Non-linear Model Predictive Control for Guidance of a Fixed-Wing UAV in Precision Deep Stall Landing

Siri Holthe Mathisen, Thor I. Fossen and Tor A. Johansen

Abstract— **In order to achieve a high degree of operational flexibility, it is often required to recover small UAVs without infrastructure such as a runway. To help overcome this difficulty, deep stall landing can be used as a landing method on small space. The UAV is in a deep stall when the angle of attack is beyond the stall angle, and the condition causes the UAV to loose height rapidly. In the post-stall phase the angle of attack is controlled and the UAV can land with relatively low speed.**

In this article we focus on the highly non-linear longitudinal dynamics, and employ a non-linear model predictive control (MPC) to find a control sequence to guide a model of a fixed-wing UAV into a deep stall to land at a given location and given path angle with minimum speed.

Index Terms— **Deep Stall, Optimal Control, Multiple Shooting, MPC, Landing, UAV**

I. INTRODUCTION

Unmanned aerial vehicles (UAVs) are used for tasks involving human risk, hazardous areas or repetitive tasks. It is common to divide between fixed-wing and rotary-wing UAVs, where the fixed-wing UAV is the type that will be considered in this article. The main advantage of fixed-wing UAVs is their ability to perform long range missions. A longer flight time makes them well suited for autonomous missions involving surveillance or transport, where the area of interest is far away from the point of departure. However, this area of interest might be too far away from land, forcing the need for take-off and landing on a ship, or it has inconvenient landing conditions, without space for a landing field. It is possible to launch a small fixed-wing UAV from a catapult [1],[2] or by hand [3], and bypass the need for a runway for the take-off. To recover the UAV without a runway, arrest systems can be used and the most common option among these is to use net landing, see [2] and references therein. However, this requires more equipment, which complicates the operation and can be a limiting factor on UAV missions. Deep stall landing is another method that can be used to land a UAV withouth a runway and without additional equipment. It exploits the reduced lift and increased drag of a high angle of attack in order to reduce speed during landing.

Centre for Autonomous Marine Operations and Systems (AMOS), Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), Trondheim, Norway

In the articles [4] and [5] the aerodynamics of a deep stall are discussed, and in [6] an analysis of deep stall landing for a small UAV is made. To go into a deep stall means to increase the angle of attack beyond the stall angle. In the deep stall, the lift coefficient decreases and the drag coefficient increases [4], [5]. This causes the UAV to fall, but at a relatively low speed. The steep path angle of the UAV reduces the need of a landing field and enables the UAV to land on a small area. In the article [1], a hybrid system for safe deep stall landing of a delta wing longitudinal model is described, with an embedded safety mechanism that can abort the landing if necessary. The authors describe how this landing method is particularly convenient for small autonomous flying vehicles, but it is difficult to predict where the UAV will land, and when to initiate the landing. A limiting factor is that wind disturbance rejection performance may be low as increased drag leads to more sensitivity to wind, and reduced speed reduces control surface effectiveness.

This article describes a case study where a non-linear constrained Model Predictive Control (MPC) is used to control the path angle of a fixed-wing UAV in deep stall. MPC is chosen since it can effectively handle non-linearities, constraints, and find optimal solutions. Though MPC has been used for various applications, the use of MPC to control a deep stall landing appears to be a novel idea, not reported in the literature. Through simulations of a longitudinal model of a Aerosonde UAV [7], which has a mass of 13.5 kg and a wing span of 2.9 meters, an algorithm is developed that guides the UAV in deep stall to a given landing point. The algorithm consists of two steps - cruise and landing - and controls the path angle of the UAV to reach the given touch down location. In the landing phase, the UAV aims to follow a constant path angle down to the landing spot. In addition to tracking the path given, the speed of the UAV is taken into consideration. To ensure a take down with minimum impact and risk, the speed of the UAV is minimized during the landing. In this algorithm, the ratio between the vertical and the horizontal velocity of the UAV is given by the path angle needed to reach the given point of impact with minimal speed. The aim in this scenario is to land the UAV on a small place, like the deck of a ship, with good accuracy and minimal speed. It will risk some damage on the UAV, but this is minimized by a minimal landing speed. To solve the numerical optimization problem, an open-source software package called CasADi is used in a Python environment. CasADi is a tool that, among other similar areas of use, can make efficient implementations of algorithms for non-linear optimization and calculations of the numerical solutions of these optimization problems [8]. A possible platform for real time implementation of non-linear MPC is ACADO [9].

The outline of the rest of the article is as follows: Part II presents the longitudinal 3-DOF model for the UAV. Part III reviews the theory of MPC and non-linear MPC, the advantages of warm starting and the theory of direct multiple shooting. Part IV defines the non-linear optimal control problem derived from the model and the need to land in a given location. The objective function is time varying and balanced between several minimization objectives and there are several constraints present. Part V discusses the simulation results and the interpretations of these results. Part VI presents the conclusion of this article.

II. Model

The kinematics and kinetics of a 6-DOF model for a UAV is given in [7, p.36]. The longitudinal model has 3 DOF:

$$\begin{bmatrix} \dot{p}_n \\ \dot{p}_d \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} u \\ w \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} \dot{u} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} -qw \\ qu \end{bmatrix} + \frac{1}{m} \begin{bmatrix} f_x \\ f_z \end{bmatrix} \tag{2}$$

$$\dot{\theta} = q, \tag{3}$$

$$\dot{q} = \frac{1}{J_y} m_a. \tag{4}$$

Considering a straight path, $\mathbf{p} = [p_n \; p_d]^\top$ is the position vector of the UAV containing longitudinal and vertical components, u and w are the velocity components in the body frame, $\mathbf{f} = [f_x \; f_z]^\top$ is the force vector acting on the UAV in the body frame, θ is the pitch angle, q is the pitch rate, J_y and m_a are respectively the moment of inertia about the y axis and the pitching torque, which acts on the UAV about the y axis, and m is the mass of the UAV.

The forces and torques that act on the UAV in the 6 DOF model are elaborated in [7, p.57]. In the longitudinal model, only the x and z components of the total forces are needed, and only the torque about the y axis. These forces and this torque are the only ones mentioned in the previous equations, and they are all given in the body frame. The equations for the forces are written as:

$$\mathbf{f} = \begin{bmatrix} f_x \\ f_z \end{bmatrix} = \mathbf{f}_g + \mathbf{f}_a + \mathbf{f}_p, \tag{5}$$

where the gravitational force is given by:

$$\mathbf{f}_g = \begin{bmatrix} -mg\sin(\theta) \\ mg\cos(\theta) \end{bmatrix}, \tag{6}$$

where g is the acceleration of gravity. The aerodynamic force is given by:

$$\mathbf{f}_a = \frac{1}{2}\rho V_a^2 S \begin{bmatrix} C_X(\alpha) + C_{X_q}(\alpha)\frac{c}{2V_a}q + C_{X_{\delta_e}}(\alpha)\delta_e \\ C_Z(\alpha) + C_{Z_q}(\alpha)\frac{c}{2V_a}q + C_{Z_{\delta_e}}(\alpha)\delta_e \end{bmatrix}, \tag{7}$$

where $C_{X_q}(\alpha)$, $C_{X_{\delta_e}}(\alpha)$, $C_{Z_q}(\alpha)$ and $C_{Z_{\delta_e}}(\alpha)$ are functions of the angle of attack α and constant parameters for the specific UAV [7, p.58]. ρ is the air density, V_a is the airspeed, S is the surface area of the wing and c is the mean aerodynamic chord of the wing. δ_e is the control signal for the elevator deflection of the UAV.

$$C_X(\alpha) \triangleq -C_D(\alpha)\cos\alpha + C_L(\alpha)\sin\alpha, \quad (8)$$

$$C_Z(\alpha) \triangleq -C_D(\alpha)\sin\alpha - C_L(\alpha)\cos\alpha, \quad (9)$$

$$C_L(\alpha) = (1-\sigma(\alpha))[C_{L_0} + C_{L_\alpha}\alpha] + \sigma(\alpha) \quad (10)$$
$$[2\mathrm{sgn}(\alpha)\sin^2\alpha \ \cos\alpha],$$

$$\sigma(\alpha) = \frac{1 + e^{-M(\alpha-\alpha_0)} + e^{M(\alpha+\alpha_0)}}{(1+e^{-M(\alpha-\alpha_0)})(1+e^{M(\alpha+\alpha_0)})}, \quad (11)$$

α_0 and M are positive constants and C_{L_0}, C_{L_α} and C_{D_p} are parameters specified for the UAV model used. C_L is the lift coefficient. In 1985, [10] published a book which contains aerodynamic data for different airfoils of the NACA 44XX series. As the wing aspect ratio of the Aerosonde is 15.24, the best fitting drag coefficient graphs are the ones for an aspect ratio of 14.88, which are marked ∞ in the article because of the section of the wind tunnel. A roughly fitted polynomial for the drag coefficient as a function of the angle of attack is:

$$C_D(\alpha) = -1.4238 \times 10^{-11}\alpha^6 + 5.1773 \times 10^{-9}\alpha^5$$
$$-6.3337 \times 10^{-7}\alpha^4 + 2.3320 \times 10^{-5}\alpha^3$$
$$+5.2458 \times 10^{-4}\alpha^2 - 0.0073\alpha + 0.0275 \quad (12)$$

The lift and drag coefficients for different values of α are shown in Fig. 1. The deep stall occures when the lift coefficient decreases and the drag coefficient increases, where the drag coefficient is largest when $\alpha = 90^o$.

The propulsion force is given by:

$$\mathbf{f}_p = \frac{1}{2}\rho S_{prop} C_{prop} \begin{bmatrix} (k_{motor}\delta_t)^2 - V_a^2 \\ 0. \end{bmatrix} \quad (13)$$

where δ_t is the control signal for the throttle deflection. The pitching moment on the UAV is given by:

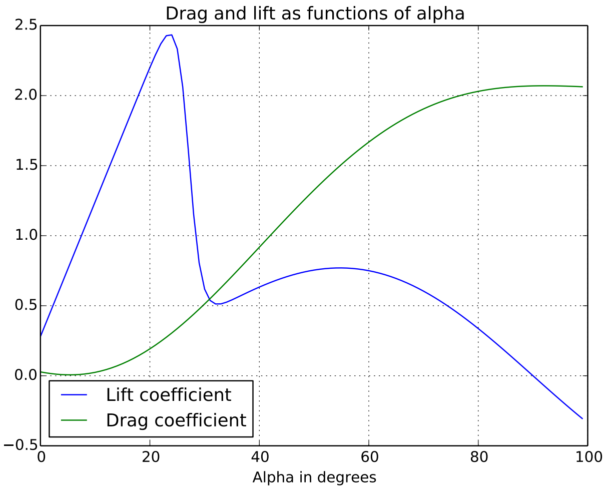


Fig. 1: CL and CD as functions of α

$$m_a = \frac{1}{2}\rho V_a^2 Sc(C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{c}{2V_a}q + C_{m_{\delta_e}}\delta_e), \quad (14)$$

where C_{m_0}, C_{m_α}, C_{m_q} and $C_{m_{\delta_e}}$ are parameters specified for the UAV model used.

The longitudinal model accounts for wind, just like the full 6-DOF model by [7]. The wind velocity vector in the body frame is given by:

$$\mathbf{v}_w^b = \begin{bmatrix} u_w \\ w_w \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} w_{n_s} \\ w_{d_s} \end{bmatrix} + \begin{bmatrix} u_{w_g} \\ w_{w_g} \end{bmatrix}, \quad (15)$$

which is a sum of the gust component in the body frame (u_{w_g}, w_{w_g}) and the inertial steady component (w_{n_s}, w_{d_s}), rotated into the body frame [7]. The gust component is simulated using the Dryden wind turbulence model described in [11]. This model represents the turbulence as a spectral density and simulates the turbulence as the output of a linear filter with white noise as input. The spectral density of the turbulence influencing the UAV varies with the speed of the vehicle and the altitude.

The velocity vector is given in (2), and taking the wind velocity into account, the relative velocity vector in the body frame is given:

$$\mathbf{v}_a = \begin{bmatrix} u_r \\ w_r \end{bmatrix} = \begin{bmatrix} u - u_w \\ w - w_w \end{bmatrix}. \quad (16)$$

The angle of attack, α, and the airspeed, V_a, are then given by:

$$\alpha = \tan^{-1}\left(\frac{w_r}{u_r}\right) \tag{17}$$

and

$$V_a = \sqrt{u_r^2 + w_r^2}. \tag{18}$$

A vector $\mathbf{u}$ is formed with the longitudinal control variables, δ_e, and δ_t. The control signal δ_t is a pulse-width-modulated command, and should be between $0-1$.

$$\mathbf{u} = [\delta_e \ \delta_t]^\top \tag{19}$$

All states from (1), (2), (3) and (4) are placed in a state vector:

$$\mathbf{x} = [p_n \ p_d \ u \ w \ \theta \ q]^\top \tag{20}$$

III. Methods

A. Model Predictive Control

This section refers in general to [12] and [13]. MPC is a controller that generates a sequence of controls over a time horizon. For each time step, an optimization problem is solved and a control plan for the prediction horizon is generated. Only the first control action from the plan is used as a control input to the system, and the response of the system is given as feedback to the controller in the next time step. Then the optimization is repeated and the prediction horizon is moved one step further. In a linear MPC, the objective function is assumed to be quadratic and the process model is linear. This means that the optimization problem can be solved with convex quadratic programming (QP). In the non-linear case, with a general problem as given in (21)–(25), convex QP is no longer an option, as the problem is no longer convex.

$$\min_{\mathbf{x}(\cdot),\mathbf{u}(\cdot)} \int_{t_0}^{t_0+T} F(\mathbf{x}(t),\mathbf{u}(t))\mathrm{d}t + E(\mathbf{x}(T)) \tag{21}$$

subject to

$$\mathbf{x}(t_0) = \mathbf{x}_0, \tag{22}$$

$$\dot{\mathbf{x}}(t) - f(\mathbf{x}(t),\mathbf{u}(t)) = 0, \tag{23}$$

$$\mathbf{u}_{min} \le \mathbf{u}(t) \le \mathbf{u}_{max}, \tag{24}$$

$$\mathbf{x}_{min} \le \mathbf{x}(t) \le \mathbf{x}_{max}, \forall t \in [t_0, t_0+T]. \tag{25}$$

To solve the non-linear MPC, a common method is to transform the optimal control problem into a non-linear program (NLP). The direct methods can be characterized as "first discretize, then optimize" [14].

B. Direct Multiple Shooting

This section refers in general to [12], [13] and [15]. In both direct single shooting and direct multiple shooting, the control horizon of length T is divided into N discrete control intervals, with a corresponding piecewise constant control. The difference between the two shooting methods is that while single shooting solves the intervals sequentially, the multiple shooting solves the problem simultaneously: Single shooting solves one single ODE which is parametrized by the complete input sequence. Multiple shooting, however, solves one ODE for each time interval in the sequence and has an additional constraint, demanding that solutions of each time interval should be linked to the next. Multiple shooting is illustrated in Fig. 2. The NLP in multiple shooting problem is given in (26)–(29), where $F(\mathbf{s}_i,\mathbf{q}_i)$ is the objective function at time step i, $\mathbf{s}_i$ is the optimization variable for the states, $\mathbf{q}$ is the optimization variable for the control and $E(\mathbf{s}_N)$ is the end term of the objective, corresponding to $E(\mathbf{x}(T))$ in 21.

$$\min_{\mathbf{s},\mathbf{q}} \sum_{i=0}^{N-1} F_i(\mathbf{s}_i,\mathbf{q}_i) + E(\mathbf{s}_N) \tag{26}$$

subject to

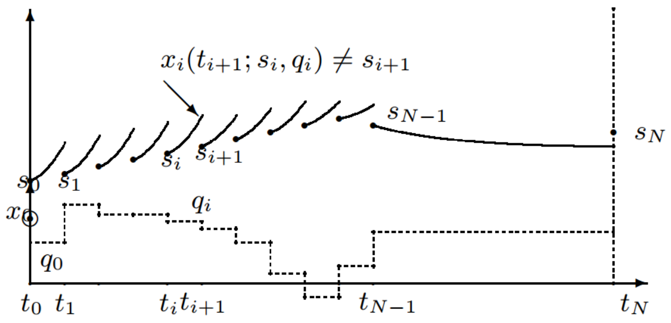
$$\mathbf{x}_0 - \mathbf{s}_0 = 0, \tag{27}$$

$$\mathbf{x}_i(t_{i+1};\mathbf{s}_i,\mathbf{q}_i) - \mathbf{s}_{i+1} = 0, \quad i = 0,...,N-1, \tag{28}$$

$$h(\mathbf{s}_i,\mathbf{q}_i) \le 0, \quad i = 0,...,N, \tag{29}$$

Equation (27) shows the initial value, which affects all other variables and controls as each consecutive state is a function of the previous state and the previous control. Equation (28) is the continuity constraint, and is necessary as each optimization interval is dependent on the previous one. Equation (29) represents the discretized path constraint.

Multiple shooting is chosen over single shooting because it is a simultaneous approach, which



$x_i(t_{i+1}; s_i, q_i) \neq s_{i+1}$
s_{N-1}
s_N
s_0
s_1
s_i
s_{i+1}
x_0
q_i
q_0
t_0 t_1 $t_i t_{i+1}$ t_{N-1} t_N

Fig. 2: Illustration of the control intervals and the optimization variables at iteration i [15].

means that the optimization variables contain not only the control variables, but also the state of the model in all control steps of the trajectory. That makes warm starting easier, meaning that the solver is initialized with the output variables of the previous optimization. Besides this obvious advantage, it is easier to add hard constraints on the states when they are included as optimization variables, which is the case for multiple shooting. Since the optimization method used is a *direct approach*, it means that it commences with a discretization and then optimizes the discrete model [14, p.77]. The discretization is performed with an Explicit Runge-Kutta algorithm of 4th order (RK4).

IV. Problem Definition

The main objective of this optimization is to land the UAV in the given location, following a given path angle, at a minimal speed. This is achieved by cruising the UAV at a constant altitude for a certain distance, and then guide the UAV to the landing location. This is shown in Fig. 3, where $p1$ is the initial position, $p2$ is the position where the cruise is ended and the landing is initiated and $p3$ is the landing location. $p2$ is calculated by first choosing the desired initial landing path angle, $\hat{\gamma}$, which is a negative angle. The next step is to intersect the line going at angle $-\hat{\gamma}$ from $p3$, with the horizontal line going from $p1$. This means that the UAV will cruise at altitude h until the distance in the x-direction is equal to $d = h/\tan(-\hat{\gamma})$. Then it will initiate the landing.

To generate a sequence of control that optimizes the trajectory of the UAV with respect to the

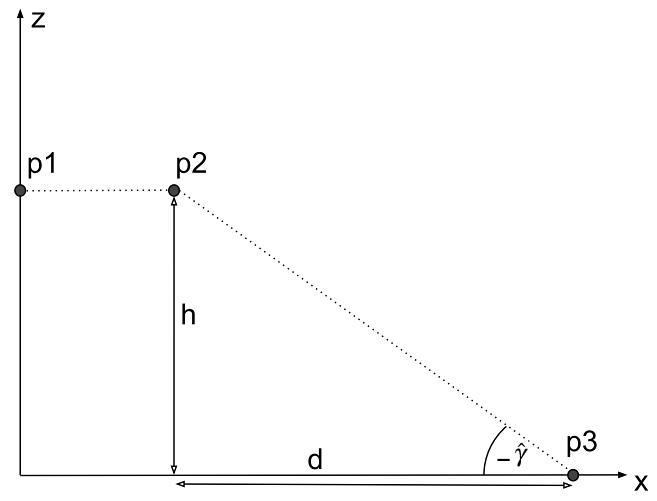


Fig. 3: The landing procedure: Cruise from $p1$ to $p2$, then land by tracking $p3$.

objective, a finite horizon optimization problem is represented by following equations:

$$\min_{\mathbf{x},\mathbf{u}} \sum_{k=0}^{N-1} F(\mathbf{x}_k)^T Q_1^T Q_1 F(\mathbf{x}_k) + (\mathbf{u}_{k+1} - \mathbf{u}_k)^T R^T R(\mathbf{u}_{k+1} - \mathbf{u}_k) + E(\mathbf{x}_N)^T Q_2^T Q_2 E(\mathbf{x}_N), \tag{30}$$

subject to

$$\mathbf{x}_{k+1} = f_d(\mathbf{x}_k, \mathbf{u}_k) \tag{31}$$

$$\text{given } \mathbf{x}_0 \tag{32}$$

where functions F and E in (30) vary with the objective of control and f_d is the right-hand side of the discretized differential equation for the model. $\mathbf{x}$ and $\mathbf{u}$ are the state and control variables of optimization. In addition, certain bounds were introduced to the variables to prevent them from challenging the physics of the UAV model. The absolute value of the pitch angle cannot exceed 90^o and there is a bound on the control signals. Besides, to prevent the angle of attack from containing a singularity, it has been limited within 60^o in absolute value. A bound for the the body velocity is introduced, limiting both u and w to 25 m/s.

$$\frac{-\pi}{2} < \theta < \frac{\pi}{2} \tag{33}$$

$$\frac{-\pi}{6} < \delta_e < \frac{\pi}{6} \tag{34}$$

$$-\frac{\pi}{3} < \alpha < \frac{\pi}{3} \tag{35}$$

$$0 \leq \delta_t \leq 1 \tag{36}$$

$$-25m/s \leq u, v \leq 25m/s \tag{37}$$

Since there is a strict limit on α, it is also included as an optimization variable and requires a second constraint:

$$\text{subject to}$$
$$\alpha_k = \tan^{-1}\left(\frac{w_r}{u_r}\right) \tag{38}$$

Since α is upwards constrained to be less than 60^o, this will necessarily affect the drag coefficient, which is dependent on alpha. According to Fig. 1, the maximum value of the drag coefficient occurs approximately at 85 degrees, which means that the constraint is active as it prevents the drag coefficient to reach its maximum.

To cruise the UAV, the difference between the UAV's altitude and its initial altitude h is minimized. At the same time, the change in the control is minimized:

$$\begin{aligned} \min_{x,u} \sum_{k=0}^{N-1} & (p_{n_k} - h)Q_1^2(p_{n_k} - h) + \alpha_k Q_2^2 \alpha_k \\ & + (\mathbf{u}_{k+1} - \mathbf{u}_k)^T R^T R(\mathbf{u}_{k+1} - \mathbf{u}_k) \\ & + (p_{n_N} - h)Q_3^2(p_{n_N} - h) + \alpha_N Q_4^2 \alpha_N, \\ & Q_1 = Q_2 = Q_3 = Q_4 = 10,\ R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{39}$$

Let the UAV's path angle be written as γ and the desired path angle as $\hat{\gamma}$. The path angle of the UAV is the angle with which it moves relative to the ground. It can be calculated by:

$$\gamma = -\tan^{-1}\left(\frac{\dot{p_d}}{\dot{p_n}}\right) \tag{40}$$

where $\dot{p_d}$ and $\dot{p_n}$ are found from (1). To land the UAV, the path angle is controlled to be as close as possible to $\hat{\gamma}$, which is the path angle from the UAV to the target.

$$\hat{\gamma} = -\tan^{-1}\left(\frac{p_d - \text{target}_z}{p_n - \text{target}_x}\right) \tag{41}$$

In addition, some weight on the speed is imposed:

$$\begin{aligned} \min_{x,u} \sum_{k=0}^{N-1} & (\gamma_k - \hat{\gamma})Q_1^2(\gamma_k - \hat{\gamma}) + V_k Q_2^2 V_k \\ & + (\mathbf{u}_{k+1} - \mathbf{u}_k)^T R^T R(\mathbf{u}_{k+1} - \mathbf{u}_k) \\ & + (\gamma_N - \hat{\gamma})Q_3^2(\gamma_N - \hat{\gamma}) + V_N Q_4^2 V_N, \\ & V_k = \sqrt{u_k^2 + w_k^2},\quad k = 0, ..., N, \\ & R = [R_1\ R_2]^\top \end{aligned} \tag{42}$$

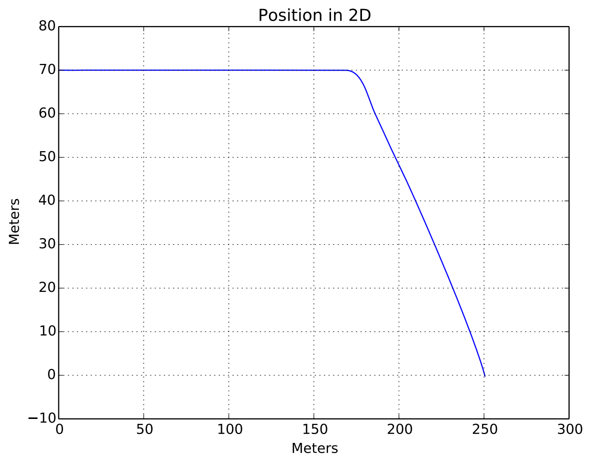
where the size of Q_1, Q_2, Q_3, Q_4 and R depend on the size of $\hat{\gamma}$ in order to minimize the objective function correctly. The deep stall landing is continued until the altitude of the UAV reaches the altitude of the target.

V. Simulation

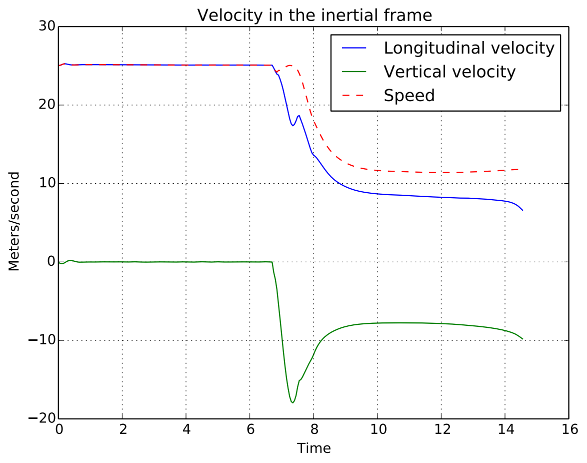
The simulation program is written in python, aided by the libraries *numpy* [16] and *CasADi* [8]. An optimization horizon of $T = 2$ seconds is chosen, and the number of equidistant control intervals is $N = 40$, which leads to a corresponding discretization interval of 50 ms. To perform the non-linear optimization, CasADi uses the non-linear program solver NlpSolver. For more details on the CasADi framework, see [8]. In the initialization of the program, the optimization variables are declared as $\mathbf{X} = [\mathbf{x}_0\ \alpha_0\ \mathbf{u}_0\ \mathbf{x}_1\ \alpha_1\ \mathbf{u}_1\ \ldots\ \mathbf{x}_N\ \alpha_N\ \mathbf{u}_N\ \mathbf{x}_{N+1}\ \alpha_{N+1}]$ and the boundaries of the variables are stated in (33), (34), (35), (36) and (37). Besides, an initial guess for the optimization variables is given. This is made by repeating the initial value of the state variables, α and the control variables, $\mathbf{x}_0$, α_0 and $\mathbf{u}_0$, N times and then adding a last occurrence of $\mathbf{x}_0$ and α_0:

$$\mathbf{x}_0 = \begin{bmatrix} 0 \\ -70 \\ 25 \\ 1.32 \\ 0.05 \\ 0 \end{bmatrix}, \mathbf{u}_0 = \begin{bmatrix} -0.08 \\ 0.33 \end{bmatrix}, \alpha_0 = 0.05 \tag{43}$$

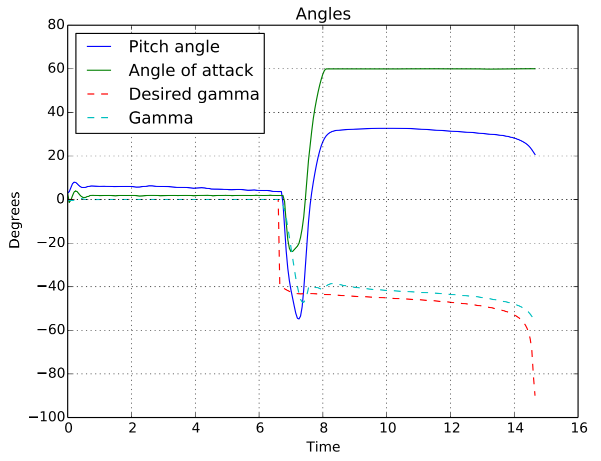
The target landing position is given by two coordinates in the longitudinal coordinate system: $target_x$ and $target_z$. The position is set to $(250, 0)$.



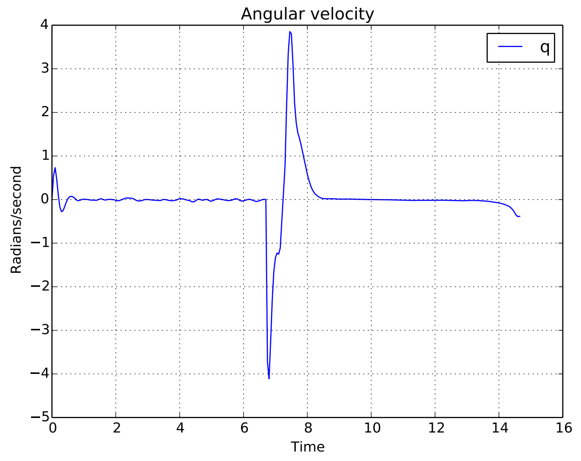
(a) Position in two dimensions



(b) Velocity in the inertial frame

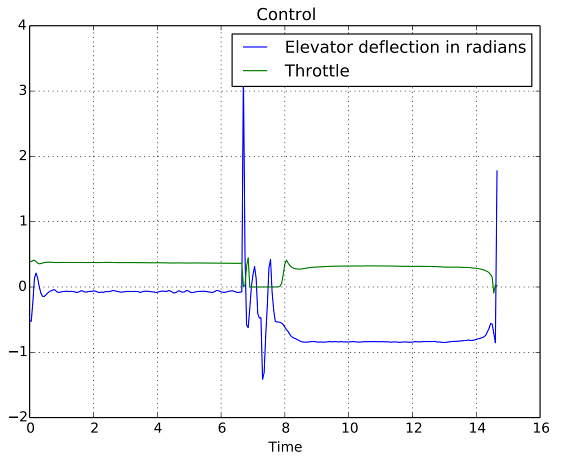


(c) Angles

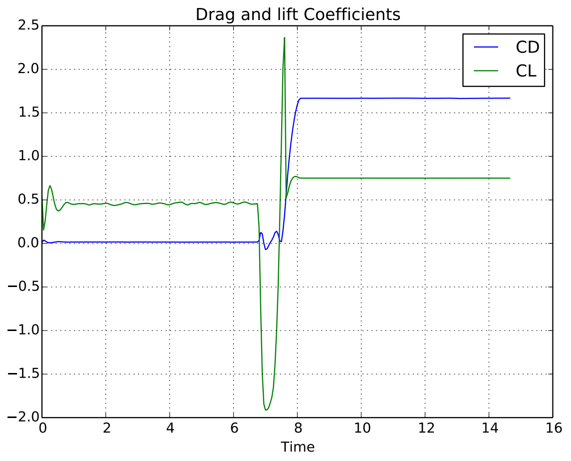


(d) Angular Velocity

Fig. 4: The position and inertial velocity, angles and angular velocity of the UAV during the deep stall landing.



(a) The control signals



(b) The drag and lift coefficients on the UAV

Fig. 5: The the control values, drag and lift coefficients of the UAV during the deep stall landing.

TABLE I: $\hat{\gamma}$ weighing parameters

$\hat{\gamma}$	Q_1	Q_2	$R_1 = R_2$	Deviation between landing position and $target_x$
30^o	400	1	40	0.00984 m
40^o	600	1	60	0.44345 m
50^o	700	1	70	0.68845 m
60^o	200	1	20	5.47832 m

The steady component of the wind is constant throughout the simulation, but the gust follows the Dryden turbulence model, as follows:

$$\begin{bmatrix} w_{n_s} \\ w_{d_s} \end{bmatrix} = \begin{bmatrix} -3 \\ 2 \end{bmatrix}, \begin{bmatrix} u_{w_g} \\ w_{w_g} \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} \tag{44}$$

Although the steady component is kept constant throughout the simulation, the gust component varies with each time step. After the optimization is performed with the present gust values, the gust is updated with a simulation of the Dryden model, and this new gust value is used in the simulation of the UAV model to iterate one time step. This way, the measurement delay of a real UAV's wind measurements is simulated. The consequence is that the simulation of the UAV model will not give exactly the same values as the optimized values, although the same control signals are used. This is an unknown disturbance to the system, and can cause results which are not perfect. An average performance over 10 tests is presented later in this section.

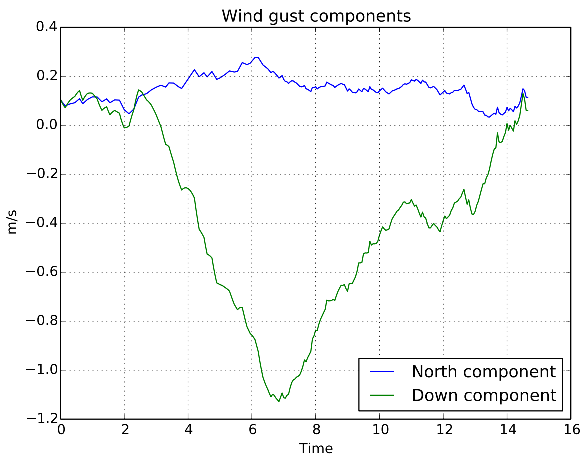
The first optimization part of the program consists of cruising in a constant height from $p1$ to $p2$ as illustrated in Fig. 3. Point $p2$ is defined by quantifying $\hat{\gamma}$ and letting the line from $p3$ intersect the horizontal line from $p1$. The value of $\hat{\gamma}$, for which the model has been successfully tested to give deep stall measurements, can be between -30^o and -60^o, although the controller also works very well for $-30^o < \hat{\gamma} < 0$, which results in no deep stall, but still low speed landing. The plots included and discussed in this article use a reference path angle of $\hat{\gamma} = -40^o$. The relationship between $\hat{\gamma}$ and the weighing parameters that tracks the desired $\hat{\gamma}$ best and at the same time keep the UAV in deep stall, are given in Table I.

In the cruising phase, an optimization with the NlpSolver is repeated, where the objective function is as given in (39) and the constraints are as given in (31). In this optimization the *IPOPT* [17] solver is used. The maximum number of iterations is 100, to prevent a long computation time. Every time the optimization is repeated, it is warm started, using the output from the last optimization as the initial value of the next optimization, only shifted one position.

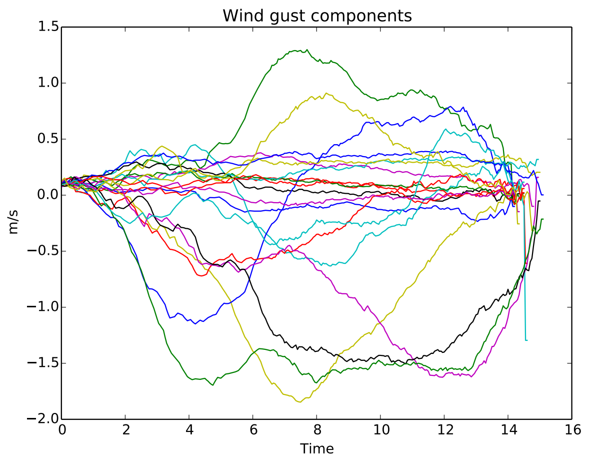
After the UAV has reached $p2$, the MPC switches objective. The NlpSolver is initiated again just like in the cruising part, only with an objective as given in (42). For the simulation whose plots are included in this article, the weighing parameters are set to $Q_1 = 600$, $Q_2 = 1$ and $R = 60$.

The simulated behaviour of the UAV is plotted in Figs. 4 and 5, with the gust component of the wind plotted in Fig. 6a. The average number of iterations for the cruising phase was 9 and the average computation time on the optimizations was 0.223 seconds. The landing phase found the optimal solution at every optimization, using an average of 20 iterations and in total 154 optimizations. The average time spent on optimization for the landing phase was 0.584 seconds. The processor used for these optimizations is a 4^{th} gen Intel® Core™ i7-4610M. It is worth mentioning that the position is plotted with *altitude* instead of the usual *down* variable, simply by inverting it and using the ground as zero. The velocity in the inertial frame, shown in Fig. 4b, therefore has altitude velocity, showing the change of altitude, instead of down velocity with the change of the down variable.

To see how much the stochastic variables of the wind gust influence the end results, 10 simulations were performed with the same parameters as mentioned above. The deviation between the landing position and the target had an average of 0.440 meters and standard deviation of 0.008 meters, and the landing speed had an average of 11.8 m/s and a standard deviation of 0.212 m/s, flying deep stall throughout the landing phase. The



(a) Gust component of the wind



(b) The two wind gust components of 10 simulations

Fig. 6: The gust components of the wind during one simulation and 10 simulations

gust components of these simulations is shown in Fig. 6b.

A. *Discussion of the Results*

Already when the path angle is -60^o, the deviation between the landing position of the UAV and the target position is significant, and when the landing speed and path angle are higher, the target tracking is no longer working, giving an oscillating angle of attack.

According to Fig. 4a, the landing position was successfully tracked with $\hat{\gamma}$. The landing position error was approximately 0.43 meters, which is reasonably small compared with the length of the UAV. At the landing, the speed is approximately 11.82 m/s, which is a reduction of more than 50% from the cruising speed. The longitudinal component of the inertial velocity is 6.7 m/s at the touch down moment, and the vertical velocity is 9.75 m/s, pointing downwards. The speed is relatively constant, but increasing slightly at the end, due to an increasing difference between desired path angle and real path angle. The wind can influence the inertial speed of the UAV a lot if the magnitude of the wind is large, pushing the UAV in the direction of the wind. If the UAV flies against the wind, this effect is exploited to reduce the speed of the UAV. The gust, however, is not known by the model at each time instance, and therefore influences optimization of the UAV, causing an error in the path angle tracking. However, as the previous section says, in average, the differences are minimal. In [1] and [6], one characteristic of the deep stall landing was that the longitudinal velocity was much smaller than the vertical velocity. In the simulations presented in this paper, this difference is not that remarkable, since a specific path angle has to be followed, thus forcing a relationship between the velocity components.

When the cruising phase is done, the plots show a brief but rather unsteady transition to the landing phase. Both the pitch angle and the angle of attack in Fig. 4c, the pitch rate q in Fig. 4d and the elevator deflection in Fig. 5a show clear signs of this. With a different tuning of the weights in the optimization, this could have been smoothed, but that would have sacrificed the tracking of the desired path angle and minimizing the speed of the UAV later. A good solution for this problem would be to diverge from using two phases, first cruise and then landing, and instead follow a smooth transition from a constant altitude flight until the landing is finished.

Looking at the control plot in Fig. 5a, the elevator deflection is smooth and does not vary much after the unsteady initial transient phase. The throttle is smoother than the elevator deflection and does not vary much. It is saturated in zero for a short while during the landing, corresponding to the part in Fig. 4c where the actual path angle oscillates slightly, from approximately 6.5 seconds to 8 seconds. The oscillations in the elevator deflection results in oscillating angles in Fig. 4c, angular velocity in Fig. 4d and oscillating lift in Fig. 5b in the transient phase. The last few iterations, both control signals deviate from their stable positions, which reflects the moment when the altitude of the UAV reaches zero and the deviation between γ and $\hat{\gamma}$ gets large. The simulation is stopped only when the altitude crosses zero. This gives an end path angle that is larger than 90^o, and magnifies the deviation between desired and real path angle and demands a high elevator deflection.

Apart from the transient phase, the angle of attack is constantly high in Fig. 4c, roughly 60 degrees, due to a constraint in allowed angle of attack for the controller that is meant to avoid singularities in calculations and potential loss of controllability. If this constraint were higher, set for instance to 85^o degrees, then the angle of attack would increase and saturate there instead. The maximum drag coefficient of this UAV is approximately 2.1, which is achieved with $\alpha = 85^o$, but even at $\alpha = 60^o$, the drag coefficient is significantly larger than the lift coefficient.

VI. Conclusion

The model that is given in Section II is able to perform a precision landing at low speed with the aid of an accurately controlled deep stall. This is achieved with a non-linear MPC that first controls the altitude and then minimizes the velocity significantly and controls the path angle of the UAV. Simulations of longitudinal dynamics and control have been performed, indicating that the precision landing is possible, and that the impact will happen at a relatively low speed. The known steady wind influences the landing speed and the unknown wind gust influences the landing position. With the use of warm start, every optimization needs only a minimum of iterations to find the optimal solution. Lateral control is also important for a deep stall, but has for simplicity not been included in this paper.

Acknowledgements

This work has been carried out at the Centre for Autonomous Marine Operations and Systems (AMOS), supported by the Research Council of Norway through the Centres of Excellence funding scheme, grant number 223254. The Norwegian Research Council is acknowledged as the main sponsor of AMOS.

References

[1] W. Pointner, G. Kotsis, P. Langthaler, and M. Naderhirn, “Using formal methods to verify safe deep stall landing of a MAV,” pp. 5D11–5D110, 2011.

[2] R. Skulstad, C. L. Syversen, M. Merz, N. Sokolova, T. I. Fossen, and T. A. Johansen, “Net Recovery of UAV with Single-Frequency RTK GPS,” in *Proc. of the IEEE Aerospace Conference, Big Sky, Montana*, 2015.

[3] A. M. Goncalves-Coelho, L. C. Veloso, and V. J. A. S. Lobo, “Tests of a light UAV for naval surveillance,” in *Oceans 2007 Europe International Conference*, 2007.

[4] A. G. Sim, “Flight Characteristics of a Manned, Low-Speed, Controlled Deep Stall Vehicle,” *NASA Technical Memorandum*, 1984.

[5] D. A. Spera, “Models of lift and drag coefficients of stalled and unstalled airfoils in wind turbines and wind tunnels,” *Nasa/CR*, 2008.

[6] H. Taniguchi, “Analysis of deepstall landing for UAV,” vol. 3, pp. 2498–2503, 2008.

[7] R. Beard and T. McLain, *Small unmanned aircraft: Theory and practice*. Princeton University Press, 2012.

[8] J. Andersson and J. Gillis, “CasADi.” July 2014.

[9] B. Houska, H. J. Ferreau, and M. Diehl, “An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range,” *Automatica*, vol. 47, pp. 2279–2285, 2011.

[10] C. Ostowari and D. Naik, *Post-Stall Wind Tunnel Data for NACA 44XX Series Airfoil Sections*. Solar Energy Research Institute, January 1985.

[11] MathWorks, “Dryden wind turbulence model (continuous).” Following Military Specification MIL-F-8785C, April 2015.

[12] M. Diehl, H. J. Ferreau, and N. Haverbeke, *Nonlinear Model Predictive Control - Towards New Challenging Applications*, ch. Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation, pp. 391–417. Springer Berlin Heidelberg, 2009.

[13] M. Diehl, H. Bock, J. P. Schlder, R. Findeisen, Z. Nagy, and F. Allgwer, “Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations,” *Journal of Process Control*, vol. 12, no. 4, pp. 577 – 585, 2002.

[14] M. Diehl, “Script on ”Numerical Optimal Control”.” Resource on It’s Learning, June 2011.

[15] M. Diehl, H. Bock, H. Diedam, and P.-B. Wieber, “Fast direct multiple shooting algorithms for optimal robot control,” *Lecture Notes in Control and Information Sciences*, vol. 340, pp. 65–93, 2006. cited By 14.

[16] Numpy devlopers, “Numpy homepage.” 2013.

[17] A. Wächter and L. T. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” *Mathematical Programming*, vol. 106, pp. 25–57, March 2006.