Method Comparison

Thomas Berry

28 March 2020

# The System

The goal is to plan a trajectory for a 1-D vehicle defined by the following dynamic system

with initial and final conditions are given by

The obtained trajectory must also minimize the following running cost:

Matlab’s fmincon function will be used to solve the optimisation problem.

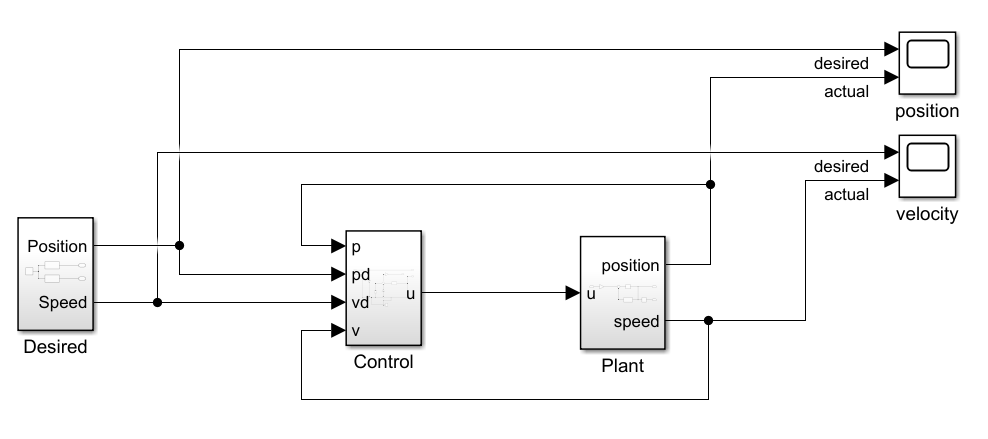
In both proposed methods, the goal is to find a set of control points for a Bernstein Polynomial that will define the trajectory.

# Trajectory Tracking

The variable to optimise contains only the control points for the position.

The initial conditions of speed and position and final condition of position was implemented as a linear constraint.

The dynamics don’t impose constraints on the optimisation process because this method sort of works like shooting: the control points define the desired position over time which is the reference of the controlled system as shown in the following simulink model.



What the cost function does is simulate the all of the state variables for the given desired input given by the control points, and squares and integrates the necessary states according to the running cost function. This process can basically be translated into solving the ordinary differential equation given by the following functions:

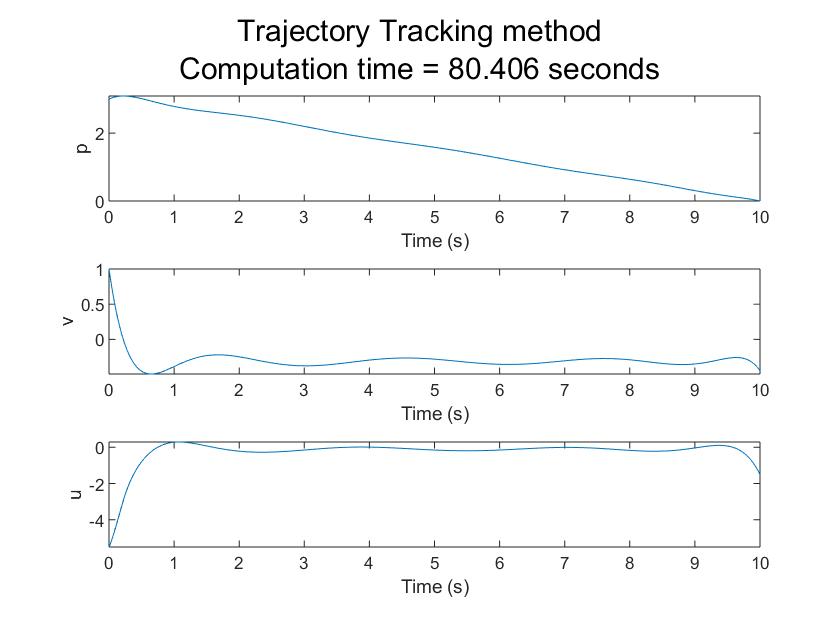
Where and are constants that belong to the trajectory tracking controller.

, and are the desired acceleration, speed and position of the vehicle, which, in this case, are given by the control points.

The value of the running cost will be the final value over time of .

The MATLAB function ode45 was used for this problem.

The result of the simulation is shown in the following figure



Note: the control points weren’t plotted because they compressed the vertical axis too much.

Why so slow?

For each optimisation step an ODE has to be solved. This is a real no no in terms of speed.

Obtaining the trajectory of each state variable via tracking is absolutely necessary otherwise it is impossible to perform calculations over them.

The ODE solver of choice, and the only one I know about (so far), cannot obtain parameterized trajectories, i.e., only some time samples. Therefore, the integral of the running cost has also to be done by the solver.

# Fully defined States

The optimisation variable is given by control points for p, v and u.

The cost is calculated by using the properties of Bernstein polynomials, i.e., by a sum of the values of the control points.

I assume that linear constrains are faster and easier for fmincon to deal with, therefore, I always aim to use linear constraints as much as I can.

2 linear constraints can be defined here:

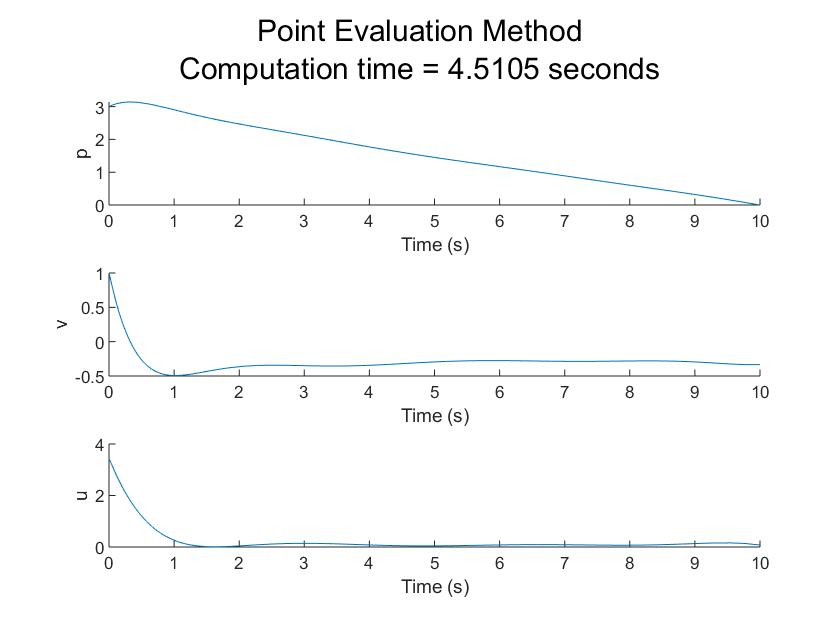
* Initial and final conditions (implemented like in the trajectory tracking method)

Where stands for control points.

The nonlinear dynamic constraint for the derivative of v cannot be imposed by a matrix multiplication. The steps to impose the constraint as the following:

1. Find the control points for the left side of the equation  , in our case, for
2. Perform degree elevation for all state variables, in our case, p, v and u
3. Evaluate the state variables and derivatives in times corresponding to the control points of the “elevated” polynomials
4. Perform the non linear calculations on the right side of the previous equation
5. Match the points in left and right sides by subtracting, taking the norm of the subtraction and equate it to zero.

First immediate thing to note is that the state variables will necessarily have to be polynomials because they are all defined by control points. These solutions will only be a subset of all possible solutions because an output of a nonlinear system whose input is a polynomial may not be a polynomial as well.



# Pros and Cons of each

Trajectory tracking is slow but allows solutions that may not be polynomials.

The other method has more variables, solution can only be polynomials but is fassssssst.