Noname manuscript No.
(will be inserted by the editor)

Precision Deep-Stall Landing of Fixed-Wing UAVs using Nonlinear Model Predictive Control

Siri Mathisen · Kristoffer Gryte · Sebastien Gros · Tor Arne Johansen

Received: date / Accepted: date

Abstract To be able to recover a fixed-wing unmanned aerial vehicle (UAV) on a small space like a boat deck or a glade in the forest, a steep and precise descent is needed. One way to reduce the speed of the UAV during landing is by performing a deep-stall landing manoeuvre, where the lift of the UAV is decreased until it is unable to keep the UAV level, at the same time as the drag is increased to minimize the speed of the UAV. However, this manoeuvre is highly non-linear and non-trivial to perform with high precision. To solve this, an on-line non-linear model predictive controller (NMPC) is implemented to guide the UAV in the landing phase, receiving inputs from the autopilot and guiding the UAV using pitch and throttle references. The UAV is guided along a custom path to a predefined deep-stall landing start point and performs a guided deep-stall. The simulation results show that the NMPC guides the UAV in a deep-stall landing with good precision and low speed, and that the results depend on a correct prediction model for the controller.

Keywords Unmanned aerial vehicles · Real-time nonlinear model predictive control · Deep-stall landing · Autonomous landing · Guidance

Mathematics Subject Classification (2010) MSC 90C90

This work is part of a project partly funded by the Research Council of Norway through the Centers of Excellence funding scheme, project number 223254. Centre for Autonomous Marine Operations and Systems (AMOS), Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), Trondheim, Norway

S. Mathisen · K. Gryte · S. Gros · T. A. Johansen
Centre for Autonomous Marine Operations and Systems (NTNU AMOS)
Department of Engineering Cybernetics
NTNU, Norwegian University of Science and Technology
S. Mathisen
SINTEF Energy Research
Tel.: +47 41 10 01 79
E-mail: siri.mathisen@sintef.no

1 INTRODUCTION

During the recent years, autonomous flight has received a lot of attention in research (e.g. [1], [2], [3]) and the main challenges that arise are autonomous take-off, navigation and landing. This paper focuses on autonomous landing of fixed-wing unmanned aerial vehicles (UAVs), as this subgroup of UAVs generally has longer range and endurance, which makes them preferable in many operations compared to rotary-wing UAVs. Autonomous landing of UAVs has been studied before, see e.g. [4], [5], [6] and, more recently [7] and [8]. However, to further increase the usage of fixed-wing UAVs, autonomous landing without a runway is desired. This is motivated by maritime operations, where a UAV should be able to land on a ship deck, which normally has limited space. One solution to this problem is to land the UAV in a net (e.g. [9], [10], [11]) though, while it simplifies the recovery and enable landing on a ship, it yields significant structural loads on the UAV and net, and requires infrastructure to support the net.

A decelerated landing of a UAV is achieved by increasing the angle of attack until the air flow separates from the airfoil of the wing, called the stall angle, and then beyond to reach a deep-stall [12]. A deep-stall is the state of the UAV with a stable trim, with an angle of attack beyond the stall angle. More details about the aerodynamics of deep-stall can be found in [13], [14], [15], [16], [17] and [18]. Landing an aircraft with the use of a deep-stall has been investigated in the literature: [19] presents a study of a manned aircraft deep-stall and [20] presents experimental results from deep-stall landing with a UAV. In [21] a deep-stall landing is considered as a promising way of retrieving a UAV after flight tests in the field. In [22] linear-quadratic regulator trees are presented as a way of controlling a UAV in deep-stall landing, with implementation, flight tests and promising results. In [23] the navigation part of the deep-stall landing and the safety aspect are considered. In [24] a nonlinear model predictive controller for real-time autolanding in deep-stall of a UAV is described, built on the work presented in [25]. The ACADO framework is used for numerical optimization, just like in the current paper, but the simulations are performed on the prediction model with no disturbances. In [26] the longitudinal aerodynamic characteristics of a deep-stall are investigated. In [27] a study on the use of flare for autonomous landing of a fixed-wing UAV is presented, and [28] creates a trajectory for perched landing of a small fixed-wing UAV.

To guide a fixed-wing UAV in a deep-stall landing, this paper suggests the use of a nonlinear model predictive controller (NMPC). An MPC is an optimal controller that provides control action rather than an explicit control law. Based on a prediction model for the vehicle dynamics and the latest measurements or estimates of the model's state variables as well as disturbances, a control trajectory for a given prediction horizon is calculated, optimizing a cost function. The first control action from the trajectory predicted by the NMPC is used as input to the UAV and the next measurement from the UAV is used as input for the state estimator supporting the MPC. A linear MPC has a linear model and constraints, and a quadratic cost function. Nonlinear MPC is more computationally demanding, as it requires solving a generic, possibly con-convex Nonlinear Program, accurate computation of gradients and other numerical issues that prevents

the problem to fall into the linear category [29]. Calculating an iterative solution within a convergence criterion will be time consuming, while the alternative solution would be to abort the calculations before convergence has been reached to stay inside a time budget [30]. For more details about the NMPC, see [30] or [31]. MPC has had great success in the process and petroleum industry, but due to its computationally costly nature, embedded MPC is not trivial [29]. A substantial research work has been done for improving the online embedded MPC, including improvements of the algorithms for solving the MPC faster (e.g. [32], [33], [34], [35]), software for embedded numerical optimal control using different algorithms, e.g. qpOASES [36], CVXGEN [37], HPMPC/MPMPI [32],[33] and environments like ACADO [38], or Acados [39], [40] where code generation is an important feature (e.g. [34], [41], [42]).

In general, the control hierarchy of a UAV is divided into layers. Navigation, guidance and control form different layers handling state estimation, reference computation and control, respectively. Examples of control objective are path following or energy minimization [43], [44]. MPC is applicable to several layers of UAV control. In path planning, MPC can plan and re-plan the UAV trajectory over a long prediction horizon, and this plan can be used as an input reference for a lower control layer. In the lower control layers it is critical to get feedback. Depending on the computing power of the embedded computer, it can be possible to solve the optimization problem on-board the UAV [45]. A viable alternative is to perform the computations on a remote computer and send the control input to the UAV over a radio communication link as long as delays are small.

As the models of UAVs are inherently non-linear, linear MPCs for UAVs are typically linearized around the trim states ([46], [47] and [48]). For regular flight conditions, the angle of attack α is usually much smaller than α_0, where α_0 is the stall angle. At low α, the drag, lift and pitch moment of the UAV can be described as linear functions of α. In stall conditions, however, the drag, lift and pitch moments are higly nonlinear functions of the angle of attack, and linear models are not sufficient [44], [23]. Therefore, for transition into stall, linearization around trim values are not useful and the full nonlinear prediction model is used.

Although there has been a substantial progress in computation time of NMPC due to efficient cropping of algoritms and progress in embedded hardware [49], an NMPC is nevertheless a control technique with high computational consumption due to a high sampling frequency. In order to keep the computational power consumption as low as possible, it is desirable to use the NMPC only when the benefits from using it outweighs the costs. One way of minimizing the computation is to decouple lateral and longitudinal guidance and control. This has been previously published in [48], where a longitudinal MPC controls the UAV directly as an augmentation to a longitudinal autopilot and in [47], where an MPC using a longitudinal model tracks a glide slope with the aim to land the UAV.

In [50], an NMPC is used in the lowest layer of a three-layered guidance, navigation and control system. A path planner provides waypoints and sensor information to a navigation system, which converts the waypoint path references to speed, flight path and course angle references for the NMPC. The flight con-

trol system uses a genetic algorithm to solve the NMPC problem. This system has elevator, aileron, throttle and rudder as output from the NMPC flight control system. In [51] the same output as the previous paper is used, based on a 6-DoF model of the UAV and an NMPC for attitude control. In [52], an NMPC uses a 6-DoF model of the UAV to control the aircraft in extreme manoeuvres, using a reference trajectory for the position.

The opposite approach concerning abstraction level would be to plan a reference path in real time with the NMPC, and provide the autopilot with way points and airspeed references to follow this path. In [53], an on-line MPC on the ground station tracks a pre-planned flight path and communicates the control signals to the autopilot through the payload computer, using a radio link. The MPC controls both the attitude of an attached gimbal and the path, providing the autopilot with waypoints and desired airspeed and the gimbal with pan and tilt commands.

If the objective of the relevant manoeuvre also requires control of the UAV at lower level than the flight path, then waypoint and speed references are not sufficient, and something between the full model and the waypoint solution is required. In [54] this is solved by creating a control system with a layered structure: the goal is to fly a mid-sized fixed-wing UAV from a given point in the air to a landing point on the ground. An on-line NMPC plans the whole horizon of the landing path at each time step. This path consists of two different optimization problems forming two different landing phases. The whole-horizon NMPC plans a landing trajectory for the UAV, and the path is followed using a high-level $\mathcal{H}_2/\mathcal{H}_\infty$ cascaded with a low-level PD controller. A similar structure is also seen in [55] and further detailed in [56], where a three-layer structure uses an attitude controller, a path-following guidance module based on MPC and a reference trajectory on top, which is followed by the guidance module. The MPC model is nonlinear, but the proposed optimization algorithm proceeds with the linearisation of the prediction model around a set of pre-computed control values, obtained by another guidance algorithm, where an estimator is included to account for disturbances. The output of the guidance layer is the flight path angle, bank angle and airspeed reference. This approach yields a sufficiently low abstraction level to control the UAV into a deep-stall, without controlling the attitude of the UAV. In [57] a high-level NMPC for path following, speed stabilization and constraint handling is used, while the low-level attitude and rates control is performed by the autopilot. In the present paper, we will follow the same idea, and use the NMPC for path control, whereas the low level attitude control and rates control is performed by the autopilot.

This paper focuses on on-board, on-line NMPC. To reduce the computation time without compromising the result, we decouple the lateral and the longitudinal guidance, performing the deep-stall NMPC guidance only in the longitudinal plane. The lateral guidance will use a separate controller that is not based on NMPC, and while it is included in the simulations it will not be the focus of this paper. As a result, instead of a full 6-DoF prediction model, we use a 3-DoF prediction model of the UAV for the NMPC calculations. The main contributions of this paper are the implementation of a longitudinal NMPC in a custom software package intended for an on-board payload computer in small fixed-wing UAVs, and a path planner for safe deep-stall landing. The presented NMPC can guide

the UAV in a deep-stall, driving the autopilot of the UAV with pitch and throttle references from the NMPC. The autopilot provides low-level attitude control. This paper makes a novel combination of NMPC and deep-stall landing. It continues the work done by [25] and [58], where a simulated modified Aerosonde fixed-wing UAV was guided in a deep-stall with an NMPC, and the work done by [59], where a path planner is implemented for precision delivery with a small fixed-wing UAV. In this present paper, the NMPC system is taken one step further than in [58] towards flight testing. Where [58] uses lateral as well as longitudinal NMPC guidance, we here present only longitudinal NMPC guidance combined with a lateral line-of-sight-based guidance module in order to reduce computational complexity. The emphasis is placed on migration towards actual flight testing and integration with a realistic physical environment, including the autopilot software, whereas [58] guides a naïve UAV-simulator using the same model as the controller and testing it with software-in-the-loop (SITL) simulation. The updated MPC in this paper uses code generation with an online active set strategy solver, compared to the interior point method used in [58]

This paper is organized as follows: Section 2 presents a path planning procedure for the landing, introduces the three manoeuvres involved in the plan and derives an algorithm for the computation of a start point for the deep-stall phase of the landing. Section 3 presents the real-time environment for the NMPC calculations and the control system structure for the on-board flight control system, while Section 4 explains the deep-stall NMPC implementation and the UAV model. Section 5 presents the details of the SITL simulation and the simulation results. In Section 6, the results are discussed and some suggestions for future work are given, before the paper is concluded in Section 7.

2 Path Planner

To execute a safe deep-stall landing, a landing plan is composed. The main purpose of the landing plan is to calculate a point for deep-stall landing initiation, to guide the UAV to this point with the correct start state and to guide the UAV in the deep-stall such that it lands at a defined target position with minimum speed. Besides, the plan must consider when a deep-stall should be aborted. The UAV's lateral landing path is a truncated Dubins path, shown in Fig. 1: First, the UAV aims for a geographical reference s and loiters around it with radius r until it has the appropriate start altitude and start velocity for the deep-stall, and is directed at the target, in point p. To minimize the ground speed in the landing, the UAV aspires to always land against the average wind direction. However, the wind changes with a change in the position of the UAV and time. Therefore, the wind speed and direction are estimated utilizing on-board sensors or ground infrastructure before planning the landing path in Fig. 1. Further changes in the wind do not affect the lateral landing plan but will be used by the NMPC calculations. In the case of no wind, the landing direction should be decided by the geographical landing conditions. After a short distance D with level flight, the NMPC takes over the longitudinal guidance of the UAV. The NMPC first guides the UAV a short distance in level to ensure the numerical algorithm is suitably initialized for the NMPC before the nonlinear deep-stall operation is initiated. Once the UAV

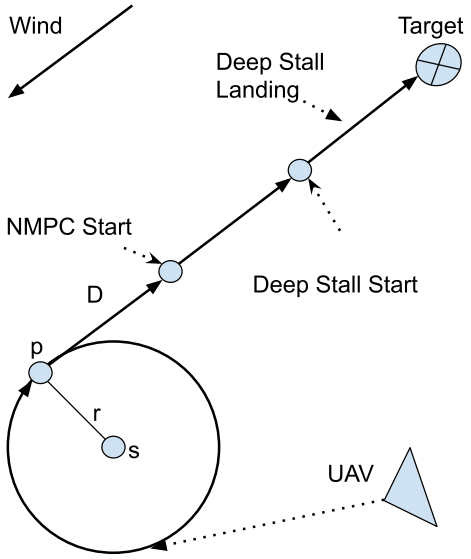


Fig. 1: The lateral path for the landing. Before longitudinal NMPC guidance is initiated, the UAV is guided with line-of-sight-based lateral and longitudinal controllers.

reaches the *Deep-Stall Start* point, it initiates the deep-stall mode and stays in deep-stall until it touches the ground or until the predicted landing error is too large, in which case the UAV aborts the landing. The landing is initiated when the UAV receives the geographical coordinates of the landing target. Based on the target coordinates, the wind velocity estimates and the UAV's deep-stall path angle, the start point of the deep-stall can be computed. Once the deep-stall start point is known, the truncated Dubins path is trivial to calculate. This path is used in [59] and described further in [60].

The landing plan consists of three manoeuvres: the *Landing Preparation*, the *Deep-Stall Landing* and the *Safety Abort*. Each manoeuvre has a longitudinal and a lateral path controller, guiding the UAV with desired roll, pitch and throttle commands sent to the autopilot. The autopilot performs attitude and propulsion control. The NMPC is only used for the longitudinal control in the *Deep-Stall Landing* manoeuvre; the two other manoeuvres use a longitudinal cascaded LOS-based controller. This controller consists of an altitude controller and a climb rate and speed controller. At the point NMPC Start in Fig. 1, the manoeuvre *Landing Preparation* ends and the manoeuvre *Deep-Stall Landing* initiates. The transition between the two manoeuvres means that the algorithm for calculating desired pitch and throttle commands changes. Any adverse effects of the switch between the two guidance methods is minimized by proper initialization of the MPC to achieve bumpless transfer, and the switch does not cause any stability issues in itself since it is a single event. A reference line with a start point and an end point,

both using geographical coordinates and altitude, is sent to the altitude controller. It uses line-of-sight (LOS) guidance theory from [61] to produce a desired climb rate, which together with the desired speed is sent to an underlying total energy control-based controller in the autopilot. The output of the climb rate and speed controller is desired throttle and pitch, the same as for the longitudinal NMPC used for the deep-stall manoeuvre. The lateral LOS-guidance controller, which is also based on [61], produces desired roll commands for the autopilot, by investigating the lateral acceleration necessary to eradicate the cross-track error.

If the NMPC is unable to control the UAV in a safe deep-stall landing, the landing must be aborted, and the *Safety Abort* manoeuvre will initiate. This manoeuvre guides the UAV using LOS-based controllers with a waypoint interface to the user. Reasons for aborting the deep-stall landing include an infeasible solution from the model predictive controller or a too high predicted landing error at a critical height above the ground. Just like in the point NMPC Start in Fig. 1, this transition involves a change of algorithms to calculate desired pitch and throttle commands, and of course a different desired waypoint. The effects of this switch of guidance methods is not considered beyond getting the UAV to a safe height. If the *Safety Abort* manoeuvre is initiated while the UAV is in a deep-stall, it is in a state with critically low lift forces acting on the aircraft. One common way of getting out of a deep-stall is to command sufficient use of throttle, requiring a high-powered UAV and sufficient available throttle potential during the deep-stall.

2.1 Computation of Deep-Stall Start Point

The *Deep-Stall Landing* manoeuvre consists of an initial phase where the UAV has level flight, and a deep-stall phase where the UAV has a high angle of attack and a steep path angle, see Fig. 3. The relative path angle for a deep-stall, $\gamma_{rel}^{\mathrm{DS}}$, is model dependent and its value is known from experiments. In a deep-stall landing, the conditions of a stable trim for are fulfilled for the pitch moment C_m [18]:

$$C_m = 0, \quad \frac{dC_m}{d\alpha} \leq 0 \tag{1}$$

allowing the UAV to passively reject minor longitudinal perturbations. A plot of the pitch moment, drag coefficient and lift coefficient with respect to the angle of attack is shown in Figure 2.

This also means that once the UAV enters the deep-stall, it is costly to change its path angle. Ideally, the UAV should have been able to follow a dynamic path angle in a deep-stall, as was shown in [25] and [58] with much simpler simulations and no modelling error. However, since this may be difficult to achieve with a more advanced simulation environment or in flight tests, the start point where the UAV enters the deep-stall will strongly influence the target error. Indeed, if the deep-stall starts too early, the UAV will be short of the target, and if it stalls too late, it will land beyond the target. Additional to the predicted landing error, disturbances may cause an error between the reference path angle and the path angle of the UAV, which can in turn lead to a landing error. However, the same effect that causes planned changes in the path angle of the UAV to be costly,

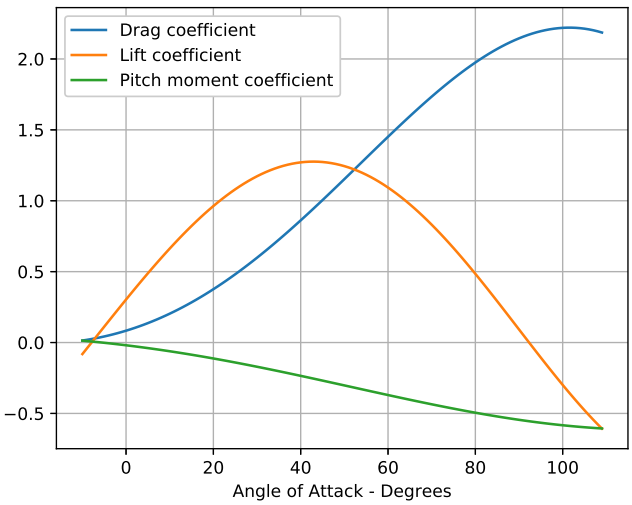


Fig. 2: The drag coefficient, the lift coefficient and the pitch moment coefficient of the UAV model used in this paper, with respect to the angle of attack.

means that disturbance inflicted changes in the path angle of the UAV demands large disturbances.

The deep-stall occurs at a high angle of attack, α, as this typically gives a high drag force and a low lift force. The angle of attack is the rotation about the body y-axis between the body x-axis and the air velocity vector, and is thus defined as:

$$\alpha = \arctan\left(\frac{w - w_w}{u - u_w}\right), \tag{2}$$

see e.g. [44], where u and w are the longitudinal and down velocities in the body frame and u_w and w_w are the corresponding components of the airspeed vector in the body frame. The relationship between the angle of attack, the pitch angle θ and the relative path angle is:

$$\gamma_r = \theta - \alpha, \tag{3}$$

see e.g. [44]. A constant angle of attack will therefore be obtained if the relative path angle γ_r is also constant, as the pitch angle is independent on wind. γ_r is expressed as

$$\gamma_r = -\arctan\left(\frac{V_z^{\text{rel}}}{V_g^{\text{rel}}}\right), \tag{4}$$

where V_z^{rel} is the relative down speed in the inertial frame and V_g^{rel} is the relative ground speed in the inertial frame. A constant γ_r will demand constant relative ground speed and constant relative down speed, which means that the absolute path angle γ of the deep-stall can, when the wind velocity, relative ground speed and relative path angle is know, be expressed as:

$$\gamma^{\text{DS}} = -\arctan\left(\frac{V_z^{\text{rel}} + V_z^{\text{wind}}}{V_g^{\text{rel}} + V_g^{\text{wind}}}\right), \tag{5}$$

$$V_z^{\text{rel}} = -V_g^{\text{rel}} \tan\left(\gamma_r^{\text{DS}}\right) \tag{6}$$

where V_g^{wind} is the ground wind velocity in the inertial frame and V_z^{wind} is the down wind velocity in the inertial frame, and is dependent on the wind strength: For a landing against the wind, a stronger wind and constant airspeed will lead to steeper path angle. As the absolute path angle and the relative deep-stall path angle are known, we can calculate where the deep-stall start point of Fig. 3 ought to be placed.

However, the UAV will not transition from the steady flight to the deep-stall phase instantaneously. Between these two phases there is a transition phase, which is highly nonlinear and nontrivial to calculate or predict. With sufficiently precise model parameters representing the UAV, including the autopilot, a path planner could plan the whole landing offline with the UAV's start altitude and start velocity, and use the planned path to compute a start point. Yet simulations show a considerable model mismatch between the NMPC's model and the UAV's model, so a simpler approach has to be used. A guess for the transition start point is a wind dependent distance d_2 ahead of the deep-stall start point, given by:

$$d_2 = \Delta_T V_g^{\text{rel}}, \tag{7}$$

where Δ_T is an empirical value and and V_g^{rel} is the ground speed of the UAV in level flight relative to the surrounding air masses. Once d_2, the start height h and the relative deep-stall angle γ_r^{DS} are known, we can compute the transition start point. The UAV should initiate the deep-stall once the longitudinal angle between the UAV and the target is steeper than a limit $\angle_{\text{lim}}$, given by:

$$\angle_{\text{target}}^{\text{lim}} = -\arctan\left(\frac{h}{d_2 + d_3}\right), \tag{8}$$

$$d_3 = -\frac{h}{\tan\left(\gamma^{\text{DS}}\right)} \tag{9}$$

After the UAV has initiated the deep-stall and started descending, it soon reaches a stationary phase with a constant path angle and constant ground velocities. Then the predicted landing error e, meaning the distance between the landing coordinates and the target coordinates, is calculated, and if the predicted landing error is larger than a limit e^{MAX}, the UAV aborts the landing. The distance d_2 is then updated with the predicted error, assuming perfect lateral guidance:

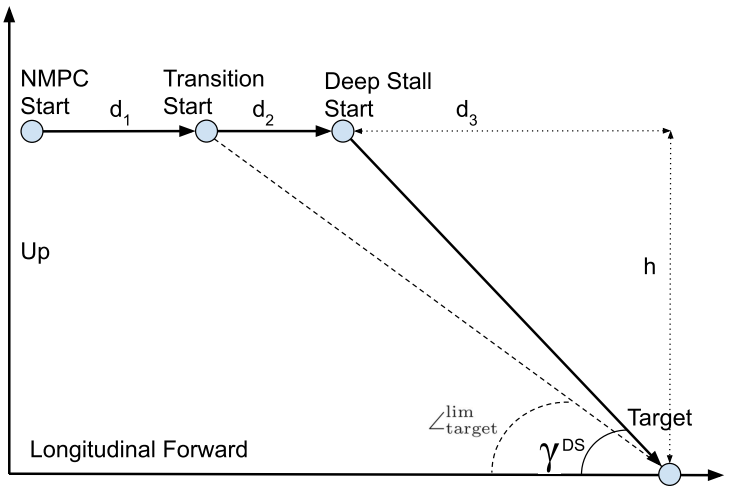


Fig. 3: The longitudinal reference path for the deep-stall landing

$$d_2^{\text{new}} = d_2^{\text{old}} + e \tag{10}$$

The computation of the deep-stall start point is summarized in Algorithm 1.

Algorithm 1 Computation of deep-stall start point

Require: Airspeed, γ_r^{DS}, V_z^{wind}, V_g^{wind}
  Compute V_z^{rel} using (6)
  Compute γ^{DS} using (5)
  Compute a guess for d_2 using (7)
  Compute $\angle_{\text{target}}^{\text{lim}}$ using (8)
  while $e \geq e^{\text{MAX}}$ **do**
    Update d_2 using (10)
  end while

3 Real-time Integration and Control Structure

Testings of the NMPC formulation in a non-real time environment (CasADi, [62], [63], [64]) in [25] and [58] have delivered satisfactory results. Tests in a real-time environment and an implementation of the controller is considered here. The real-time optimization environment selected for this work is the ACADO software referred to previously. Promising results from simulations of this problem using ACADO are presented in [24]. We use the automatic code generation feature provided by ACADO, which makes a stand-alone optimization package suited for embedded use. The QP solver used is qpOASES, based on an online active set strategy [65]. After successfully testing the NMPC with a simple simulator using

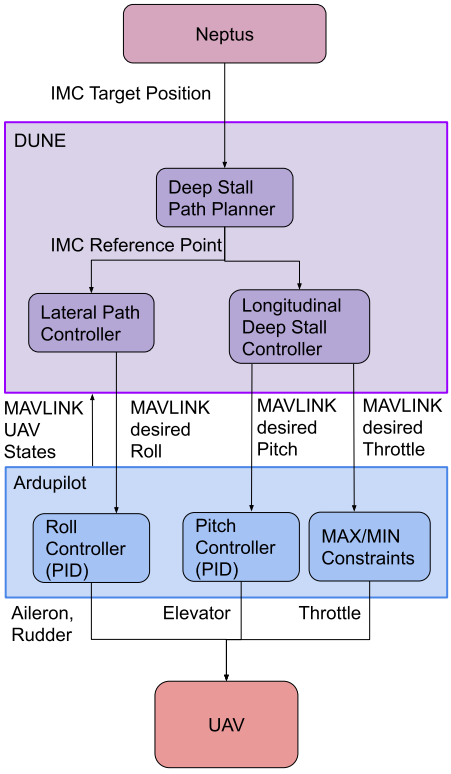


Fig. 4: Control system structure for deep-stall manoeuvre.

our model and the Runge Kutta method, the controller is tested in a more advanced simulation environment.

The flight control system is implemented as described in [66] with minor upgrades and revisions. It contains a Pixhawk autopilot using Ardupilot flight control software onboard a small, stable fixed-wing UAV with aerodynamic properties suited for a controlled deep-stall. The software onboard the embedded control platform uses the LSTS toolchain [67] to organize the software. This toolchain uses a GLUED based Linux operating system and runs a modular software called DUNE that implements customized algorithms, communicates with a ground station and can, in autonomous mode, command the autopilot. DUNE also facilitates a modular guidance, navigation and control (GNC) system, which we exploit by controlling our UAV in various modes for different purposes, with various controllers within the different modes. The LSTS toolchain provides an Inter-Module Communication (IMC) protocol and a graphical user interface (Neptus).

An overview of the modular control design for the *Deep-Stall Landing* manoeuvre is shown in Fig 4. Implicitly, the autopilot provides all modules on the experimental flight computer with state feedback. The UAV receives a landing

command including the coordinates of the landing location, and the message is processed by a top-level unit, the landing plan. This module assigns the NMPC with the longitudinal path control, and the lateral path control to the LOS-based lateral controller described in Section 2. Both controllers receive the geographical coordinates of the landing location and the coordinates for the start point, which the lateral controller uses to create a reference line for the LOS. The output of the lateral path controller are roll (ϕ) commands, whereas the deep-stall manoeuvre outputs pitch and throttle (θ, δ_t) commands. These commands are received by the autopilot, which performs low-level attitude control of the UAV and commands its control surfaces and propeller. The output from the guidance algorithms are sent to an autopilot communication module that translates the references to a MAVlink protocol messages (see [66] for more details).

4 NMPC Implementation

The discrete-time NMPC is formulated as follows:

$$\min_{x,u} \quad \sum_{k=0}^{N-1} \|h\left(x_k, u_k\right) - \tilde{y}_k\|_{W_k}^2 + \|h_N\left(x_N\right) - \tilde{y}_N\|_{W_N}^2 \tag{11a}$$

$$\text{s.t.} \quad x_0 = \hat{x}_0, \tag{11b}$$

$$x_{k+1} = F\left(x_k, u_k, z_k\right), \qquad \text{for } k = 0, ..., N-1 \tag{11c}$$

$$u_{\min} \leq u_k \leq u_{\max}, \qquad \text{for } k = 0, ..., N-1, \tag{11d}$$

$$C_{\min} \leq f\left(x_k\right) \leq C_{\max}, \qquad \text{for } k = 0, ..., N. \tag{11e}$$

A time varying reference vector $\tilde{y}_k$ is provided to the NMPC scheme, as well as the external input variables, $z_k = \left[u_w \; w_w \; \dot{p}_d^{\text{ref}}\right]$, where u_w and w_w are wind velocity with respect to the inertial frame expressed in the body frame, and $\dot{p}_d^{\text{ref}}$ is the desired down velocity in the inertial frame. The dynamics referred to in (11c) are numerical discretizations of the longitudinal model for the fixed-wing, unmanned aircraft described in [44], recalled here:

$$\begin{bmatrix} \dot{p}_n \\ \dot{p}_d \\ \dot{u} \\ \dot{w} \\ \dot{\theta} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} \cos\left(\theta\right) u + \sin\left(\theta\right) w \\ -\sin\left(\theta\right) u + \cos\left(\theta\right) w \\ -qw + \frac{1}{M} f_x \\ qu + \frac{1}{M} f_z \\ q \\ \frac{1}{J_y} m \end{bmatrix} \tag{12}$$

The variables p_n and p_d are the position in the inertial landing direction and down position of the UAV, u, w are the body frame velocities measured along the longitudinal and vertical body frame axes, θ is the pitch angle, q is the pitch rate along the body frame y-axis, M is the mass of the UAV while flying and J_y is the moment of inertia about the body frame y-axis. The forces acting on the body frame f_x and f_z are given by:

$$\begin{bmatrix} f_x \\ f_z \end{bmatrix} = \mathbf{f}_g + \mathbf{f}_a + \mathbf{f}_p, \tag{13a}$$

$$\mathbf{f_g} = \begin{bmatrix} -Mg\sin(\theta) \\ Mg\cos(\theta) \end{bmatrix}, \tag{13b}$$

$$\mathbf{f_a} = \frac{1}{2}\rho V_a S \begin{bmatrix} C_X(\alpha) + C_{X_q}(\alpha)\frac{c}{2V_a}q + C_{X_{\delta_e}}(\alpha)\delta_e \\ C_Z(\alpha) + C_{Z_q}(\alpha)\frac{c}{2V_a}q + C_{Z_{\delta_e}}(\alpha)\delta_e \end{bmatrix}, \tag{13c}$$

$$\mathbf{f_p} = \frac{1}{2}\rho S_{\text{prop}} C_{\text{prop}} \begin{bmatrix} (k_{\text{motor}}\delta_t)^2 - V_a^2 \\ 0 \end{bmatrix} \tag{13d}$$

where $C_X(\alpha)$, $C_{X_q}(\alpha)$, $C_{X_{\delta_e}}(\alpha)$, $C_Z(\alpha)$, $C_{Z_q}(\alpha)$ and $C_{Z_{\delta_e}}(\alpha)$ are functions of the aerodynamic drag and lift coefficients $C_D(\alpha)$ and $C_L(\alpha)$. The pitch moment m is given by

$$m = \frac{1}{2}\rho S c[C_m(\alpha) + C_{m_q}\frac{c}{2V_a}q + C_{m_{\delta_e}}\delta_e], \tag{14}$$

where the airspeed is:

$$\begin{aligned} V_a &= \sqrt{u_r^2 + w_r^2}, \\ \begin{bmatrix} u_r \\ w_r \end{bmatrix} &= \begin{bmatrix} u - u_w \\ w - w_w \end{bmatrix}, \end{aligned} \tag{15}$$

assuming perfect lateral guidance.

The control of the UAV in the longitudinal plane is based on manipulating the throttle δ_t and the deflection of the elevator δ_e. However, our system structure requires that pitch angle and throttle command references are sent to the autopilot, see Fig. 4. Whereas the NMPC is intended to guide the UAV in the longitudinal plane, the PID controller in the autopilot oversees the attitude control, using the elevator deflection to control the pitch angle of the UAV. The relationship of the autopilot between the desired pitch angle and the elevator command is given by a PID controller. To account for this in the prediction model, we introduce δ_e as an additional state instead of as an output of the controller, and an additional input θ_d for the desired pitch:

$$\dot{\delta_e} = -K_p\left(\dot{\theta}_d - q\right) - K_d\left(\ddot{\theta}_d - \dot{q}\right) - K_i\left(\theta_d - \theta\right). \tag{16}$$

We assume that $\ddot{\theta}_d = 0$. The NMPC scheme generates two inputs for the autopilot, gathered in the control vector $u_k = \left[\dot{\delta}_t \ \dot{\theta}_d\right]$. The cost function in (11a) considers errors in $h(x_k, u_k)$ with weighting W_k, and end term $h_N(x_N)$ with weighting W_N. The purpose of the NMPC is to produce outputs that can both keep the UAV in level flight and in a precise deep-stall. We thus use the following terms in the function h:

$$h = \begin{bmatrix} \dot{p}_n \\ \rho \\ \theta \\ \delta_t \\ \dot{\delta}_t \\ \dot{\theta}_d \end{bmatrix}, \tag{17}$$

$$\dot{\rho} = \dot{p}_d - \dot{p}_d^{\text{ref}}. \tag{18}$$

The mismatch between the NMPC model and the actual UAV dynamics yield an offset between the measured states and the references. Offsets can also result from sensor biases. To remove the offset, we add the ρ term in h, which is the integral of the deviation of the vertical speed in the inertial frame, $\dot{p}_d$, from the desired vertical speed $\dot{p}_d^{\text{ref}}$ in the inertial frame. This term introduces integral action in the NMPC scheme for the vertical speed, controlling the path angle of the UAV. As there are many ways to achieve correct $\dot{p}_n$ and $\dot{p}_d$, δ_t and θ are included in h to aid the solver in the NMPC. The two last terms of h, $\dot{\delta}_t$ and $\dot{\theta}_d$, penalize large changes in the control variables δ_t and θ_d. The complete state vector reads as:

$$x_k = \begin{bmatrix} p_n & p_d & u & v & \theta & q & \delta_e & \delta_t & \theta_d & \rho \end{bmatrix}^T \tag{19}$$

In the *Cruise* phase of the *Deep-Stall Landing* manoeuvre, the reference vectors are constant and defined as:

$$\tilde{y}_k = \begin{bmatrix} v_x^{\text{cruise}} + v_x^{\text{wind}}, & 0, & \alpha^{\text{cruise}}, & \delta_t^{\text{cruise}}, & 0, & 0 \end{bmatrix}^T, \tag{20}$$

for $k = 0, \ldots, N$, and

$$\dot{p}_d^{\text{ref}} = -v_x^{\text{cruise}} \tan\left(\gamma_r^{cruise}\right) + v_z^{\text{wind}}, \tag{21}$$

$$= 0, \tag{22}$$

as the relative path angle for cruise is zero. In the *Deep-Stall* phase, the reference vectors are as:

$$\tilde{y}_k = \begin{bmatrix} v_x^{\text{stall}} + v_x^{\text{wind}}, & 0, & \alpha^{\text{stall}} + \gamma_r^{\text{DS}}, & \delta_t^{\text{stall}}, & 0, & 0 \end{bmatrix}^T, \tag{23}$$

for $k = 0, \ldots, N$. The parameters v_x^{cruise} and v_x^{stall} are decided by the user and appropriate values are found through tuning, and α^{cruise} and α^{stall} are decided by the definition of level flight and deep-stall. The input variable is defined as

$$\dot{p}_d^{\text{ref}} = -v_x^{\text{stall}} \tan\left(\gamma_r^{DS}\right) + v_z^{\text{wind}}. \tag{24}$$

The initial state $\hat{x}_0$ of the NMPC (11b) at each time step comes from the autopilot. Since the NMPC is an Earth-fixed longitudinal controller, the input from the autopilot is translated to a longitudinal frame where the UAV moves against the wind direction, as it was first estimated when the plan was made. The wind velocity estimate is translated to body wind velocity before it is fed to the NMPC, and translated to the Earth-fixed longitudinal frame for the references. The initial

guess for the control variables $\dot{\delta}_t$ and $\dot{\theta}_d$ is zero. The box constraints $C_{\min}$ and $C_{\max}$ in (11d) and (11e) are constant constraints given by:

$$\begin{aligned} -40^o \leq\ \theta\ &\leq 40^o \\ -10^o \leq\ \alpha\ &\leq 80^o \\ -40^o \leq \delta_e &\leq 40^o \\ 0 \leq \delta_t &\leq 1 \\ -35^o \leq \theta_d &\leq 35^o \end{aligned} \tag{25}$$

$$\tag{26}$$

where $f\left(x_k\right) = \left[\theta\ \alpha\ \delta_e\ \delta_t\ \theta_d\right]$.

As the aim of this research is to deep-stall land a UAV with an on-line, on-board NMPC, fast code is imperative and the code generation tool from ACADO is used. To be able to use the NMPC in real-time, a Real-Time Iteration in the form described in [49] is used, implementing a single linearization and QP solution at each NMPC sampling time, together with a preparation-feedback split, see Algorithm 2. The reference vector $\tilde{y}_k$ is dynamic and its value will change according

Algorithm 2 NMPC Implementation

```
NMPC Initializing
x_0 = x_measured
NMPC Preparation step
for each sampling period do
    NMPC Feedback step
    Apply u_0
    x_0 = x_measured
    NMPC  Preparation step
end for
```

to which phase of the landing the UAV is currently in.

5 SITL Simulation

In SITL simulation, the software of the autopilot is included in the simulations, to allows testing of the interaction between the autopilot and the controller before a flight test. The SITL simulation uses the ArduPlane autopilot software intended for SITL, the LSTS toolchain [67] with the DUNE unified navigation environment, inter-module communication (IMC) protocol and Neptus user interface, and JSBsim physical environment and flight dynamics simulator running a modified Aerosonde UAV, with the experimental flight computer program running on a laptop computer. See Fig. 4, where JSBsim simulates the UAV. The wind in the simulator follows the Dryden wind turbulence model [68]. One realization of the lateral path of the UAV is shown in Fig. 5. All constants for the model referred to in (12) are found in [44], values for the Aerosonde. The lift, drag and pitch moment coefficients $C_D(\alpha)$, $C_L(\alpha)$ and $C_m(\alpha)$ are given by polynomial expressions corresponding to the NACA 4415 airfoil [69]. More details about the model can

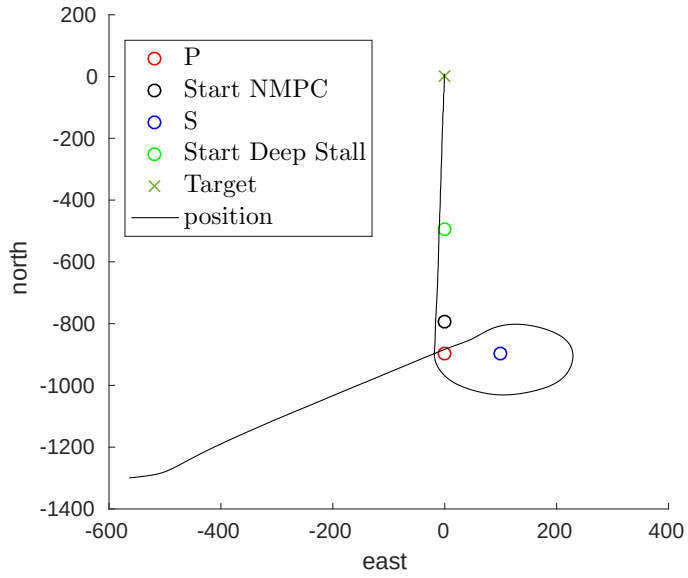


Fig. 5: The lateral flight path of the UAV before and in the deep-stall, with 4 m/s wind.

Table 1: Weighing parameters

penalized term	**value**
$\dot{p}_n$	10.0
ρ	0.5
θ	1.0
δ_t	1.0
$\dot{\delta}_t$	1.0
$\dot{\theta}_d$	200

be found in [58].

The controller is tuned for low wind speeds, using the parameters given in Table 1 and in Table 2. As the UAV follows the ground speed reference with a bias, the actual ground speed is used for v_x^{stall} in (24). The start height for the deep-stall is $h = 200.0$ m and the parameter $\Delta_T = 5$ s in (7). The NMPC step length is $\Delta t = 0.05$ s, the horizon length is $N = 100$ and the discretization type is multiple shooting. One challenge in the simulations is the demand for a high sampling frequency due to the short NMPC step length, combined with an NMPC computation time that can be as high as 0.2 s in extreme cases. To overcome this challenge simulation is not run real-time, but slowed down to 5 times real time, although 2 times real time was adequate in many cases.

Table 2: Reference values for SITL with Aerosonde

Reference	**Cruise**	**Deep-Stall**
v_x	24.0 m/s	6.5 m/s
δ_t	0.62	0
γ_r	0^o	35.7^o
α	0^o	44.0^o

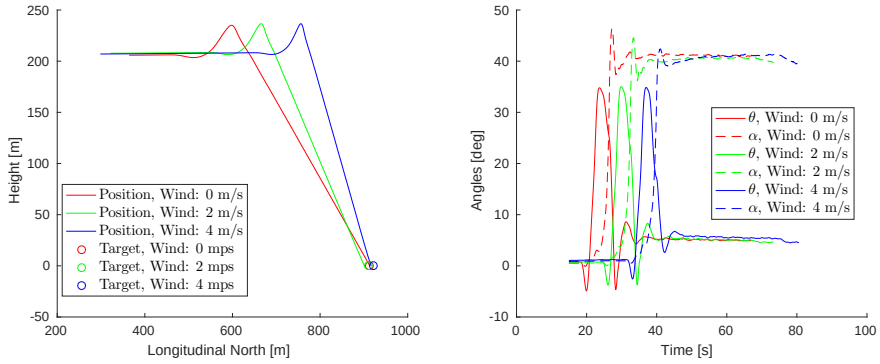
Table 3: Deep-stall landing results

Steady Wind		0 m/s	2 m/s	4 m/s
Longitudinal error	**Mean value**	6.2076 m	6.7386 m	3.6351 m
	Standard deviation	3.0446 m	3.4994 m	3.4978 m
Lateral error	**Mean value**	0.0464 m	0.0688 m	−0.2306 m
	Standard deviation	1.6313 m	1.5748 m	2.2647 m
Landing speed	**Mean value**	9.6700 m/s	8.4749 m/s	7.4619 m/s
	Standard deviation	0.0389 m/s	0.0282 m/s	0.0307 m/s

For each steady wind speed, the controller is first tuned to find d_2, using $e^{\mathrm{MAX}} = 10.0$. Then, using the autopilot's wind estimates subject to light turbulence, a larger error is tolerated and $e^{\mathrm{MAX}} = 50.0$. The results obtained for the deep-stall landing for various steady wind speeds with a light turbulence are shown in Fig. 6. For a small set of 10 simulations for each of three different wind speeds, with a light turbulence, the landing average precisions and velocities are shown in Table 3. To illustrate the disturbances a turbulence exerts on the NMPC, Fig. 7 contains data from one landing with known, constant wind and one landing with a light turbulence and wind estimations received by the autopilot. Fig. 8 shows the lateral error of simulations with different wind speeds and with same wind speed with and without turbulence and wind estimation.

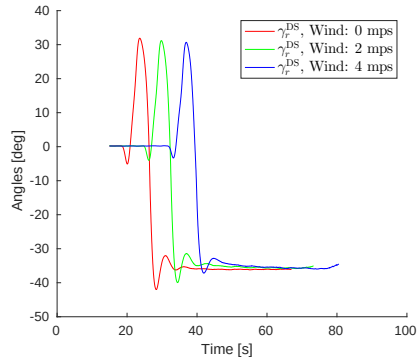
6 Discussion

The results from Table 3 give a good indication of the common precision and landing speed, although there are few samples. For 2 m/s wind speed we see that the UAV lands short of the target every time, which indicates that d_2 was too large. As the tuning of d_2 accepts target errors up to 10 m with only steady, known wind, the accepted error will be the expected mean value of all landing positions. According to Table 3, the longitudinal error clearly outweighs the lateral error, and future works could minimize this work further, for instance by tuning d_2 further. However, the current results mean that using the Aerosonde, which is not particularly intended for deep-stalling, allows landing in spaces with a radius of 10 meters in the considered wind conditions. In Fig. 6, we see that except for the position plots, which shows an increasingly steep path angle of the deep-stall landing, all curves show the same behaviour for the three wind speeds, just shifted in time. The time shift is necessary to achieve the steeper angle for higher wind speeds, in order to achieve the constant relative path angle. The controller is tuned to work with small wind speeds and light turbulence. The controller rejects light

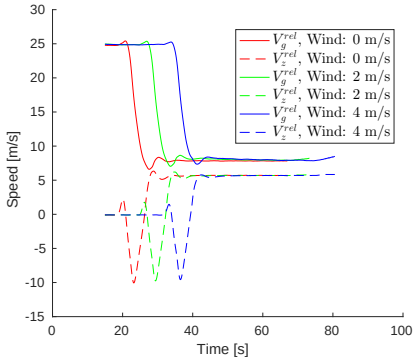


(a) Longitudinal position

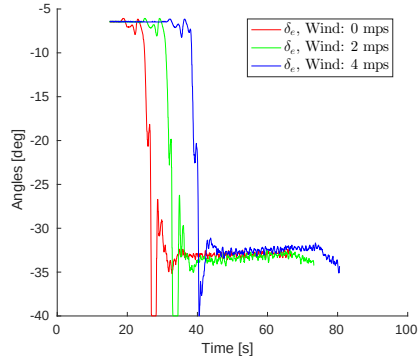
(b) Pitch angle and angle of attack



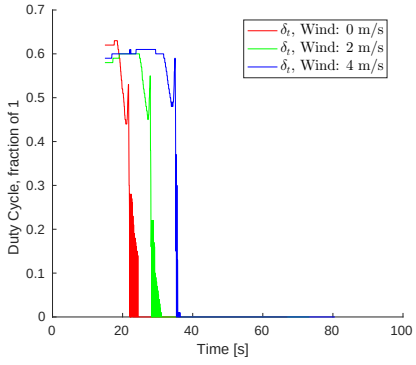
(c) Relative path angle



(d) Relative velocities in the inertial frame

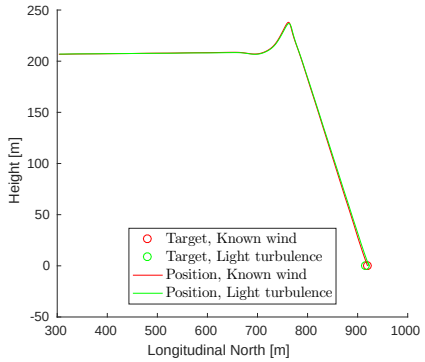


(e) Elevator deflection

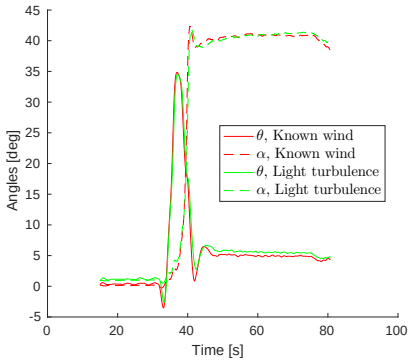


(f) Throttle usage

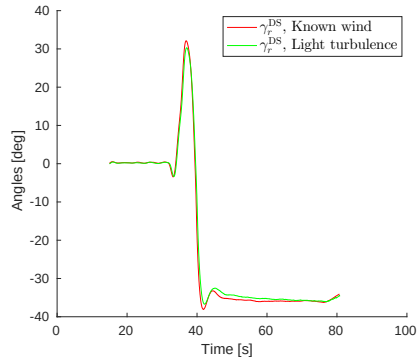
Fig. 6: Estimated variables of the UAV while controlled by an NMPC from cruise to deep-stall, for steady wind equal to 0 m/s, 2 m/s and 4 m/s, with a light turbulence.



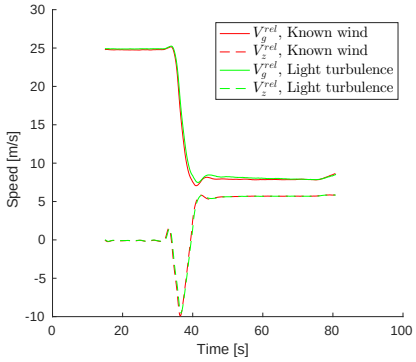
(a) Longitudinal position



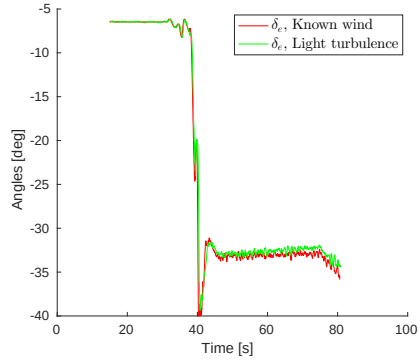
(b) Pitch angle and angle of attack



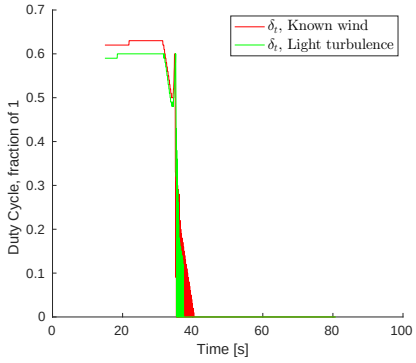
(c) Relative path angle



(d) Relative velocities in the inertial frame

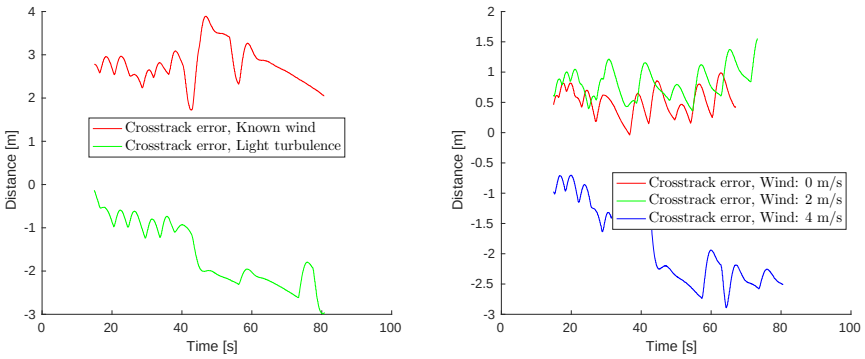


(e) Elevator deflection



(f) Throttle usage

Fig. 7: Estimated variables of the UAV while controlled by an NMPC from cruise to deep-stall, with 4 m/s wind and no or light turbulences.



(a) 4m/s wind speed. (b) Varying wind and a light turbulence

Fig. 8: Lateral error of the simulations

turbulence well. Simulations for poor wind estimates or high turbulence show that the UAV states do not converge in the deep-stall, and the controller thus predicts a too large target error, and is forced to abort the landing. The abortion of deep-stall landing is a subject for future work.

The prediction model used in this controller is a simple model of the kinematics and dynamics of a UAV and of the forces and moments acting on it. The model used by the simulator is not transparent, although it allows a lot of parameter specifications, and parameter identification has only been performed to a very small degree. The simulator model contains more information about the plane, which adds to the model mismatch between the prediction and simulation. The autopilot creates another layer on top of the simulation model, as it performs attitude control. The NMPC prediction model contains a very simple PID controller for the pitch, tuned with the same gains as the PID controller of the autopilot, but the autopilot controller is more complex with a roll compensation and an airspeed scaler. The airspeed scaler includes division by the airspeed, and could lead to an disproportional high gain in the controller for the low airspeeds of a stall, especially if there are oscillations involved. Another limitation of this is the use of decoupling between lateral and longitudinal control under for extreme and highly nonlinear manoeuvres.

To be able to control a UAV real-time, it is necessary to use an real-time iteration-based NMPC instead of e.g a sequential quadratic programming strategy. However, as there is a considerable modelling error, the RTI does not necessarily solve the problem to a sufficient convergence, even when it is warm-started with a fairly accurate solution from the previous time step. To increase the success of the NMPC, a high sampling frequency is used in this paper, which in turn demands a fast NMPC. The computation time and power is an issue of real-time control with an NMPC, and it is non-trivial to speed up the calculations. However, using an interior point solver and parallelization will speed up the computation times. 5 times real time is not far from real time, so the prospects are optimistic. Addi-

tionally, it will be wise to invest time in the modelling of the UAV, to ensure a model that is as realistic as possible and demands a lower sampling frequency.

7 Conclusion

The NMPC presented in this paper is able to guide a simulated UAV in a deep-stall landing to a target location with mean target error less than 10 m and mean landing speed less than 10 m/s. The conclusion of this paper is that it is possible to deep-stall land a fixed-wing UAV with an NMPC, but the results depend on a correct prediction model for the controller. It is therefore advisable to invest time in a precise model, as this will minimize problems.

ACKNOWLEDGMENT

This work is part of a project partly funded by the Research Council of Norway through the Centers of Excellence funding scheme, project number 223254. Centre for Autonomous Marine Operations and Systems (AMOS), Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), Trondheim, Norway

References

1. L. Sun, R.W. Beard, in *American Control Conference* (2013). DOI 10.1109/ACC.2014.6859132
2. C. Goerzen, Z. Kong, B. Mettler, Journal of Intelligent Robotic Systems: Theory and Applications (2010). DOI 10.1007/s10846-009-9383-1
3. A. Mcfadyen, L. Mejias, Progress in Aerospace Sciences (2016). DOI doi.org/10.1016/j.paerosci.2015.10.002
4. A. Gautam, P. Sujit, S. Saripalli, in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)* (2014). DOI 10.1109/ICUAS.2014.6842377
5. W. Kong, D. Zhang, X. Wang, Z. Xian, J. Zhang, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2013). DOI 10.1109/IROS.2013.6696776
6. P. Riseborough, in *5th Asian Control Conference* (2004)
7. M. Noor, M. Ismail, M. Khyasudeen, A. Shariffuddin, N. Kamel, S. Azzuhri, Frontiers in Artificial Intelligence and Applications **299** (2017). DOI 10.3233/978-1-61499-828-0-459
8. D. Zhang, X. Wang, Journal of Intelligent and Robotic Systems: Theory and Applications (2017). DOI 10.1007/s10846-017-0512-y
9. R. Skulstad, C. Syversen, M. Merz, N. Sokolova, T. Fossen, T. Johansen, IEEE Aerospace and Electronic Systems Magazine **30**(5), 18 (2015). DOI 10.1109/MAES.2015.7119821
10. K. Klausen, T.I. Fossen, T.A. Johansen, Journal of Field Robotics (2017). DOI DOI: 10.1002/rob.21772
11. H.J. Kim, M. Kim, H. Lim, C. Park, S. Yoon, D. Lee, H. Choi, G. Oh, J. Park, Y. Kim, IEEE/ASME Transactions on Mechatronics (2013). DOI 10.1109/TMECH.2013.2247411
12. M.D. White, G.E. Cooper, in *NASA Conference on Aircraft Operating Problems* (1965)
13. M.V. Cook, *Flight Dynamics Principles: A Linear Systems Approach to Aircraft Stability* (Butterworth-Heinemann, 1997)
14. I.H. Abbott, A.E. von Doenhoff, *Theory of Wing Sections* (Dover Publications Inc., 1959)
15. J.D. Anderson, *Fundamentals of Aerodynamics*, 2nd edn. (McGraw-Hill Inc., 1991)
16. A.C. Kermode, *Mechanics of Flight*, 11th edn. (Pearson Prentice Hall, 2006)
17. J.J. Bertin, R.M. Cummings, *Aerodynamics for Engineers*, 6th edn. (Person Education Limited, 2014)

18. W.F. Phillips, *Mechanics of Flight* (Wiley, 2004)
19. A.G. Sim, Flight characteristics of a manned, low-speed, controlled deep stall vehicle. Nasa technical memorandum, NASA Aeronautics and Space Administration (1984)
20. H. Tanaguchi, in *26th International Congress of the Aeronautical Sciences*, vol. 3 (2008), vol. 3, pp. 2498–2503
21. W. Crowther, K. Prassas, in *14th Bristol International Unmanned Air Vehicle Systems Conference* (1999)
22. J. Moore, R. Cory, R. Tedrake, Bioinspiration and Biomimetics (2014)
23. W. Pointner, G. Kotsis, P. Langthaler, M. Naderhim, in *2011 IEEE/AIAA 30th Digital Avionics Systems Conference (DASC)* (2011)
24. E. Kanellis, G. Vasov, Real time nmpc for fixed wing uav applications. Master's thesis, Aalborg university, Electronics and IT (2016)
25. S.H. Mathisen, T.I. Fossen, T.A. Johansen, in *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on* (IEEE, 2015). DOI 10.1109/ICUAS.2015.7152310
26. B. Cheng, Z. Guo, (2018), pp. 269–274. DOI 10.1109/CMAME.2017.8540179
27. H.Z.I. Khan, J. Rajput, S. Ahmed, J. Riaz, (2019). DOI 10.1109/IBCAST.2019.8666962
28. A. Waldock, C. Greatwood, F. Salama, T. Richardson, Journal of Intelligent and Robotic Systems: Theory and Applications (2018). DOI 10.1007/s10846-017-0696-1
29. T.A. Johansen, IEEE Systems Jounal (2017). DOI 10.1109/JSYST.2014.2368129
30. M. Diehl, H.J. Ferreau, N. Haverbeke, *Nonlinear Model Predictive Control* (Springer, 2009), chap. Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation, pp. 391–417. DOI 10.1007/978 − 3 − 642 − 01094 − 1_32
31. R. Findeisen, L. Imsland, F. Allgower, B.A. Foss, European Journal of Control (2003). DOI 10.3166/ejc.9.190-206
32. G. Frison, D.K.M. Kufoalor, L. Imsland, J.B. Jørgensen, in *IEEE Conference on Control Applications (CCA)* (2014). DOI 10.1109/CCA.2014.6981589
33. G. Frison, H.B. Sørensen, B. Dammann, J.B. Jørgensen, in *European Control Conference (ECC)* (2014). DOI 10.1109/ECC.2014.6862490
34. D.K.M. Kufoalor, S. Richter, L. Imsland, T.A. Johansen, M. Morari, G.O. Eikrem, in *2014 22nd Mediterranean Conference of Control and Automation (MED)* (2014). DOI 10.1109/MED.2014.6961399
35. S. Richter, C.N. Jones, M. Morari, IEEE Transactions on Automatic Control **57**(6), 1391 (2012). DOI 10.1109/TAC.2011.2176389
36. H.J. Ferreau, H.G. Bock, M. Diehl, International Journal of Robust and Nonlinear Control **18**(8), 816 (2008). DOI 10.1002/rnc.1251
37. J. Mattingley, S. Boyd, Optimization and Engineering **13**, 1 (2012). DOI 10.1007/s11081-011-9176-9
38. B. Houska, H.J. Ferreau, M. Diehl, Optimal Control Applications and Methods (2010). DOI 10.1002/oca.939
39. acados, https://docs.acados.org/index.html
40. R. Verschueren, G. Frison, D. Kouzoupis, N. van Duijkeren, A. Zanelli, R. Quirynen, M. Diehl, in *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control (NMPC)* (2018)
41. M. Vukov, A. Domahidi, H.J. Ferreau, M. Morari, M. Diehl, in *Proceedings of the 52nd IEEE Conference on Decision and Control* (2013)
42. B. Houska, H.J. Ferreau, M. Diehl, Automatica (2011). DOI 10.1016/j.automatica.2011.08.020
43. J.H. Kim, S. Sukkarieh, S. Wishart, *Field and Service Robotics* (Springer, 2006), *Springer Tracts in Advanced Robotics*, vol. 24, chap. Real-Time Navigation, Guidance, and Control of a UAV Using Low-Cost Sensors, pp. 299–309. DOI 10.1007/10991459_29
44. R.W. Beard, T.W. McLain, *Small Unmanned Aircraft Theory and Practice* (Princeton University Press, 2012)
45. M. Vukov, S. Gros, G. Horn, G. Frison, K. Geebelen, J. Jørgensen, J. Swevers, M. Diehl, Control Engineering Practice **45**, 64 (2015). DOI https://doi.org/10.1016/j.conengprac.2015.08.012. URL http://www.sciencedirect.com/science/article/pii/S0967066115300095
46. H.F. Erdogan, A. Kural, C. Ozsoy, Aircraft Engineering and Aerospace Technology **89**(2), 193 (2017). DOI http://dx.doi.org/10.1108/AEAT-03-2015-0074
47. S. Koo, S. Kim, J. Suk, IFAC-PapersOnLine pp. 59–64 (2015). DOI 10.1016/j.ifacol.2015.06.464

48. J. Gripp, U.P. Sampaio, in *International Conference on Unmanned Aircraft Systems (ICUAS)* (IEEE, 2014), pp. 1219–1224. DOI 10.1109/ICUAS.2014.6842378
49. S. Gros, M. Zanon, R. Quirynen, A. Bemporad, M. Diehl, International Journal of Control pp. 1–19 (2016). DOI http://dx.doi.org/10.1080/00207179.2016.1222553
50. L.D. Filippis, G. Guglieri, F.B. Quagliotti, Aircraft Engineering and Aerospace Technology: An International Journal **86**(3), 198 (2014). DOI http://dx.doi.org/10.1108/AEAT-01-2013-0016
51. D. Reinhardt, T.A. Johansen, in *International Conference on Unmanned Aircraft Systems (ICUAS)* (IEEE, 2019). DOI 10.1109/ICUAS.2019.8798229
52. S. Gros, R. Quirynen, M. Diehl, in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)* (2012). DOI 10.1109/CDC.2012.6426439
53. E. Skjong, S.A. Nundal, F.S. Leira, T.A. Johansen, in *International Conference on Unmanned Aircraft Systems (ICUAS)* (IEEE, 2015), pp. 904–913. DOI 10.1109/ICUAS.2015.7152377
54. M.A. Masri, S. Dbeis, M.A. Saba, Journal of Intelligent & Robotic Systems **86**(2), 255 (2017). DOI 10.1007/s10846-016-0455-8
55. F. Gavilan, R. Vazquez, S. Esteban, IFAC-PapersOnLine **48**(9), 132 (2015). DOI https://doi.org/10.1016/j.ifacol.2015.08.072
56. F. GAVILAN, R. VAZQUEZ, E.F. CAMACHO, IEEE Transactions on Aerospace and Electronic Systems **51**(3), 2406 (2015). DOI 10.1109/TAES.2015.140153
57. T. Stastny, R. Siegwart, in *International Conferenco on Unmanned Aircraft Systems (ICUAS)* (2018). DOI 10.1109/icuas.2018.8453377
58. S.H. Mathisen, K. Gryte, T.I. Fossen, T.A. Johansen, in *AIAA Infotech @ Aerospace* (AIAA Scitech Forum, 2016). DOI https://doi.org/10.2514/6.2016-0512
59. S.H. Mathisen, V. Grindheim, T.A. Johansen, in *IFAC-PapersOnline* (2017). DOI https://doi.org/10.1016/j.ifacol.2017.08.624
60. S.G. Mathisen, F.S. Leira, H.H. Helgesen, K. Gryte, T.A. Johansen, Autonomous Robots (2019). URL http : //folk.ntnu.no/torarnj/prec_drop.pdf
61. D.I. You, Y.D. Jung, S.W. Cho, H.M. Shin, S.H. Lee, D.H. Shim, in *AIAA Guidance, Navigation, and Control Conference* (2012). DOI 10.2514/6.2012-4674
62. J. Andersson, J. Åkesson, M. Diehl, *Recent Advances in Algorithmic Differentiation* (Springer, Berling, Heidelberg, 2012), chap. CasADi: A Symbolic Package for Automatic Differentiation and Optimal Control, pp. 297–307
63. J.A.E. Andersson, J. Gillis, G. Horn, J.B. Rawlings, M. Diehl, Mathematical Programming Computation (In Press, 2018)
64. CasADi, https://web.casadi.org/
65. QPOases, https://projects.coin-or.org/qpOASES
66. A.P. Zolich, T.A. Johansen, K.P. Cisek, K. Klausen, in *Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)* (2015). DOI 10.1109/RED-UAS.2015.7441026
67. J. Pinto, P. Dias, R. Martins, J. Fortuna, E. Marques, J. Sousa, in *MTS/IEEE OCEANS - Bergen* (2013). DOI 10.1109/OCEANS-Bergen.2013.6608148
68. MathWorks, Dryden wind turbulence model (continuous) (2015). Following Military Specification MIL-F-8785C
69. C. Ostowari, D. Naik, Post-stall wind tunnel data for naca 44xx series airfoil sections. Tech. rep., Solar Energy Research Institute (1985)