Abstract

This document presents a model prective control – MPC – algorithm aiming to autonoumously drive an Unmanned Surface Vehicle – USV – towards a set of waypoints. The design of the algorithm has been thinked to be robust to environmental disturbances encountered in the marine environment.

The modelling of the USV and disturbances have been simplified as this work is aimed to be a proof of concept: a more accurate modelling should be considered for a real world implementation.

Introduction

This work follows a previous one where the goal was to autonomously control an USV over a set of waypoints. A MPC algorithm with constant position reference - i.e. the current waypoint - tracking was implemented but this control scheme was not achieving good tracking performances. The problem could come from the optimization that was done over the position variables in the global frame which expressions are non-linear. Thus it was decided to recast the MPC strategy to make the optimization simpler using angular and forward speed variables – which expression are fully linear in the boat's frame - into the optimization problem.

In this new version, the algorithm tries to track both angle and forward speed references over a constant time horizon. The boat model was rewritten in a simpler but yet accurate way and an "intelligent" reference providing algorithm was written to make the system more robust to disturbances.

The first point that will be addressed is the modelling of the USV, followed by the tracking strategy used. Then the reference providing algorithm - including dealing with disturbances - will be explained and finally the MPC algorithm and the results obtained.

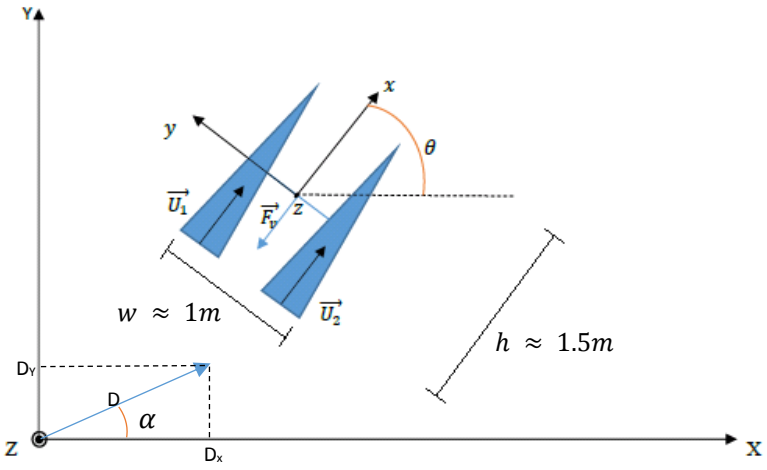
USV modelling

This part explains the physics used to describe the USV behavior and how this was modeled so it can be used into an MPC scheme.

Physical description

The USV studied here has a catamaran-like structure with two static propellers at the end of each hull.

The figure below provides a graphic view of the boat and the forces that applies to him.



Modelling

Using Newton's second law in translation and rotation the linear and angular accelerations can be written in the boat local frame:

$$\begin{cases} m\ddot{x} = U_2 + U_2 - k\dot{x} \\ I\ddot{\theta} = \dfrac{D}{2}(U_1 - U_2) \end{cases}$$

The MPC algorithm will need to track a forward-speed and an angle reference, thus a model where those are available is required.

The state-space formalism will be used to describe the system and the state and input vectors are stated as follow[1]:

$$X = \begin{pmatrix} \ddot{\theta} \\ \dot{\theta} \\ \theta \\ \dot{x} \\ \ddot{x} \end{pmatrix}, U = \begin{pmatrix} U_1 \\ U_2 \end{pmatrix}$$

Then a simple discrete-time state-space system is written:

$$X_{k+1} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ T_e & 1 & 0 & 0 & 0 \\ 0 & T_e & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & T_e \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} X_k + \begin{pmatrix} \dfrac{D}{2I} & \dfrac{-D}{2I} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \dfrac{1}{m} & \dfrac{1}{m} \end{pmatrix} U_k = AX_k + BU_k$$

As explained in [1] an optimization made on input increments will be more tractable for tracking problems. So the system was extended as follow:

$$X = \begin{pmatrix} \ddot{\theta} \\ \dot{\theta} \\ \theta \\ \dot{x} \\ \ddot{x} \\ U_1 \\ U_2 \end{pmatrix}, \Delta U = \begin{pmatrix} \Delta U_1 \\ \Delta U_2 \end{pmatrix}$$

$$X_{k+1} = \begin{pmatrix} A & B \\ 0 & I \end{pmatrix} X_k + \begin{pmatrix} 0 \\ I \end{pmatrix} \Delta U_k$$

[1] The indices used for discrete time variables have been ignored in order to simplify the equations.

Tracking

The past section dealt with variables described in the boat local frame. But the boat position needs to be estimated - or measured - in order to drive it towards the waypoint. To achieve this a technique known as "dead reckoning" has been used.

Dead reckoning

Technically dead reckoning is equivalent to integrating the boat's speed over one time step with the actual position as initial condition.

As the velocity of the boat is obtained in its own local frame, its position in the global frame depends on the angle of the boat.

Again, a state-space system will be used to implement this. However, this system will be non-linear[2]:

$$X = \begin{pmatrix} X \\ Y \\ \theta \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{x} \\ \ddot{x} \end{pmatrix}, U = \begin{pmatrix} U_1 \\ U_2 \end{pmatrix}$$

$$f(X,U) = \begin{pmatrix} X + T_e \dot{x} \cos\theta \\ Y + T_e \dot{x} \sin\theta \\ \theta + T_e \dot{\theta} \\ \dot{\theta} + T_e \ddot{\theta} \\ \ddot{\theta} + \frac{D}{2I}(U_1 - U_2) \\ \dot{x} + T_e \ddot{x} \\ \ddot{x} + \frac{1}{m}(U_1 + U_2 - k\dot{x}) \end{pmatrix}$$

This set of state equations are linearized around the boat current angle to obtain:

$$X_{k+1} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & T_e \cos\theta & 0 \\ 0 & 1 & 0 & 0 & 0 & T_e \sin\theta & 0 \\ 0 & 0 & 1 & T_e & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & T_e & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & T_e \\ 0 & 0 & 0 & 0 & 0 & -k/m & 1 \end{pmatrix} X_k + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{D}{2I} & -\frac{D}{2I} \\ 0 & 0 \\ \frac{1}{m} & \frac{1}{m} \end{pmatrix} U_k$$

Kalman filters

The previous paragraph provides a model to estimate the boat's position. However in real world applications a GPS and a compass could be used to obtain this data.

A Kalman filter could be developed to compute a better estimate of the position using both model and sensors data. This is known as sensor fusion.

[2] The indices used for discrete time variables have been ignored in order to simplify the equations.

Reference providing algorithm

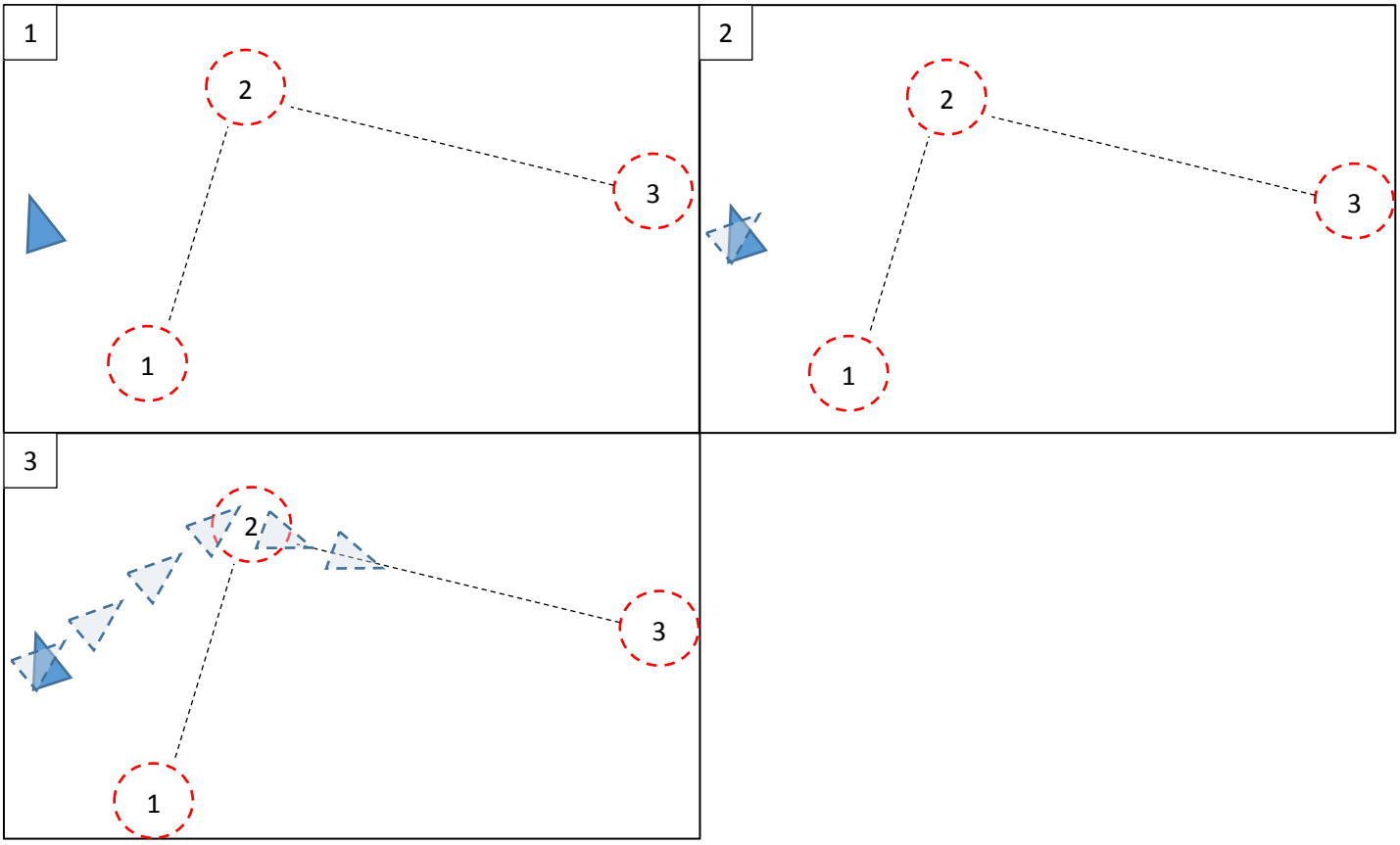
When using a MPC algorithm, a reference over the prediction horizon is needed. The goal of this algorithm is to provide such a reference and to anticipate the environmental disturbances that could be encountered. In this algorithm an angle reference is needed and the speed reference will simply be a constant.

The algorithm goal is to provide an angle reference, knowing the USV position and the environmental disturbances. It works on a fairly simple way:

- $While\ i < n_y$:
  - $Compute\ the\ angle\ between\ the\ USV\ position\ and\ the\ next\ waypoint.$
  - $Create\ a$ virtual $state\ space\ model\ linearized\ around\ the\ angle\ previously\ computed.$
  - $Input\ to\ the\ "virtual"\ model\ two\ equal\ thrusts\ computed\ to\ have\ the\ desired\ forward\ speed.$
  - $Add\ the\ angle\ to\ the\ reference\ vector.$
  - $If\ the\ boat\ arrived\ to\ the\ waypoint, consider\ the\ next\ waypoint.$
  - $Increment\ i$
- End

Where n_y is the MPC prediction horizon.

An illustrated example is provided below to help the understanding.



1. The USV is in a random position.
2. Compute the angle between the USV and the next waypoint and create a linearized model of the USV – represented with dashed lines.
3. Input to this model constant and equal thrusts until it arrives to the waypoint and then recompute the angle to the next waypoint and so on.

This algorithm outputs an angle between minus pi and pi, thus, when the difference between the current angle and the next computed is greater than pi, this means that there is a shorter path to go than what the algorithm outputs. A second algorithm is here to correct this.

It works as follow:

- $Difference = abs(angle - next\_angle)$
- $If\ difference > pi$
  - $If\ angle < 0$
    - $Next\_angle = angle - (2 * pi - difference)$
  - $Else$
    - $Next\_angle = angle + (2 * pi - difference)$
  - End
- End

Disturbances

In order to see the algorithm robustness to external disturbances, some were added to the model.

It was assumed that the disturbances were constant and that they could be measured with an arbitrary precision, which is obviously false in a real scenario but more convenient here.

The disturbances in the global frame can be expressed as:

$$D_k = \begin{pmatrix} D_X \\ D_Y \end{pmatrix}$$

Let α be the angle of the D_k vector from the X axis[3]. So in the boat frame this vector become:

$$D_k = \begin{pmatrix} D_x \\ D_y \end{pmatrix} = \begin{pmatrix} D_X \cos(\alpha - \theta) \\ D_Y \sin(\alpha - \theta) \end{pmatrix}$$

y axis velocity and acceleration had to be add to the state-space system used to provide the reference; so it became:

$$X = \begin{pmatrix} X \\ Y \\ \theta \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \end{pmatrix}, U = \begin{pmatrix} U_1 \\ U_2 \end{pmatrix}$$

[3] See the USV figure in the « USV modelling » part for details.

With this state vector, the equations became:

$$f(X,U) = \begin{pmatrix} X + T_e\dot{x}cos(\theta) - T_e\dot{y}sin(\theta) \\ Y + T_e\dot{x}sin(\theta) + T_e\dot{y}cos(\theta) \\ \theta + T_e\dot{\theta} \\ \dot{\theta} + T_e\ddot{\theta} \\ \ddot{\theta} + \frac{D}{2I}(U_1 - U_2) \\ \dot{x} + T_e\ddot{x} \\ \ddot{x} + \frac{1}{m}(U_1 + U_2 - k\dot{x} + D_x) \\ \dot{y} + T_e\ddot{y} \\ \ddot{y} + \frac{1}{m}(D_y - k\dot{y}) \end{pmatrix}$$

And linearizing around θ gives the following Jacobians:

$$F_X = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & T_e\cos(\theta) & 0 & -T_e\sin(\theta) & 0 \\ 0 & 1 & 0 & 0 & 0 & T_e\sin(\theta) & 0 & T_e\cos(\theta) & 0 \\ 0 & 0 & 1 & T_e & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & T_e & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & T_e & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -k/m & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & T_e \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k/m & 1 \end{pmatrix}$$

$$F_U = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{D}{2I} & -\frac{D}{2I} \\ 0 & 0 \\ \frac{1}{m} & \frac{1}{m} \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

So the following recursive equation is obtained:

$$X_{k+1} = F_X X_k + F_U U_k + \frac{1}{m}\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ D_x \\ 0 \\ D_y \end{pmatrix}$$

Using this system into the reference providing algorithm will automatically take disturbances into account as far as they are measured.

MPC algorithm

This part will describe the MPC algorithm and how it has been implemented in Matlab.

The previous MPC control scheme was optimizing the boat's position in the plane to drive it to the next waypoint. This method has not given good results as the trajectory was not optimal in any sense.

It is probable that this was produced by a bad optimization problem, as the boat position in the global plane was given by a non-linear system – due to the referential change between the boat frame and the global frame.

Thus it has been decided to use variables easily expressed in the boat's frame. It has been noticed that boat angle and cruise speed should be enough to control the boat. And as they can be expressed easily in the boat's frame they should give a more tractable optimization problem.

The extended state-space system described in the modelling part will be used for the MPC algorithm – with the D_x term of the disturbance added. The standard A, B notations will be used for its matrixes even if they were already used for the non-extended system.

Also, the output matrix has been chosen to match the optimized variables:

$$C = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Cost function

The cost function used is the same than the one used in [1] for the tracking problem:

$$J = \frac{1}{2}(r_{k+N} - y_{k+N})^T S(r_{k+N} - y_{k+N}) + \frac{1}{2}\sum_{n=0}^{N-1}\left((r_{k+n} - y_{k+n})^T Q(r_{k+n} - y_{k+n}) + \Delta u_{k+n}^T R \Delta u_{k+n}\right)$$

Replacing the output term with its equation from the state-space representation gives:

$$J = \frac{1}{2}\left(r_{k+n_y} - CX_{k+n_y}\right)^T S\left(r_{k+n_y} - CX_{k+n_y}\right) + \frac{1}{2}\sum_{n=0}^{n_y-1}\left[(r_{k+n} - CX_{k+n})^T Q(r_{k+n} - CX_{k+n}) + \Delta U_{k+n}^T R \Delta U_{k+n}\right]$$

With S, Q and R gain matrices.

It is show in [1] that the cost function can be expressed in a form that can be used into the $quadprog$ function of Matlab:

$$J = \frac{1}{2}x^T H x + f^T x$$

Where x is the optimization variable – i.e. the n_u future input increments written as $\overrightarrow{\Delta U}$.

It is also shown in [1] that H and f can be expressed as follow:

$$\begin{cases} H = \overline{C}^T \overline{Q}\,\overline{C} + \overline{R} \\ f^T = \begin{bmatrix} X_k{}^T & r^T \end{bmatrix} \begin{bmatrix} \overline{A}^T \overline{Q}\,\overline{C} \\ -\overline{T}\,\overline{C} \end{bmatrix} \end{cases}$$

With:

$$\overline{Q} = \begin{pmatrix} C^T QC & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & C^T QC & 0 \\ 0 & \cdots & 0 & C^T SC \end{pmatrix}$$

$$\overline{C} = \begin{pmatrix} B & 0 & \cdots & 0 \\ AB & B & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ A^{n_y-1}B & \cdots & AB & B \end{pmatrix}$$

$$\overline{R} = \begin{pmatrix} R & 0 & \cdots & 0 \\ 0 & R & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & R \end{pmatrix}$$

$$\overline{A} = \begin{pmatrix} A \\ A^2 \\ \vdots \\ A^{n_y} \end{pmatrix}$$

$$\overline{T} = \begin{pmatrix} QC & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & QC & 0 \\ 0 & \cdots & 0 & SC \end{pmatrix}$$

And X_k the current state vector and r the reference vector.

Constraints

Constraints can be easily added to an MPC scheme as they are often included into the optimization software. The $quadprog$ function of Matlab has been used to solve the optimization problem, so constraints of the following form have been added:

$$A\overrightarrow{\Delta U} \leq B$$

As shown in [2], let's pose:

$$\begin{cases} X \leq \overline{X} \\ X \geq \underline{X} \end{cases}$$

So

$$\begin{cases} X \leq \overline{X} \\ -X \leq -\underline{X} \end{cases}$$

What can be written as:

$$\begin{pmatrix} I_{n_y} \\ -I_{n_y} \end{pmatrix} \vec{X} \leq \begin{pmatrix} \overline{X} \\ \vdots \\ \overline{X} \\ - \\ -\underline{X} \\ \vdots \\ -\underline{X} \end{pmatrix}$$

Where $\vec{X}$ represents the vector of the n_y future states.

The following notation will be used:

$$C_X \vec{X} \leq d_X$$

And, as state predictions are given by:

$$\vec{X} = \overline{A} X_k + \overline{C}\,\overrightarrow{\Delta U}$$

The previous inequality can be written as:

$$C_X(\overline{A} X_k + \overline{C}\,\overrightarrow{\Delta U}) \leq d_X$$

Which gives:

$$C_X \overline{C}\,\overrightarrow{\Delta U} \leq d_X - C_X \overline{A} X_k$$

So in our constraint equation:

$$A = C_X \overline{C}$$

$$B = d_X - C_X \overline{A} X_k$$

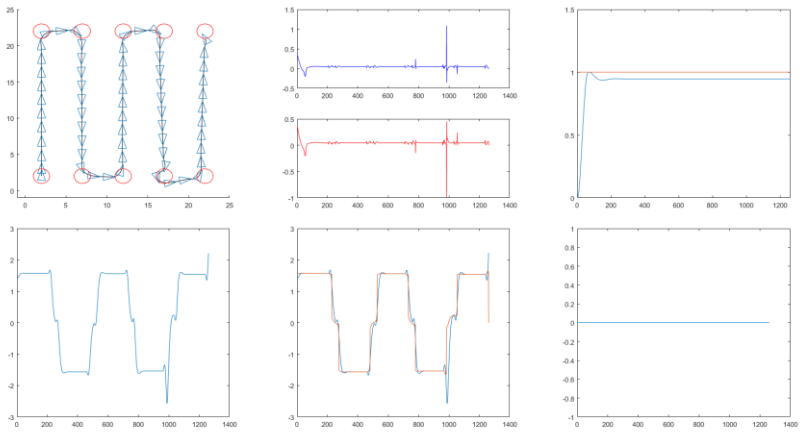
Results

This part will present the results obtained with the MPC algorithm as described in this document.

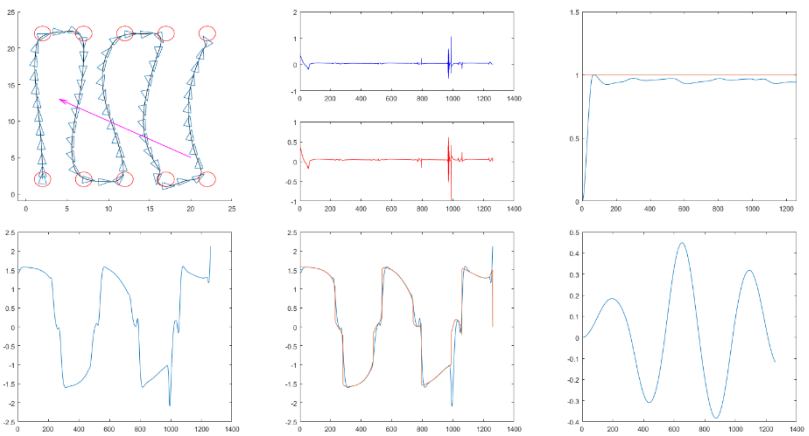
The simulations used to validate this algorithm are based on a benchmark trajectory of 10 waypoints. The USV has to reach the 10 waypoints in a defined order. This test has been made with and without disturbances – on the second figure the pink arrow represents the direction of the disturbance.

In the figures below are represented – from left to right and up to down: the USV trajectory with the waypoints, the thrust applied on each motor, the forward speed and its reference, the angle from the state vector used in the reference providing algorithm, the angle from the state vector used in the MPC algorithm and the y axis boat speed.

In the first figure below – simulation without disturbances – the main objective is clearly reached: the algorithm drives the boat around all the waypoints in a fairly optimized way. However an offset is observed on the forward speed of the boat – adding an integral gain may solve the problem. And an unexplained angle drift arises around the 1000th time step.



In the second figure – simulation with disturbances – the main objective is also clearly reached. However, besides the two kinks pointed out in the previous paragraph, two more issued arises: as the distance from the "optimal" path is not optimized, the MPC algorithm doesn't care of its trajectory as long as all the waypoints are reached thus leading to a non-optimal trajectory. Also, the y speed of the boat is not an optimization criterion so all the disturbances pass through the MPC and directly affects the boat. The problem is that the two motors are not controllable in angle, making the correction of the y axis speed difficult.



References

[1] Z. Hurák, "Optimal and Robust Control course at Faculty of Electrical Engineering, Czech Technical University," [Online]. Available: https://moodle.fel.cvut.cz/mod/folder/view.php?id=63655.

[2] J. A. Rossiter, "Constrained Predictive Control 5_3 - including constraints into GPC," [Online]. Available: https://www.youtube.com/watch?v=mkVgvA_94HM&index=3&list=PLs7mcKy_nInGHNeV1RRTo-tGFyzqOr7Nv.

Appendices

The code is available on GitHub:

https://github.com/ccalas/mpc/blob/master/final_mpc.m