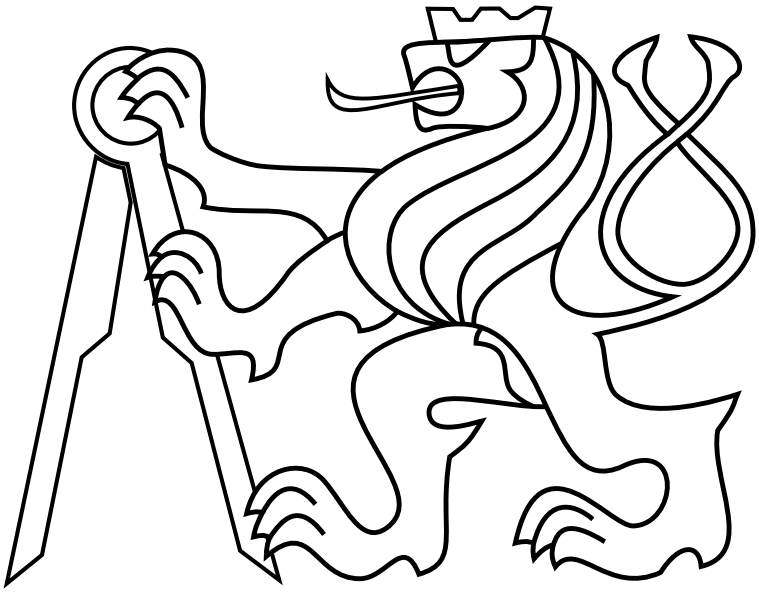
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

MASTER'S THESIS



Diego A. Saikin

Predictive Control of an Unmanned Aerial Vehicle with a Time-Variable Mass

Department of Control Engineering

Thesis supervisor: **Dr. Martin Saska**

Author statement for master thesis

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date..... .....

Acknowledgements

I would first like to thank my thesis advisor, Dr. Martin Saska, from the Deparment of Cybernetics of the Faculty of Electrical Engineering at the Czech Technical University, who is also the head of the Multi-Robot Systems team. Dr. Saska provided me an opportunity to join his popular and prestigious team, and gave access to the laboratory and research facilities.

Besides my advisor, I thank Dr. Zdeněk Hurák, and Dr. Martin Gurtner, from the Department of Control Engineering of the Faculty of Electrical Engineering at the Czech Technical University, for their insightful comments and encouragement, but mainly for giving me a so much needed push at the early stages of this thesis.

I thank my lab-mates from the Multi-Robot Systems team whose help with the implementation of my project was invaluable. Specially, among the teammates, I would like to mention Tomaš Bača, without whose help this project would have never worked. His hard work, broad knowledge and dedication are no less than admirable.

A would also like to thank my office mate Lynn Puzzo, whose writing skills and deep knowledge of the English language led to an unmeasurable improvement in the legibility and understandability of this document.

I would like to express my deep appreciation to my parents, who inculcated me the values of hard work and thought me not to be afraid of challenges.

Finally, I must express my very profound gratitude to my wife Olga, for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis.

Seriously, thank you.

Diego A. Saikin

In Memoriam, Chicha y Po Z"L

Abstract

The ultimate purpose of this thesis is to bring closer to reality, the use of fully autonomous unmanned aerial vehicle to extinguish wildfires. In order to achieve that goal, the approach taken consists in designing a solution to an optimal control problem for which a UAV can deploy a payload (e.g. fire retardant or water) above a wildfire. This solution also avoids structural damage to the UAV (due to heat exposure), and minimize the fire retardant dissipation. This is achieved by minimizing both and the releasing distance from the fire and by maximizing the speed upon releasing the payload. In addition, the optimizer takes into account environmental parameters such as wind and terrain gradient. The novelty in this method is to deliberately cause the drone to fly outside of its safe flight envelope. This means, by tilting the UAV to high pitch angles, allowing it to engage in a sort of controlled free fall, while the thrust is pointed almost horizontally, and thus achieving higher horizontal speeds than it would normally be able to. By doing so, the high heat exposure time can be minimized and the payload can be dropped closer to the fire epicenter. The optimization process takes into account the expected change in mass (due to the payload release above the fire), and allows it to engage in a *risky* maneuver, assuming that after dropping the payload, the vehicle will be lighter and thus able to recover without impacting on the terrain. The output of the optimizer consists of a full state trajectory for the whole planned maneuver. These outputs were tested with both simulators and real platforms.

Contents

List of Figures

1 Introduction

1.1 Motivation

Wildfires have been on the rise during the last few decades. Researches have concluded that 84% of wildfires between the years 1992 and 2012 were caused by humans [7]. In addition, global warming is suspected to have contributed to the intensity and frequency of those fires by making forests drier and more likely to burn [1]. That trend is not expected to change in the forthcoming years and therefore, more frequent and intense wildfires are to be expected in the future.

Small fires are easier to put out than big ones. Reaching a wildfire early is crucial to increase probability that a small, manageable fire will not become a large, unmanageable one. The key advantage of aircrafts is their speed and ability to access remote or otherwise difficult to reach fires.

Figure 1: Wildfires, aerial firefighting and this proposal.

Aerial firefighting is expensive and risky for the flight crews, and in addition it suffers from many limitations such as the inability to operate at night. Autonomous UAV's would

seem like a good alternative to regular aircrafts because they would be more efficient in payload terms, they would require significantly less training, they would pose no risk to the operators and there should be no limitation in using them during night time.

This thesis will explore the possibility of enabling UAV's to do the main task autonomously: Extinguishing the flames.

1.2 State of The Art

Although UAV's have recently begun to be employed in wildfire fighting, they are mostly used for *secondary* tasks such as mapping the fires and igniting prescribed fires [8]. Some patents attempt to use single multicopters or swarms of them to hold a hose and enable reaching close to the fire source in cases when the fire focus is hard or too risky to reach [29].

Recently, a company began the testing of a '*Kaman K-MAX*' helicopter (synchropter) which was refurbished for unmanned firefighting [16]. On [28], a broad analysis of the possible uses of UAV's in firefighting tasks is discussed in detail. However, no mention of directly using the UAV's to extinguish the fires themselves was made; only uses such as help in coordinating ground task forces, fire detection and surveillance are currently considered. It seems that the academic and engineering community is skeptical of this being even possible. However, strictly from the engineering point of view, in order to prove the feasibility of this concept, three different capabilities must be demonstrated:

a. Dropping a passive payload so that it precisely and accurately hits a target (in this case a fire).

b. Planning a trajectory that accounts for a substantial variation of mass over the time (due to the payload release), preferentially in real-time.

c. Having the ability to perform aggressive maneuvers for which linear controllers may not be sufficient.

Dropping a payload which weight accounts for a considerable percentage of the overall weight presents a challenge. Extensive amounts of literature have been written about how to overcome the difficulties caused by large mass variations. For example, in [9] an $\mathcal{H}_\infty$-based method is proposed to increase the system robustness for such a variation. Controllers like the one proposed in [15] attempt to increase the system's robustness by including the variations in the center of mass in the dynamic model, in order to suppress the unwanted effect. In general, it can be seen that mass variation (and inertia) are considered disturbances which have to be dampened or overcome.

The authors of [22] compared an MPC with a regular PID controller and a gain-scheduled PID controller for a UAV that drops a payload *"weighing one-fourth of its total weight"* at a predetermined time. They arrived to the conclusion that the gain-scheduled PID controller, despite the need to tune it, was the best option due to the lack of computational power to run an MPC with constraints on board. In addition, they used a linear model which is not suitable for the high speeds and high tilt angles which this thesis attempts to reach. In any case the article discussed a trajectory tracking but not planning.

Two other interesting master theses, [17] and [10], whose authors cooperated, explored the planning aspect of a trajectory in order to reach the optimal release point. However, these theses worked on a fixed-wing aircraft, the payload was supposed to be released from high altitudes and therefore the heat exposure criterion did not affect that planning. In addition, contrary to our case, the payload mass does not play a role in the planning because it is negligible compared to the vehicles mass. Similarly, [11] describes an interesting trajectory planning that manages to converge in about two seconds but it has no variation of any kind in any of the model parameters and the UAV is a fixed wing aircraft.

Regarding aggressive maneuvers, the authors of [19] and [18] demonstrate the generation of aggressive and complex trajectories for quadrotors as sequences of simple ones. Another good example would be [27] where aggressive maneuvers are generated by solving a constrained optimal control problem. Nevertheless, these trajectories have the aim to avoid obstacles or perch on inclined surfaces, but not to precisely drop any payload. In consequence, these trajectories do not include a change in the model's dynamics. Another good example for aggressive maneuvers generations is [12], on which the change in mass is compensated by an L_1 based adaptive controller. Nonetheless, as the other researched resources, the mass release is not planned ahead.

1.3 Progress Beyond

Both, helicopters and airplanes have advantages and disadvantages regarding aerial firefighting. Helicopters can (and must) hover high above the flames in order to drop water or fire retardant. Normally they use a bucket hanging from a long cable as shown in Fig. 1. This adds constraints to the control and stability needs, and limits the maneuverability. Airplanes, on the contrary, can carry larger payloads for longer distances and are more fuel efficient; but achieving precision in dropping the payload comes to the cost of endangering the plane because the required maneuvers are dangerous for various reasons such as, the variations in the center of mass caused by sloshing; and consequently the increased risk of a stall. Nevertheless, perhaps the largest advantage of helicopters over airplanes is their ability to gather water from shallow and small ponds or rivers. In order to refill water, some firefighting airplanes have the ability to scoop up water but they require the water source to be a large, and relatively calm, extension such as a lake, sea or a large river, so

it can perform a water landing.

This thesis attempts to generate a payload dropping trajectory which will be autonomous, precise and real-time; combining some of the advantages of VTOL and fixed-wing UAV's. The Czech Technical University's MRS team, with which the project was developed, possesses expertise with multicopters and platforms for field evaluation; and for that reason, the ODE model and the constraints were tailored to that platform.

1.4 Problem Definition

The main goal consists in making the UAV deliver the payload safely, precisely and quickly to the target. In this problem the target is a wildfire, whereas the payload is water or some kind of fire retardant. If the water is dropped from a high altitude it will evaporate or disperse before reaching the fire. For that reason it must be dropped from a as low a height as possible. Unfortunately, flying at low height will increase the heat exposure of the vehicle and consequently the risk of damage. However, the UAV is carrying the maximum amount of water it can hold. Therefore, it can barely tilt without having the vertical component of the total thrust becoming insufficient to hold the combined weight of the vehicle and payload in the air. The maximum tilt angle limitations derive in limitations of the horizontal speeds. It can be assumed that, given the payload mass accounts for a large portion of the total mass, the vehicle should become more maneuverable after dropping the payload, due to a larger power to weight ratio. If the UAV begins its trajectory from a high enough point, neglecting the need to maintain altitude, it could increase its pitch angle more than the *maximum* (i.e. the maximum pitch angle for which the vertical component of the thrust still remains enough to counteract its own weight); therefore most of its thrust would now be exploited to accelerate further. As long as the payload is released as planned above the target, the larger power to ratio available should allow the UAV to recover from the large vertical speed downwards acquired during the acceleration phase. It must be pointed that reaching to the initial state of the trajectory, as well as taking over after reaching its final point, is out of the scope of this thesis.

It is expected that the OCP solution will yield a trajectory which will resemble (in its shape), the *dive-bombing* maneuver like the ones practiced during WWII, depicted in Fig. 2.

The trajectory should be generated in a short time and the feasibility of a solution determined within the time it takes to perform the maneuver (a few seconds). The following assumptions are made:

- The UAV position, velocity and attitude are assumed to be accurately known; as well as its mass and aerodynamic drag coefficient.
- The UAV is equipped with an on-board controller that acts as a buffer between

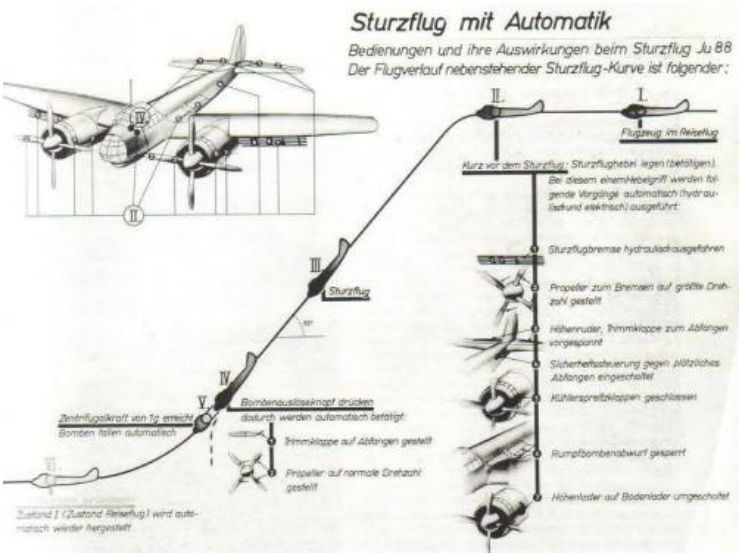


Figure 2: Manual of the dive-bombing maneuver of a German Ju-88. Source: Pinterest

the reference trajectory, generated by this optimizer, and the electric motors that drive it. The on-board controller which is responsible for making the drone follow the trajectory, is based on a controller described in [13].

- Providing the on-board controller with reference states before and after performing the maneuver is the responsibility of an MPC tracker described in [5, 3, 2].
- The target is defined as a single point on the ground.
- The ground is assumed to be any **constant gradient** plane.
- There are no obstacles.
- The trajectory is confined to a plane (2D). A *pseudo-3D* approach is implementer by rotating the trajectory plane around the vertical (Z) axis to determine the optimal heading angle, considering terrain slope and wind. The reason to take that approach are computational cost and the conviction that even a fully 3D approach would yield a similar solution. o
- The wind is known and it is assumed to move parallel to the terrain plane.
- The payload, once released, is assumed to follow an ideal ballistic trajectory (no air drag or diffusion).
- The payload is assumed to be a single rigid point, so it does not affect the UAV behavior in any way other than altering its total mass and moment of inertia (e.g. no sloshing or changes in the center of mass).

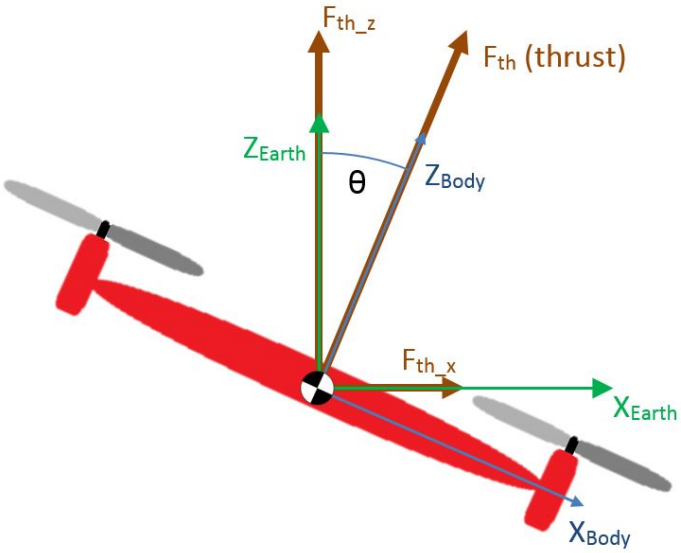


Figure 3: Multicopter reference frames, the total thrust F_{th} can be decomposed into two components: One vertical, which counteracts the gravity force; and one horizontal, which causes the UAV to accelerate (in this case towards the right). These two components can be calculated by multiplying the total thrust by $cos(\theta)$ and $sin(\theta)$ respectively, where θ is the pitch angle.

2 Dynamic Model Description

The differential equations that describe a multicopter motion can be derived from Newton's first law of motion. The acting forces are thrust, aerodynamic drag and gravity. The thrust vector norm depends on the power of the motors, while its direction is aligned with the UAV vertical axis (z_{body}). In order to induce changes in its horizontal velocity, the multicopter has to be tilted so that the horizontal components of the thrust vector point towards the desired direction of acceleration. This is clearly depicted in Fig. 3.

The UAV described here, during the maneuver, will drop a payload which could be water, foam or a fire retardant. For that reason its mass will change over time by a significant amount.

In an environment without obstacles and in which the wind has a constant velocity, the reduction of the solution to a 2D plane is justifiable. For that reason, and in order to simplify the OCP description; the model will be reduced to two dimensions only, vertical and horizontal. In addition, the equations describing the motion on both vertical and horizontal axes will be decoupled.

For the horizontal axis, the vehicle's motion is given by

$$\dot{v_x} = \frac{F_{th}\sin(\theta) - F_{D\_x}(v_x)}{m(t)}; \tag{1}$$

whereas for the vertical axis, assuming the positive direction of the z axis is upwards, the vehicle's motion is given by

$$\dot{v_z} = \frac{F_{th}\cos(\theta) - F_{D\_z}(v_z)}{m(t)} - g, \tag{2}$$

where:

- F_{th} is the total thrust.
- θ is the pitch angle around y axis (from z to x).
- $F_D(v_x)$ and $F_D(v_z)$ are the aerodynamic drag forces on the horizontal and vertical directions respectively.
- $m(t)$ is the drone mass.
- g is the gravity force set to 9.80665 m/s^2.

2.1 Aerodynamic Drag

The equation that describes the simplest form of aerodynamic drag is given by

$$F_{D\_i}(v_i) = \rho C_{D\_i} A_i v_{AS\_i}^2 * sign(v_{AS\_i}), i \in \{x, z\}, \tag{3}$$

where:

- ρ is the air density which, although it depends on altitude (and other meteorological parameters), will be considered a constant since its variation throughout the proposed maneuver can be neglected.
- C_D is the drag coefficient which depends on the physical shape of the UAV and will be inferred from empirical data.
- A is the cross-section area of the UAV which, although it is dependent on the pitch angle, will be considered a constant.
- v_{AS} is the drone airspeed given by $v_{AS\_i} = v_i - v_{wind\_i}$, where v_{wind_i} is the velocity of the wind relative to the earth in the ith axis.
- i is the axis index (x, y or z) in the *body* reference frame.

No relation between the UAV attitude relative to the air flow and the drag was modeled. No induced drag of any kind, nor friction, vortices or any other kind of drag was modeled either.

2.2 Rotational Kinematics

The control scheme proposed in this thesis is to be implemented on a multicopter which has a layered control architecture. For that reason the scope of this thesis did not include the modeling of components such as motors, propellers or rotational kinematics. Instead, a non linear controller placed on a lower layer, takes care of driving the multicopter motors in such a way that the pitch angle behaves as a first order linear system according to

$$\dot{\theta} = F_\theta(\theta_{ref} - \theta), \tag{4}$$

where, θ is the actual pitch angle of the multicopter, F_θ is the pitch angle transient factor which is dependent on the vehicle mass and θ_{ref} is the reference pitch angle of the multicopter.

The pitch angle transient factor can be estimated from recording a large amount of angles θ and θ_{ref} from the real system or from a simulator, over a relatively large timespan and a wide range of values, and applying the Moore–Penrose pseudo-inverse formula. The relation between θ and θ_{ref} in discrete time is given by

$$\theta[k+1] = F_{\theta Disc}\theta[k] + (1 - F_{\theta Disc})\theta_{ref}[k], \tag{5}$$

where $\theta[k]$ is the actual pitch angle of the multicopter at the k-th time step, $F_{\theta Disc}$ is the pitch angle transient factor **in discrete time**, which is dependent on the vehicle mass and it is subject to $0 \leq F_\theta \leq 1$ and $\theta_{ref}[k]$ is the reference pitch angle of the multicopter at the k-th time step.

This formula, for many time steps can be converted into

$$\begin{pmatrix} \theta[1] - \theta_{ref}[1] \\ \theta[2] - \theta_{ref}[2] \\ \vdots \\ \theta[k] - \theta_{ref}[k] \end{pmatrix} F_{\theta Disc} = \begin{pmatrix} \theta[2] - \theta_{ref}[1] \\ \theta[3] - \theta_{ref}[2] \\ \vdots \\ \theta[k+1] - \theta_{ref}[k] \end{pmatrix}. \tag{6}$$

Once $F_{\theta Disc}$ is calculated, if the sampling time T_s is given, F_θ can be easily derived by transforming the system from discrete time into continuous time according to the relation

$$F_\theta = -\frac{\ln(F_{\theta Disc})}{T_s}. \tag{7}$$

2.3 State Equation

Merging all the variables that describe the UAV kinematics into a state vector

$$X \stackrel{\text{def}}{=} \begin{pmatrix} p_x \\ v_x \\ p_z \\ v_z \\ \theta \end{pmatrix} \tag{8}$$

and their respective ODE's (eq.1, 2 and 4) to the form $\dot{X} = \mathbb{F}(X)$, we obtain

$$\begin{pmatrix} \dot{p_x} \\ \dot{v_x} \\ \dot{p_z} \\ \dot{v_z} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} v_x \\ \frac{F_{th}\sin(\theta) - F_D(v_x)}{m(t)} \\ v_z \\ \frac{F_{th}\cos(\theta) - F_D(v_z)}{m(t)} - g \\ F_\theta(\theta_{ref} - \theta) \end{pmatrix}, \tag{9}$$

where p_x is the UAV horizontal position, p_z is the UAV altitude, and θ_{ref} and F_{th} are the control inputs, i.e. the reference pitch angle and thrust respectively.

3 Optimal Control Problem

In order to remain airborne, the vertical component of the thrust must be larger or equal to the UAV weight. For that reason the pitch angle must always comply with

$$\theta_{max\_no\_descent} \leq \arccos\left(\frac{m(t)g}{F_{th\ max}}\right). \tag{10}$$

This limitation defines the *flying envelope* of the UAV. The maximal speed achievable for a certain weight is reached when the horizontal component of the thrust equals the aerodynamic drag force. For that reason, the limitation on the maximum tilt angle described in eq. (10) limits also the maximal speed it can reach. This maximal speed can be calculated by setting eq. (1) to zero, and the pitch angle $\theta = \theta_{max\_no\_descent}$. By substituting the aerodynamic drag force F_D with eq. (3), and then isolating v_x we obtain

$$v_{x\ max\_no\_descent} = \sqrt{\frac{F_{th\ max}\sin(\theta_{max\_no\_descent})}{\rho C_{Dx} A_x}}. \tag{11}$$

In simple terms, eq. (10) and (11) imply that the heavier the UAV is, the slower it can fly. If the UAV will fly above a wild fire at a low speed, the long exposure to the intense heat may damage it.

The thesis goal is to generate a maneuver such that:

- The UAV is able to accelerate to more than the *maximum no descent speed* before releasing the load, so the time spent by the vehicle in close proximity to the flames is minimized.
- The payload can be released at a low enough altitude.
- The payload, after being released, follows a free fall trajectory and hits the target precisely.
- The vehicle does not crash while performing the maneuver (assuming the payload was successfully released).

If the *maximum no descent speed* constrain is overridden by allowing the pitch angle to exceed $\theta_{max\_no\_descent}$, thus flying outside the flight envelope (meaning that the UAV will inevitably loose altitude), it will increase it's speed considerably higher than $v_{x\ max\_no\_descent}$. For such a pitch angle, the maximum theoretical speed can be calculated by setting eq. (1) to zero, and the pitch angle this time to $\theta = \pi/2$ (pointing the thrust directly forward). Again, by substituting the aerodynamic drag force F_D with eq. (3), and then isolating v_x we obtain

$$v_{x\ max} = \sqrt{\frac{F_{th\ max}}{\rho C_{Dx} A_x}}. \tag{12}$$

However, this poses a danger to the vehicle because of the large vertical speed it gains. Nevertheless, by the time the *recovery* stage is attempted, the UAV is expected to be lighter because it will not be carrying the payload anymore.

3.1 Optimal Control Problem Description

The aim of any optimal control problem, is to find values for a set of variables Ω_L which minimize a cost function $J(\Omega_L)$, while these variables are subject to several constraints. Hopefully the reached cost value will be the global minimum but if the problem is non-convex, there are no guarantees this will happen.

The desired solution to the OCP treated in this thesis is composed of a full-state trajectory, the corresponding input control signals and the general heading angle on which this trajectory is oriented. The problem is subject to a set of constraints dictated by the UAV dynamics, by the task goal and by the required safety measures. Naturally, it is of a highly non-convex nature and it has a relatively large time interval. The chosen method to solve this OCP is the *multiple shooting* method. A significant advantage of this method is that it is relatively easy to parallelize. In this method, the trajectory is discretized into finite N time-steps.

3.1.1 The Stages

The trajectory was divided into three consecutive stages: *Approach*, *Release* and *Recovery*. It is not the responsibility of this maneuver to determine how to reach the starting point or what to do after the end of the trajectory is reached. However the initial and final system state must be feasible by the on-board controller.

The approach stage is the segment of the trajectory from the start point until the UAV reaches the release point, which is the location where the payload begins to be released. During this phase it accelerates towards the release point while losing altitude. The release point is calculated assuming the payload will follow an ideal ballistic trajectory.

The release stage is the segment of the trajectory from the release point until all the payload is completely released. This phase can be modeled in many ways depending on the release mechanism.

The recovery stage is the segment that begins after the end of the payload release until the end of the trajectory. The purpose of this stage is in essence, to diminish the downwards vertical speed acquired during the approach stage, in order to ensure that the vehicle will not crash against the ground after releasing the payload. In practice, although it must be planned in order to ensure the feasibility of the OCP solution, it is not imperative for the vehicle to follow the whole trajectory until its final time step. Once the recovery stage is reached, the control of the UAV could be ceded at any time before the formal end of the trajectory, as long as it is determined that there is no danger of collision with the terrain.

Each stage was given a constant amount of time steps $N_{approach}$, $N_{release}$ and $N_{recovery}$ which account for the total amount of time steps N.

3.1.2 The Optimized Variables

The set of optimized variables Ω_L, also called '*optimization vector*' is composed of:

a. The sequence of states
$$\Omega_X = \{X[0], X[1], \ldots, X[N]\}, \tag{13}$$
that fully describe the movement of the UAV through the whole maneuver in discrete time steps.

b. The sequence of control inputs
$$\Omega_U = \{U[0], U[1], \ldots, U[N-1]\}, \tag{14}$$
where each vector $U[n]$ is composed of the thrust F_{th} and reference pitch angle θ_{ref} for a specific time step.

c. The time step lengths $t_{s\_approach}, t_{s\_release}$ and $t_{s\_recovery}$, which differ for each trajectory stage.

d. The payload free fall time t_{ff}.

e. The trajectory heading vector H_{DG}, a $\mathbb{R}^2$ unit vector defined as $[cos(\psi), sin(\psi)]^\top$, where ψ is the heading angle.

The payload release time $t_{release}$ takes place after the last time step of the approach stage and thus, is given by $t_{release} = t_{s\_approach} N_{approach}$.

3.1.3 The Cost Function

The cost function used in this optimization problem has three main terms:

a. The total amount of time it takes to complete the maneuver, which is to be minimized.

b. The altitude at which the payload is released, which is to be minimized.

c. The horizontal velocity at which the payload is released, which is to be maximized.

Each one of the components is multiplied by a weight factor. Different weight factors alter the final trajectory depending on their relative values. The cost formula can be defined as

$$J(\Omega_L) = W_T \sum_{0}^{t_f} t + W_{hr} p_z[N_{approach}] + W_{vr} v_x[N_{approach}] \tag{15}$$

where, W_T is the weight of the completion time of the trajectory, W_{hr} is the weight of the release altitude component and W_{vr} is the weight of the release speed component of the cost function. $p_z[N_{approach}]$ and $v_x[N_{approach}]$ are the UAV altitude and speed in the X axis respectively, at the time step on which the releasing stage begins.

3.1.4 The Constraints

The constraints to which the OCP is subject limit the set of solutions. These can be divided in two subtypes, *fixed* and *parametric*. The fixed constraints are those which are essential to the problem solution and must always be met. On the other hand, the parametric constraints can be adjusted prior to running the simulation in order to achieve different effects on the trajectories planing. For example, in order to help the optimizer to converge to a good solution, this problem the constraints were tuned so that the UAV always performs the maneuver moving forward.

3.1.4.1 Fixed Constraints

- The state propagation according to the dynamic model as described by the ODE's in eq. (9).
- The state values continuity over time. For each time step the model's ODE is propagated for the length of t_s using a classical 4th Order Runge-Kutta method. The resulting state must be equal to the initial state of the next time step, as defined in the *Multiple Shooting* method.
- The released payload, after following a ballistic trajectory through time t_{ff}, must hit the target. This is calculated for the initial time of the payload release regardless of the releasing mode. The free-fall time is given by

$$t_{ff} = \frac{v_{z\_release} + \sqrt{v_{z\_release}^2 + 2gp_{z\_release}}}{g}, \tag{16}$$

  where $p_{z\_release} = p_z[N_{approach}]$ and $v_{z\_release} = v_z[N_{approach}]$, i.e. the altitude and vertical velocity upon release respectively.

  The location where the payload will hit the ground is given by

$$p_{x\_hit} = p_{x\_release} + v_{x\_release}t_{ff}, \tag{17}$$

  where p_{x_hit} is the position in the trajectory's x axis where the payload hits the ground, which must coincide with where the target is located (0 in this case) and $p_{x\_release} = p_x[N_{approach}]$ and $v_{x\_release} = v_x[N_{approach}]$.
- The heading vector must be a unit vector, $\|\mathbf{H}_{DG}\| = 1$.

3.1.4.2 Parametric Constraints

For all the optimized variables, there are constraints limiting minimum and maximum values they can reach. These limitations may be differ for different stages of the trajectory. In fact, some of the state variables and control signals at the initial state, during the trajectory and at the final state are constrained to different ranges of values. Some of these limitations can be changed in order to achieve different effects, but most of them remain constant regardless of the different scenarios. For example, though the whole trajectory, the height with respect to the terrain was constrained to remain above a minimal value; and the pitch angle θ was constrained to $\frac{-\pi}{2} \leq \theta \leq \frac{\pi}{2}$ in order to eliminate the possibility that the UAV would direct its thrust downwards. However, only at the final state, the vehicle's velocity was constrained to be positive both in the horizontal and vertical directions, $0 \leq v_x[N], v_z[N] \leq \infty$, meaning that the UAV is climbing so it is clear of danger from collision with the terrain. Another example is the maximum thrust used for planning $F_{th\_max\_plan}$, which was constrained to a percentage of the maximum theoretical value (70% - 80%,

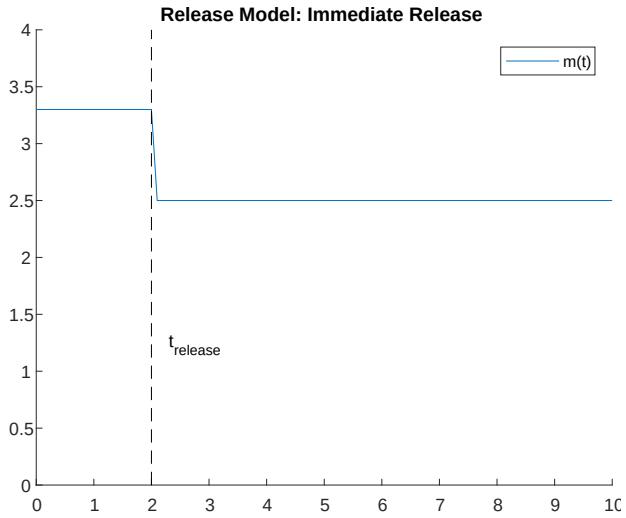


Figure 4: The mass over time of a 2.5 kg UAV carrying a 0.8 kg payload and releasing it immediately at t=2 s.

empirically determined). This constraint, $0 \leq F_{th\_max\_plan} \leq 0.8max(F_{th})$ was imposed in order to make sure that the UAV has spare power to exert control and the thrust does not reach saturation.

3.1.5 The Releasing Models

The releasing models are expressed in the way the total mass $m(t)$ evolves in time. Three releasing mechanisms were considered for modeling and testing. One mimics a mechanic arm or solenoid, in which all the payload is immediately released; the second mechanism mimics a constant flux pump; and the third, mimics a container with a hole at its bottom through which the payload flows out.

3.1.5.1 "Immediate Release" Model

In this release model, the mass change is represented by a negative delayed Heaviside step function $\mathbb{U}(t)$ and thus, the total mass along the trajectory is given by

$$m = m_{UAV} + m_{payload}(1 - \mathbb{U}(t - t_{release})), \tag{18}$$

where m_{UAV} is the vehicle mass and $m_{payload}$ is the total payload mass. In practice, this was modeled by imposing a discontinuity in the mass parameter of eq. (9). Fig. 4 depicts the value of the total mass through time before and after the releasing time.

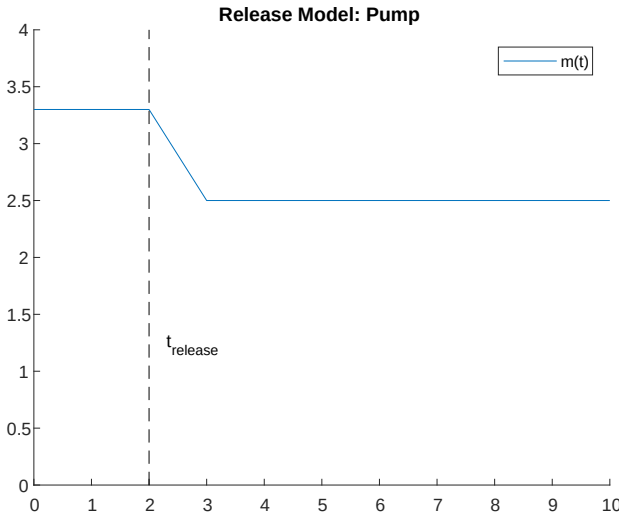


Figure 5: The mass over time of a 2.5 kg UAV carrying a 0.8 kg payload and releasing it through a 0.8 l/s pump, beginning at t=2 s.

3.1.5.2 "Pump" Model

In this model, the mass decreases at a constant rate $\dot{m}$ from $m_{UAV} + m_{payload}$, over a period of time until it reaches m_{UAV}. The mass decrease rate is a constant defined by the properties of the modeled pump. In order to achieve this, the ODE state equation described in eq. (9) was expanded with the formula

$$\dot{m} = \begin{Bmatrix} const < 0, stage = release \\ 0, otherwise \end{Bmatrix}. \tag{19}$$

Fig. 5 depicts the value of the total mass through time before, during and after the releasing time.

3.1.5.3 "Hole in a Bucket" Model

This can be modeled by a special case of Bernoulli's equation in which the fluid is incompressible; and viscosity and friction are neglected. The velocity at which the fluid leaves the bucket is given by a modification of the potential/kinetic energy equation

$$v_f = \sqrt{2gh}, \tag{20}$$

where v_f is the fluid exit velocity through the hole in the bottom of the bucket, g is the gravity force (*felt* by the fluid), and h is the height of the fluid column. $\dot{m}$ can be calculated by multiplying the fluid velocity by the hole cross-section area and the fluid density thus

providing us with an ODE that describes the evolution of mass along the time. The fluid column height (h) is also dependent on the remaining amount of fluid in the bucket; and on the bucket's geometrical characteristics. It is given by the quotient of the fluid volume divided by the bucket cross-section area. If the fluid volume is replaced by the division of its mass and density, we obtain

$$h = \frac{m(t)}{\rho a_{bucket}}, \tag{21}$$

where h is the fluid column height, ρ is the fluid density, $m(t)$ is the fluid mass and a_{bucket} is the bucket cross-section area. The gravity force felt by the bucket is in fact, the addition of earth's gravity and the vehicle vertical acceleration. Assuming a uniform bucket cross-section (e.g. cylindrical), the resulting ODE describing the dependence of the mass over time is given thus by

$$\dot{m} = -a_{hole}\sqrt{2(g + \dot{v_z})\frac{\rho m_{payload}(t)}{a_{bucket}}}, \tag{22}$$

where a_{hole} is the cross-section area of the hole through which the fluid exits the bucket, a_{bucket} is the cross-section area of the bucket. This can be developed further, if $\dot{v_z}$ is substituted by eq. (2), assuming the mass is $m_{payload} + m_{UAV}$, the pitch angle $\theta = 0$, and the vertical speed v_z is zero as well thus, canceling the aerodynamic drag; we obtain

$$\dot{m} \approx -a_{hole}\sqrt{\frac{2\rho F_{th} m_{payload}(t)}{a_{bucket}(m_{UAV} + m_{payload}(t))}}. \tag{23}$$

Fig. 6 depicts the value of the total mass through time before and during releasing.

3.1.6 Simulation Scenarios

The trajectory planner was designed to cope with two typical environmental parameters *terrain gradient* and *wind*. These parameters have diverse effects and can be set to create different simulation scenarios. The heading vector is optimized taking into account the wind strength and direction, as well as the terrain gradient norm and direction, in order to determine the optimal heading angle for approaching the target. That means, the heading angle for which the cost function (eq. 15) will yield the lowest possible value.

3.1.6.1 Terrain Gradient

The terrain gradient $\nabla_{terrain} \in \mathbb{R}^2$ was modeled as a constant vector for each particular scenario. More complex terrain models such as DTM or the inclusion of obstacles were left out of this thesis' scope because it would make the problem much more non-convex and

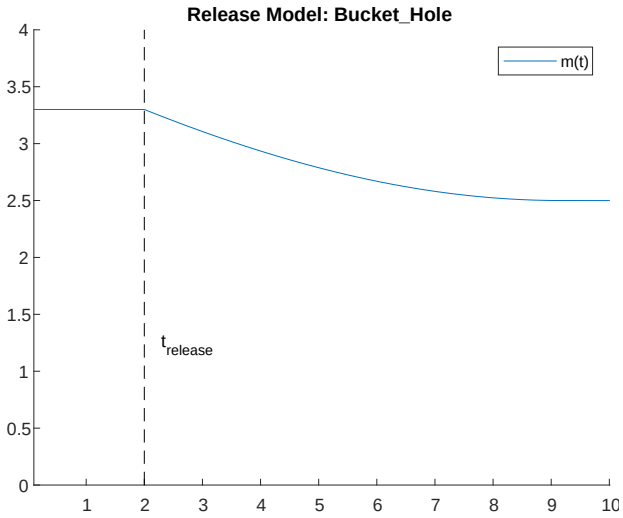


Figure 6: The mass over time of a 2.5 kg UAV carrying a 0.8 kg payload on a bucket with a base area of 0.1 m^2 and releasing it through a 0.05 m^2 opening at the bottom, beginning at t=2 s.

it would also be very computationally expensive. Nevertheless, an extension to the proposed method would be straight forwards. More complex terrains could be approximated for example with polynomial approximations.

Since the target is considered to be always at elevation zero, at the coordinates' origin, the vehicle height can be calculated according to

$$h = p_z - p_x \mathbf{H}_{DG}^{\top} \cdot \nabla_{terrain}. \tag{24}$$

3.1.6.2 Wind

The wind $\mathbf{v}_{wind} \in \mathbb{R}^3$ was modeled as a vector which represents the wind velocity and remains constant for each particular scenario. The wind is modeled so that it always flows parallel to the terrain. That means that the vertical component of the wind $\mathbf{v}_{wind\_z}$ is defined by

$$\mathbf{v}_{wind\_z} = [\mathbf{v}_{wind\_x}, \mathbf{v}_{wind\_y}]^t \cdot \nabla_{terrain}. \tag{25}$$

No wind gusting, whirlwinds, wind shear, turbulence, thermal or any other meteorological phenomena were modeled, since these are too computationally costly, these may vary too frequently relatively to the time it takes to solve the OCP and perform the maneuver; and in addition, usually such information on local conditions is not available and neither be estimated from measurements taken by the sensors on board. The wind is assumed to be known a-priori and it can be divided in two different components, one parallel to the trajectory plane, and a crosswind component. The cross wind was modeled as a disturbance

which causes a reduction in the vehicle thrust on the trajectory plane. This is caused due to the need of the UAV to adjust its roll angle, in order to counteract the lateral force that the cross wind exerts on it, leaving thus less thrust available for maneuvering in the trajectory plane. The available thrust, after compensating for cross wind is given by

$$F_{th\_max} = \sqrt{1 - \left(\frac{F_D(v_{crosswind})}{max(F_{th})}\right)^2}. \tag{26}$$

$v_{crosswind}$ is calculated by projecting the wind vector into another unit vector, which is perpendicular to the heading, according to

$$v_{crosswind} = \mathbf{v}_{wind\_x,y}^{\top} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \mathbf{H}_{DG}. \tag{27}$$

3.2 Optimal Control Problem Solution

One of the best suited tools to solve this OCP is the *CasADi* toolbox. This software has multiple solvers and allows a large amount of freedom to describe this kind of problems, and particularly the *multiple shooting*. In addition, this package has a Matlab API that eases the description of the problem and its solution. The solver used for the task is the IPOPT (**I**nterior **P**oint **OPT**imizer) which forms part of the CasADi package and is well suited for a large-scale nonlinear optimization problem such as this one. For all the simulations, in order to simplify the problem description, the target was placed at the axes' origin, so that the whole maneuver revolves around it.

4 Offline Simulations and Analysis

The simulations were performed first in Matlab where various combinations of parametric constraints, environmental parameters and releasing models were tested. In addition, the effects of different initialization values for the optimization vector Ω_L were analyzed[1]. Then, the method was verified by performing simulations with a software package called *'Gazebo'*, which is a physics simulator, that can also render 3D real-time visualizations of the modeled scenario. The simulations performed in Gazebo are discussed in more detail on Section 5.

Parameters common for all the simulations:

- $m_{UAV} = 2.5$ kg
- $m_{payload} = 0.8$ kg
- $\frac{-\pi}{2} < \theta, \theta_{ref} < \frac{\pi}{2}$
- $F_{th\_max} = 34.32$ N $= (3.5 \text{ kg} \times g)$, $F_{th\_max\_plan} = 0.8 F_{th\_max}$ in order to plan assuming there is less thrust available than in reality. That *pessimistic* approach allows the UAV to have spare power left to maneuver. $F_{th\_max\_plan}$ is not even enough to maintain both the UAV and its payload airborne but it is enough just for the UAV.
- The simulated altitude is the mean sea level, that causes the drag coefficient to be 0.0902 according to eq. (3).
- The *max no descent* speed for the loaded vehicle is 11.26 m/s, based on eq. (11).
- The *max no descent* speed for the unloaded vehicle is 16.32 m/s, based on eq. (11).
- The maximum theoretical speed at $\theta = 2\pi$ is 19.51 m/s, based on eq. (11) if θ is set to $\pi/2$.
- The total amount of time steps was set to $N = 100$ distributed in the following manner:
  1. $N_{approach} = 40$, $N_{release} = 0$ and $N_{recovery} = 60$, for *immediate release* payload releasing mode.
  2. $N_{approach} = 40$, $N_{release} = 10$ and $N_{recovery} = 50$, for all the other payload releasing modes.

4.1 Simulation #1 (flat terrain, no wind)

This simulation was run using the environmental parameters:

[1] A video containing animations of the Matlab simulations can be found at http://mrs.felk.cvut.cz/mpc-with-time-variable-mass

		$p_x[m]$	$v_x[m/s]$	$p_z[m]$	$v_z[m/s]$	θ
x_0	max	-10	0	200	0	0
	min	-110	0	$-\infty$	0	0
x	max	∞	∞	200	∞	2π
	min	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-2π
x_f	max	∞	∞	200	∞	0
	min	15	$\frac{v_{x\ max\_no\_descent}}{2}$	$-\infty$	1	$-\theta_{max\_no\_descent}$

Table 1: Boundaries of the state vector in simulation #1. x_0 is the initial state, x is the state through the course of the maneuver and x_f is the final state

- $\|\nabla_{terrain}\| = 0$
- $\|\mathbf{v}_{wind}\| = 0$ m/s

All the parametric constraints are according to table 1.The minimum allowed height above the terrain (which in this specific scenario is the same as p_z because $\|\nabla_{terrain}\| = 0$) was set to 5 m.

The weights of the different terms of the cost functions were set as follows:

- $W_{hr} = 50$ (for height upon releasing).
- $W_{vr} = 300$ (for horizontal speed upon releasing).

Both releasing models were tested (*pump* and *immediate release*) and there were no significant differences between their results.

For the *immediate release* payload releasing model, the horizontal speed at the release time was 16.83 m/s, 49% above $v_{max\_no\_descent\_loaded}$ and the release altitude was 35.96 m.

For the *pump* release mode, the horizontal speed at the release time was also 16.83 m/s, 49% above $v_{max\_no\_descent\_loaded}$ and the payload was released between altitudes 39.93 m and 26.33 m.

In general, it can be appreciated observing Fig. 8 and Fig. 9 that neither the state variables nor the control signals have any major differences between the release models other than the mass variation. The path followed by the UAV can be appreciated in Fig. 7. In this case the different heading angle ψ can be due to the *hot start* values or some local minimum. There is no factor favoring any specific heading angle.

4.2 Simulation #2 (moderate terrain gradient, no wind)

This simulation was run using the following environmental parameters:

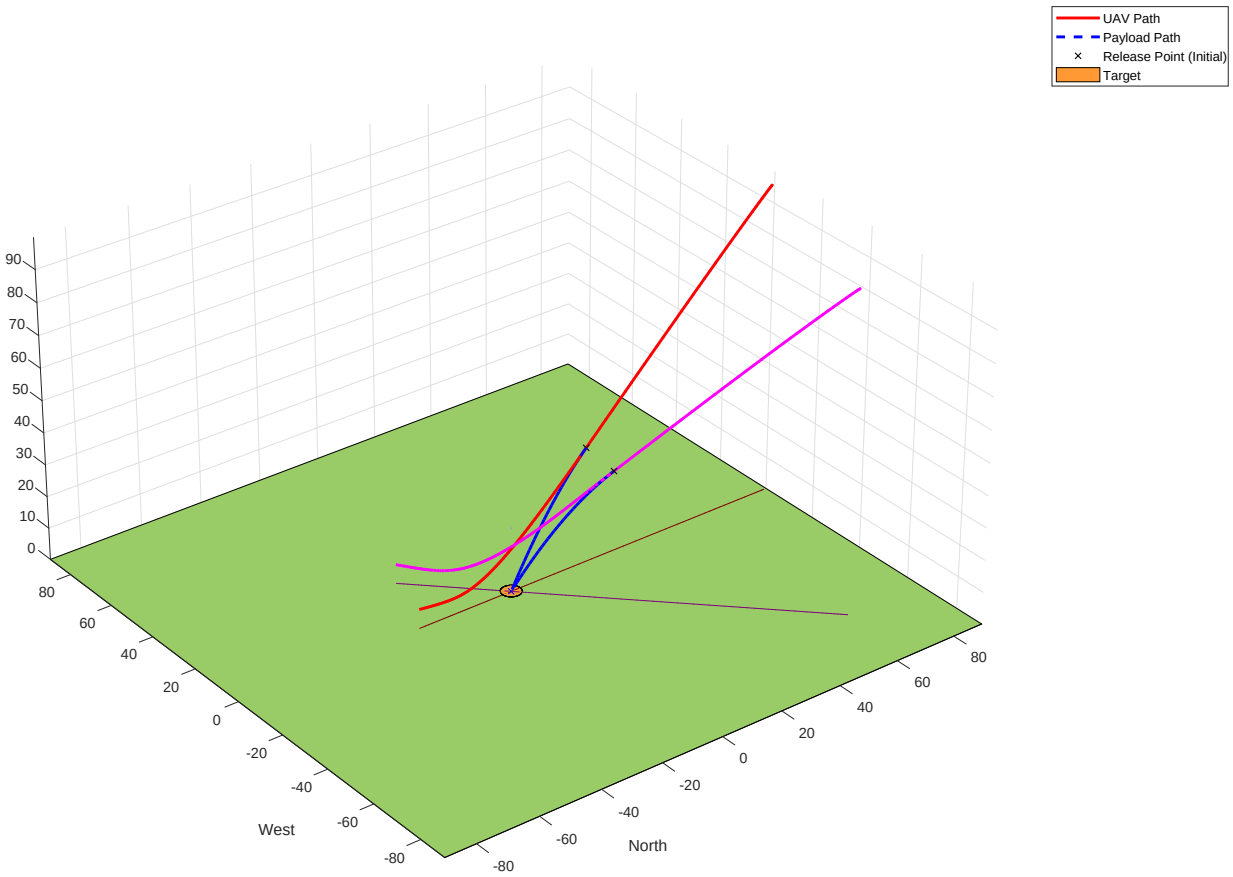


Figure 7: Trajectory followed by the UAV on both releasing model (red-immediate release and magenta-pump). It can be seen that their shape does not differ much except for the release point. The difference in the heading between releasing models is due to the lack of any direct relation between the heading chosen and the cost function, so both models converge to random heading angles (or pre-biased, depending on the hot start values).

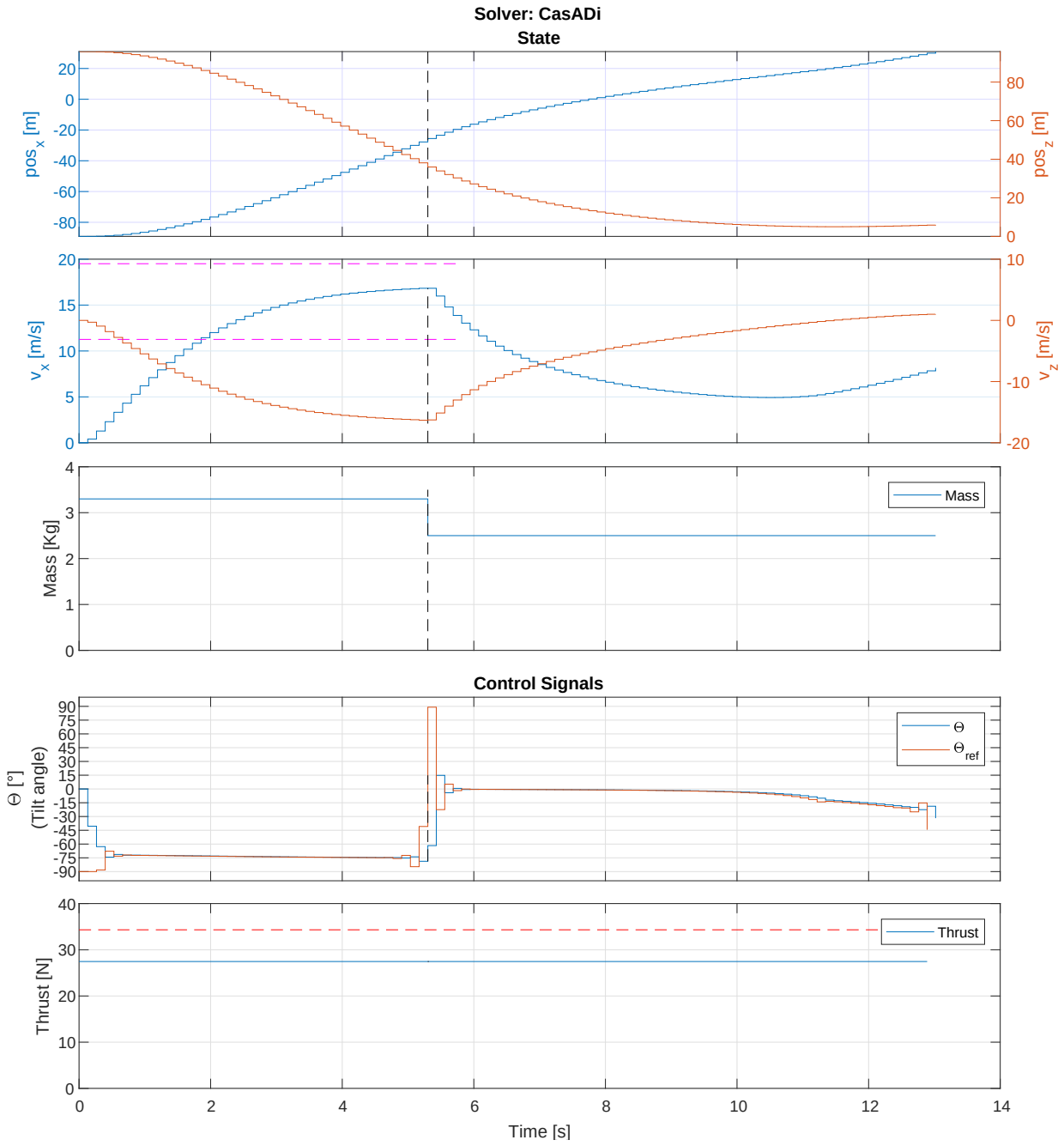


Figure 8: The state and control signals of the vehicle through a trajectory for the *immediate release* payload releasing model. The lower and higher magenta dashed lines on the velocity graph represent the *max no descent* speed and the maximal speed achievable at $\theta = 2\pi$ respectively

.

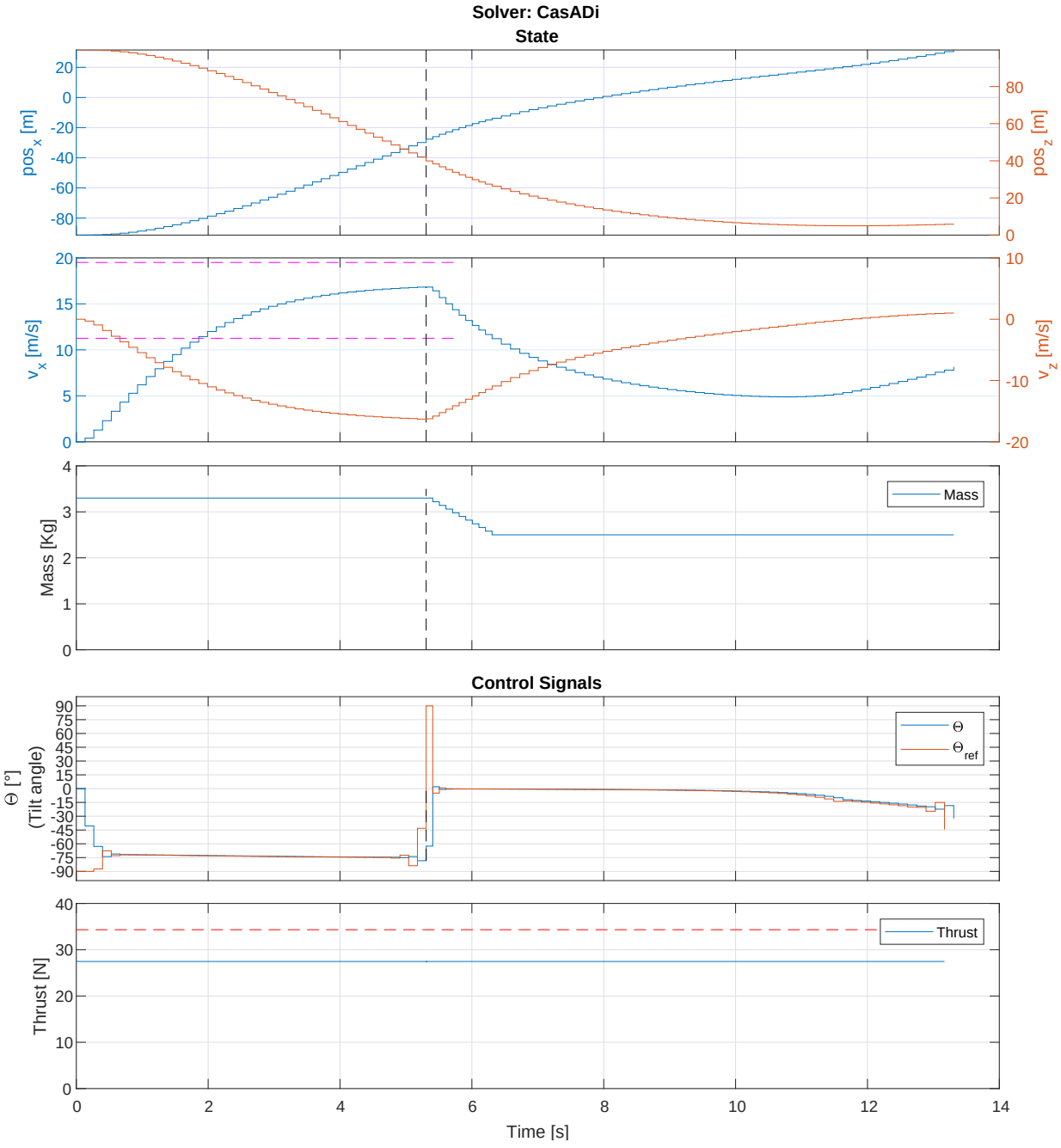


Figure 9: The state and control signals of the vehicle through a trajectory for the *pump* payload releasing model. The lower and higher magenta dashed lines on the velocity graph represent the *max no descent* speed and the maximal speed achievable at $\theta = 2\pi$ respectively

.

- $\|\nabla_{terrain}\| = 0.6$ (approx. 31°), Oriented to the north.
- $\|\mathbf{v}_{wind}\| = 0$ m/s.

All constraints and cost weights were the same as in section 4.1. Both releasing models were tested (*pump* and *immediate release*) and there were no significant differences between their results.

For the *immediate release* payload releasing model, the horizontal speed at the release time was 17.09 m/s, 52% above $v_{max\_no\_descent\_loaded}$ and the release altitude was 20.14 m.

For the *pump* release mode, the horizontal speed at the release time was also 17.07 m/s, 52% above $v_{max\_no\_descent\_loaded}$ and the payload was released between altitudes 23.19 m and 8.59 m.

Also in this simulation, the similarity between releasing models is noticed. It can be appreciated observing Fig. 12 and 11 that neither the state variables nor the control signals have any major differences between the release models other than the mass variation. However, in both cases, the optimizer converged to a heading angle of $\psi = 180°$ which is exactly opposed to the gradient, i.e. approaching the target downhill, as it can be clearly appreciated in Fig. 10.

4.3 Simulation #3 (steep terrain gradient, no wind)

This simulation was run using the following environmental parameters:

- $\|\nabla_{terrain}\| = 1.2$ (approx. 50°), Oriented to the north
- $\|\mathbf{v}_{wind}\| = 0$ m/s

All constraints, and cost weights were the same as used in section 4.1. Both releasing models were tested (*pump* and *immediate release*) and there were no significant differences between their results.

For the *immediate release* payload releasing model, the horizontal speed at the release time was 17.22 m/s, 52% above $v_{max\_no\_descent\_loaded}$ and the release altitude was 17.97 m. The optimal heading angle to perform the trajectory was found to be $\psi = 226.49°$, i.e. 46.49° from straight-downhill.

For the *pump* release model, the horizontal speed at the release time was also 17.22 m/s, 52% above $v_{max\_no\_descent\_loaded}$ and the payload was released between altitudes 19.82 m and 4.16 m. The optimal heading angle to perform the trajectory was found to be $\psi = 137.56°$, i.e. 42.44° from straight-downhill.

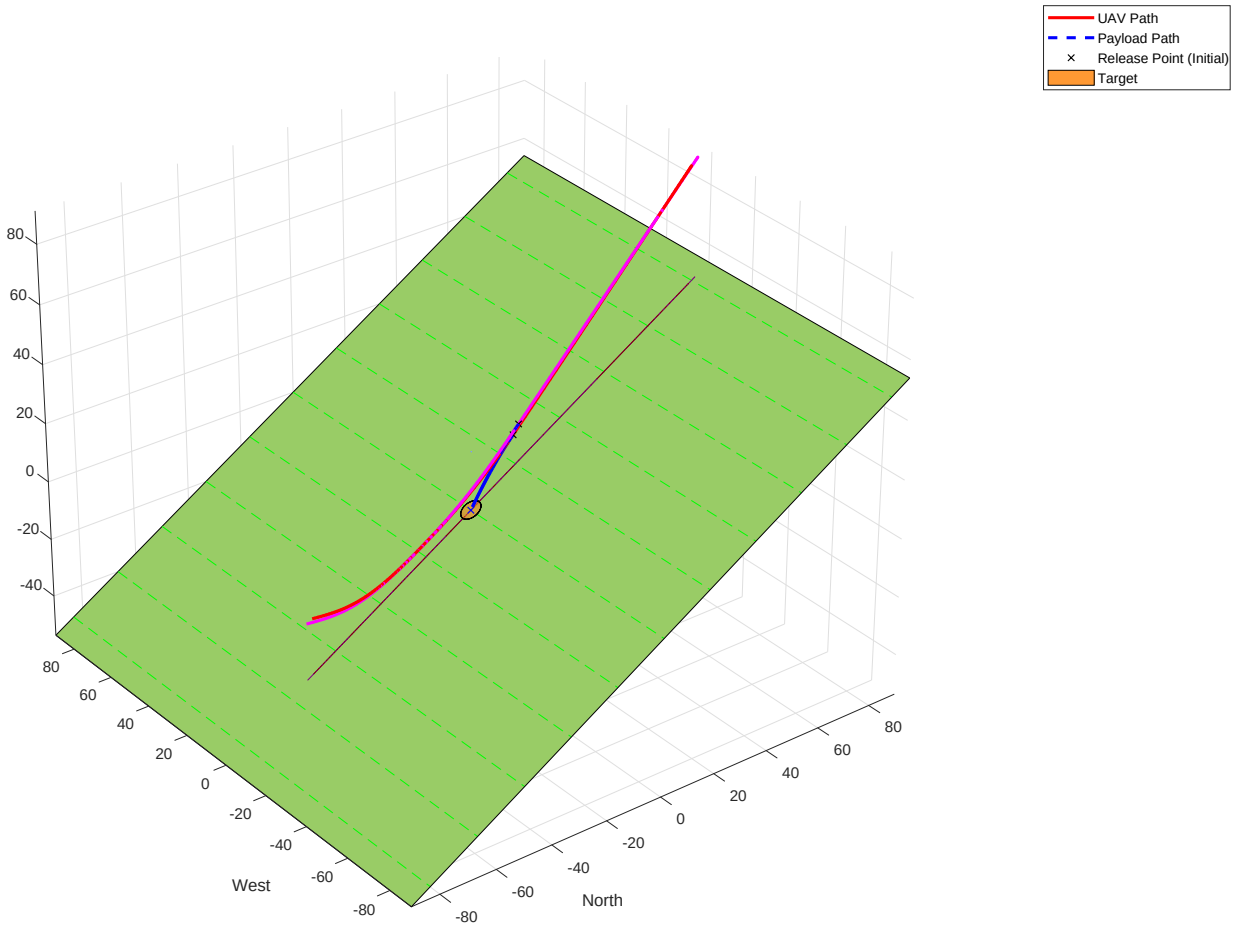


Figure 10: Trajectory followed by the UAV on both releasing model (red-immediate release and magenta-pump).It can be seen that they do not differ much except for the release point. Even the heading angle is almost the same.

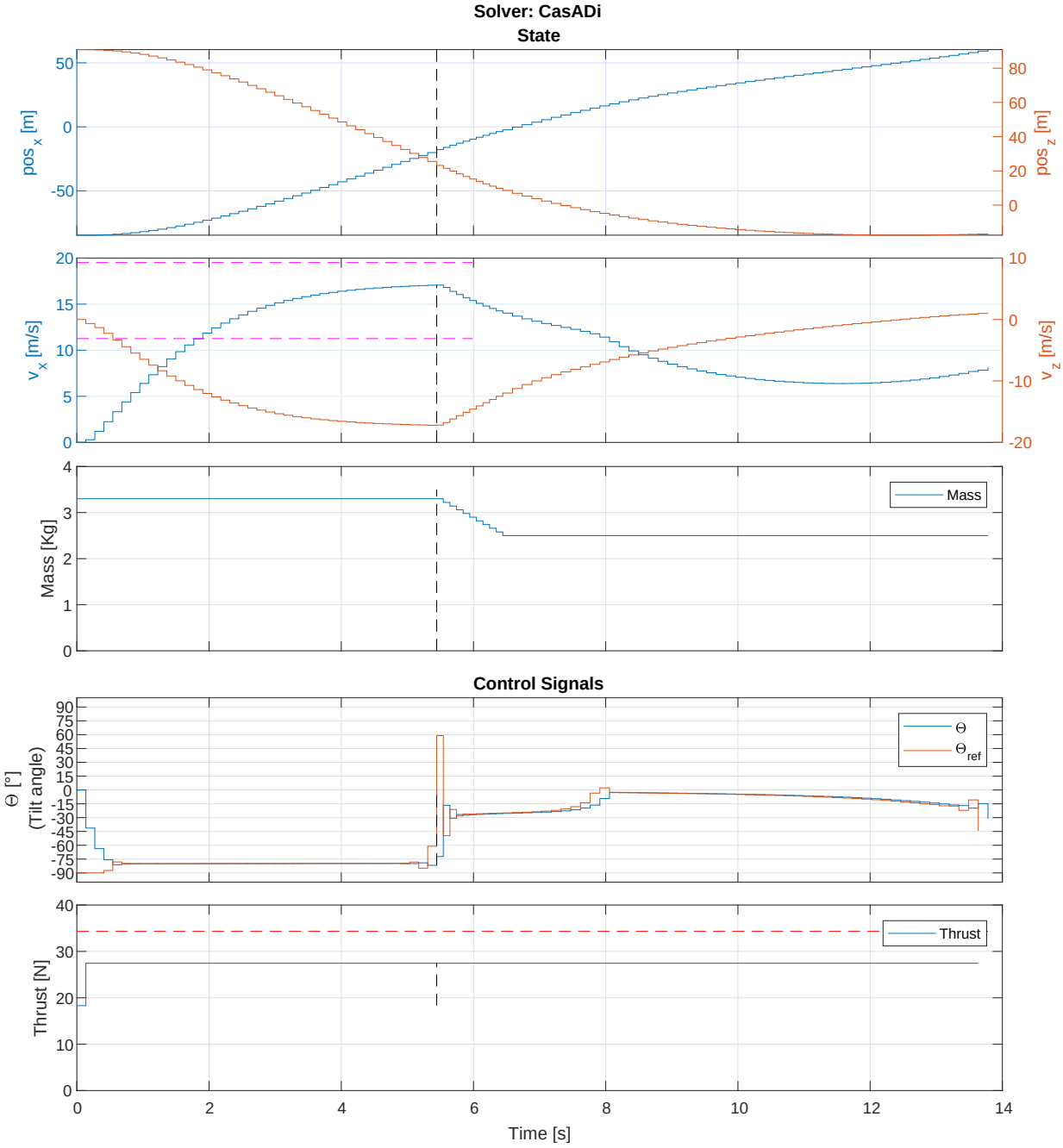


Figure 11: The state and control signals of the vehicle through a trajectory for a *pump* payload releasing model. The lower and higher magenta dashed lines on the velocity graph represent the *max no descent* speed and the maximal speed achievable at $\theta = 2\pi$ respectively.

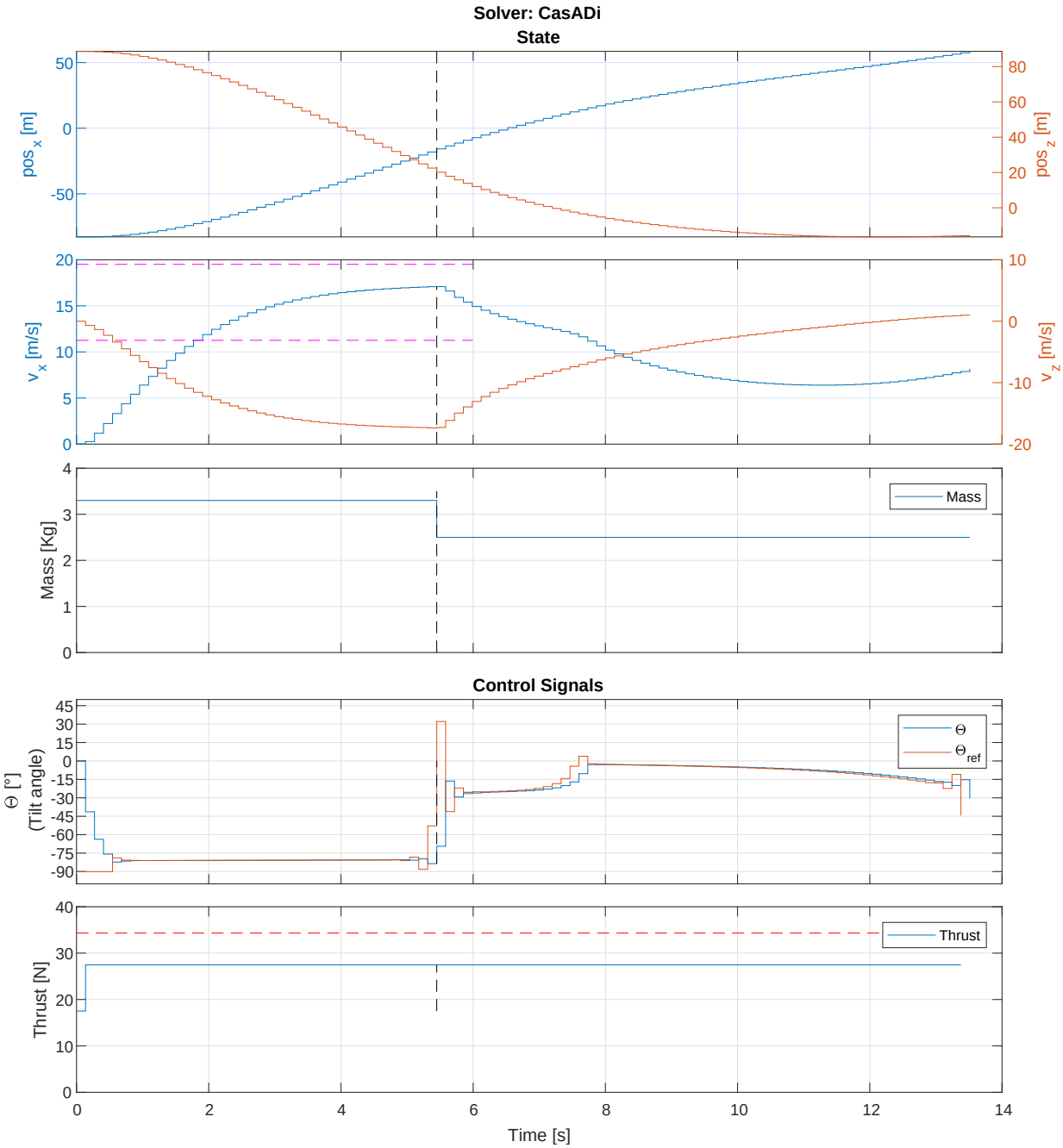


Figure 12: The state and control signals of the vehicle through a trajectory for the *immediate release* payload releasing model. The lower and higher magenta dashed lines on the velocity graph represent the *max no descent* speed and the maximal speed achievable at $\theta = 2\pi$ respectively.

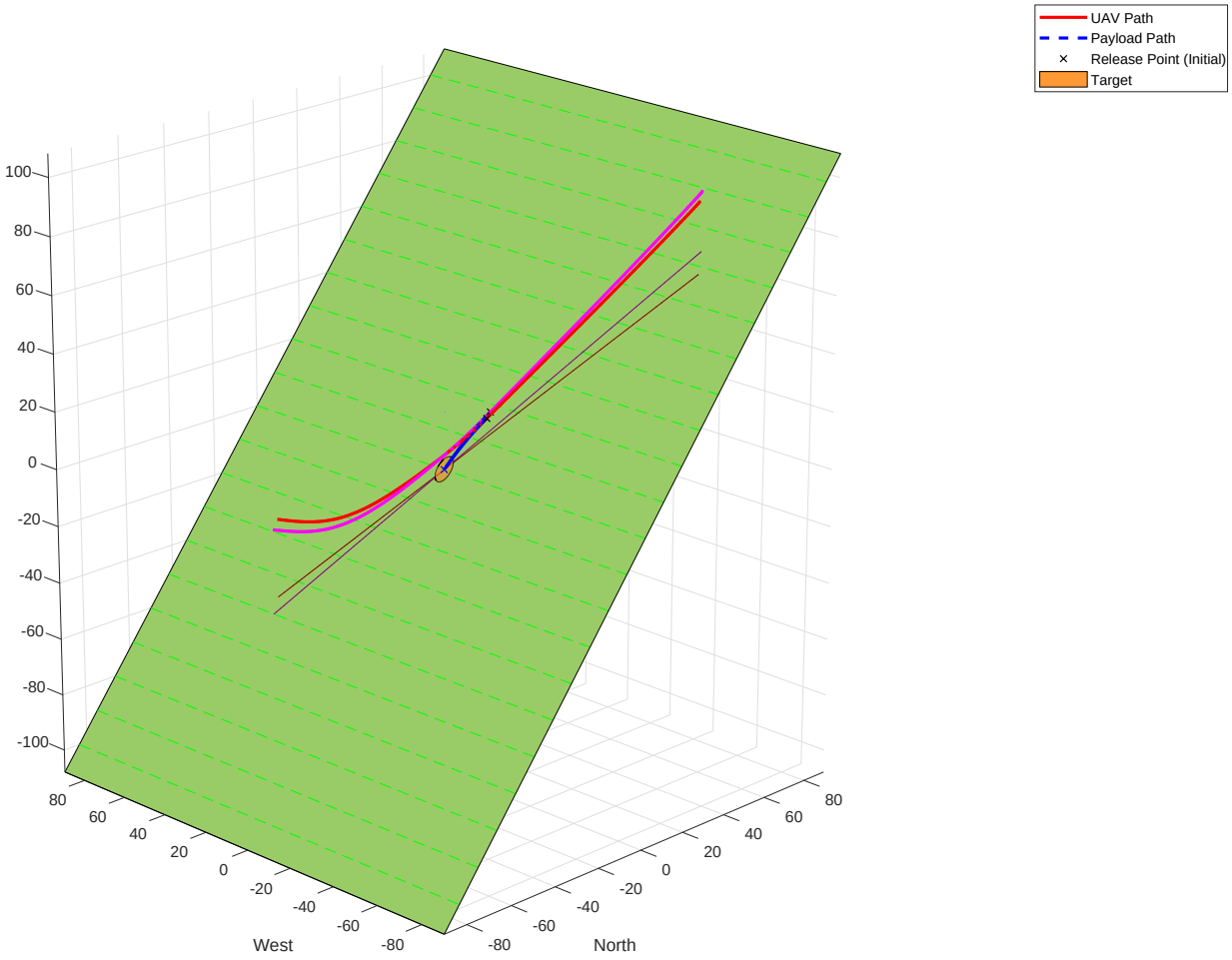


Figure 13: Trajectory followed by the UAV on both releasing model (red-immediate release and magenta-pump). It can be seen that their trajectories do not differ much except for the release point. The heading angle is slightly different.

In this case it can be observed that, for both releasing models, the optimized did not approach the target in a heading opposite to the gradient but in a diagonal manner. The projection of the terrain gradient vector into the heading is approx. 0.7 meaning that there is a limit to the advantage of approaching the target downhill. In addition it can be noted that both models reached different local minima heading-wise; i.e. both models chose a heading approx. 45° from the anti-gradient, as can be clearly appreciated in Fig. 13. It must be noticed that the fact that both trajectories cut the gradient in the same angle is due to random causes. They could have been in mirrored angles with respect to the gradient. Nevertheless, also in this simulation, the similarity between releasing models is noticed. It can be appreciated observing Fig. 15 and Fig. 14 that neither the state variables nor the control signals have any major differences between the release models other than the mass variation.

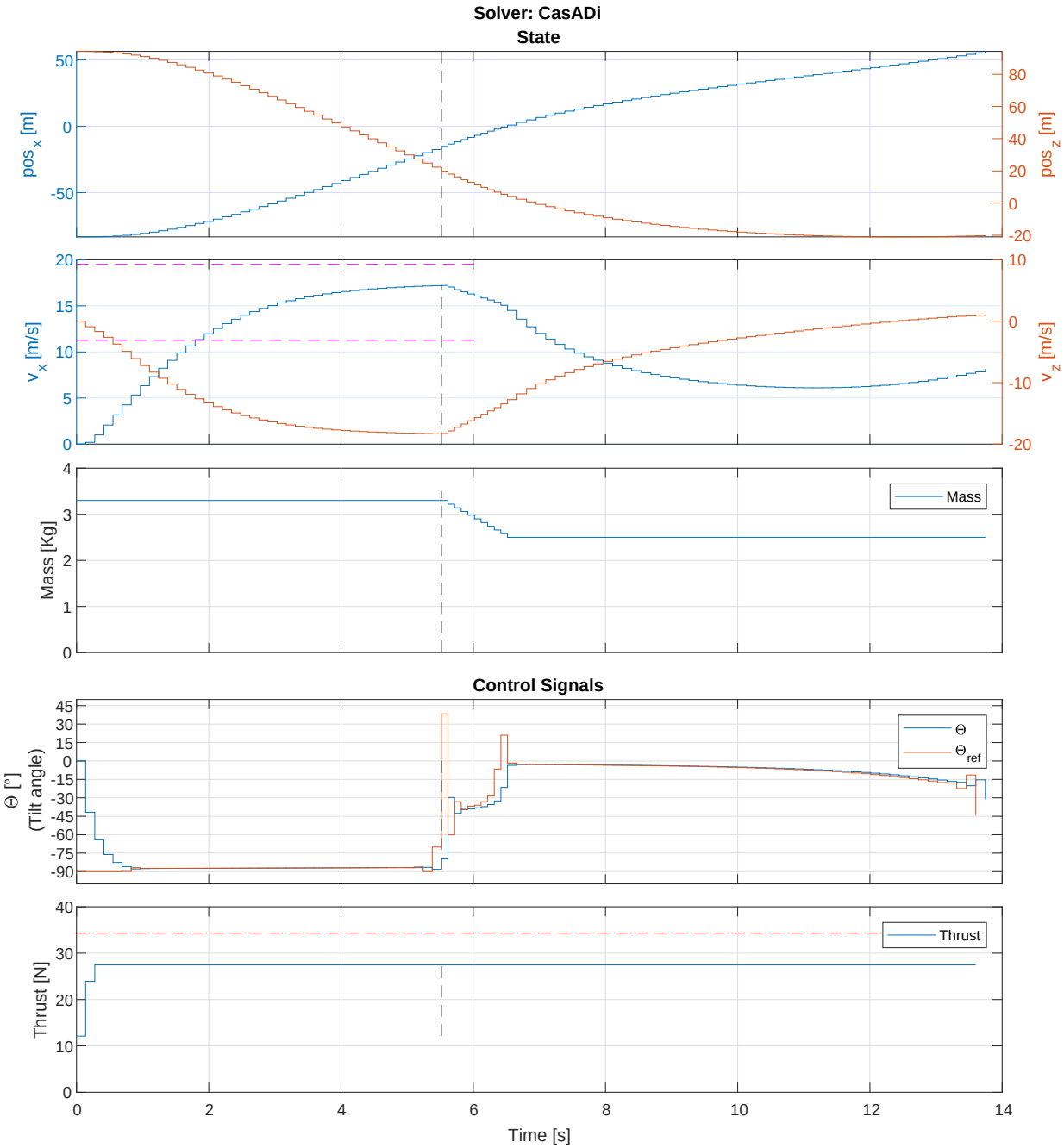


Figure 14: The state and control signals of the vehicle through a trajectory for the *pump* payload releasing model. The lower and higher magenta dashed lines on the velocity graph represent the *max no descent* speed and the maximal speed achievable at $\theta = 2\pi$ respectively.

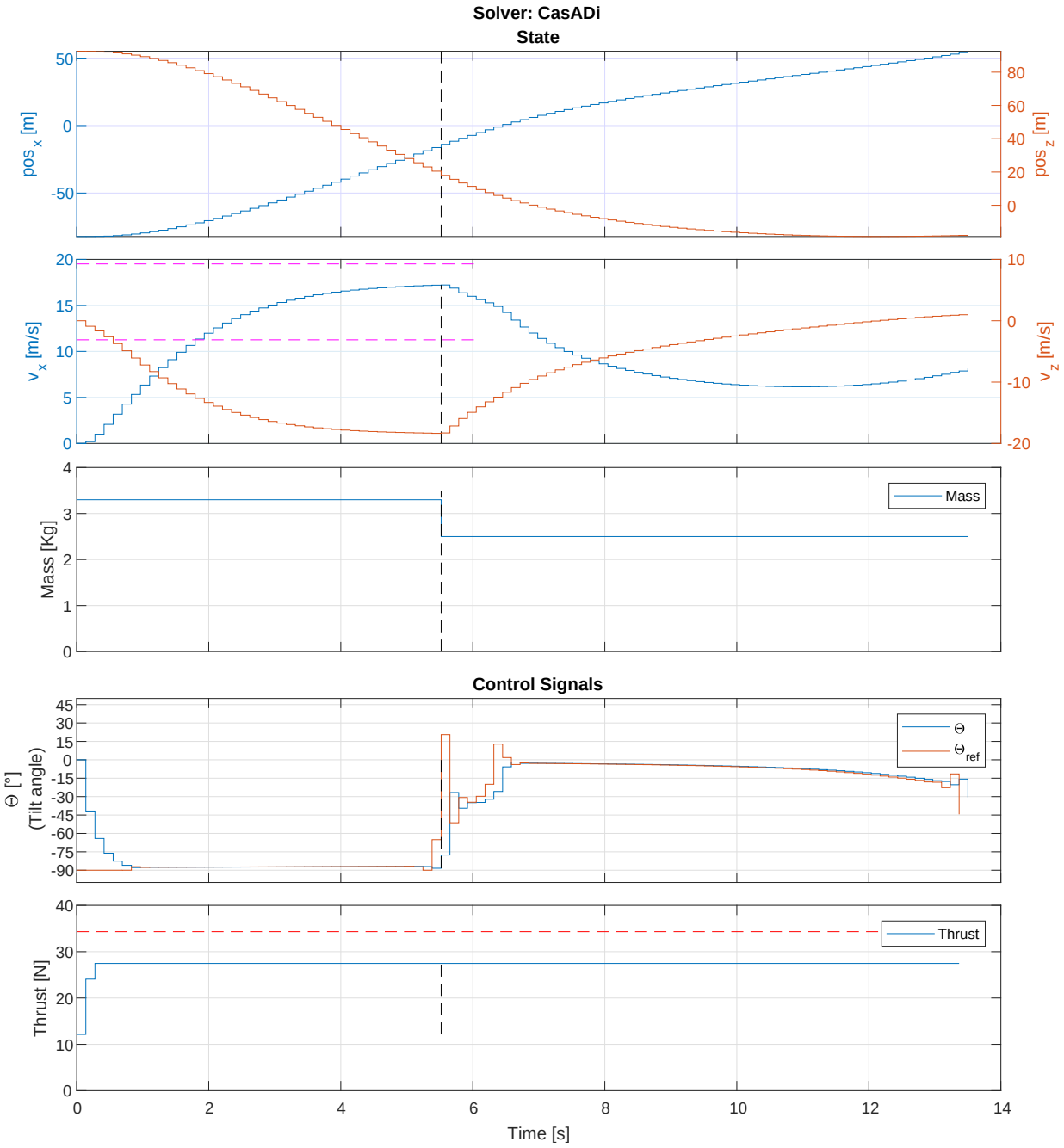


Figure 15: The state and control signals of the vehicle through a trajectory for the *immediate release* payload releasing model. The lower and higher magenta dashed lines on the velocity graph represent the *max no descent* speed and the maximal speed achievable at $\theta = 2\pi$ respectively.

4.4 Simulation #4 (no terrain gradient, moderate wind)

This simulation was run using the following environmental parameters:

- $\|\nabla_{terrain}\| = 0$
- $\|\mathbf{v}_{wind}\| = 5$ m/s from the East (90°)

All constraints, and cost weights were the same as used in section 4.1. Both releasing models were tested (*pump* and *immediate release*) and there were no significant differences between their results. For the *immediate release* payload releasing model, the horizontal speed at the release time was 21.62 m/s, 92% above $v_{max\_no\_descent\_loaded}$ and the release altitude was 35.86 m.

For the *pump* payload releasing model, the horizontal speed at the release time was also 21.55 m/s, 92% above $v_{max\_no\_descent\_loaded}$, and the payload was released between altitudes 39.75 m and 26.20 m.

For both releasing modes, the optimal heading angle to perform the trajectory was found to be $\psi = 270°$, i.e. exactly with the same direction of the wind, which maximized the speed at the release point, as it can be clearly appreciated in Fig. 16.

As well as in the previous simulations, the similarities between both releasing models are obvious. It can be appreciated observing Fig. 18 and 17 that neither the state variables nor the control signals have any major differences between the release models other than the mass variation. It must be noted that the achieved speed was approximately the same as in section 4.1, with the wind speed added.

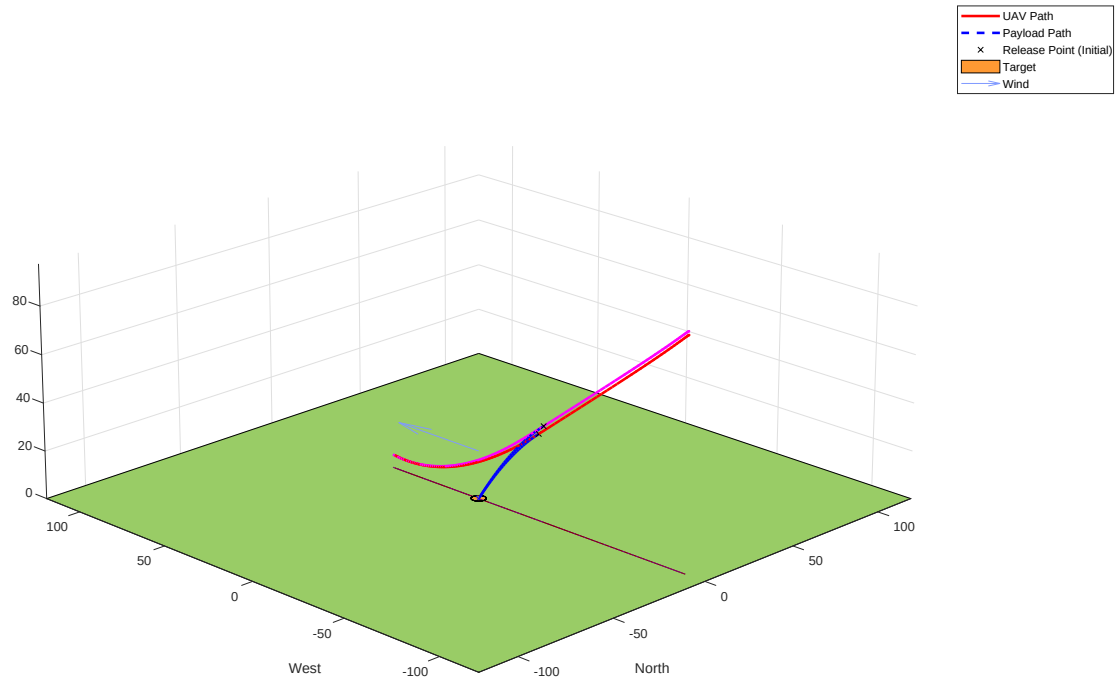


Figure 16: Trajectory followed by the UAV on both releasing model (red-immediate release and magenta-pump).It can be seen that their do not differ much except for a a slight difference in the release point.

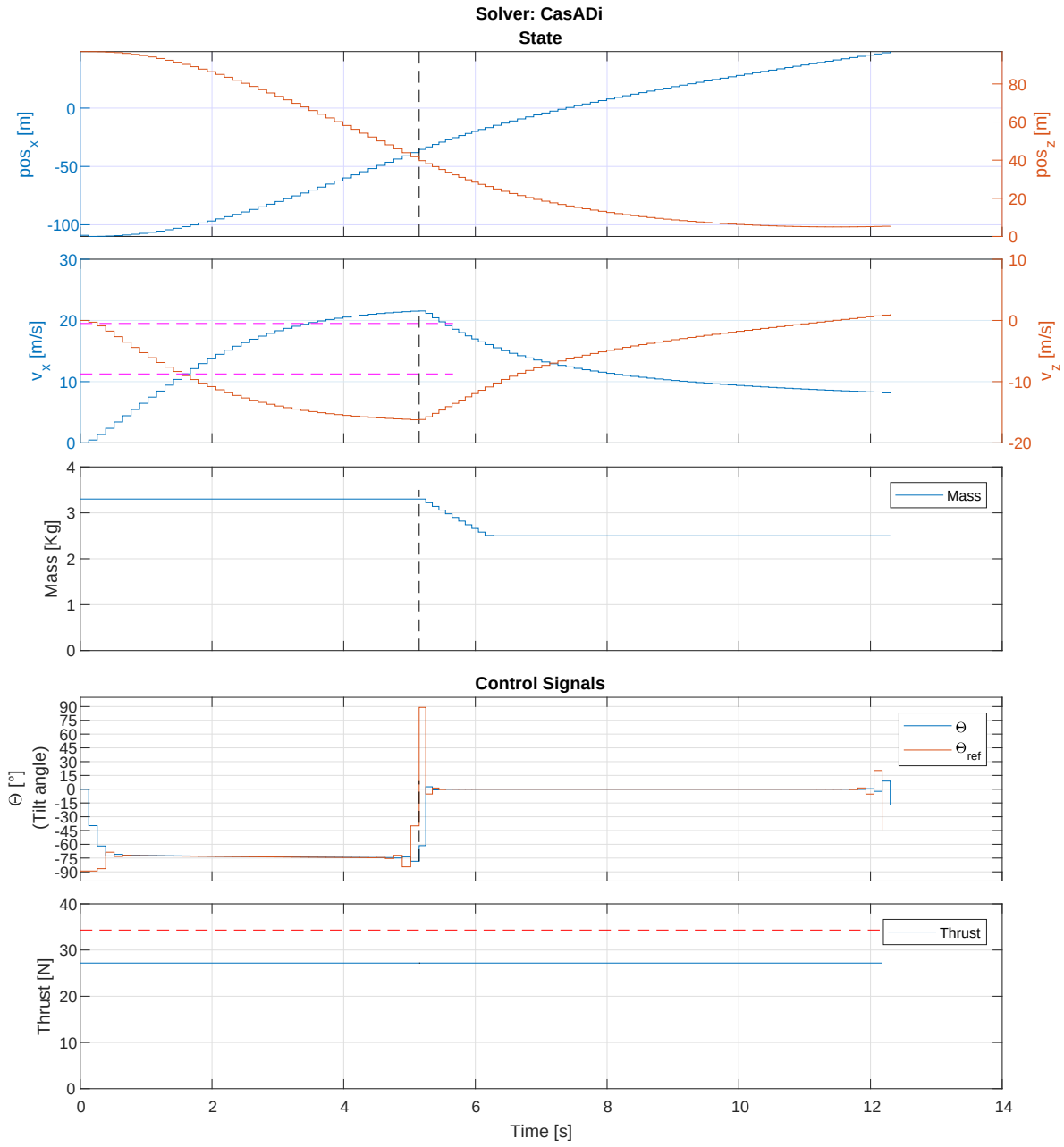


Figure 17: The state and control signals of the vehicle through a trajectory for the *pump* payload releasing model. The lower and higher magenta dashed lines on the velocity graph represent the *max no descent* speed and the maximal speed achievable at $\theta = 2\pi$ respectively. Note that due to tail wind the reached speed was higher than the

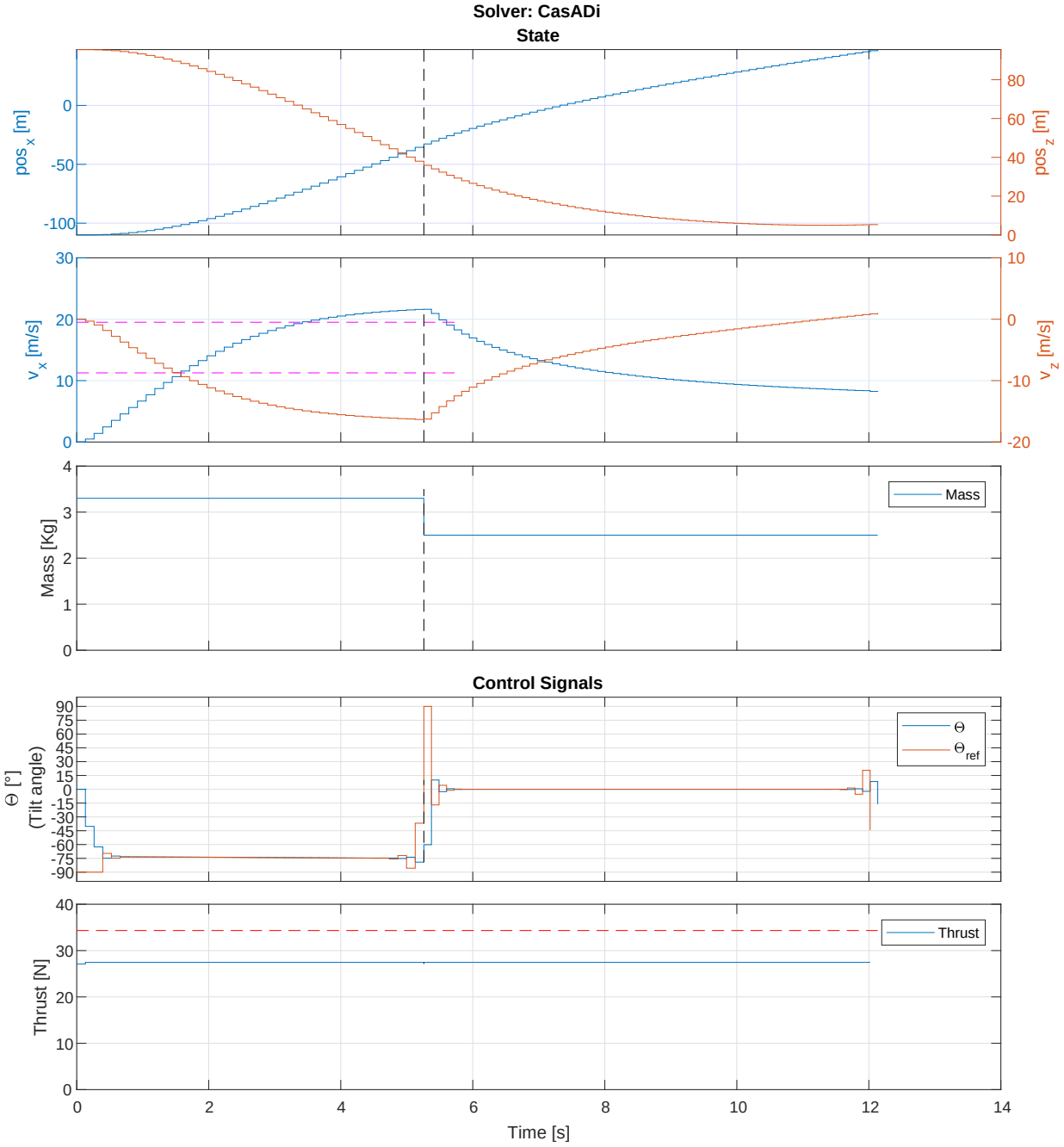


Figure 18: The state and control signals of the vehicle through a trajectory for the *immediate release* payload releasing model. The lower and higher magenta dashed lines on the velocity graph represent the *max no descent* speed and the maximal speed achievable at $\theta = 2\pi$ respectively

4.5 Environmental Parametric Sweep Simulations

In the previous simulations, the effects of gradient and wind on the optimal solution were tested, each one separately. In general, it can be stated that the optimal solution tends to yield a heading opposite to the terrain gradient direction and in the same direction of the wind. However, due to the endless combinations of possible simulation scenarios, it would require an inconvenient amount of simulations to properly describe the way the optimizer finds the best heading angle. Specially interesting cases would be those where these two environmental constraints collude with each other in some degree. The following simulations will attempt to provide an insight of the effects of wind and terrain gradient when the optimizer has to trade off between both criteria. Due to the highly non-convex nature of the problem the solution for similar environmental and mechanical parameters may vary depending on the initialization values. For example, there may be symmetrical solutions relatively to the terrain gradient if wind is parallel or almost parallel to the terrain gradient. Another example would be the lack of both slope and wind, in which case the heading angle would not have any effect in the cost function and thus, any heading angle would be an optimal solution; as demonstrated in section 4.1. In addition, the computation times vary considerably (about two orders of magnitude) for hot or cold starts. The same computation time variance can be observed between hot-started optimizations, depending on the magnitude of the variations in the environmental and mechanical parameters between runs.

4.5.1 Parametric Sweep - Terrain Gradient Magnitude

In this simulation, depicted in Fig. 19, a moderate wind of 3 m/s was modeled originating at 135° (flowing to 315°, i.e. slightly uphill on purpose) whereas the magnitude of the terrain gradient vector $\|\nabla_{terrain}\|$ (pointing to 0°) was increased from 0 to 4. The results show that for lower terrain gradients the optimizer favors approaching the target from the same direction as the wind (Fig. 20a) but as the gradient increases, it opts for favoring more the downhill direction rather than the wind(Figs. 20b, 20c). Eventually, for a higher gradient magnitude, due to the increase in the vertical component of the wind (described in eq. (25)), the heading angle begins to align with the wind direction until finally, for high enough gradient magnitudes ($\|\nabla_{terrain}\| > 2.2$, meaning the terrain is almost vertical), the trajectory although still feasible, loses its characteristic *dive-bombing* shape (Fig. 20d).

4.5.2 Parametric Sweep - Wind Speed

In this simulation, depicted in Fig. 21, the effect on varying wind speed can be appreciated. The terrain gradient was set to 0.5 towards the north. For low wind speeds,

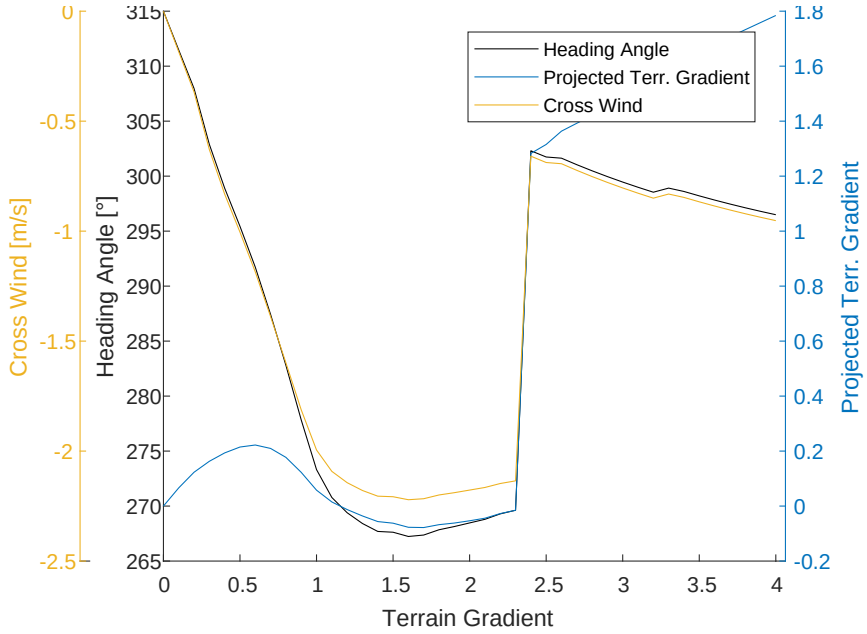
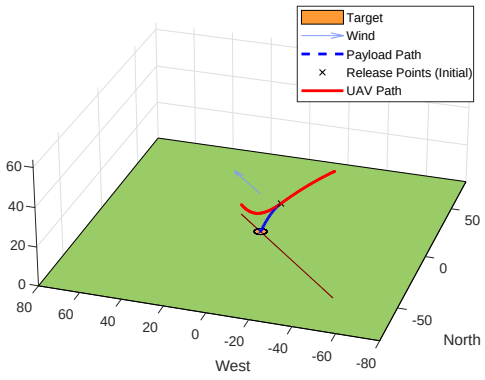
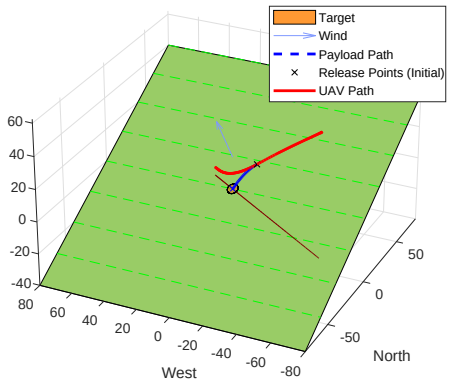


Figure 19: The effect of increasing the terrain gradient magnitude on the resulting heading angle found by the optimizer, for wind 3 m/s flowing 45° apart from the terrain gradient.

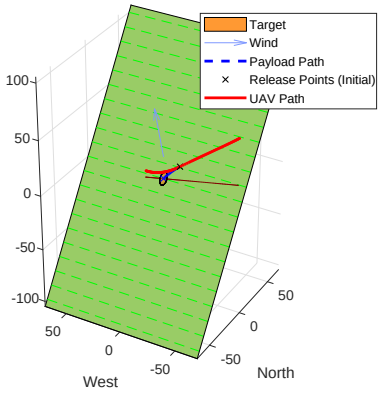
the optimizer chooses an approach direction opposite to the terrain gradient, i.e. 180° (Fig. 22a), but as the wind speed $\|\mathbf{v}_{wind}\|$ increases, it overweights the effect of the terrain gradient (Figs. 22b, 22c). For $\|\mathbf{v}_{wind}\| > 1.5$ m/s, the influence of the wind grows up to the point that the UAV chooses to approach the target uphill. As long as the wind speed increases further, the heading vector $\mathbf{H}_{DG}$ becomes almost perfectly aligned with the wind velocity vector $\mathbf{v}_{wind}$ (Fig. 22d).



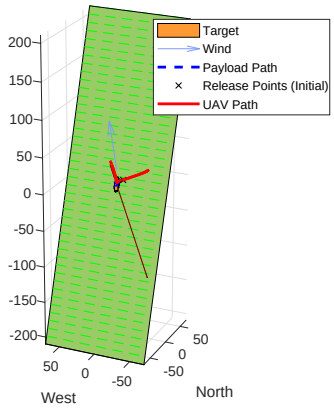
(a) $\|\nabla_{terrain}\| = 0$



(b) $\|\nabla_{terrain}\| = 0.5$



(c) $\|\nabla_{terrain}\| = 1.5$



(d) $\|\nabla_{terrain}\| = 3$

Figure 20: Resulting trajectories for various terrain gradient magnitudes. The wind remains unchanged proceeding from 135° at a speed of 3 m/s.

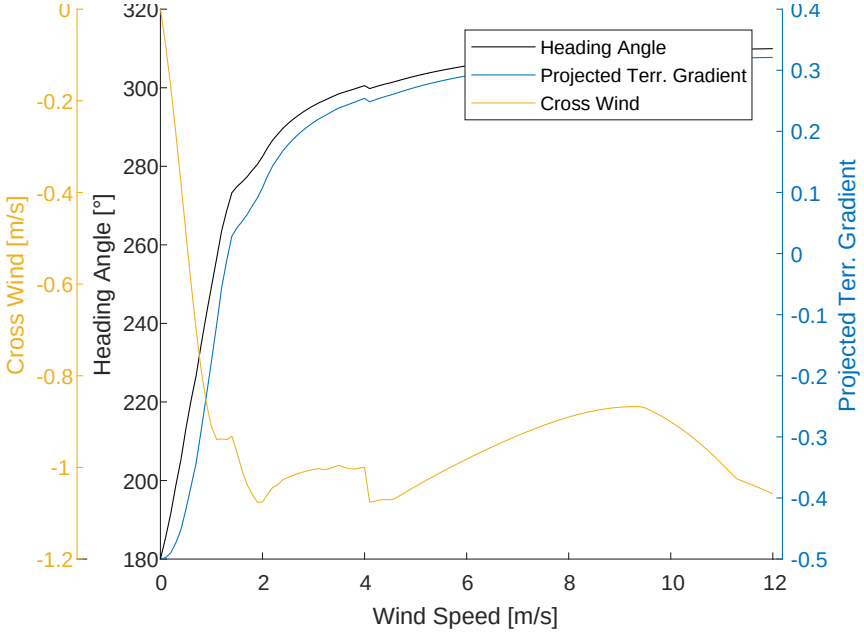
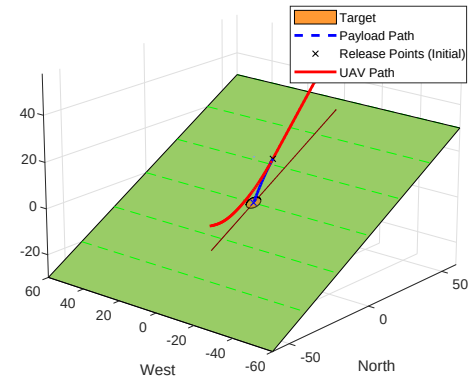
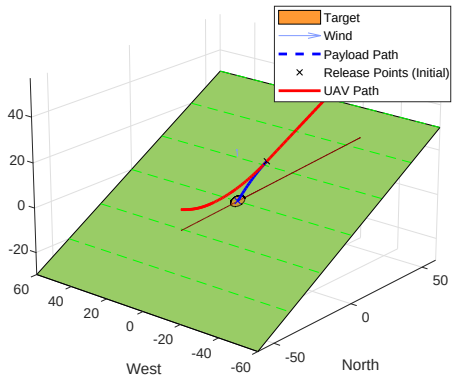


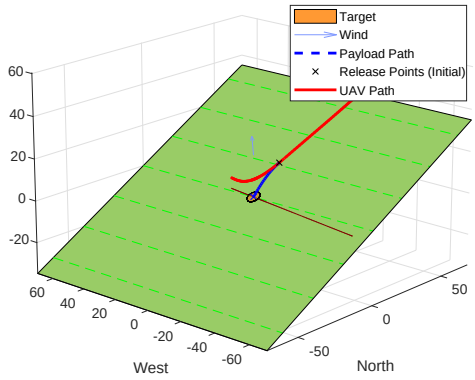
Figure 21: The effect of increasing the wind speed on the choice of heading angle by the optimizer, given that the wind is flowing at 45° from the terrain gradient which has a magnitude of $\|\nabla_{terrain}\| = 0.5$.



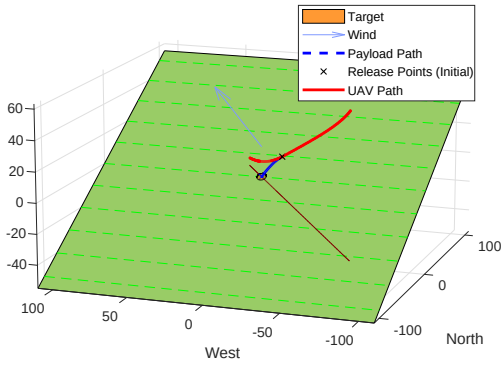
(a) $\|\mathbf{v}_{wind}\| = 0$ m/s



(b) $\|\mathbf{v}_{wind}\| = 0.5$ m/s



(c) $\|\mathbf{v}_{wind}\| = 1.5$ m/s



(d) $\|\mathbf{v}_{wind}\| = 8$ m/s

Figure 22: Resulting trajectories for various wind speeds. The wind direction remains unchanged proceeding from 135° and $\nabla_{terrain}$ remains fixed as well at 0.5 pointing north.

4.6 Trajectory Generation Time

Several factors influence the overall time that takes to generate the trajectories for the maneuvers. Naturally, the more variables to optimize the more time it takes to converge. That can be affected also by the trajectory resolution. In addition the more complex the model becomes the more non-convex the cost function gets. Trying to account for more phenomena (e.g. loss of forward thrust due to crosswind) comes with a cost in real-time terms.

Ambiguity in the variables can have a specially large impact in the convergence time. For example, if the terrain gradient is zero, and there is no wind (as simulated in 4.1), then $\mathbf{H}_{DG}$ does not affect the cost function and consequently, it prolongs to convergence time. In fact, the difference in the convergence time can be 3 orders of magnitude. But even if the terrain has a gradient, still if the wind is zero or parallel to it, the cost function would be the same, regardless of the oblique heading on which the maneuver would cross these vectors (to the right or to the left). A possible solution for it would be to penalize some aspect of the heading angle such as the X axis component (north) to the cost function with a small weight that would not affect it. That could be interpreted as "if any heading angle is the same, then choose north". Last, the hot or cold start aspect regarding the initialization values of the variables. If the overall difference between the initialization and final values is small, so will be the time it will take the optimizer to converge.

Table 2 shows the convergence times for some optimization example scenarios. The optimizer was run in an Intel "Core i7-6820HQ" processor which has 8 cores running at 2.7GHz. The running operating system is Windows 7. By comparing rows 9 and 10, it becomes evident that the optimization time can be reduced by about two orders of magnitude, just by initializing the trajectory to values close to the optimal solution.

It can also be seen that if the heading variable is confined to a given value, and it cannot be optimized the optimization time does not differ from other cases where the existing terrain gradient and wind help the optimizer to converge faster.

The amount of time steps N has a direct effect on the optimization time. This can be seen by comparing rows 1, 5, 7 and 8. Given the finite nature of the problem, a shrinking horizon model for real-time control (SHMPC) could be more appropriate than a receding horizon one. It can be expected then that as long as the UAV comes closer to the releasing point the optimizer will work faster enabling real-time control when it becomes more critical.

Test#	Optimization Conditions	Conv. Time [s]	#Iter.
1	Wind and terrain gradient, pump release, hot start	0.88	18
2	Wind and terrain gradient, pump release, cold start	12.24	257
3	Wind and terrain gradient, immediate. release, hot start	0.78	20
4	Wind and terrain gradient, imm. release, cold start	14.85	383
5	Wind and terrain gradient, imm. release, hot start, 2 x resolution	2.92	42
6	Wind and terrain gradient, imm. release, cold start, 2 x resolution	422.64	409
7	Wind and terrain gradient, imm. release, hot start, 1/2 x resolution	0.36	16
8	Wind and terrain gradient, imm. release, hot start, 1/4 x resolution	0.15	12
9	No wind, no terr. gradient, pump release, cold start	1638.38	35094
10	No wind, no terr. gradient, pump release, hot start	62.66	1291
11	No wind, no terr. gradient, imm. release, hot start	69.96	1952
12	No wind, no terr. gradient, imm. release, cold start	571.53	15583
13	No wind, no terr. gradient, imm. release, hot start, heading fixed	0.80	21
14	No wind, no terr. gradient, imm. release, cold start, heading fixed	10.65	265

Table 2: The time and amount of iterations it takes to CasADi to solve the OCP and generate a trajectory. The optimizer was run in an Intel "Core i7-6820HQ" processor which has 8 cores running at 2.7GHz.

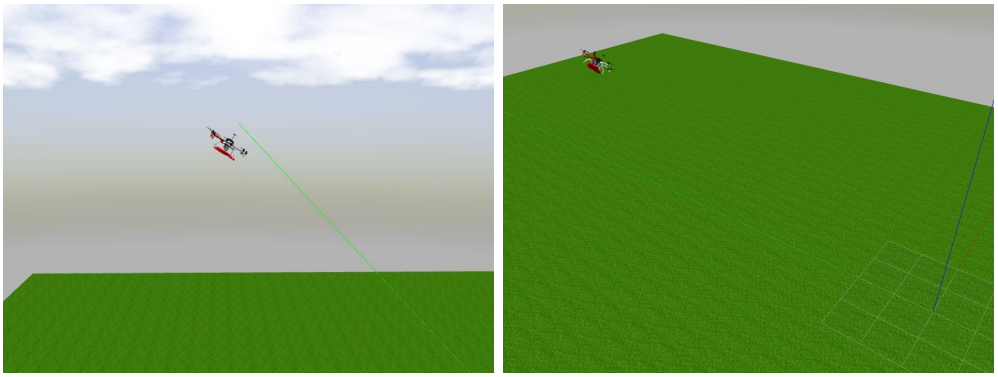
5 Implementation

The implementation phase aims to bring in the theoretical concept to reality. This requires several tasks such as programming dedicated ROS nodes or re-tuning the on-board controller. To achieve that, online simulations can be performed in *'Gazebo'*[2].

Gazebo runs on a PC and is connected to a ROS instance that can run all the lower control layers with the same algorithms that run in reality, and simulate more accurately the scenario. On Gazebo, however, due to technical limitations, only a scenario with flat terrain and without wind was tested. Simulating the maneuver in Gazebo is in fact, very close to reality in the aspect that the modules controlling the virtual UAV are exactly the same as in reality. This allows problems, that may arise when performing the real maneuver, to be spotted without any risk, and to visualize the whole procedure to better understand it such as depicted in Fig. 23.

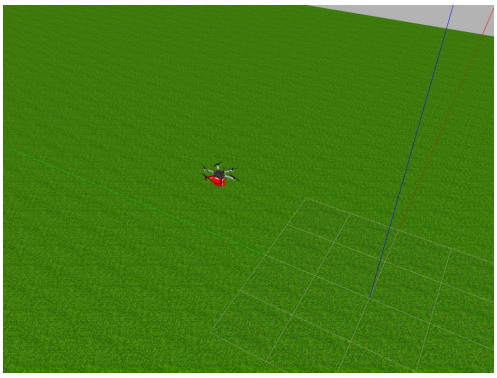
A successful simulation on gazebo is a crucial to validate the concept and proceed to real platform experiments given the inherent risk of the maneuver to the UAV integrity.

[2]A video containing the simulations performed in Gazebo can be found at http://mrs.felk.cvut.cz/mpc-with-time-variable-mass

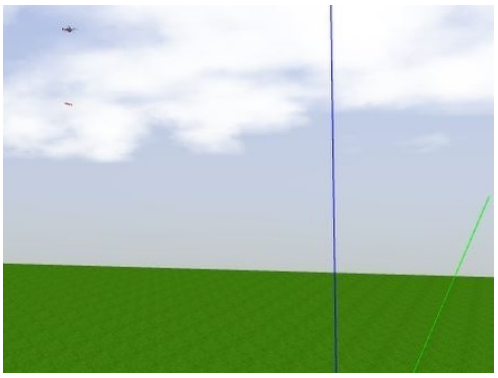


(a) Accelerating

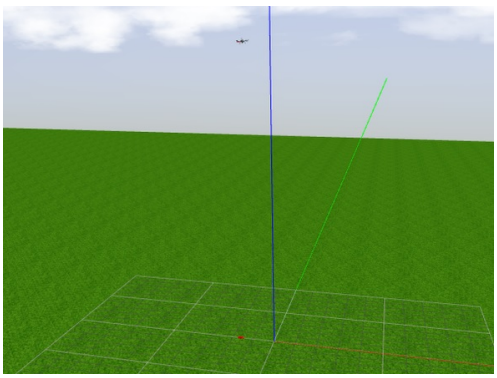
(b) Accelerating



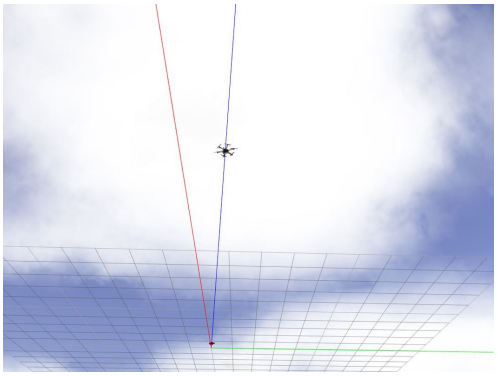
(c) Before dropping



(d) After dropping



(e) Recovering



(f) Recovering (view from below ground)

Figure 23: Visualization of the maneuver in Gazebo. In 23a, 23b and 23c, the payload is attached to the UAV. In 23d the UAV is recovering while the payload is falling. In 23f and 23f, the payload has fallen about 1 m after the axes origin while the UAV is recovering.

5.1 ROS Nodes, Topics and Services

Some changes had to be made in the ROS platform that runs on the UAV in possession of the MRS team. This consisted in developing two dedicated ROS nodes and embedding them within the existing platform.

5.1.1 Trajectory Generation ROS Node

In order to make the UAV follow a reference trajectory, which is generated by the optimizer; the state values (plus the accelerations, which can be derived from the velocities) must be fed to the on-board controller at a 100 Hz rate. It was decided that the trajectory will be exported to a CSV file and the values in it, interpolated in time so that they can be fed, to the on-board controller, which is in charge of maneuvering the UAV, at the mentioned rate. An example of such a file is shown in Fig. 24. A dedicated ROS node called *'CSV_tracker'*, whose function is to read the state values listed in the CSV file and feed them to the on-board controller was created. This node takes command of the UAV for the duration of the maneuver and resigns it upon its end.

5.1.2 Payload Releasing ROS Node

The second ROS node implemented was the *'Releaser'*, which controls the gripper that physically releases the payload. Normally, the control signals generated by the optimizer upon planning have little resemblance to those generated in real-time. That is due to disturbances and inaccuracies in the UAV model. Indeed, similarly to the other control signals (pitch angle and thrust) the releasing signal must be altered. In this case, due to differences between the reference trajectory and the performed one, and also due to the fact that there is a considerable delay between the generation of the *release* signal and the actual physical releasing of the payload. In order to deal with those two conveniences, the *releaser* iteratively and constantly performs two real-time calculations:

a. Where would the payload fall if requested to be released at this moment?

b. Would the UAV crash into the terrain if a request to release the payload were not issued at this moment?

The term *'request to release'* means that for both calculations, in order to account for the delay in the releasing process, the releaser node extrapolates the state in time and bases these calculations on the predicted state of the UAV *after* the releasing delay. The state prediction in time is a simple linear extrapolation that is calculated only on the position and velocity components neglecting accelerations. It was empirically found that

```
-36.61 36.54 0.00  0.00 2.03 -1.81  0.00 27.46 0.00
-36.61 36.54 0.02 -0.02 2.03 -1.81 -0.06 27.46 0.00
-36.61 36.54 0.04 -0.04 2.03 -1.81 -0.12 27.46 0.00
-36.61 36.54 0.06 -0.05 2.03 -1.81 -0.18 27.46 0.00
-36.61 36.54 0.08 -0.07 2.03 -1.81 -0.24 27.46 0.00
-36.61 36.54 0.10 -0.09 2.03 -1.81 -0.30 27.46 0.00
-36.61 36.54 0.12 -0.11 2.03 -1.81 -0.36 27.46 0.00
-36.61 36.54 0.14 -0.13 2.38 -1.95 -0.42 27.46 0.00
-36.61 36.54 0.17 -0.15 4.35 -2.72 -0.47 27.46 0.00
-36.61 36.53 0.21 -0.17 4.35 -2.72 -0.49 27.46 0.00
-36.60 36.53 0.25 -0.20 4.35 -2.72 -0.51 27.46 0.00
-36.60 36.53 0.30 -0.23 4.35 -2.72 -0.53 27.46 0.00
-36.60 36.53 0.34 -0.26 4.35 -2.72 -0.55 27.46 0.00
-36.59 36.52 0.38 -0.28 4.35 -2.72 -0.57 27.46 0.00
-36.59 36.52 0.43 -0.31 4.35 -2.72 -0.59 27.46 0.00
-36.59 36.52 0.47 -0.34 4.46 -2.79 -0.61 27.46 0.00
```

Figure 24: An example of a CSV file (although the values are not properly separated by *commas*, as the *CSV* acronym suggests, but by *spaces* instead). Each column contains a series of values for a specific placeholder required by the on-board controller over time. There is a column for each of the state values which are: position (p_x and p_z), velocity (v_x and v_z), acceleration (which has to be derived from the velocity, also for x and z axes), pitch angle θ and thrust f. The last column is the *'release'* signal which was later overridden by an on-line algorithm described in section 5.1.2

the delay between the release command and the physical payload release is, in average, about 1.8 s. This delay varies between the real platform and the gazebo simulation, and there are also indications that it may vary slightly depending on the payload weight but no serious analysis of this behavior was conducted. The release position is thus given by

$$p_{i\_release} = p_i + \Delta t_{release} v_i, i \in \{x, y, z\}, \tag{28}$$

where p_i is the current UAV position in the i-th axis, v_i is the current velocity in the i-th axis and $\Delta t_{release}$ is the time delay between the release command and the physical payload release. The location where the payload would hit the ground can be calculated by eq. (16) and (17).

The question, whether or not the UAV would still be able to recover if the payload is not released, is needed because these trajectories are risky maneuvers designed on purpose with narrow safety margins. The UAV, upon releasing the payload, would be flying close to the terrain with a high negative vertical speed, and its available thrust may not be enough to recover from it on time. This node has the capability of predicting such a scenario by performing a series of calculations. These calculations consist in comparing the portion of the UAV kinetic energy due to the vertical speed, to the available residual *work*, i.e. the remaining thrust after discounting the UAV weight times the UAV height above the

terrain. In other words, if the statement

$$\frac{m_{UAV}v_z^2}{2} \geq (F_{th\_max\_plan} - m_{UAV}g)p_z \tag{29}$$

is *true*, then the UAV will certainly crash, unless the payload is released immediately. In order to prevent that outcome, this node has the authority to impose a payload release, at any time during the maneuver, even at the cost of not hitting the target. It will do so, if it determines that the danger of terrain collision is significant.

Another task this node is entitled with consists in updating the on-board controller about the change in the total mass after releasing the payload. It does so by publishing a ROS message.

5.2 Other Required Modifications

The on-board controller gains had to be re-adjusted (mostly increased) and safety switches had to be disabled in order to allow more aggressive maneuvers than those allowed for standard operation [21, 20, 25, 23, 24]. These changes should imply no serious risk because the designers of the controller demonstrated in their paper that "*... this controller is particularly useful for complex, acrobatic maneuvers of a quadrotor UAV, such as recovering from being initially upside down.*" [13].

During the simulations in Gazebo, differences in performance between runs were noticed. These differences seemed to be dependent on available computing power. Apparently, the reason was that both, ROS and the physical world simulation (Gazebo), ran both in real-time but the latter requires much more computational power. For example, the UAV could succeed in the maneuver or crash against the terrain depending solely on what computer the simulation was ran on; or whether real-time graphs were plotted or not. In addition, it was noted that the UAV aerodynamic model of the Gazebo simulator had some flaws. The main flaw affected the vertical airspeed v_{AS_z}. In reality, due to aerodynamic drag, the vertical airspeed v_{AS_z} should converge to a fixed value however, in the Gazebo simulations it kept rising to values much higher than expected. For those reasons, prior to the field test, and in order to reduce the risk of impacts on the ground, a dedicated *milder* trajectory was generated. This trajectory was generated after performing two modifications on the optimizer:

a. The aerodynamic drag coefficient *only* on the vertical axis, C_{D_z}, of eq. (3), was decreased by about 20% causing a correspondent reduction in the aerodynamic drag force F_{D_z} acting in eq. (2). That way, the optimizer *assumes* that the vertical speed reached during the maneuver should be higher and thus, it should *rely* less on the aerodynamic drag, generating a less extreme (and consequently safer) maneuver. This is especially

important prior to the first real tests in order to minimize the chances of damaging the UAV.

b. The weights of the cost function weight parameters W_{hr} and W_{vr} were modified to favor a low releasing height instead of high releasing speed. This way, the UAV should release the payload at a lower height, meaning that it should have less room to recover and consequently, the optimizer should plan the trajectory so that the UAV reaches lower vertical speeds.

Figure 25: An MRS team's *DJI FlameWheel550* UAV equipped with a magnetic gripper carrying a metallic disc as payload, flying close to a colorful pole which marks the target.

6 Testing and Results

The platform on which the algorithm was tested is based on a *"DJI FlameWheel550"* of six rotors which is also simulated in Gazebo. The UAV is equipped with wireless communication modules and several sensors such as a differential GPS, an IMU, cameras, a ground laser range-finder [14]. In addition, the UAV is equipped with a *NUC* computer which runs ROS and several proprietary nodes that control the UAV, the sensor fusion [4, 3, 6], and the other stacks such as logging and communication [26]. The used payload was a stack of metallic discs glued to each other. The payload weight was regulated by stacking more or less metallic discs. The releasing mechanism is an electro-magnet with high hysteresis that can hold the payload without constantly consuming energy [14]. The UAV with the gripping mechanism and the payload are depicted in Fig. 25. Because of that the more practical releasing model to test is the *immediate release* explained in Section 3.1.5.1.

6.1 Feasibility Tests

The first test was intended to confirm the feasibility of the maneuver. That includes the proper functioning of the on-board controller with the re-adjusted control gains, and the switching between the trackers. In order to reduce the risk of an accident the first test did not carry a payload to be released and the trajectories were slightly modified with some additional safety parameters.

These extra safety parameters were:

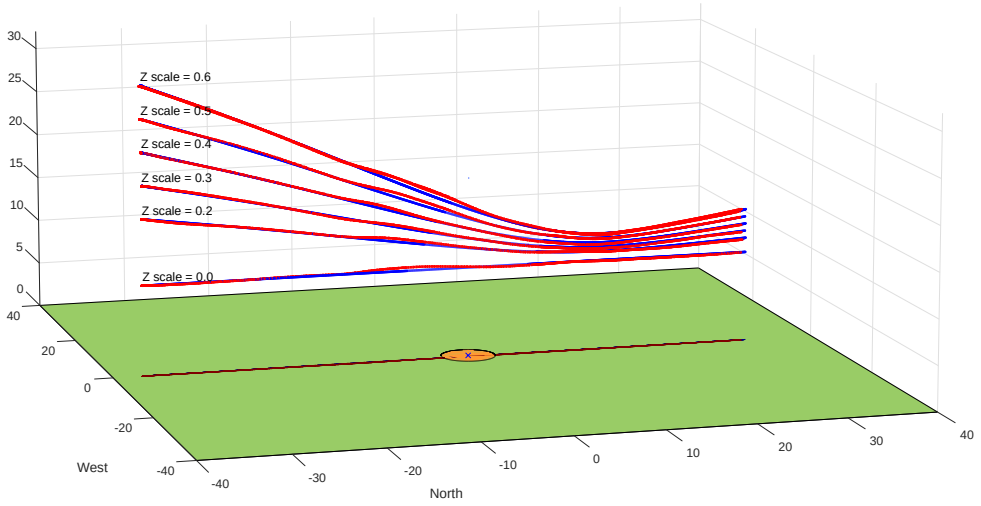


Figure 26: Progressively increasing the Z-scale factor from 0.0 to 0.6. An altitude offset of 10 m was added. The blue lines represent the reference trajectory and the red lined the performed one.

- An added offset in the trajectory altitude, to allow more time to recover in case of a malfunction.
- A down-scaling of all the positions and its derivatives in the Z axis. All the values in the trajectory corresponding to the Z axis were multiplied by a factor ranging between 0 and 1 to ensure milder maneuvers.

These parameters were gradually reduced until it was proven that the UAV would be able to handle the maneuver with a payload.

6.2 Payload Releasing Tests - Overview

The releasing tests consisted in allowing the UAV to perform the whole maneuver and assess the results based on the payload fall locations and the flight performance achieved.

Test#	Fall Position		**Payload**	Extra Safety Parameters		$v_{x\_max\_no\_descent}$	$v_{x\_release}$	**diff**
	X[m]	**Y**[m]	**Weight [kg]**	**Path Offset [m]**	**Path Scale Factor Z**	**[m/s]**	**[m/s]**	**[%]**
1	4.00	-10.21	0.55	10	0.9	11.56	10.93	-5.5
2	-3.90	-4.31	0.55	7	0.9	11.56	NaN	NaN
3	6.11	-2.01	0.70	7	1.0	10.25	11.38	11.0
4	4.01	3.26	0.70	2	1.0	10.25	11.88	15.9
5	3.64	4.01	0.70	0	1.0	10.25	12.33	20.3
6	-4.15	-0.03	0.86	0	1.0	8.81	11.70	32.7
7	-8.00	-0.71	0.86	0	1.0	8.81	12.96	47.1
8	-6.80	-0.98	0.86	0	1.0	8.81	13.22	49.9
9	-2.33	4.71	0.86	0	1.0	8.81	12.63	43.3
10	-6.23	2.14	0.86	0	1.0	8.81	12.49	41.6
11	4.72	3.59	0.86	0	1.0	8.81	11.56	31.1

Table 3: Payload Releasing Test parameters and Results. From test#6 and on, there are no extra safety parameters (the Z axis offset is zero, the Z axis scale is 1.0 and the weight is maximal).

Some examples of the UAV approaching the target can be seen in Fig. (27), and some examples of payload droppings can be seen in Fig. (28).

The UAV managed to follow the trajectory precisely. For all the release tests, the average maximum deviation from the reference path was 1.28 m. Apart from one single outlier, which shown a maximum deviation of 2.86 m, no deviations higher than 1.58 m were observed for any other test. An example can be seen in Fig. (29).

In total, eleven drop tests were performed[3]. The first five had gradually reduced extra safety measures (Z axis scale, Z axis offset) and lighter payload; whereas the last six had none. During the last six tests the UAV carried full weight payloads and the reference trajectory was not offset nor scaled down. The parameters and fall locations of the payload through the different tests are described in Table 3. The exact fall locations can be seen in Fig. (30).

6.3 Payload Releasing Tests - Statistics

The average fall location had an error of -0.81 m in the X axis and -0.05 m in the y axis. That is not surprising because the whole maneuver was carried parallel to the X axis and thus, the only thing that affected the free-fall trajectory of the payload in the Y

[3]A video containing a compilation of the payload releasing tests can be found at http://mrs.felk.cvut.cz/mpc-with-time-variable-mass

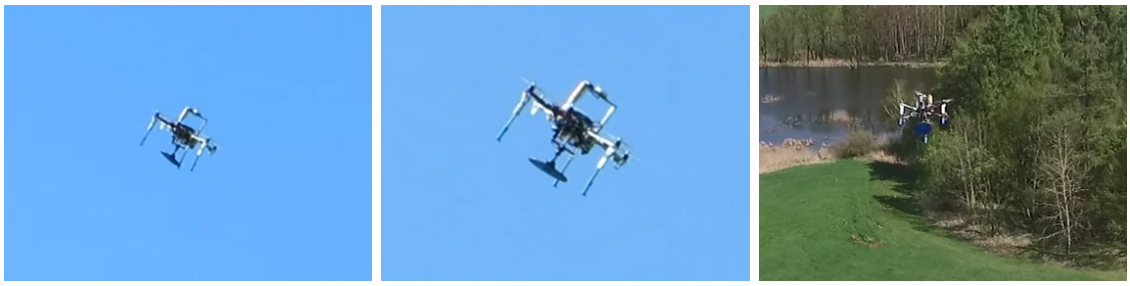


Figure 27: The UAV accelerating during the *approach* stage.

Figure 28: The UAV dropping the payload in flight. The target and the previous falling locations can be seen on the right side of the image at bottom-right.

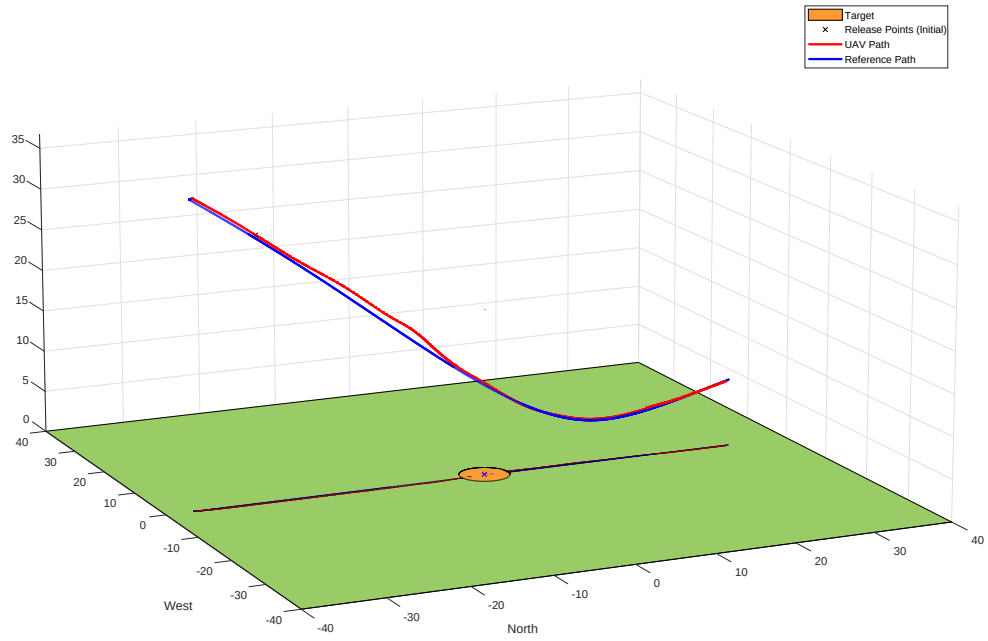
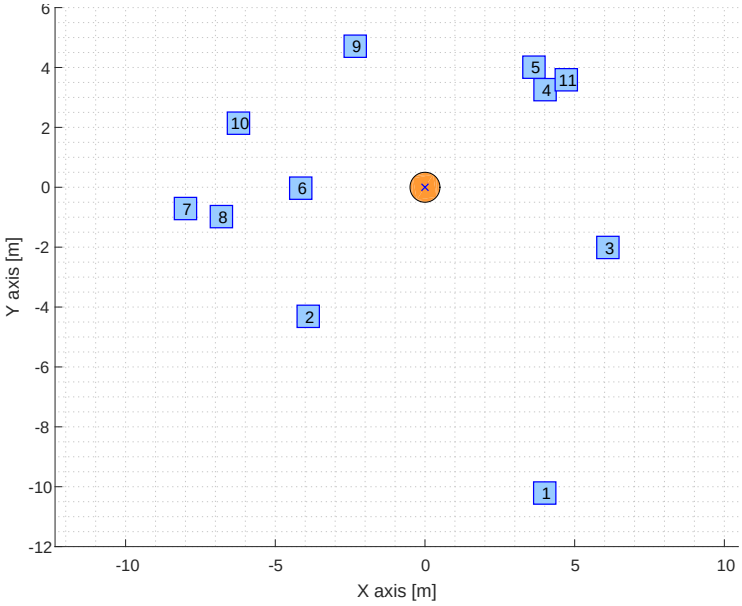


Figure 29: Reconstruction of a path followed by the UAV on a release test.

(a) Aerial photo of the payload falling locations. The large circle in the center that casts a shadow is the coordinates origin (where the target is). Tests 9, 10 and 11 do not appear.



(b) All the payload fall locations. Measured with a differential GPS

.

Figure 30: Payload releasing tests fall locations.

axis is the way the planned trajectory is carried out, but not the releasing point during the trajectory. The standard deviation for all the tests was 6.65 m meaning that for around 68% of the tests, the payload fell within a circle of the mentioned radius.

The first test was a clear outlier. Its parameters can be seen in Table 3, 2^{nd} row. It can be clearly seen on the test footage that the payload *'glided'* instead of following a ballistic trajectory[4]. The reason apparently was the payload shape (a very thin disc), which in combination with its relatively low weight (550 g), enabled such an erratic behavior. During the tests following this one, such a phenomenon did not seem to occur at least in such an obvious manner, although aerodynamic drag surely played a part in deviating the payloads. In fact, although the planning was performed assuming that the payload would follow a ballistic trajectory after being released, it was expected that this would not be true in reality. Payloads with higher ballistic coefficients could have been used, but this would not have served the purpose of studying the feasibility of using UAV's to drop water or fire retardant, as these have even lower ballistic coefficients than the metallic disc used as a payload.

Another source of precision loss could be the releasing mechanism. Despite the measured delay in the gripping artifact, explained in section 5.1.2, there could be variations in the delay depending on the payload weight. Those variations were not confirmed nor accounted for. In addition, the extrapolation in time for the decision when to release the payload, assumes zero acceleration which could not be the case.

Regarding the releasing speeds, except for the two first tests, where the trajectory was scaled down with the explicit purpose of reducing them, all the maximum achieved velocities were on average 32.6% higher than $v_{max\_no\_descent}$.

6.4 Releasing Malfunction Test

As explained in Sec. 3.1.3, the aim of the optimizer is to release the payload at the highest speed possible and at the lowest possible height (leaving some safety margin from the terrain, 5 m in this case). During this maneuver, the UAV gains high vertical speed because it is using almost the totality of its thrust to propel itself forward rather than maintaining its own weight airborne. It is literally falling. The UAV engages in this *risky* maneuver, counting on that by the time it will attempt to recover from the fall, the payload will already be released and the UAV will be lighter. If the payload is indeed released as planned, the UAV will be able to recover and continue flying.

When a malfunction was simulated in Matlab, the UAV 'crashed' into the terrain as expected(Fig. 31). During the last real test an attempt was made to replicate this

[4]A video showing the whole payload releasing test with the unusual path followed by the payload can be found at http://mrs.felk.cvut.cz/mpc-with-time-variable-mass

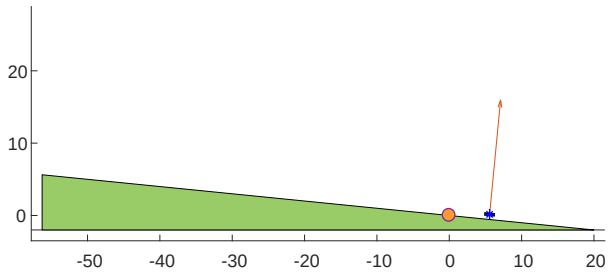


Figure 31: Terrain collision simulated in Matlab after a simulated malfunction in the releasing mechanism. The blue object represents the UAV. The red vector originating at it represents the reference thrust and pitch angle. The orange circle is the target.

malfunction by mechanically avoiding the release of the payload. The aim of this test is to prove that the planning is indeed minimizing the cost to the minimum possible. The weight were tied to the gripping arm with adhesive tape.

The UAV survived the test[5]. In fact, the trajectory deviation from the reference trajectory was barely different from those in previous payload releasing tests. It seems the UAV was underestimated and its maximal thrust is higher than previously thought. As a result, a new series of tests, with less exaggerated safety measures, is being planned.

[5] A video containing a simulation of the release malfunction test in Matlab and a recording of the *release malfunction* test can be found at http://mrs.felk.cvut.cz/mpc-with-time-variable-mass

7 Conclusion

In this thesis it was verified that planning a trajectory counting on a substantial reduction of mass is possible. It was also verified that by allowing large tilt angles and controlled falls, higher speeds than those dictated by the safe flight envelope can be reached.

Defining an OCP in such a way that horizontal velocity at the releasing point must be maximized, and altitude at the same spot, minimized, yields a trajectory that resembles the *dive-bombing* maneuver as predicted.

The *hole in a bucket* release model was not tested either in the simulations or in the real experiments. The main reasons are, its computational complexity, and the suspicion that it will lead to a large dispersion of water, making it impractical. In addition, the small difference between those releasing models that were tested indicated that no special insights would be obtained from testing this releasing model as well. It was verified that for the amount of mass released (about 25% of the overall weight), the mass releasing method does not have a large impact on the trajectory shape. Other unaccounted factors such as the delay imposed by the gripper in charge of physically dropping the payload, have a much larger influence.

The precision of dropping is a merely few meters even though the hardware used was not designed for this purpose. Anyway, in case of a real wildfire with larger UAV's dropping larger amounts of water, should become negligible.

8 Proposed Future Work

This chapter proposes ideas for continuing the work initiated in this thesis.

Implementing the trajectory planning in real-time on board. Transmit terrain data, and wind direction and intensity to the UAV for this purpose.

Creating a SHMPC controller to update the trajectory in real-time to enable maximum exploitation of the UAV's resources.

Attempting to model the payload free fall with a more advanced model such that takes into account the aerodynamic drag acting on the payload to increase drop precision.

Using a faster reacting gripper or planning the trajectory taking into account the delay caused by the current gripper.

Implementing and testing the *'hole in a bucket'* releasing model. Include perhaps the water dispersion in the cost function as a parameter to minimize.

Attempting the trajectory planning in real 3D movement, not constrained to a single plane.

During the development of the project a new idea arouse. It consisted of making the cost function directly dependent on the heat exposure. That means making the cost the integration of the inverse of the square distance between the UAV and the fire through the course of the trajectory $J = \sum_{t_o}^{t_f} \frac{W}{\Delta x^2 + \Delta z^2}$, where W is a constant representing the heat intensity of the fire. From the control point of view this is interesting because it consists in integrating the inverse of a state component.

References

[1] John T. Abatzoglou and A. Park Williams. Impact of anthropogenic climate change on wildfire across western us forests. *Proceedings of the National Academy of Sciences of the United States of America*, 113(42):11770–11775, 7 2016. doi: 10.1073/pnas.1607171113.

[2] T. Baca, D. Hert, G. Loianno, M. Saska, and V. Kumar. Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of unmanned aerial vehicles, 2018. (submitted to IEEE Robotics and Automation Letters).

[3] T. Baca, G. Loianno, and M. Saska. Embedded model predictive control of unmanned micro aerial vehicles. In *21st International Conference on Methods and Models in Automation and Robotics (MMAR)*, 2016.

[4] T. Baca, P. Stepan, and M. Saska. Autonomous landing on a moving car with unmanned aerial vehicle. In *The European Conference on Mobile Robotics (ECMR)*, 2017.

[5] Tomas Baca, Daniel Hert, Giuseppe Loianno, Martin Saska, and Vijay Kumar. Coastal areas division and coverage with multiple uavs for remote sensing. *conditionally accepted in IEEE Robotics and Automation Letters*, 2018.

[6] Tomas Baca, Petr Stepan, Vojtech Spurny, Daniel Hert, Robert Penicka, Martin Saska, Justin Thomas, Giuseppe Loianno, , and Vijay Kumar. Autonomous landing on a moving vehicle with an unmanned aerial vehicle, 2017. (submitted to the special issue on "MBZIRC 2017 - Challenges in Autonomous Field Robotics").

[7] Jennifer K. Balch, Bethany A. Bradley, John T. Abatzoglou, R. Chelsea Nagy, Emily J. Fusco, and Adam L. Mahood. Human-started wildfires expand the fire niche across the united states. *Proceedings of the National Academy of Sciences of the United States of America*, 114(11):2946–2951, 3 2017. doi: 10.1073/pnas.1617394114.

[8] Evan Beachly. An unmanned aerial system for prescribed fires. Master's thesis, University of Nebraska-Lincoln, Lincoln, Nebraska, USA, 12 2017.

[9] L. Chang and Y. Jia. Robust $\mathcal{H}_\infty$ control for tanker station-keeping with mass and inertia variation. In *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pages 340–345, Aug 2017.

[10] Simen Fuglaas. Precision airdrop from a fixed-wing unmanned aerial vehicle. Master's thesis, Norwegian University of Science and Technology, Trondheim, Norway, 5 2014.

[11] C. K. Lai, M. Lone, P. Thomas, J. Whidborne, and A. Cooke. On-board trajectory generation for collision avoidance in unmanned aerial vehicles. In *2011 Aerospace Conference*, pages 1–14, March 2011.

[12] M. Landolfi, S. Bandyopadhyay, J. P. de la Croix, and A. Rahmani. Autonomous guidance navigation and control for agile quadrotors using polynomial trajectory planning and li adaptive control. In *2017 25th Mediterranean Conference on Control and Automation (MED)*, pages 1041–1046, July 2017.

[13] T. Lee, M. Leok, and N. H. McClamroch. Geometric tracking control of a quadrotor uav on se(3). In *49th IEEE Conference on Decision and Control (CDC)*, pages 5420–5425, Dec 2010.

[14] G. Loianno, V. Spurny, T. Baca, J. Thomas, D. Thakur, T. Krajnik, A. Zhou, A. Cho, M. Saska, and V. Kumar. Localization, grasping, and transportation of magnetic objects by a team of mavs in challenging desert like environments. *IEEE Robotics and Automation Letters*, 2018.

[15] S. Di Lucia, G. D. Tipaldi, and W. Burgard. Attitude stabilization control of an aerial manipulator using a quaternion-based backstepping approach. In *2015 European Conference on Mobile Robots (ECMR)*, pages 1–6, Sept 2015.

[16] Vertical Mag. Unmanned k-max completes firefighting demo, October 2015. [Online].

[17] Siri Holthe Mathisen. High precision deployment of wireless sensors from unmanned aerial vehicles. Master's thesis, Norwegian University of Science and Technology, Trondheim, Norway, 5 2014.

[18] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, 31(42):664–674, 4 2012.

[19] Daniel Warren Mellinger. *Trajectory Generation and Control for Quadrotors*. PhD thesis, University of Pennsylvania, Pennsylvania, USA, 1 2012. Accessible Penn Dissertations. 547.

[20] Robert Pěnička, Jan Faigl, Petr Váňa, and Martin Saska. Dubins orienteering problem. *IEEE Robotics and Automation Letters*, 2(2):1210–1217, April 2017.

[21] Robert Pěnička, Martin Saska, Christophe Reymann, and Simon Lacroix. Reactive dubins traveling salesman problem for replanning of information gathering by uavs. In *European Conference of Mobile Robotics (ECMR)*, pages 259–264, 2017.

[22] Iman Sadeghzadeh, Mahyar Abdolhosseini, and Youmin M. Zhang. Payload drop application of unmanned quadrotor helicopter using gain-scheduled pid and model predictive control techniques. In Chun-Yi Su, Subhash Rakheja, and Honghai Liu, editors, *Intelligent Robotics and Applications*, pages 386–395, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[23] M. Saska, T. Baca, J. Thomas, J. Chudoba, L. Preucil, T. Krajnik, J. Faigl, G. Loianno, and V. Kumar. System for deployment of groups of unmanned micro aerial vehicles in GPS-denied environments using onboard visual relative localization. *Autonomous Robots*, 41(4):919–944, 2017.

[24] M. Saska, V. Vonásek, J. Chudoba, J. Thomas, G. Loianno, and V. Kumar. Swarm distribution and deployment for cooperative surveillance by micro-aerial vehicles. *Journal of Intelligent & Robotic Systems.*, 84(1):469–492, 2016.

[25] M. Saska, V. Vonasek, T. Krajnik, and L. Preucil. Coordination and Navigation of Heterogeneous MAV–UGV Formations Localized by a ‘hawk-eye’-like Approach Under a Model Predictive Control Scheme. *International Journal of Robotics Research*, 33(10):1393–1412, 2014.

[26] Vojtěch Spurný, Tomáš Báča, Martin Saska, Robert Pěnička, Tomáš Krajník, Giuseppe Loianno, Justin Thomas, Dinesh Thakur, and Vijay Kumar. Cooperative autonomous search, grasping and delivering in a treasure hunt scenario by a team of uavs, 2017. (submitted to the special issue on "MBZIRC 2017 - Challenges in Autonomous Field Robotics").

[27] Justin Thomas, Giuseppe Loianno, Morgan Pope, Elliot W. Hawkes, Matthew A. Estrada, Hao Jiang, Mark R Cutkosky, and Vijay Kumar. Planning and control of aggressive maneuvers for perching on inclined and vertical surfaces. In *ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 5C: 39th Mechanisms and Robotics Conference, 2015.

[28] Andrija Vidovic. Possibility of implementing unmanned aerial vehicles in firefighting operations, Apr 2014.

[29] Todd Michael Whitalker and Michael Corson. Tethered unmanned aerial vehicle fire fighting system, Sep 2017. Patent (US9764839B2).

Appendix A List of Abbreviations and Acronyms

Abbreviation	Meaning
API	application programming interface
CSV	comma separated values
DTM	digital terrain model
GPS	global positioning system
IMU	inertial measurement unit
MPC	model predictive control
MRS	multi-robot systems
NUC	next unit of computing
OCP	optimal control problem
ODE	ordinary differential equation
PC	personal computer
PID	proportional integral derivative
ROS	robot operating system
SHMPC	shrinking horizon model predictive control
UAV	unmanned air vehicle
VTOL	vertical take of and landing

Appendix B CD Content

In Table 4 are listed names of all root directories on the attached CD.

Directory name	**Description**
Thesis Document	Master's thesis in PDF format.
Source Code for ROS	source code for the implementation in ROS.
Matlab Source Code	source code for the Matlab optimization and plotting.
Videos	Videos of simulations and real tests.

Table 4: CD Content