Tecnológico de Monterrey

Research stay #5

Final report

Trajectory generation and tracking for an Unmanned Underwater Vehicle

Victor Sebastian Martínez Pérez

A01232474

Advisor: Herman Castañeda

06/19/2022

Introduction

Path and trajectory planning are crucial problems in the broad field of robotics, as they provide references of motion for robotic systems. The first problem addresses the issue of generating paths, without considering temporal constraints, different from the latter problem, as it is concerned on path generation with time limitations. Path planning algorithms are usually divided according to the methodologies used to generate the desired path; some examples are (Gao et al., 2020):

- Graph-search based planners:
    - Dijkstra, A*, D*, AD*, state lattice algorithms
- Sampling based planners:
    - Rapidly-Exploring Random Tree (RRT)
    - RRT*, PRM, etc.
- Interpolating Curve planners
    - Bézier, Clothoid, polynomial, spline curves, etc.
- Cell decomposition algorithms
- Artificial potential methods

On the other hand, the trajectory planning problem is usually addressed by using (Gao et al., 2020):

- Curve planners
    - Clothoid, polynomial, spline curves, s-curves, etc.
- Optimization techniques

The trajectory planning problem is especially relevant, as for many applications it is desired to generate references based on some objectives, such as time optimality, energy efficiency, and jerk/snap minimization. The latter objective is concerned on generating smooth references that do not produce excessive wear of the actuators, or to ensure quality readings from sensors (Richter et al., 2016). The mentioned task can be solved by techniques that minimize the high order kinematic variables, such as the jerk and snap. A novel method that addresses the mentioned issue are S-curve based trajectory planning. This method utilizes a piecewise sigmoid function to establish a jerk profile such that the generated trajectories are infinitely continuously differentiable under the constraints on velocity, acceleration, and jerk. Time optimality is ensured, as the kinematic profiles are based on the kinematic constraints (Fang et al., 2019).

In this report, a solution for the problem of 2D trajectory generation is addressed in simulations using ROS, with the creation of a motion planning and control library for an Unmanned Underwater Vehicle. First, the path planning problem will be addressed. Next, the implementation of the trajectory generation will be discussed. Then, the design of a trajectory tracking controller shall be reviewed. Finally, results of the integration of the mentioned methods will be shown. It is worth noting that most of the effort of this stay project was focused on programming the libraries.

Development

1. Path planner

For the path planning problem, any algorithm capable of yielding a set of waypoints to follow is suitable to address the problem. The *Open Motion Planning Library* (OMPL) was chosen to develop the solution as it includes many state-of-the-art sampling-based motion planning algorithms, provides a high level of abstraction to make it easier to integrate it into larger robot software systems, and has a good documentation. From the available planning algorithms, the RRT* planner was chosen.

The code for the path planner can be found here:

https://github.com/vanttec/vanttec_motion_planners/tree/main/path_planners/sampling_based

2. Trajectory planner

The S-Curve trajectory planning method addressed in (Fang et al., 2019) is used to solve the trajectory generation problem.

- **Method description**

First, it is important to describe the sigmoid function, shown in equation (1) with distinctive properties. The first one is that has a definition in the domain $[-\infty, +\infty]$, with values growing from 0 to 1, and centering at 0.5. The second property is that is differentiable tractable, as the slope is approximately linear over the middle range of the interval around x=0.

$$f(x) = \frac{1}{1+e^{-x}} \qquad 1$$

The sigmoid function is adopted to constitute a continuous jerk profile that can be integrated three times to obtain the displacement trajectory. Using this method, analytic optimal solutions are derived without relying in iterative optimization processes. The jerk profile is described in equation (2).

$$j(t) = sign(D) \cdot \begin{cases} J_{\max} \dfrac{1}{1+e^{-a(1/(1-\tau_i)-1/\tau_i)}} & t_0 \le t < t_1, t_{12} \le t < t_{13} \\ J_{\max} & t_1 \le t < t_2, t_{13} \le t < t_{14} \\ J_{\max} \dfrac{1}{1+e^{a(1/(1-\tau_i)-1/\tau_i)}} & t_2 \le t < t_3, t_{14} \le t \le t_{15} \\ 0 & t_3 \le t < t_4, t_7 \le t < t_8, t_{11} \le t < t_{12} \\ -J_{\max} \dfrac{1}{1+e^{-a(1/(1-\tau_i)-1/\tau_i)}} & t_4 \le t < t_5, t_8 \le t < t_9 \\ -J_{\max} & t_5 \le t < t_6, t_9 \le t < t_{10} \\ -J_{\max} \dfrac{1}{1+e^{a(1/(1-\tau_i)-1/\tau_i)}} & t_6 \le t < t_7, t_{10} \le t < t_{11} \end{cases}$$

Where $t_i\ (i = 0,\ 1,\ \ldots,\ 15)$ stands for the time boundary at each of the 15 trajectory segments. $\tau_i = \frac{t - t_{i-1}}{t_i - t_{i-1}}$ represents the normalized time variable in the interval $[t_{i-1},\ t_i]$, and $a = \frac{\sqrt{3}}{2}$ is a positive constant parameter controlling the variation rate. This jerk profile is infinitely differentiable. Fig. 1 shows the kinematic profiles based on the S-curve model, and the different time segments can be appreciated. Since the derivatives are assumed to be restricted as absolute values, T1=T3=T5=T7 and T2=T6 should hold for the acceleration stage [t0,t7] to have a null acceleration at t7. Correspondingly, the deceleration stage [t8,t15] has similar relations T9=T11=T13=T15 and T10=T14. Meanwhile, the velocity profile is of symmetric type for point-to-point motions, thus the time intervals allocated for the acceleration and deceleration stages are equal. Therefore, the shape of the trajectory depends completely on the actual maximum jerk and four-time intervals: the varying jerk interval Ts, the constant jerk interval Tj, the constant acceleration interval Ta and the constant velocity interval Tv, during which the maximal snap, jerk, acceleration and velocity are achieved respectively (Fang et al., 2019). An important aspect to note is that this model considers that the kinematic differentials start and end at zero.

The problem then is to calculate the times Ts, Tj, Ta and Tv to obtain the desired displacement between the initial and final position within the shortest time under consideration of motion constraints.

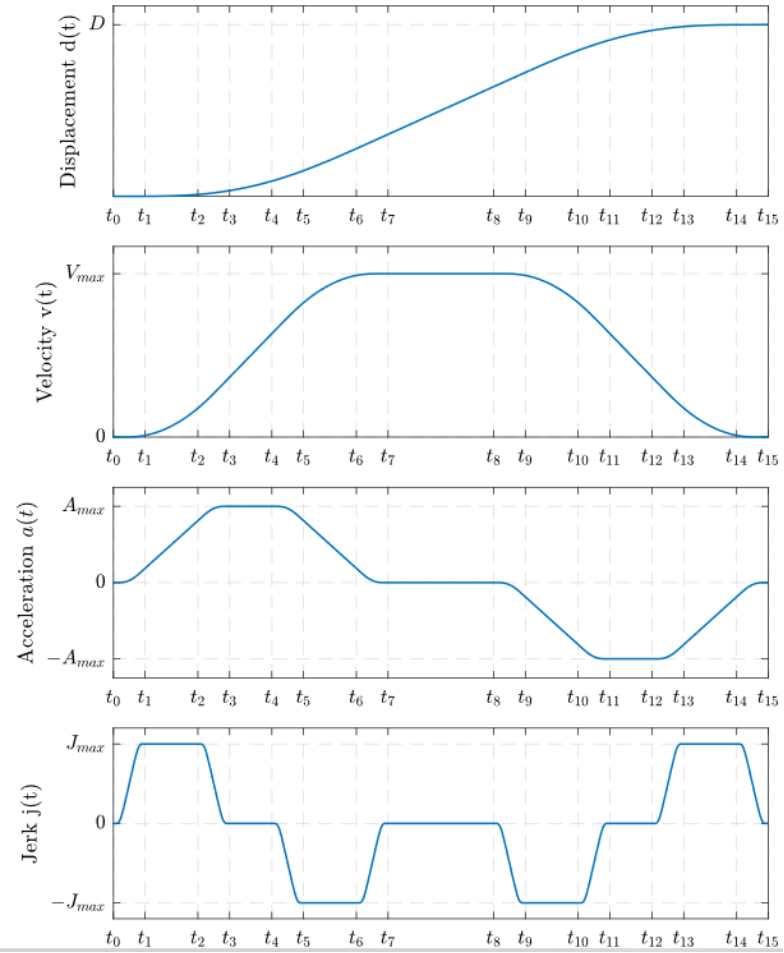

Fig. 1. Kinematic profiles of S-curve model

- **Calculation of time parameters**

To calculate the time periods Ts, Tj, Ta and Tv, the authors in (Fang et al., 2019) propose a method, shown in Fig. 2, to calculate each period, starting with Ts until Tv, to calculate their values based on defined maximum kinematic constraints Smax, Jmax, Amax and Vmax, for the snap, jerk, acceleration, and velocity, respectively. In summary, the algorithm calculates each time period depending on the number of kinematic constraints to be resolved for. For example, Ts is first calculated four times, one dependent on the displacement, other on Vmax, other dependent on Amax and a final one on Jmax. From those calculations, the minimum one is chosen as the final value for Ts. This is done to ensure respecting the constraint of the lower order kinematic, and that would end up respecting the constraints of the higher ones accordingly. Also, this ensures that a minimal number of calculations can be made. Similar to this, the rest of the time periods are calculated. Further details can be found on the paper.

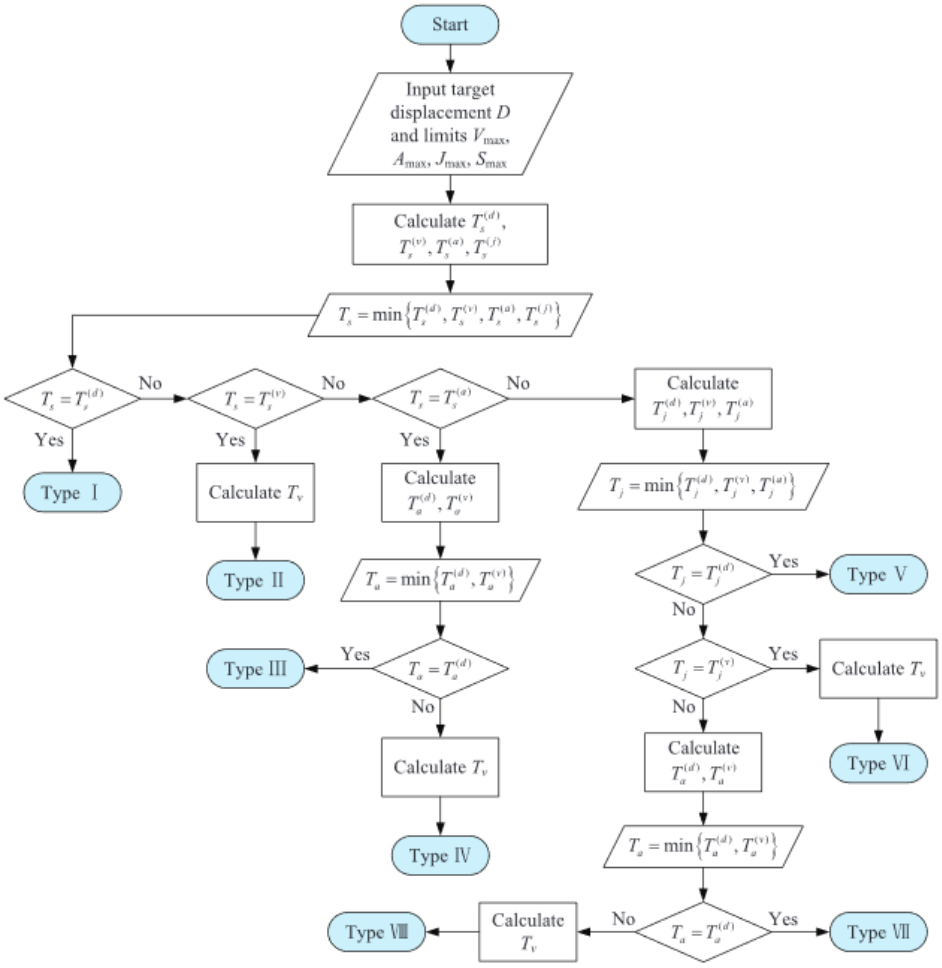

Fig. 2. Flowchart of the time parameters determination process

- **Time synchronization**

Synchronize times for the different degrees of freedom (DOF) is an important task, as without it, the overall desired motion may not be achieved. For example, in the case of following a single line, if only the time periods of each DOF are calculated and used as they are, it may happen than a DOF reaches the destination faster than the rest, leaving the final traversed path as a curve, instead than a line.

Three methods are proposed to synchronize the trajectory: time synchronized, minimum jerk, and velocity based. The first one relies on finding the maximum possible total execution time (sum of all time segments) among every individual DOF. Once this synchronization time is determined, the rest of the motion profiles should be adapted to this time, by dividing by a factor composed of the division of the synchronization time by the execution time of the individual axis, all to the power of the kinematic order. The second method relies on minimizing the peak value of the jeark for each individual DOF and is done through searching for the jerk limit that prolongs the duration of the trajectory

to the synchronization time in the range of [0, Jmax], while remaining the maximum snap and other kinematic constraints unchanged. The third method is concerned on finding the minimum kinematic limits per each motion amongst all DOFs; and the lower ones found per motion, are used so the same minimum set is shared among all DOFs.

- **Implementation**

Determination of kinematic constraints

First, the kinematic constraints of the VTec-U4 were determined based on the dynamic model, in Matlab. Motions focused on surge, sway and heave were analyzed, with the maximum thrust available at each motion [35 N], to define their maximum snap, jerk, acceleration, and velocity.

Fig. 3 demonstrates the speed profile of each DOF, the same was done for the rest. Table 1 shows all the kinematic limits.

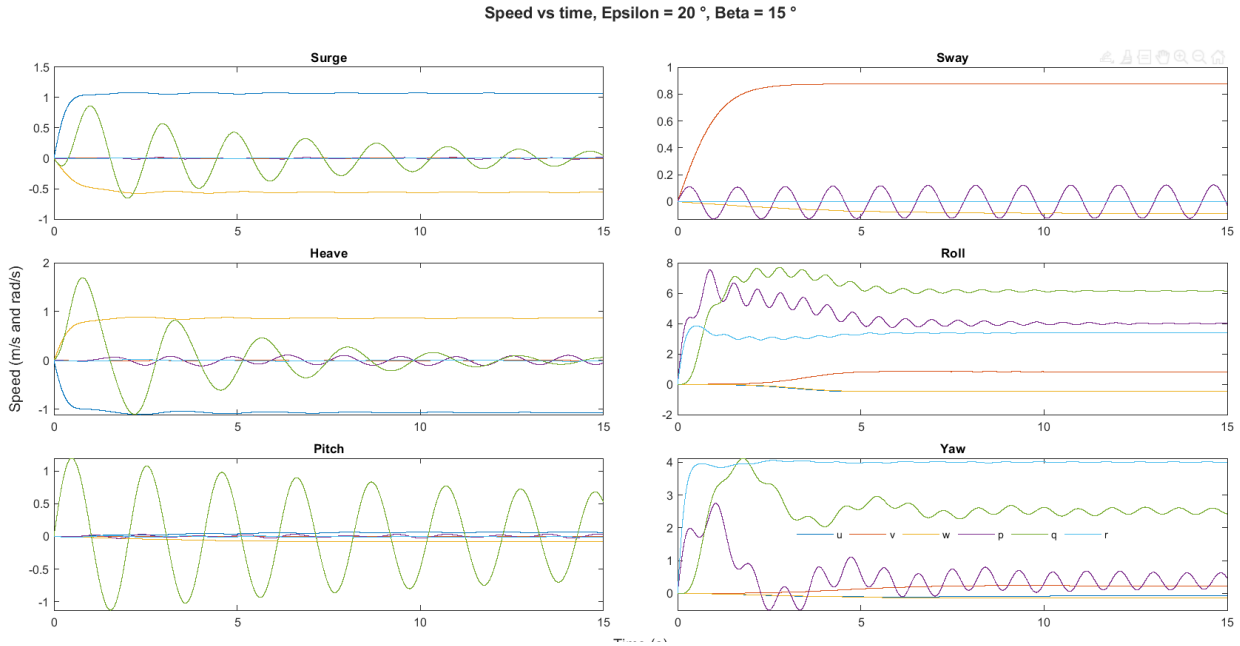

Fig. 3. VTec U4 speed profile per degree of freedom

Table 1. Maximum kinematic constraints per degree of freedom

	Max speed (m/s)	Max acceleration (m/s^2)	Max jerk (m/s^3)	Max snap (m/s^3)
Surge	1.08	3.112	7	17.3
Sway	0.8752	0.769	0.51	1.19
Heave	0.88	1.96	3.7	21

Time synchronization method

Among the three methods, the velocity method is chosen as it was the supposedly best one on synchronizing the motion, but analyzing the algorithm, one can simply give the lowest kinematic values beforehand, without requiring a specialized algorithm to synchronize them. For the VTec U4, the sway constraints are the lowest ones, so those are shared as the max values of the rest.

Implementation

The model was first programmed in Matlab to validate the motion profiles for waypoints $p1 = [1\ 1\ 1],\ p2 = [1.5\ 0.5\ 2], p3 = [1\ 1.5\ 1.5]$ and $p4 = [3\ 3\ 3]$. Fig. 4 shows the generated profiles for each degree of freedom

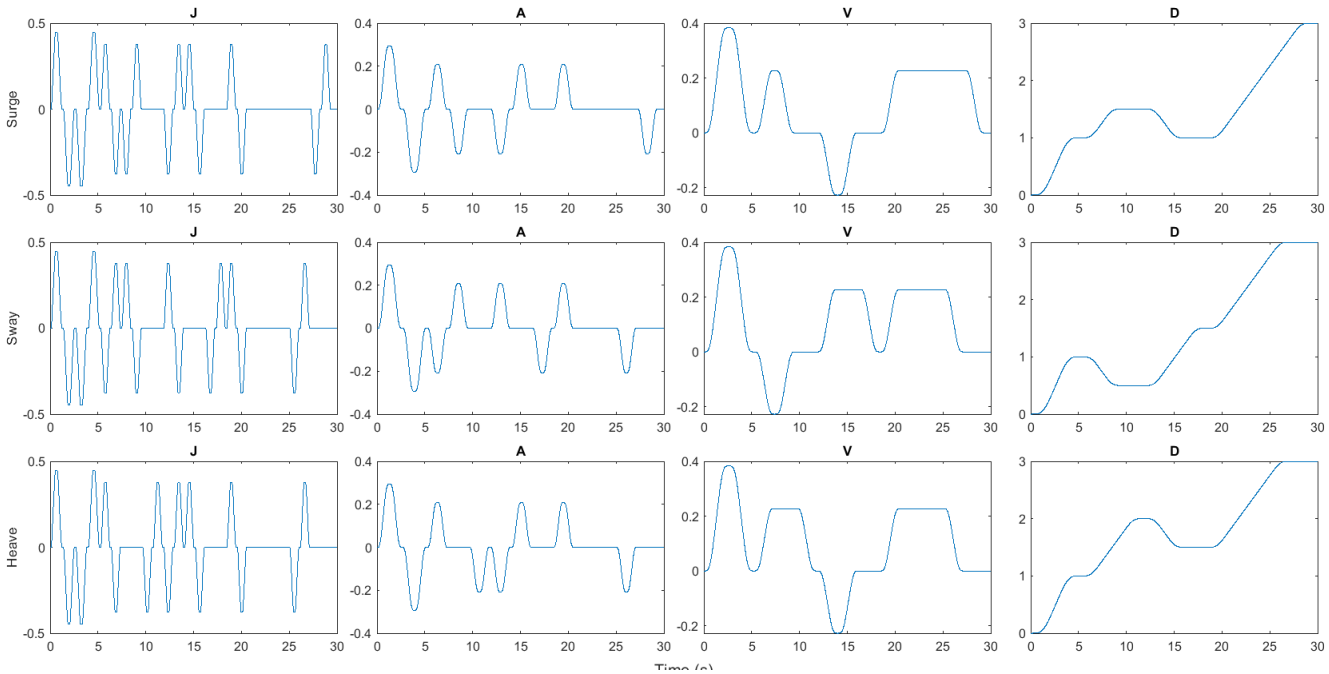

Fig. 4. Kinematic profiles per each degree of freedom

As appreciated, the maximum values of each kinematic are the same, what changes are the profiles' time periods as they are dependent on the distances, which are different. Nevertheless, smooth references are attained, as the jerk profile presents a smooth curve, and therefore, the accelerations and velocities as well. The displacement of each degree of freedom is appreciated to be valid, as all the waypoints are reached.

Next, to validate the model, it is required to provide these signals to the ANTSMC controller as trajectory references. Fig. 5 and Fig. 6 show the responses of the UUV against the desired references for the linear DOFs. It can be noted that the position reaches the desired waypoints. Moreover, the velocity profile is continuous and smooth. Fig. 7 showcase the control signals for the desired motion. It is perceived that there is no present chattering, nor saturation at any actuator; smooth signals are achieved.

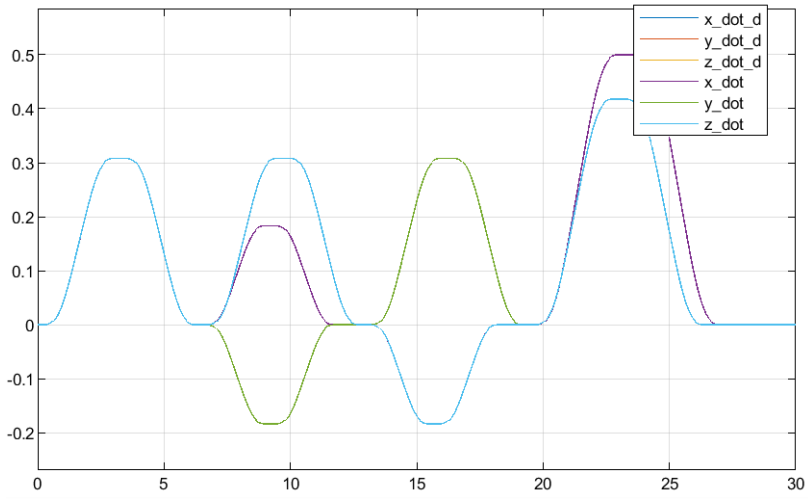

Fig. 5. Linear velocity reference vs UUV response

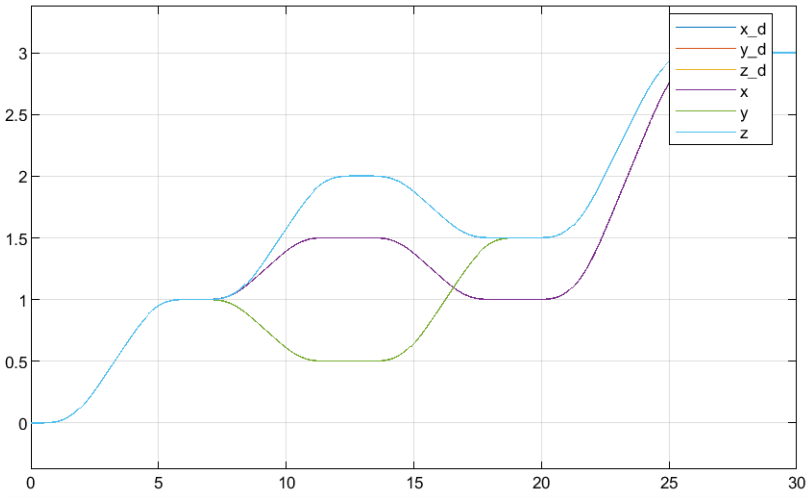

Fig. 6. Linear displacement reference vs UUV response

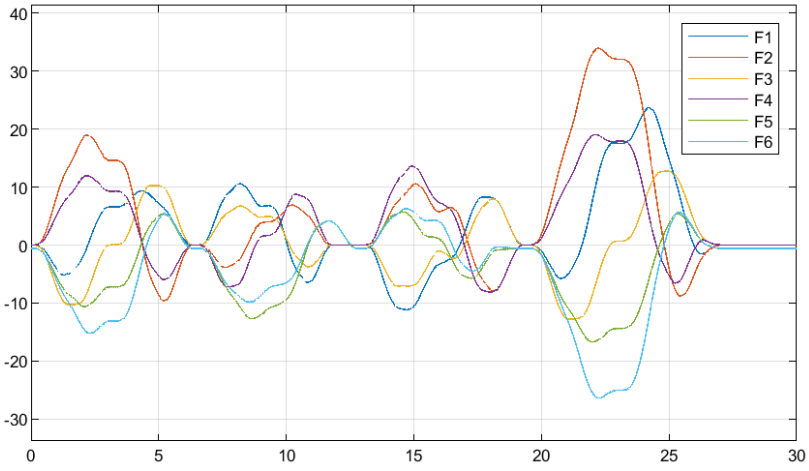

Fig. 7. Control signals

- **Library programming**

After validating the method in MATLAB/Simulink, it was programmed in C++ and integrated with ROS. The code for the s-curve trajectory generator can be found here: https://github.com/vanttec/vanttec_motion_planners/tree/main/trajectory_planners/s_curve

3. Trajectory Tracker Controller

With the trajectory generated, a trajectory tracker controller is required to drive the vehicle to reach the desired reference. For this purpose, a second order feedback linearization PID controller was designed and implemented.

- Modeling and tracking control design

The kinematic and dynamic 6-DOF model can be reviewed in (Martinez-Perez et al., 2022), as the same models are used.

For the 6-DOF PID control design for the UUV, consider a second order nonlinear dynamic system with external disturbances and model uncertainties as described below:

$$\dot{\boldsymbol{\chi}}_1 = \boldsymbol{\chi}_2$$

$$\dot{\boldsymbol{\chi}}_2 = \boldsymbol{f}(\boldsymbol{\chi}_1, \boldsymbol{\chi}_2) + \boldsymbol{g}(\boldsymbol{\chi}_1, \boldsymbol{\chi}_2)\boldsymbol{u}$$

where $\boldsymbol{\chi}_1 = \boldsymbol{\eta}$ and $\boldsymbol{\chi}_2 = \dot{\boldsymbol{\eta}}$ are the system states, $\boldsymbol{u} = \boldsymbol{\tau}$ is the control input, and the nonlinear functions $\boldsymbol{f}(\boldsymbol{\chi}_1, \boldsymbol{\chi}_2)$ and $\boldsymbol{g}(\boldsymbol{\chi}_1, \boldsymbol{\chi}_2)$ are:

$$\boldsymbol{f}(\boldsymbol{\chi}_1, \boldsymbol{\chi}_2) = -\bar{\boldsymbol{M}}^{-1}\left(\bar{\boldsymbol{C}}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})\dot{\boldsymbol{\eta}} + \bar{\boldsymbol{D}}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})\dot{\boldsymbol{\eta}} + \boldsymbol{g}(\boldsymbol{\eta})\right)$$

$$\boldsymbol{g}(\boldsymbol{\chi}_1, \boldsymbol{\chi}_2) = -\bar{\boldsymbol{M}}^{-1}$$

Tracking errors are defined as $\boldsymbol{e} = \boldsymbol{\chi}_{1,d} - \boldsymbol{\chi}_1$ and $\dot{\boldsymbol{e}} = \boldsymbol{\chi}_{2,d} - \boldsymbol{\chi}_2$ where $\boldsymbol{\chi}_{1,d}$ and $\boldsymbol{\chi}_{2,d}$ are the desired references. Finally, the controller is formulated as second order feedback linearized PID:

$$\boldsymbol{u} = \boldsymbol{g}(\boldsymbol{\chi}_1, \boldsymbol{\chi}_2)^{-1}(\dot{\boldsymbol{\chi}}_2 - \boldsymbol{f}(\boldsymbol{\chi}_1, \boldsymbol{\chi}_2) - \boldsymbol{u}_a)$$

$$\boldsymbol{u}_a = -\boldsymbol{K}_p\boldsymbol{e} - \boldsymbol{K}_i\int \boldsymbol{e}\,\boldsymbol{dt} - \boldsymbol{K}_d\dot{\boldsymbol{e}}$$

where $\boldsymbol{K}_p, \boldsymbol{K}_i, \boldsymbol{K}_d$ are diagonal matrices corresponding to the proportional, integral, and derivative gains.

- Library programming

The implementation of the dynamic model and PID controllers can be found here.

Results and discussion

The map shown in Fig. 8 was used to validate the correct functioning of the trajectory planner between two points defined by the user. In the same figure, a solution with the RRT* algorithm is shown between the points [] and [].

The s-curves method was programmed to receive any type of route as input and produce as output a trajectory (references in jerk, acceleration, velocity, and position). To achieve this, the limitations on the maximum affordable values of the kinematics of Y, since these were the smallest within the set of the three degrees of freedom, which always ensures that the trajectories would be suitable to the submarine, since larger trajectories than which the submarine is capable of tracking would be generated. Something to highlight is that the submarine could move slower than it could really be able to move due to the previous limit. In Fig. 8. the trajectory solution is shown. It can also be perceived that in straight-line references, a DOF reached the reference before the other, which resulted in a slight curve when the trajectory was generated.

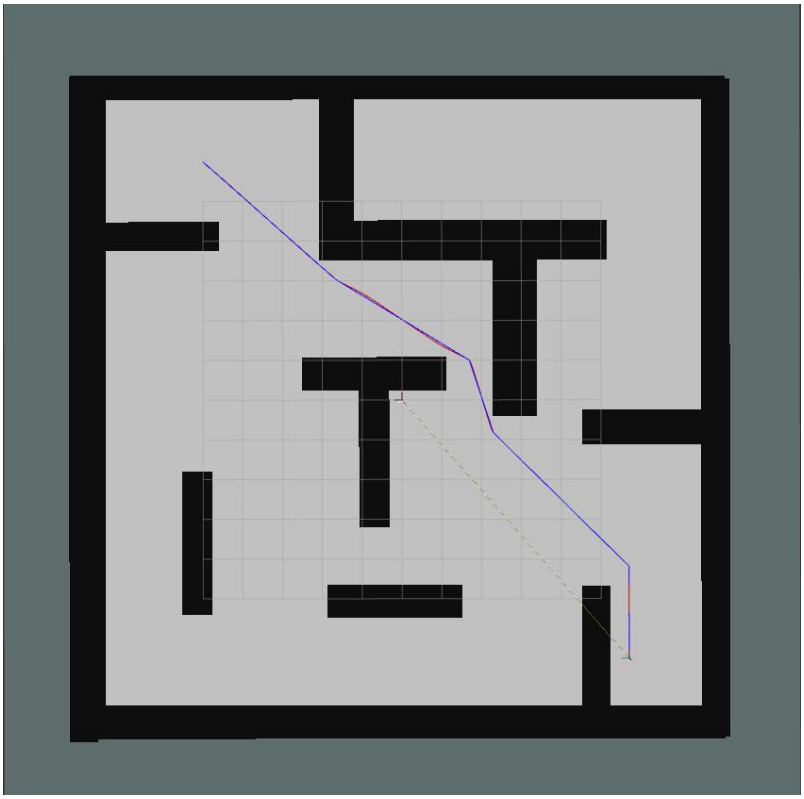

Fig. 8. 2D Map, RRT* solution (blue) and generated trajectory (red)

With the generated trajectory, it was the turn of the controller to drive the UUV states to reach the references. The trajectory generator was designed so the trajectory is calculated all at once, giving the whole reference to the controller. Fig. 9 shows the UUV tracking the trajectory, with the green line being the UUV path, and it can be noted that

the trajectory is tracked accurately. Also, the UUV stops when reaching every waypoint, which is to be expected, as the method considers initial and final kinematic values of zero.

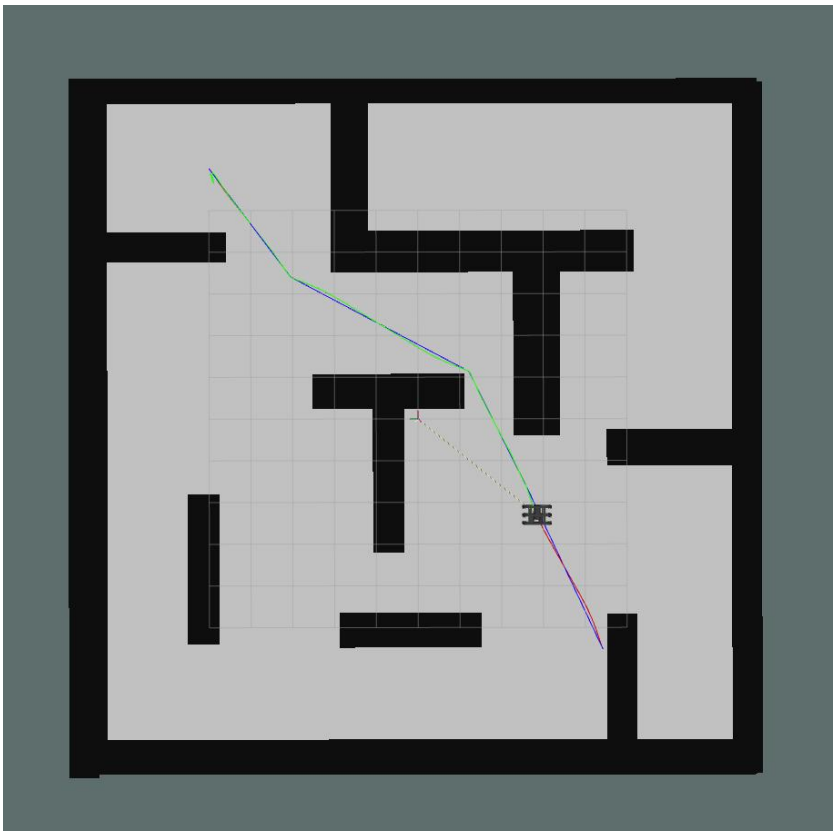

Fig. 9. Trajectory (red) vs UUV path (green)

Conclusions

A novel trajectory planning method based on sigmoid curves for the generation of an infinitely differentiable jerk profile was addressed. The method provides means to generate smooth profiles as references for a controller, that are also time optimal since the maximum kinematic constraints on snap, jerk, acceleration, and velocity are considered to calculate time periods that compose the jerk profile. The method was used to generate profiles for a UUV, based on the vehicle's kinematic constraints for each degree of freedom, and finally passed as references for a non-linear PID controller. The results show that smooth trajectories were generated for the controller that successfully drove the vehicles states towards the reference. The base s-curve method presents some drawbacks that could be addressed in the future.

First, the author claims that the time synchronization method can solve the overall synchronization method, but after some testing, the response of all DOF against a straight-line desired reference was not attained, as some DOFs reached the reference before others, so no straight line was achievable. Second, reviewing the velocity method, really no algorithm/technique is required, as the user could simply input the lowest kinematic constraints as all are known beforehand for all DOFs, which was what was

done. Finally, overall time optimality on multipoint navigation is not guaranteed, as the original method states that all the derivatives start and end at zero, so for example, if a path with multiple waypoints is used, the vehicle would be stopping at each individual waypoint, which may not be wanted. Based on this, the current method could still be improved.

On the programming of the control and motion planning libraries, the task was sometimes challenging, as thorough testing was required to ensure that the method works correctly. Overall, the control library is capable of computing control signals in six degrees-of-freedom, based on the UUV model, and the trajectory planning library is currently restricted to two dimensions, without considering changes in yaw, and with static maps. Based on these drawbacks, in the 2D context, future work can address the implementation of costmaps, like the ones used in the ROS Navigation Stack which, generally speaking, has the advantage of providing a modified map that can account for dynamic obstacles and safety areas around obstacles. Also, the implementation of yaw profiles with the s-curve method can also be addressed. After these issues are solved, it could be worth generating 3D trajectories for the vehicle, considering 3D motion is possible thanks to the controller available. Finally, multithreading solutions could be addressed, to speed up the calculation of trajectories, control signals and solution of the model.

References

Fang, Y., Hu, J., Liu, W., Shao, Q., Qi, J., & Peng, Y. (2019). Smooth and time-optimal S-curve trajectory planning for automated robots and machines. *Mechanism and Machine Theory*, *137*. https://doi.org/10.1016/j.mechmachtheory.2019.03.019

Gao, S., Huang, S., Xiang, C., & Lee, T. H. (2020). A REVIEW OF OPTIMAL MOTION PLANNING FOR UNMANNED VEHICLES. *Journal of Marine Science and Technology*, *28*(5).

Martinez-Perez, V. S., Sanchez-Calvo, A. E., Gonzalez-Garcia, A., & Castañeda, H. (2022). Adaptive Non-Singular Terminal Sliding Mode Tracking Control of an UUV Against Disturbances. *IFAC-PapersOnLine*, *55*(31). https://doi.org/10.1016/j.ifacol.2022.10.402

Richter, C., Bry, A., & Roy, N. (2016). Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. *Springer Tracts in Advanced Robotics*, *114*. https://doi.org/10.1007/978-3-319-28872-7_37