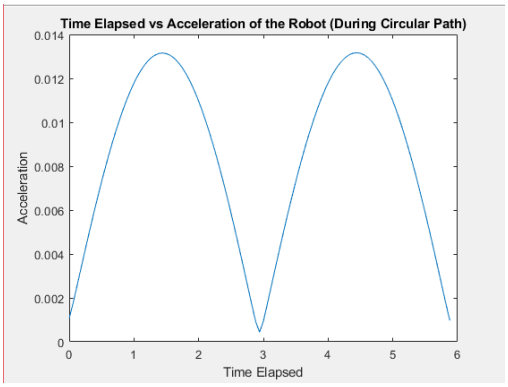




Path Control Velocity and Acceleration Values

The table of values for magnitude of velocity and acceleration, and additionally their respective angles of direction, is too long to be included in this document and has been attached in a separate file called *Table of Values for Circular Path*. Below are plots for the magnitude of the velocity and acceleration with respect to time during the circular path travel.





My plan for creating the MatLab script changed from Step 2. For question (c), I needed to determine a method to calculate a new set of values for the x and y positions of the robot during the circular journey. I also wanted to have a high number of data points so the values for velocity and acceleration would be accurate. Thus, I needed to use 2 complicated for loops that calculated 100 evenly-spaced points on the curve: $x^2 + (y - 1.2)^2 = 1.2^2$. To testify my technique, I used the command `>> plot(x_circle, y_circle)` to ensure the produced data points depicted the perfect circle of the path.

Looking Back

I know that I have adequately addressed the problem because I developed all the plots and produced enough information the question requires. My linear approximations plots demonstrate the position, speed, and acceleration values of the robot from its raw data. This is depicted in the curve as there are many jagged and sharp regions of the graphs. My curve-fitted models produce better representations of the 3 quantities above as they are still accurate compared to the original graphs yet smoothen the regions. This enables us to possibly interpolate or extrapolate the data. My final two graphs consist of the magnitude of the velocity and acceleration of the robot's journey. As the robot is travelling in a circle, it should have a velocity and acceleration graph that is symmetric, as seen in the plots. My original goal also includes producing plots with time increments small enough that the data can be derived from almost any point. This need is met as my curve-fitted plots use a time increment of only 0.06 seconds, whereas my plots in part (c) used a time increment of 0.6 seconds, both of which are very small. Therefore, the plots containing sufficient information of the trajectory of the robot and the table of exact values of velocities and acceleration will help the robotics team send signals to the motors.

I learned about how to convert from component form to polar form to represent velocities and accelerations. I also learned about how to use MatLab to create a set of evenly spaced points on the circumference of a circle using the tan function.

In conclusion, to determine the velocity and acceleration values of the robots circular journey of a diameter of 2.4 m starting from the origin (0,0), one should look at the two previous plots generated or the table of exact values provided.