Péter Rucz, Zoltán Belső, Balázs Gáti, István Koller, Antal Turóczi

DESIGN AND IMPLEMENTATION OF NONLINEAR CONTROL SYSTEMS FOR ROTARY AND FIXED WING UAVS

The aim of this paper is to present a novel approach for the design and implementation of onboard nonlinear control systems for different types of unmanned air vehicles. The essential difficulty of creating such controllers is the inherent nonlinearity of the dynamics of the system, which also introduces a great complexity. Our proposed approach relies on an automatic state dependent coefficient (SDC) factorization procedure, which is capable of handling the dynamic equations of the system on a symbolic level. The resulting linearized system representation is solved by the state dependent Riccati equation (SDRE) method. The applicability of the proposed methodology is demonstrated in this paper by means of various examples.

***Keywords**: UAV, nonlinear controller, automatic SDC factorization, state dependent Riccati equation (SDRE)*

INTRODUCTION

In unmanned air vehicles (UAVs) all flight maneuvers are carried out by a robot pilot that controls the aircraft. This means that all the delicate tasks which are performed by skilled pilots in case of ordinary aircrafts have to be handled by a computer program, i.e., the autopilot. The autopilot has to be capable of estimating the expected motion of the vehicle and must be able to produce the control signals for executing the flight plan. These tasks are performed by different computational units in the autopilot whose common structure is depicted in Figure 1.

The sensor fusion unit collects data from the navigational sensors and estimates the orientation and state of motion of the vehicle. The estimated state vector is denoted by $\hat{\mathbf{x}}$. The guidance algorithm compares the actual path of the aircraft with the flight plan and issues commands for correcting the deviations from the desired path. The commands of the guidance unit can also be interpreted as a reference state vector denoted hereafter by $\mathbf{x}_{\text{ref}}$. Finally, the control unit calculates the necessary actions and creates the control signals for the actuators based on the estimated and reference state vectors. When built into the UAV, the three units form a closed control loop together with the sensors and actuators of the aircraft. This paper, however, focuses only on the control unit in the sequel. The state estimation is discussed in more detail in the accompanying paper [1]. A possible solution to the guidance problem is addressed later in the present paper.

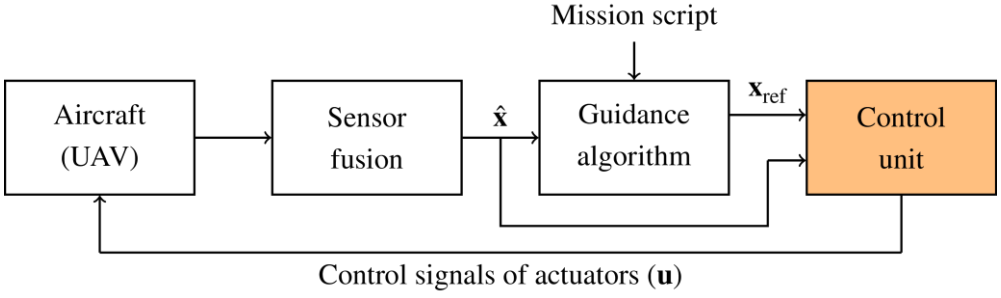

Figure 1. Common structure of a UAV autopilot

To produce the appropriate control signals **u** the control unit of the autopilot must be aware of the dynamics of the aircraft. Moreover, to arrive at a safe and robust flight control, the autopilot has to be able to mitigate the effects of unexpected disturbances such as gusts of wind. Additionally, an efficient control algorithm should produce control signals that are affordable in cost, i.e. ones that do not consume large amounts of energy from the battery. Unfortunately, these criteria are not straightforward to satisfy at once due to the complexity of the system. The main source of this complexity is the nonlinear nature of the problem at hand: both the dynamics of the aircraft (i.e. the partial differential equations relating the state variables) and the connections between the control signals and the resulting forces and moments acting on the vehicle are inherently nonlinear. Contrary to the case of linear systems there are no general methods for designing optimal controllers for systems with nonlinear dynamics. In general, the solution to the nonlinear problem is sought by linearization for which different approaches can be applied. [2][3]

In this paper we introduce a novel approach for developing and implementing nonlinear control systems for different types of UAVs. The proposed technique relies on the state dependent Riccati equation (SDRE) method [4][5] and an automatic factorization approach. This method is beneficial regarding the following aspects. First, the automatic symbolic factorization enables the generation of the system matrices directly from the partial differential equations (PDEs) and ensures that each term is represented correctly in the factorized system. Second, it is straightforward to generate low level computer code from the factorized output. The generated codes can directly be uploaded to the onboard navigational computer. We show by means of hardware-in-the-loop simulations that the proposed approach can efficiently be applied for implementing nonlinear control systems for rotary and fixed wing UAVs.

The subsequent sections of the paper are structured as follows. The next chapter briefly introduces the theory of optimal control systems and discusses the optimal solution of the linear problem and its state dependent version. The automatic factorization method and its implementation are introduced next. Then, the applications of the established nonlinear control method are demonstrated by an academic and two real-life examples. Finally, the paper is concluded by a brief summary and outlook on further research possibilities.

OPTIMAL CONTROL

In this section the mathematical model of a general nonlinear controller is introduced. Then, optimal control is discussed briefly and the special case of linear quadratic regulators (LQR) is addressed. Finally, the state dependent Riccati equation is considered.

A dynamic system can generally be described by its state and output equations which read in their common form as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t), \tag{1}$$

$$\mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{u}, t). \tag{2}$$

Hereafter **x** is the state vector, **u** denotes the input vector of the system, both depending on the time variable t. Differentiation with respect to time is denoted by the upper dot. The vector **y**

is the so-called output vector containing the measurable outputs of the system. In the most general case, both functions **f** and **g** can depend on the time variable. The state equation (1) is a general system of nonlinear differential equations in which all relations can depend on the actual state vector and the input vector.

In a state control system the objective is to generate a control signal $\mathbf{u}(t)$ that makes the state vector converge to a given reference state $\mathbf{x}_{\mathrm{ref}}$. This objective is arrived at by introducing a feedback mechanism that relates the control signal $\mathbf{u}(t)$ with the error vector $\mathbf{x}_{\mathrm{e}} = \mathbf{x} - \mathbf{x}_{\mathrm{ref}}$.

Optimal control solves the feedback problem by minimizing a predefined cost functional, denoted by $J(\mathbf{u})$ here:

$$J(\mathbf{u}) = p\left(t_c, \mathbf{x}(t_c)\right) + q \int_{t_0}^{t_c} h\left(t, \mathbf{x}(t), \mathbf{u}(t)\right)\, \mathrm{d}t. \tag{3}$$

The control is assumed to be limited to the time interval $t \in [t_0, t_c]$. The cost functional defines the penalty for the deviation of the state vector $\mathbf{x}(t)$ from the reference state by integrating it together with the control signal $\mathbf{u}(t)$ weighted by the functions h and p and the scalar q. Formally, the optimal control signal $\mathbf{u}_{\mathrm{opt}}(t)$ is the one that minimizes the functional $J(\mathbf{u})$:

$$\mathbf{u}_{\mathrm{opt}} = \arg\min_{\mathbf{u}} \{J(\mathbf{u})\}. \tag{4}$$

The special case of a linear, time invariant dynamic system for which the state and output equations read as

$$\dot{\mathbf{x}} = \mathbf{A}x + \mathbf{B}\mathbf{u}, \tag{5}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \tag{6}$$

and the cost functional is quadratic is referred to as the linear quadratic regulator (LQR) problem. The LQR problem is of special interest since this is the only case where a general optimal control can be found in closed form [6]. Figure 2 shows the control loop of the linear system. As it is seen the feedback is introduced by means of the feedback matrix **K**.

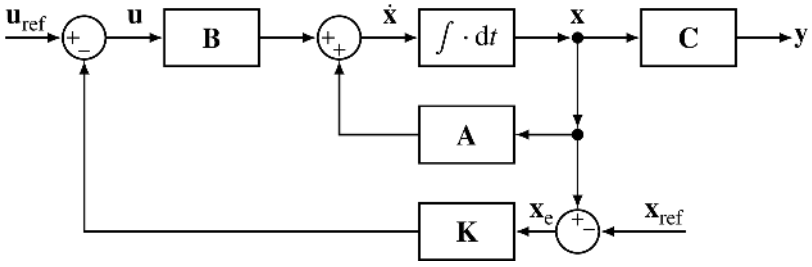

Figure 2. Feedback mechanism in the LQR system

In the LQR case with taking the limit $t_c \to \infty$ the cost functional takes the form

$$J(\mathbf{u}) = \frac{1}{2} \int_{t_0}^{\infty} \left(\mathbf{x}(t)^{\mathrm{T}}\mathbf{Q}\mathbf{x}(t) + \mathbf{u}(t)^{\mathrm{T}}\mathbf{R}\mathbf{u}(t)\right)\, \mathrm{d}t, \tag{7}$$

with **R** and **Q** being positive semidefinite weighting matrices. The feedback matrix **K** is found by solving the Riccati equation

$$\mathbf{P}\mathbf{A} + \mathbf{A}^{\mathrm{T}}\mathbf{P} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{P} + \mathbf{Q} = \mathbf{0} \tag{8}$$

for the matrix **P**. The feedback matrix **K** and the control signal $\mathbf{u}(t)$ are finally obtained as

$$\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{P} \qquad \text{and} \qquad \mathbf{u}(t) = -\mathbf{K}\mathbf{x}_{\mathrm{c}}(t) + \mathbf{u}_{\mathrm{ref}}, \tag{9}$$

where $\mathbf{u}_{\mathrm{ref}}$ is the reference control signal. In case of nonlinear systems a possible solution for finding a suitable (but theoretically not optimal) control is the extension of the LQR problem. This extension is attained by the factorization of the general system equation (1) as

$$\dot{\mathbf{x}} = \mathbf{A}(\mathbf{x}, \mathbf{u})\mathbf{x} + \mathbf{B}(\mathbf{x}, \mathbf{u})\mathbf{u}. \tag{10}$$

Note that time independence was assumed here without the loss of generality. This extension can be used if the so-called state dependent coefficient (SDC) form (10) of the problem exists. In this case the same equations (8) and (9) can be used to find the control signal $\mathbf{u}(t)$. However, the matrices **P** and **K** will also be state dependent; thus, this approach is referred to as the state dependent Riccati equation (SDRE) method [4]. In the following the SDRE method is utilized for designing control systems for different UAV applications.

Unfortunately, as discussed in the next section, it is not trivial to find a suitable SDC form of the system equations, even if such factorizations of the problem exist. To preserve the controllability of the original nonlinear system, the factorized matrices $\mathbf{A}(\mathbf{x}, \mathbf{u})$ and $\mathbf{B}(\mathbf{x}, \mathbf{u})$ must be constructed in such manner that the corresponding linear systems are controllable for all possible states **x**.

AUTOMATIC SDC FACTORIZATION

The factorization problem

To the best knowledge of the authors there is no general approach for finding the best SDC factorization of a nonlinear system. For some very simple nonlinear systems the optimal factorization can be found analytically; however, this is not the case for complex, real-life systems. The general difficulty of the factorization is that the SDC form is not unique. Even the simplest systems can have a continuum number of possible factorizations. Let us consider the following two-dimensional example

$$\begin{aligned} \dot{x}_1 &= x_1 x_2 \\ \dot{x}_2 &= -x_2 + u. \end{aligned} \tag{11}$$

The system matrices **A** and **B** can be factorized as

$$\mathbf{A}(\mathbf{x}) = \begin{bmatrix} \mu x_2 & (1-\mu)x_1 \\ 0 & -1 \end{bmatrix} \qquad \text{and} \qquad \mathbf{B}(\mathbf{x}) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{12}$$

The parameter μ can be an arbitrary real number, which means that a continuum number of possible solutions exist. However, different choices of μ result in different behaviors of the linearized system. By choosing $\mu = 1$ the linearized system becomes unconditionally uncontrollable, while any other choice of μ results in a controllable system, provided that x_1 is not zero. The uncontrollability of the former choice ($\mu = 1$) is explained by the fact that the factorized form does not represent the dependence of $\dot{x}_1$ on x_2 in this case, as the coefficient A_{12} is

zero. This way, the state variable x_1 becomes uncontrollable. In order to preserve the controllability property of the original nonlinear system it is important that all relations between the state and input variables are represented in the factorized system as well.

The proposed algorithm

Our proposed solution for the factorization problem ensures that all relations between the state and input variables in the nonlinear are also represented in the linearized system by an automatic, equal factorization of product terms. The algorithm distributes multivariate polynomial terms equally among each variable, while leaves non-polynomial terms in their original form. An important property of the algorithm is that the computation is done without evaluating the derivatives of the vector valued function **f** (i.e., the Jacobi matrix). Thus, the computational costs of the proposed method are moderate. This approach is simple, yet it can provide a suitable solution in a number of control applications as will be demonstrated in the sequel.

A simple example is given here to illustrate the mechanism of the algorithm. Let us consider a three-dimensional system with $\mathbf{x} = [x_1\ x_2\ x_3]^{\mathrm{T}}$ and a scalar control signal u. Let us define the first PDE of the system as

$$\dot{x}_1 = x_1^2 + x_1 x_2 x_3^2 + x_2^2 u. \tag{13}$$

Note that the other PDEs are indifferent with respect to the factorization of the first PDE. The right hand side of (13) consists of three multivariate polynomial terms, having one (x_1), three (x_1, x_2, x_3), and two (x_2, u) variables, respectively. Distributing the terms equally leads to the form

$$\begin{aligned}\dot{x}_1 = {} & 1 \cdot x_1 \cdot [x_1] + \\ & + \frac{1}{3} \cdot x_2 x_3^2 \cdot [x_1] + \frac{1}{3} \cdot x_1 x_3^2 \cdot [x_2] + \frac{1}{3} \cdot x_1 x_2 x_3 \cdot [x_3] + \\ & + \frac{1}{2} \cdot x_2 u \cdot [x_2] + \frac{1}{2} \cdot x_2^2 \cdot [u],\end{aligned} \tag{14}$$

where the factorization of the three terms are shown in separate lines. Here and in the following the state or control variable to which the given term is assigned is shown in square brackets for the sake of enhanced visibility. Finally, after rearranging the terms we get

$$\dot{x}_1 = \left(x_1 + \frac{1}{3} x_2 x_3^2\right) \cdot [x_1] + \left(\frac{1}{3} x_1 x_3^2 + \frac{1}{2} x_2 u\right) \cdot [x_2] + \left(\frac{1}{3} x_1 x_2 x_3\right) \cdot [x_3] + \left(\frac{1}{2} x_2^2\right) \cdot [u], \tag{15}$$

with the expressions in parentheses holding the coefficients A_{11}, A_{12}, A_{13}, and B_{11} of the matrices $\mathbf{A}(\mathbf{x}, \mathbf{u})$ and $\mathbf{B}(\mathbf{x}, \mathbf{u})$, respectively. To attain the other coefficients the same steps are repeated for the other equations.

This approach leads to the following properties of the SDC form:

1. The result $\mathbf{A}(\mathbf{x}, \mathbf{u})\mathbf{x} + \mathbf{B}(\mathbf{x}, \mathbf{u})\mathbf{u}$ equals $\mathbf{f}(\mathbf{x}, \mathbf{u})$ for all **x** and **u** vectors.
2. The hyperplanes resulting from the linearization are generally not tangential to the surfaces described by the functions $f_i(\mathbf{x}, \mathbf{u})$, with f_i denoting the i-th component of **f**.

The above properties are illustrated in Figure 3 for a simple polynomial function in one dimension. As it can be seen, the proposed factorization can result in curves having remarkably different slopes than the tangent lines. In particular, for the second linearization point at $x = 1.75$ the slopes have

different signs. Nevertheless, the detailed mathematical analysis of the proposed approach is not an objective of this paper; we focus on the possible applications of the technique instead.

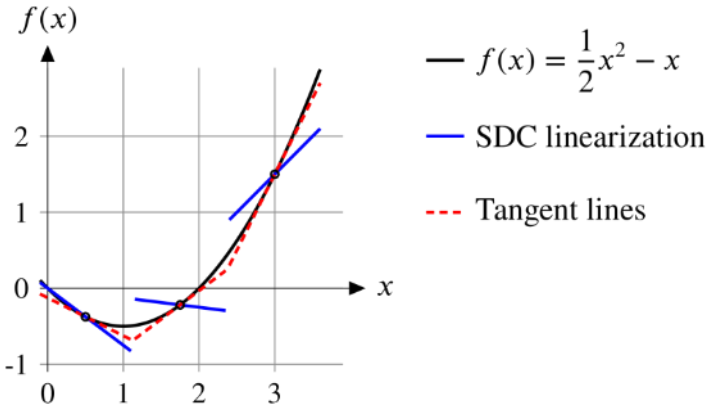

Figure 3. Result of the SDC linearization of a simple, one-dimensional function using the proposed algorithm

Implementation

The objective of the implementation introduced in this section is the automatic generation of controller codes for different UAV setups. For this purpose the proposed automatic factorization algorithm is utilized. The generated codes are intended to be used both in an embedded environment (the onboard computer of the UAV) and in laboratory simulations for testing and tuning different control algorithms. The workflow of the algorithm is illustrated in Figure 4. The main steps of the implementation are the following:

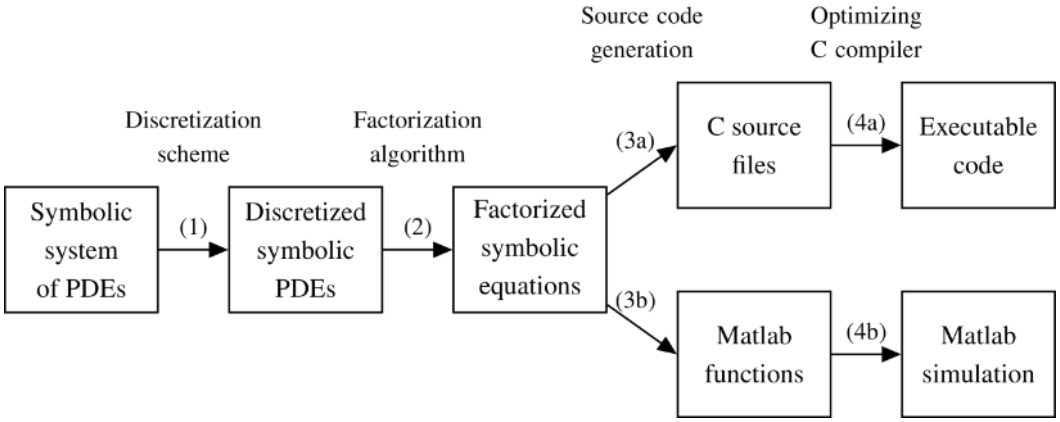

Figure 4. Workflow of control system implementation using automatic SDC factorization

(1) The input of the algorithm is the system of partial differential equations describing the dynamics of the UAV in a symbolic form. These PDEs are discretized using a predefined time stepping scheme. In general, the simple forward time scheme is suitable.

(2) The discretized PDEs are factorized by the automatic factorization algorithm. The result is a function that computes the matrices $\mathbf{A}(\mathbf{x},\mathbf{u})$ and $\mathbf{B}(\mathbf{x},\mathbf{u})$ for the given $\mathbf{x}$ and $\mathbf{u}$ inputs.

(3) The resulting functions can be used exported as (a) C source files or (b) Matlab functions. This step is performed by a source code generating routine.

(4) To be able to run the control algorithm on the embedded computer, the C sources must be compiled and an executable code must be generated. This is done by means of an

optimizing compiler that ensures the best possible performance of the executable code. Thus, a code suited for the limited computational power of the onboard computer is obtained. Alternatively, the factorized forms can readily be exported as Matlab functions for performing different types of simulations in a PC environment.

Steps (1) to (3) are carried out exploiting the capabilities of Matlab's Symbolic math toolbox [7]. The latter makes running the factorization algorithm on the symbolic level feasible.

Since the control algorithm must run real-time in the UAV, it must be capable of performing the computationally involving tasks so that the critical timing conditions are satisfied. Furthermore, beside the control unit, the sensor fusion can also contain computationally heavy steps. The guidance unit does not require much computational effort. In the penultimate chapter of this paper we show by means of hardware-in-the-loop simulations that our implementation fulfils these requirements.

In the SDRE control, the solution of the Riccati equation (8) is the computationally most expensive step. In the discretized form equation (8) takes a bit different form; however, the solution is attained in similar steps. This involves solving an eigenvalue problem of order $2n$, where n is the number of state space dimensions. There are different approaches for solving the discrete Riccati equation. A direct method is described in [8]. An alternative method that improves the numerical behavior of the implementation based on the generalized eigenvalue problem was published in [9]. The latter method was implemented in C language using the LAPACK library. As far as Matlab simulations are concerned, continuous and discrete time LQR problems can readily be solved using the functions named "lqr" and "dlqr", respectively.

AN ACADEMIC EXAMPE

In this section a simplified example is presented which illustrates the limitations of linear controllers in UAV applications. On the other hand, the solution for the same problem using a nonlinear controller is also given here. For the demonstration an academic case is chosen: a rotary wing aircraft that has only two rotors and which can move only in two dimensions. Before discussing the example application in detail, the general equations of aircraft dynamics and their nonlinear nature are addressed.

General governing equations of aircraft dynamics

In general the motion of a UAV can be described as the motion of a rigid body with six degrees of freedom: its three-dimensional translation and rotation. These quantities can be defined in different coordinate systems. Specifically, we use two different coordinate systems in the following: the Earth reference frame with North–East–Down (NED) axes, and the body frame with its axes rotating together with the vehicle. Quantities represented in the Earth and body frames are denoted by the upper indices (E) and (B), respectively.

The translation of a rigid body is governed by Newton's laws of motion, which can be written in the following form:

$$\dot{\mathbf{p}}^{(\mathrm{E})} = \mathbf{v}^{(\mathrm{E})}, \tag{16}$$

$$\dot{\mathbf{v}}^{(\mathrm{E})} = \frac{1}{m}\sum_{i=1}^{N}\mathbf{F}_i^{(\mathrm{E})} + \mathbf{G}^{(\mathrm{E})}. \tag{17}$$

Hereafter $\mathbf{p}$ and $\mathbf{v}$ denote the position and velocity vectors, respectively. The takeoff weight of the aircraft is m, and it is assumed that there are N elements on the aircraft that create the forces $\mathbf{F}_i$ $(i = 1, 2, \ldots N)$, such as propellers, lifting surfaces, and others. The effect of gravity is taken into account with $\mathbf{G}$ representing the gravitational acceleration vector. Equations (16) and (17) can also be written in the body frame; however, for the time being the Earth frame representation is used.

The rotational motion of a rigid body is described by the attitude and its dynamics. There are different possibilities for representing the attitude, such as Euler angles, rotation matrices, or quaternions. First the rotation is described using quaternions, while later an alternative description using Rodrigues parameters is also introduced. A more detailed discussion of attitude representations is found in the accompanying paper [1].

The dynamics of the attitude using the quaternion representation are written as

$$\dot{\mathbf{q}} = \frac{1}{2}\Xi\boldsymbol{\omega}^{(\mathrm{B})} = \frac{1}{2}\begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix}\boldsymbol{\omega}^{(\mathrm{B})}, \tag{18}$$

where $\mathbf{q}$ is the quaternion holding the current attitude of the body, and $\boldsymbol{\omega}^{(\mathrm{B})}$ denotes the angular velocity of the rigid body in the body frame. The rate of change of angular velocity is derived from the torques acting on the body as

$$\dot{\boldsymbol{\omega}}^{(\mathrm{B})} = \left(\mathbf{I}^{(\mathrm{B})}\right)^{-1}\left(-\boldsymbol{\omega}^{(\mathrm{B})} \times \left(\mathbf{I}^{(\mathrm{B})}\boldsymbol{\omega}^{(\mathrm{B})}\right) + \sum_{i=1}^{N}\mathbf{M}_i^{(\mathrm{B})}\right), \tag{19}$$

with $\mathbf{I}$ denoting the moment of inertia matrix of the body, and $\mathbf{M}_i$ $(i = 1, 2, \ldots N)$ representing the moments generated by the forces $\mathbf{F}_i$. The symbol × represents the cross product. Contrary to the case of translational movement, it is more useful to describe the rotational movement in body frame since in the latter frame the matrix $\mathbf{I}^{(\mathrm{B})}$ is constant.

Equations (16)–(19) are the common dynamic equations of all UAV control applications. It can directly be seen that the rotation equation (19) is inherently nonlinear in all applications of our interest. The application-specific property of these equations is the way in which the forces and moments are related to the state and control variables. Naturally, the latter can be remarkably different for different kinds of aircrafts, as it is shown in the subsequent chapters.

In a state control framework the choice of the attitude representation is important with respect to the controllability of the system. For example, rotation matrices and quaternions are not minimal representations of the attitude since their components are not independent. This means that systems with state spaces containing quantities of these kinds are not controllable after linearization. There-

fore, the quaternion representation used in equation (18) cannot be used in a SDRE control application. While Euler angles are a minimal representation of the attitude, they suffer from the problem of singularities, also known as the gimbal lock effect. In this paper we use an alternative representation, the so-called Rodrigues parameters. The latter can be defined by means of a projection of four-dimensional quaternions onto a three-dimensional hyperplane, see details in [10].

The dynamics of the Rodrigues parameters $\boldsymbol{\sigma}$ are given by the equation

$$\begin{bmatrix} \dot{\sigma}_1 \\ \dot{\sigma}_2 \\ \dot{\sigma}_3 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1+\sigma_1^2-\sigma_2^2-\sigma_3^2 & 2(\sigma_1\sigma_2-\sigma_3) & 2(\sigma_1\sigma_3-\sigma_2) \\ 2(\sigma_2\sigma_1+\sigma_3) & 1-\sigma_1^2+\sigma_2^2-\sigma_3^2 & 2(\sigma_2\sigma_3-\sigma_1) \\ 2(\sigma_3\sigma_1-\sigma_2) & 2(\sigma_3\sigma_2+\sigma_1) & 1-\sigma_1^2-\sigma_2^2+\sigma_3^2 \end{bmatrix}\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}^{(B)}. \qquad (20)$$

The direction cosine matrix (DCM, also known as rotation matrix), denoted here by $\mathbf{R}_{\mathrm{DCM}}$ can be expressed from the Rodrigues parameters as

$$\mathbf{R}_{\mathrm{DCM}} = \frac{1}{(1+n)^2}\begin{bmatrix} 4\left(\sigma_1^2-\sigma_2^2-\sigma_3^2\right)+s^2 & 8\sigma_1\sigma_2+4\sigma_3 s & 8\sigma_1\sigma_3-4\sigma_2 s \\ 8\sigma_2\sigma_1-4\sigma_3 s & 4\left(-\sigma_1^2+\sigma_2^2-\sigma_3^2\right)+s^2 & 8\sigma_2\sigma_3+4\sigma_1 s \\ 8\sigma_3\sigma_1+\sigma_2 s & 8\sigma_3\sigma_2-4\sigma_1 s & 4\left(-\sigma_1^2-\sigma_2^2+\sigma_3^2\right)+s^2 \end{bmatrix} \qquad (21)$$

Here $n = \boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{\sigma}$ is the square norm of the Rodrigues parameters and $s = 1 - n$. The Rodrigues parameter representation of the attitude provides a singularity-free description of rotations in the range from $-360°$ to $+360°$ along all three axes.

The Rodrigues parameters also have an attractive property regarding the SDC factorization: equation (20) consists only of multivariate polynomial terms. The same holds for the matrix on the right hand side of equation (21). It was already seen that such terms are handled efficiently by the automatic factorization algorithm introduced above. In the subsequent examples and applications these advantageous properties of the Rodrigues parameters are heavily exploited.

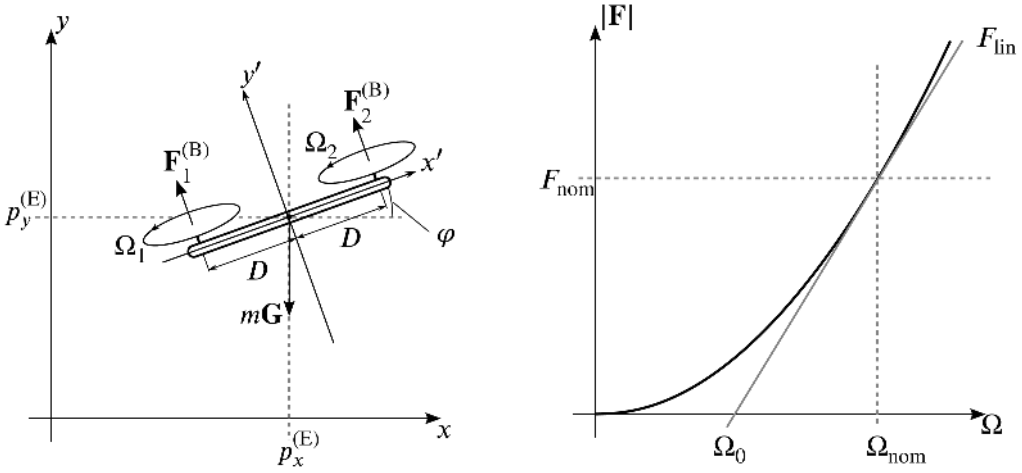

Figure 5. Schematic of the two-dimensional copter (left), linearization of lift forces (right)

A two-dimensional rotary wing aircraft

The schematic of the two-dimensional copter is shown in the left hand side of Figure 5. It is assumed that the copter moves in the x–y plane in the Earth frame. The body frame is defined

by the axes x' and y', and it is rotated by the angle φ compared to the Earth frame. For the sake of simplicity it is also assumed that the lift forces $\mathbf{F}_1$ and $\mathbf{F}_2$ generated by the rotors are parallel to the y' axis. The magnitudes of the lift forces are proportional to the angular velocities of the rotors Ω_1 and Ω_2 squared, related by the lift force coefficient C_L as

$$\left|\mathbf{F}_i\right| = C_L \Omega_i^2 \qquad i = 1, 2. \tag{22}$$

The corresponding moments are attained as $\mathbf{M}_i = \mathbf{r}_i \times \mathbf{F}_i$ with $\mathbf{r}_i$ denoting the vector from the center of gravity of the aircraft and the center of the i-th rotor. For the sake of simplicity, it is assumed that the two rotors are located at $(\pm D, 0)$ in the body frame as shown in the left panel of Figure 5. The control signals u_i drive the angular velocities of the rotors through a first order low-pass filter with the time-constant τ.

We define the ten-dimensional state space of the aircraft as $\mathbf{x} = \left[\boldsymbol{\xi}^{(\mathrm{E})}\ \mathbf{p}^{(\mathrm{E})}\ \mathbf{v}^{(\mathrm{E})}\ \sigma\ \omega\ \boldsymbol{\Omega}\right]^{\mathrm{T}}$, with $\boldsymbol{\xi}^{(\mathrm{E})}$ denoting integral of the position error. The latter is introduced to be able to follow the position reference signals without remainder errors. In the two-dimensional system the angular velocity of the body is the same in the Earth and body frames, therefore these frame notations are omitted. It is worth noting that in two dimensions equations (20) and (21) for the single Rodrigues parameter σ are written as

$$\dot{\sigma} = \frac{1}{4}\left(1 + \sigma^2\right)\omega, \tag{23}$$

$$\mathbf{R}_{\mathrm{DCM}}^{(2\mathrm{D})} = \frac{1}{1 + 2\sigma^2 + \sigma^4}\begin{bmatrix} 1 - 2\sigma^2 + \sigma^4 & 4\sigma(1 - \sigma^2) \\ -4\sigma(1 - \sigma^2) & 1 - 2\sigma^2 + \sigma^4 \end{bmatrix}. \tag{24}$$

Using the above state space choice the full system of state equations for the copter reads as

$$\dot{\boldsymbol{\xi}}^{(\mathrm{E})} = \mathbf{p}^{(\mathrm{E})} - \mathbf{p}_{\mathrm{ref}}^{(\mathrm{E})} \tag{25a}$$

$$\dot{\mathbf{p}}^{(\mathrm{E})} = \mathbf{v}^{(\mathrm{E})} \tag{25b}$$

$$\dot{\mathbf{v}}^{(\mathrm{E})} = \frac{1}{m}\sum_{i=1}^{N_{\mathrm{rot}}} \mathbf{F}_i^{(\mathrm{E})} - \mathbf{G}^{(\mathrm{E})} \tag{25c}$$

$$\dot{\sigma} = \frac{1}{4}\left(1 + \sigma^2\right)\omega \tag{25d}$$

$$\dot{\omega} = I^{-1}\left(-\omega I \omega + \sum_{i=1}^{N_{\mathrm{rot}}} M_i\right) \tag{25e}$$

$$\dot{\Omega}_i = \frac{1}{\tau}\left(u_i - \Omega_i\right) \tag{25f}$$

with N_{rot} denoting the number of rotors, and in this example $N_{\mathrm{rot}} = 2$.

The state of zero acceleration is of particular interest in the sequel, therefore we define the nominal lift force F_{nom} and the nominal angular velocity of the rotors Ω_{nom} as

$$F_{\mathrm{nom}} = m\,|\mathbf{G}| \qquad \rightarrow \qquad \Omega_{\mathrm{nom}} = \sqrt{\frac{m\,|\mathbf{G}|}{C_L N_{\mathrm{rot}}}}. \tag{26}$$

Limits for the maximal and minimal angular velocities are introduced as $\Omega_{\min} = 0$ and $\Omega_{\max} = 2\Omega_{\text{nom}}$. These constraints are enforced by the saturation of the driving signals u_i.

Both for the linear and nonlinear controllers the simple forward discretization scheme is used:

$$\mathbf{x}^{(n+1)} \approx \mathbf{x}^{(n)} + T_s \dot{\mathbf{x}}^{(n)}, \tag{27}$$

where the upper indices in parentheses refer to the n-th and $n+1$-th time steps, respectively, and T_s denotes the sampling time. In the following examples $T_s = 0.01$ s is chosen, which corresponds to the control loop frequency $f_s = 100$ Hz. The other parameters of the system were chosen as: $D = 0.35$ m, $m = 2$ kg, $C_L = 10\ \text{N}/(\text{krad/s})^2$, $I = 0.082\ \text{kgm}^2$, $\tau = 0.032$ s, and $G = 9.81\ \text{m/s}^2$.

Here, and in the following parts of the paper the system simulation is always performed using equations (16)–(19) and the simple time discretization scheme (27). Thus, in the simulations the vehicles are always treated as rigid bodies with six degrees of freedom.

Linear controller solution

First, a linearized solution is discussed, which is based on the assumption that the aircraft operates near to its stable equilibrium state. The following conditions are assumed.

1. The angle φ is assumed to be small, that is $|\varphi| \ll \pi/2$. This way, the approximations $\sin\varphi \approx \varphi$ and $\cos\varphi \approx 1$ can be applied. It is also observed that for small angles the connection with the Rodrigues parameter $\varphi \approx 4\sigma$ holds.
2. The angular velocity of the body is small, i.e. $\omega \approx 0$.
3. The velocity is also assumed to be small, so that $|\mathbf{v}| \approx 0$.

Equation (22) is linearized as shown in the right hand side of Figure 5, i.e.

$$\left|\mathbf{F}_{i,\text{lin}}\right| = C_{L,\text{lin}} \cdot \Omega_{i,\text{lin}}, \tag{28}$$

where

$$C_{L,\text{lin}} = 2C_L\Omega_{\text{nom}} \qquad \text{and} \qquad \Omega_{i,\text{lin}} = \Omega_i - \Omega_0 = \Omega_i - \frac{\Omega_{\text{nom}}}{2}. \tag{29}$$

The equations for the acceleration are linearized as

$$\dot{v}_x = -\frac{1}{m}C_L\left(\sum_{i=1}^{N_{\text{rot}}}\Omega_i^2\right)\sin\varphi \approx -4\frac{N_{\text{rot}}}{m}C_L\Omega_{\text{nom}}^2 \cdot [\sigma], \tag{30}$$

$$\dot{v}_y = \frac{1}{m}C_L\left(\sum_{i=1}^{N_{\text{rot}}}\Omega_i^2\right)\cos\varphi - |\mathbf{G}| \approx \frac{1}{m}C_{L,\text{lin}} \cdot \left[\Omega'_{1,\text{lin}}\right] + \frac{1}{m}C_{L,\text{lin}} \cdot \left[\Omega'_{2,\text{lin}}\right]. \tag{31}$$

Making use of the relation (26), the notation $\Omega'_{i,\text{lin}} = \Omega_{i,\text{lin}} - \Omega_0 = \Omega_i - \Omega_{\text{nom}}$ was introduced. Finally, the nonlinear relation for the angular acceleration is transformed as

$$\dot{\omega} = I^{-1}\left(\underbrace{-\omega I\omega}_{\approx 0} + \sum_{i=1}^{N_{\text{rot}}}\mathbf{r}_i \times \mathbf{F}_i\right) \approx -\frac{DC_{L,\text{lin}}}{I}\left[\Omega'_{1,\text{lin}}\right] + \frac{DC_{L,\text{lin}}}{I}\left[\Omega'_{2,\text{lin}}\right]. \tag{32}$$

Note that in the linear case the state variables Ω_i are replaced by the linearized variables $\Omega'_{i,\mathrm{lin}}$ in the state space. The relation of the latter variables to the driving signals remains the same as given in equation (25) if the signals $u_{i,\mathrm{lin}} = u_i - \Omega_{\mathrm{nom}}$ are introduced in place of u_i. Using the above approximations the system of equations (25) is fully linearized in the vicinity of the stable equilibrium position.

Results of an example simulation using the linear controller are shown in Figure 6. The starting state of the system was the equilibrium state except for the angle φ which was set as $\varphi(0) = 5°$. The simulation was performed on a 25 s long maneuver. The reference position signals are shown in the left panel of Figure 6 by the dashed lines. The copter is commanded to go upwards 15 meters, then to go 10 meters right, and finally to descend 15 meters. Moving in the direction of the x-axis is only possible by controlling the attitude (angle φ) of the copter. As shown in the right panel of Figure 6, in this example configuration the linear controller operates in the domain $-30° < \varphi < 30°$. Observing the actual trajectory shown by the continuous lines in the left panel of Figure 6, it is seen that the controller can follow the reference signals with a small delay and a minimal overshoot. It is worth mentioning that the delay and overshoot properties are configured in the LQR control system by tuning the weighting matrices **R** and **Q** in equation (7). This issue is addressed in the last section of this paper.

While the linear control can provide good results and very low computational costs at the same in the vicinity of the equilibrium state, it cannot operate far from the equilibrium point. In this example the angle φ is clearly a limiting factor; the controller cannot be expected to work properly if the condition $|\varphi| < \pi/2$ is violated. In the next paragraphs it is demonstrated that such limitations can be overcome by using a nonlinear control system.

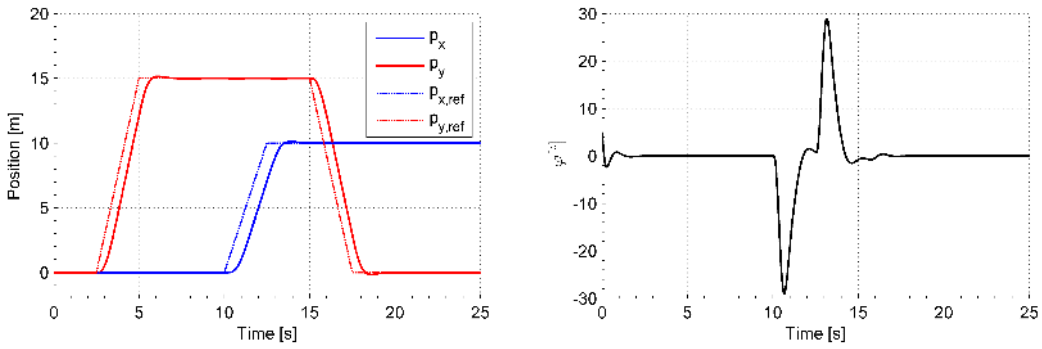

Figure 6. Example results of the linear control of a two-dimensional hypothetical copter

Nonlinear SDRE controller

In the nonlinear controller none of the linear approximations are used. Instead, the following "trick" is introduced in the equation for the acceleration. To be able to take the effect of the gravitational acceleration **G** into account a dummy variable x_d is introduced, and equation (25c) is written in the form

$$\dot{\mathbf{v}}^{(\mathrm{E})} = \frac{1}{m}\sum_{i=1}^{N_{\mathrm{rot}}} \mathbf{F}_i^{(\mathrm{E})} - x_d \mathbf{G}^{(\mathrm{E})}. \tag{33}$$

The dummy variable is included in the state space and its dynamics are expressed as

$$\dot{x}_d = -\lambda_d x_d, \tag{34}$$

where λ_d is a positive real constant, and thus x_d is seen by the controller as a globally stable state variable. For such a state variable no feedback is needed and its weight in the weighting matrix $\mathbf{Q}$ can be set to zero for the cost functional defined by equation (7). However, in each control cycle $x_d = 1$ is set manually, thus, by means of equation (33) the effect of the gravitational acceleration is taken into account correctly in the nonlinear controller.

Since equation (33) incorporates the forces in the Earth frame $\mathbf{F}_i^{(\mathrm{E})}$, while the direction of the forces depend on the attitude of the aircraft, the DCM defined in equation (24) is utilized to transform the lift forces into the Earth frame. Hence, equation (33) becomes a nonlinear relation of the variables σ and Ω_i. Equations (25d) and (25e) are also nonlinear.

To have the same control signals as in the linear case, the rotor angular velocities are shifted by utilizing the state variables $\Omega_i' = \Omega_i - \Omega_{\mathrm{nom}}$ and the control signals $u_i' = u_i - \Omega_{\mathrm{nom}}$. By this choice $u' = 0$ is obtained in the equilibrium state. As it was discussed in the linear controller case, these substitutions do not affect the dynamics of the state variables. The factorized matrices $\mathbf{A}(\mathbf{x}, \mathbf{u})$ and $\mathbf{B}(\mathbf{x}, \mathbf{u})$ are constructed using the automatic factorization algorithm discussed above.

Figure 7 demonstrates the stabilization of the 2D hypothetical copter from an extreme starting attitude $\varphi(0) = 120°$. As it can be seen the copter returns to a stable state after a few seconds of initial transient movement. In the left panel of Figure 7 the copter's response to the same reference signal that was shown in Figure 6 for the linear controller is displayed. It can be seen that in this configuration the controller gives a slower response to the reference signal and the overshoot is also increased a little. However, it should be noted that in this case the weighting matrices $\mathbf{Q}$ and $\mathbf{R}$ were chosen to achieve a wide region of stability and not to provide the smallest delay and overshoot. A possible improvement to achieve both a wide region of stability and small delay and overshoot at the same time is to adaptively tune the weighting matrices depending on the actual state of motion of the aircraft. This possibility is yet to be explored and is out of the scope of this paper.

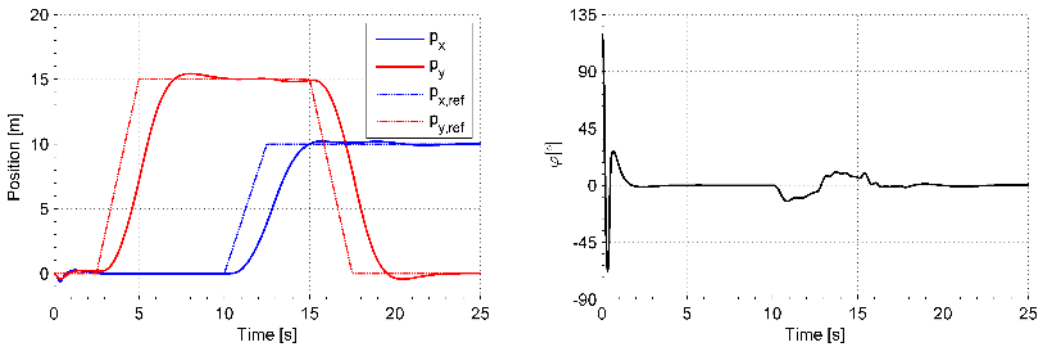

Figure 7. Stabilization of the 2D copter from extreme starting attitude using nonlinear control

The results shown in Figure 7 indicate that our proposed nonlinear control solution is applicable in UAV control systems as it can extend the region of stability of conventional linear controllers. Therefore, the applicability of the proposed methodology in real UAVs is examined in the sequel.

NONLINEAR CONTROL OF A DJI-F450 QUADROCOPTER

The DJI-F450 quadrocopter

In this section the simulation of the nonlinear control of a DJI-F450 type quadrocopter is discussed. This quadrocopter is a general purpose commercially available rotary wing airframe. The mechanical parameters of the airframe were identified as listed in Table 1. It is noted that the airframe can be mounted with different motors, rotors, and electronic devices. The values given in Table 1 correspond to a specific configuration assembled in the frame of this study. The notation $\langle\cdot\rangle$ in the table stands for a diagonal matrix of the given elements.

The controller for the quadrocopter is based on the applications of the relations given as equation (25). However, in order to get a more realistic model some modifications are introduced compared to the two-dimensional case. In the 3D model it is useful to take the effect of gyroscopic moments and drag forces into account. The force $\mathbf{F}_i$ and the torque $\mathbf{M}_i$ induced by the i-th rotor are expressed as

$$\mathbf{F}_i^{(\mathrm{B})} = C_L \mathbf{d}_i \Omega_i^2 \tag{35}$$

$$\mathbf{M}_i^{(\mathrm{B})} = \mathbf{r}_i \times \mathbf{F}_i^{(\mathrm{B})} - C_D \mathbf{d}_i s_i \Omega_i^2 + \boldsymbol{\omega}^{(\mathrm{B})} \times I_p \mathbf{d}_i s_i \Omega_i, \tag{36}$$

where $\mathbf{d}_i$ is the direction of the lift force (expressed in the body frame), s_i is the sign of the angular velocity of the i-th rotor, I_p is the inertia of a rotor and motor pair and C_D is the drag coefficient. For the sake of simplicity, it is assumed that C_L, C_D, and I_p are the same for all four motor–rotor pairs. The three terms on the right hand side of equation (36) are associated with the torque generated by lift forces, drag forces, and gyroscopic moments, respectively.

Since the system is three-dimensional, the dynamics of the Rodrigues parameters are governed by equation (20) instead of equation (25c). As it was discussed above, the Rodrigues parameters provide a singularity-free description of the attitude in the $-360°$ to $+360°$ range. To be able to perform maneuvers consisting of more than one turns, an alternative interpretation of the Rodrigues parameters is used here. A reference coordinate system represented by the rotation matrix $\mathbf{R}_{\mathrm{ref}}$ is introduced, and the Rodrigues parameters are interpreted as attitude errors compared to the reference orientation. Thus, the transformation from the Earth to the body frame is described by the DCM denoted by $\mathbf{R}_{\mathrm{DCM}}$ given as

$$\mathbf{R}_{\mathrm{DCM}} = \mathbf{R}_{\mathrm{rod}} \mathbf{R}_{\mathrm{ref}}, \tag{37}$$

where $\mathbf{R}_{\mathrm{rod}}$ is calculated from the Rodrigues parameters using equation (21). Since the attitude error is always in the $-180°$ to $+180°$ range, the description of the attitude becomes free of singularities even for maneuvers including several turns around an axis.

The eighteen-dimensional state space of the controller includes the position and the velocity of the aircraft in the Earth frame, the Rodrigues parameter triplet, the angular velocity of the UAV in the body frame, the angular velocity of each rotor, the integral of the position error along the z-axis only and the dummy variable introduced by equations (33) and (34). Similar to the case of the hypothetical copter, the control signals u_i are assumed to drive the angular velocities of the rotors through a first order low-pass filter represented by the time-constant τ. The latter

assumption may be considered as an oversimplification; however, measurements performed on the motor controller indicated that this simplification is suitable.

Parameter	Notation	Value
Takeoff weight	m	1.4 kg
Moment of inertia matrix	$I^{(\mathrm{B})}$	$\langle 0.0104, 0.0104, 0.0196\rangle$ kgm^2
Lift coefficient of rotors	C_L	27.153 N/(krad/s)^2
Drag coefficient of rotors	C_D	0.607 N/(krad/s)^2
Inertia of a rotor and motor pair	I_p	$1.375 \cdot 10^{-3}$ kgm^2
Nominal angular velocity of rotors	Ω_{nom}	3395 RPM
Rotor positions	r_i	$(\pm 0.16, \pm 0.16, 0)$ m
Lift force directions	d_i	$(0, 0, -1)$ [−]
Rotor angular velocity signs	s_i	$(-1, +1, -1, +1)$ [−]

Table 1. Parameters of the DJI-F450 airframe

Matlab simulation

The nonlinear controller for the DJI-F450 copter was first implemented in the frame of a Matlab simulation. Again, the proposed SDC factorization method was utilized to attain the system matrices. To simulate the limited computational resources available on the onboard computer, the update of the factorized matrices $\mathbf{A}(\mathbf{x}, \mathbf{u})$ and $\mathbf{B}(\mathbf{x}, \mathbf{u})$ was performed only every tenth cycle of the simulation. Thus, the control rule $\mathbf{u} = -\mathbf{K}\mathbf{x}_{\mathrm{e}}$ is calculated in each simulation cycle, whereas the feedback matrix $\mathbf{K}$ is updated (and the state dependent Riccati equation is solved) only every tenth cycle. This means a significant reduction of the required computational effort.

The simulation of an example flight path is shown in Figure 8. The sixty-second-long flight plan consists of three parts. (1) In the first 10 seconds the quadrocopter is commanded to take off from the origin and to reach the first waypoint at $(0, 10, -10)$ m. (2) Then, in the next 20 seconds, the copter travels 20 meters along the x-axis and rotates two whole rounds around the z-axis in the positive direction. (3) Finally, in the last 30 seconds the copter ascends another 10 meters while making a half circle with 10 meters radius to arrive to the last waypoint at $(20, -10, -20)$ m. All reference states except the position and the yaw angle are zero during the whole maneuver. As it is seen in Figure 8, the controller makes the copter follow the reference path with very small errors. Note that the y and the z axes are mirrored in the left panel of Figure 8 due to the limitations of the plotting program. The forward direction of the airframe is indicated by the red rotors in Figure 8.

The resulting Euler angles are shown in the right panel of Figure 8. It is seen that the copter follows the reference yaw angle with a small delay. The two complete turns around the z-axis are also performed with maintaining small errors. It can be assessed that the implemented controller is capable of driving the copter in case of usual flight commands.

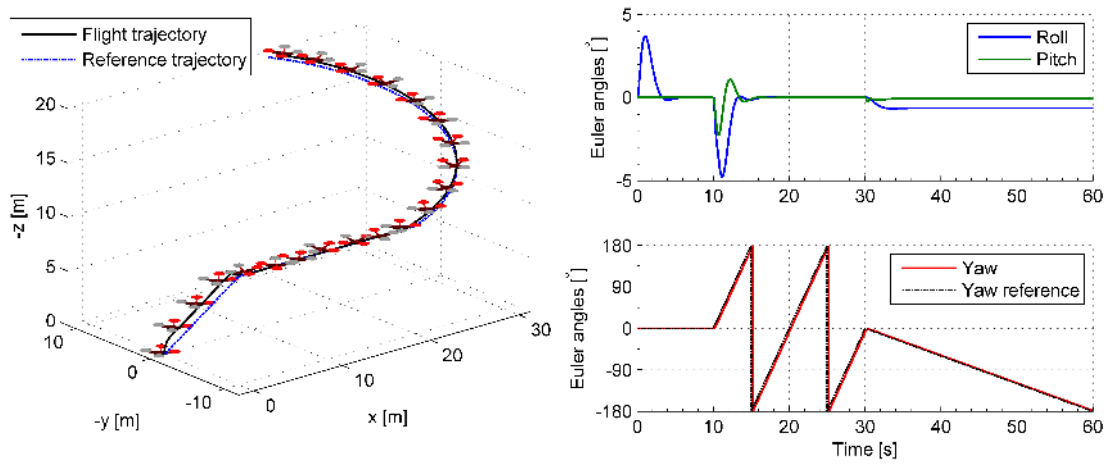

Figure 8. Simulation of flying a test trajectory with the DJI-F450 quadrocopter using SDRE control

In order to examine the robustness of the controller, another simulation was performed in which the copter was started from an instable state: the initial roll angle φ was set as $\varphi(0) = 150°$. This corresponds to a nearly upside-down attitude. For the sake of simplicity, the initial values of the other state variables correspond to the equilibrium state, which is also the reference state in this case. The results of a ten-second-long simulation are shown in Figure 9. It is seen that the copter is stabilized in a few seconds. Apparently, during the stabilization process, the aircraft only loses less than 1 meter of altitude. In the y direction the maximum deviation from the reference position is around 3.5 meters. The small loss in altitude is due to the incorporation of the position integral along the z-axis into the state space. Results illustrated in Figures 8 and 9 indicate that the proposed control method is applicable in real UAVs and can handle both normal and extraordinary flight conditions.

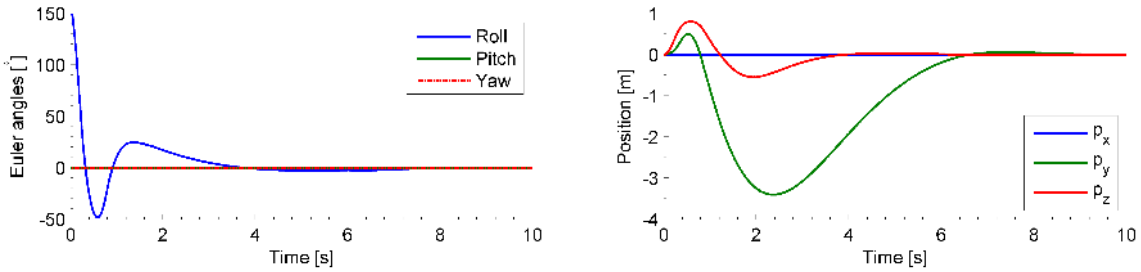

Figure 9. Stabilization of the DJI-F450 quadrocopter using SDRE control from an extreme attitude

Hardware-in-the-loop simulation

To test the control unit inside the embedded environment of the onboard computer, a hardware-in-the-loop simulation was also performed. In this case, the controller is not run in a PC simulation but an executable is generated by means of the steps (3a) and (4a) as shown in Figure 4. Contrary to the simulations presented so far, in this case the whole autopilot system was tested, i.e. all three blocks (sensor fusion algorithm, guidance, and control) were running real-time on the onboard computer as illustrated in Figure 1. Furthermore, beside the navigational tasks, a communication task is also incorporated into the system. This way, the aircraft was controlled from the user interface of the ground control station (GCS) application. Finally, it is also worth mentioning that

opposed to a real-life flight scenario the system simulation (modeling the dynamics of a rigid body with six degrees of freedom) was also implemented inside the onboard computer.

The simulation scenario consists of the following steps. First, the user can construct a flight plan composed of various mission items (waypoints, loitering etc). Then, a mission script is generated and transferred to the vehicle. Upon receiving the script the vehicle checks if all the items in the script are valid. If a valid script was received the execution of the flight plan can begin. In the simulation environment, at the beginning the simulated vehicle is transferred automatically to the home position and starts the flight from there.

During the simulated flight fictitious sensor signals corrupted by noise, offset, and distortion on purpose are generated based on the simulated state of motion of the aircraft. To model the behavior of real sensor measurements the statistical properties of the noise and offset were chosen corresponding to the information given in the data sheets of the onboard sensors. From these corrupted signals the sensor fusion algorithm [1] predicts the actual state of the vehicle. A simple guidance method generates the reference signal for the controller such that the aircraft always heads towards the current waypoint. When a waypoint is reached, the vehicle hovers and turns around the z-axis towards the next waypoint.

A snapshot of the user interface of our Ground Control Station application (GCS) during the hardware-in-the-loop simulation is shown in Figure 10. As seen, the map and the flight plan is shown in the left of the screen, with the green line indicating the path of the vehicle. In the top right part an artificial horizon is visible which helps the pilot and the operator in case of flight by remote control. The bottom right part of the screen is occupied by a customizable dashboard on which different functions and measured values can be plotted. As shown in the figure, the controller operates properly in the embedded environment and is capable of producing control signals in a real-time manner. Moreover, it is also seen that the control unit can work in cooperation with the sensor fusion and guidance units.

Figure 10. Hardware-in-the-loop simulation of the DJI-F450 copter in the ground control station

NONLINEAR CONTROL OF AN X8 RIGID WING UAV

Aerodynamic forces in the nonlinear control system

Similar to the rotary wing aircrafts discussed above, the control of a rigid wind aircraft also relies on the dynamics of a rigid body, governed by equations (16)–(19). However, in a rigid wing aircraft the forces and moments acting on the body have a much more complex dependence on the state of motion of the vehicle and the state of the actuators. For example, the deflections of different control surfaces affect the resulting lift and drag forces significantly. In the sequel the lift, drag, and propulsion forces are evaluated based on the study of Gausz & Gáti [11]. The incorporation of these forces into the SDRE controller also involves some simplifications which are discussed briefly in the following paragraphs.

The lift and drag forces are assumed to act in the aerodynamic center (AC) of the wing, as depicted in the right panel of Figure 11. The aerodynamic center of each wing is calculated making use of the mean aerodynamic chord (MAC). The aerodynamic forces are usually expressed in the non-inertial aerodynamic (or wind) coordinate system denoted here by the upper index (A). In the calculation the so-called surface coordinate system (fixed and aligned to the lifting surface) is also utilized and denoted by the upper index (S).

The relative air speed $\mathbf{V}$ of the aerodynamic center of a lifting surface is expressed in the body frame as

$$\mathbf{V}^{(\mathrm{B})} = \mathbf{v}^{(\mathrm{B})} + \boldsymbol{\omega}^{(\mathrm{B})} \times \mathbf{r}_{\mathrm{AC}}^{(\mathrm{B})} - \mathbf{V}_{\mathrm{wind}}^{(\mathrm{B})} - \mathbf{V}_{\mathrm{ind}}^{(\mathrm{B})}, \tag{38}$$

where $\mathbf{r}_{\mathrm{AC}}^{(\mathrm{B})}$ is the vector from the center of gravity (CG) to the AC, $\mathbf{V}_{\mathrm{wind}}^{(\mathrm{B})}$ is the speed of wind expressed in the body frame, and $\mathbf{V}_{\mathrm{ind}}^{(\mathrm{B})}$ is the so-called induced air speed. The aerodynamic force generated by the i-th lifting surface is expressed as

$$\mathbf{F}_i^{(\mathrm{A})} = \frac{\rho}{2} \left|\mathbf{V}_i\right|^2 S_i \begin{bmatrix} -C_{i,D} \\ 0 \\ -C_{i,L} \end{bmatrix}, \tag{39}$$

where ρ is the density of air, $\mathbf{V}_i$ and S_i are the relative air speed and the area of the i-th lifting surface, with the former obtained using equation (38). The lift and drag coefficients of the i-th surface are denoted by $C_{i,L}$ and $C_{i,D}$, respectively. The aerodynamic forces generated by each lifting surface are transformed into the body frame using the relation

$$\mathbf{F}^{(\mathrm{B})} = \mathbf{R}_{(\mathrm{S})}^{(\mathrm{B})} \mathbf{R}_{(\mathrm{A})}^{(\mathrm{S})} \mathbf{F}^{(\mathrm{A})}, \tag{40}$$

where the rotation matrices $\mathbf{R}_{(\mathrm{S})}^{(\mathrm{B})}$ and $\mathbf{R}_{(\mathrm{A})}^{(\mathrm{S})}$ denote transformations from the frame shown in the lower index to the frame shown in the upper index.

The coefficients C_L and C_D depend on the angle of attack α. Based on the study of Gausz & Gáti [11] the following approximations are introduced. For the lift forces

$$C_L \approx C_{L\alpha} \sin\alpha \approx C_{L\alpha} \frac{V_z^{(S)}}{|\mathbf{V}|} \qquad \text{pre-stall,}$$
$$C_L = kC_{L\alpha}\alpha_s \sin 2\alpha \approx kC_{L\alpha}\alpha_s \cdot 2\frac{V_x^{(S)} V_z^{(S)}}{|\mathbf{V}|^2} \qquad \text{post-stall.} \tag{41}$$

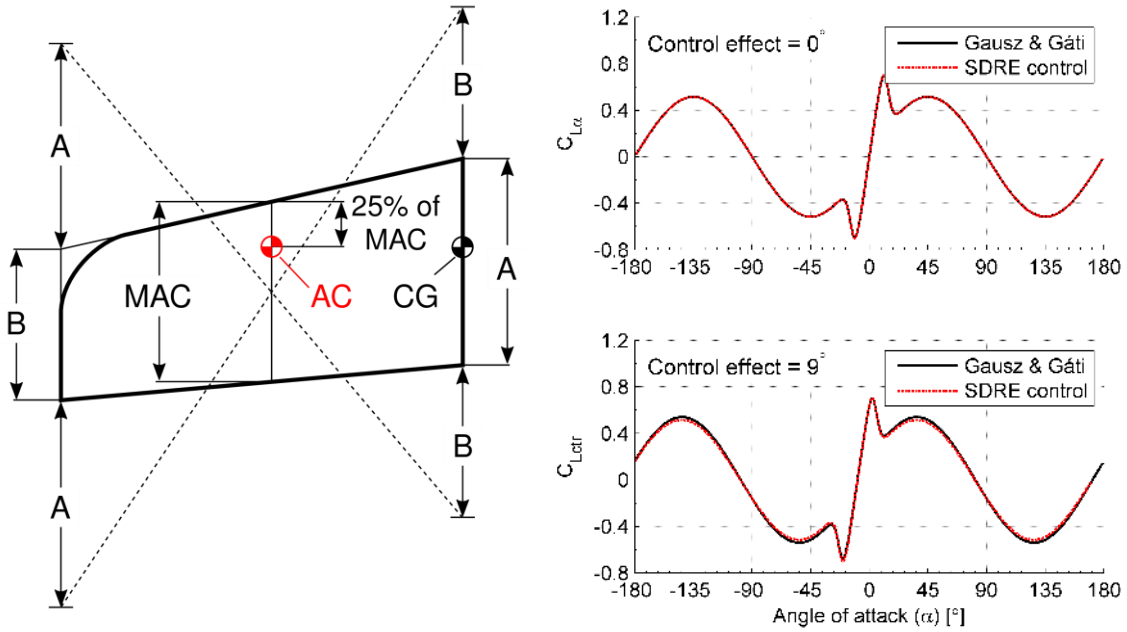

Figure 11. Definition of the aerodynamic center of a wing (left), lift forces and their approximations (right)

Here $C_{L\alpha}$ is the specific lift force coefficient, α_s is the critical angle of attack and k is a constant factor. The drag forces are approximated as

$$C_D \approx C_{D0} + C_{D\max} \cdot \left(C_{L\alpha} \frac{V_z^{(S)}}{|\mathbf{V}|} \right)^2, \tag{42}$$

with C_{D0} and $C_{D\max}$ denoting the drag force coefficients at $\alpha = 0°$ and $\alpha = 90°$, respectively.

Each wing can be mounted with control surfaces, whose effect are taken into account by the modification of the angle of attack. The modified angle α_{ctr} is attained as

$$\alpha_{\text{ctr}} = \alpha + \frac{\partial\alpha}{\partial\delta}\delta, \tag{43}$$

with δ denoting the deflection of the control surface from the equilibrium position and $\partial\alpha/\partial\delta$ is the control efficiency. The deflection of the control surfaces also result in a change of the aerodynamic forces represented by $\Delta\mathbf{F}^{(A)}$, which reads as

$$\Delta\mathbf{F}^{(A)} = \frac{\rho}{2}|\mathbf{V}|^2 S_{\text{ctr}} \begin{bmatrix} -(C_{D\text{ctr}} - C_D) \\ 0 \\ -(C_{L\text{ctr}} - C_L) \end{bmatrix}, \tag{44}$$

where S_{ctr}, $C_{D\text{ctr}}$, and $C_{L\text{ctr}}$ are the area, the drag, and the lift coefficients of the control surface, respectively. The results of the approximations introduced in equation (41) are compared to the

results of the original equations of [11] in the right panel of Figure 11. In the top diagram, the control surface is in its trimmed state, whereas in the bottom diagram a state with $\Delta\alpha = \partial\alpha/\partial\delta \cdot \delta = 9°$ is shown. As it can be seen, the introduced approximations closely follow the original curves for the whole range of the angle of attack. It is worth noting that the above simplifications were chosen to make the SDC factorization of the system feasible.

The thrust generated by the propulsion is taken into account in a similar way as in the case of the quadrocopter; however, the dependence of the thrust coefficient on the advance ratio [11] is also taken into consideration in this case. For the sake of brevity, the corresponding equations are omitted here.

From the above discussion it is clear that the deflection of the control surfaces are related to the state of motion of the rigid wing aircraft in a complex manner. In order to reduce this complexity to some extent, the following approximations are also introduced.

1. The effect of the wind and induced velocities are neglected in equation (38).
2. During normal flight, only the z-component of the angular velocity $\boldsymbol{\omega}^{(\mathrm{B})}$ can be significantly different from zero. This means a further simplification to equation (38).
3. For the calculation of the rotation matrix $\mathbf{R}_{(\mathrm{A})}^{(\mathrm{S})}$ it is assumed that $V_y^{(\mathrm{S})}$ and $V_z^{(\mathrm{S})}$ are small compared to $V_x^{(\mathrm{S})}$. This results in a remarkable simplification of the symbolic expression of the rotation matrix $\mathbf{R}_{(\mathrm{A})}^{(\mathrm{S})}$.
4. The body of the aircraft can produce both lift and drag forces in reality. However, the former are neglected in the control model.

Control of the X8 UAV

The X8 is a commercially available foam airframe. An X8 aircraft assembled in the frame of our project is shown in Figure 12. The X8 is a flying wing aircraft, controllable by a single propulsion system located at the back of the body and two hinged control surfaces, one for each wing. A small winglet is present on the end of both wings.

Figure 12. AMORES X8 aircraft during flight

In the control system, it is assumed that the control signals drive the deflections of the control surfaces and to the angular velocity of the propeller through first order low-pass filters. The control signals are introduced as

$$\dot{\Omega} = \frac{1}{\tau_p}\left(u_p - \Omega\right), \tag{45}$$

$$\dot{\delta}^{\pm} = \frac{1}{\tau_\delta}\left(u^{\pm} - \delta^{\pm}\right). \tag{46}$$

Here Ω is the angular velocity of the propeller and u_p is its driving signal. The quantities $\delta^{\pm}$ symbolize $\delta_1 \pm \delta_2$, where δ_i is the deflection of the i-th control surface. The corresponding control signals are denoted by $u^{\pm}$. The time-constants of the low-pass filters are denoted by τ_p and τ_δ, respectively. The seventeen-dimensional state space consists of the position of the aircraft, its velocity represented in the body frame, the Rodrigues parameter triplet, the angular velocity of the body, the three control state variables (Ω, δ^+, and δ^-), the integral of the position error along the z-axis and the dummy variable x_d. Since the velocity of the body is expressed in the body frame equation (17) is replaced by the relation

$$\dot{\mathbf{v}}^{(\mathrm{B})} = \boldsymbol{\omega}^{(\mathrm{B})} \times \mathbf{v}^{(\mathrm{B})} + \frac{1}{m}\sum_{i=1}^{N_{\mathrm{aero}}} \mathbf{F}_i^{(\mathrm{B})} + \mathbf{R}_{\mathrm{DCM}}\mathbf{G}^{(\mathrm{E})}, \tag{47}$$

with N_{aero} denoting the total number of components creating aerodynamic forces. In our case $N_{\mathrm{aero}} = 4$ as the X8 airframe has two wings, a body, and one propulsion system. For the implementation of the control system, precise identification of the aircraft was also required. The most important parameters of the system were identified as listed in Table 2.

The control of the X8 aircraft was simulated inside Matlab utilizing the automatic factorization procedure again. To generate the reference signals, the so-called L1 guidance algorithm was used as defined in [12]. This simple guidance algorithm provides a lateral acceleration command based on the deviation from the desired trajectory.

Parameter	Notation	Value
Takeoff weight	m	3.2 kg
Moment of inertia matrix	$I^{(\mathrm{B})}$	$\langle 0.8, 0.06, 0.86\rangle$ kgm^2
Wing span	–	2.22 m
Wing areas	S_i	0.345 m^2
Specific lift coefficient	$C_{L\alpha}$	4.37 [−]
Minimal drag coefficient	C_{D0}	0.009 [−]
Critical angle of attack	α_s	13.5°
Effective control surface area	S_{ctr}	0.175 m^2
Control efficiency	$\partial\alpha/\partial\delta$	0.3 [−]
Nominal (trim) velocity	v_{nom}	12 m/s
Nominal angular velocity of propeller	Ω_{nom}	2750 RPM
Inertia of propulsion	I_p	$1.4 \cdot 10^{-3}$ kgm^2

Table 2. Identified parameters of the X8 aircraft

The results of the simulation are displayed in Figures 13 and 14. The flight plan consists of a number of waypoints, as shown in the left panel of Figure 13. The position of the aircraft is also displayed on the right hand side of the figure, where the reference signals are notated by dashed lines. As seen, the vehicle closely follows the reference path with a maximal deviation of around 10 meters.

The resulting Euler angles are displayed in the left panel of Figure 14. The reference signals are indicated by dashed lines again. The switching between waypoints are clearly visible in the reference signals. It is noted that in this case both the reference position and Euler angles are generated online during the simulation. The control state variables are shown in the right hand side of Figure 14. As it can be seen, the control is achieved by using both the deflectors and the propulsion, as expected. It is also seen that the control signals vary smoothly in time, which indicates that the dynamics of the controller are well fitted to the dynamics of the system. Comparing the deflections of the control surfaces and the Euler angles, the effect of the control on the attitude is clearly observable.

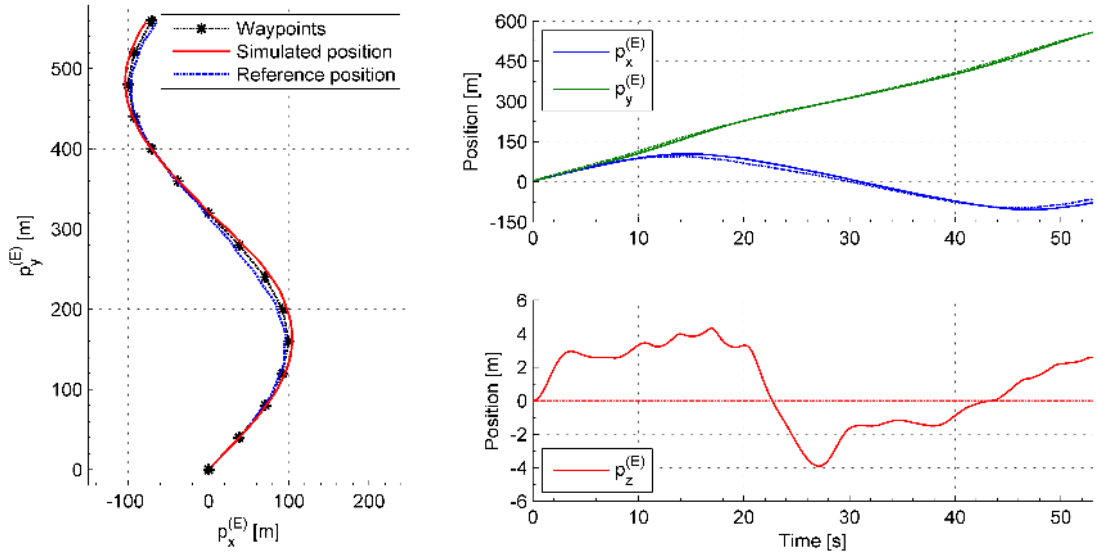

Figure 13. Simulated flight control of the X8 aircraft: waypoints and path (left), position signals (right)

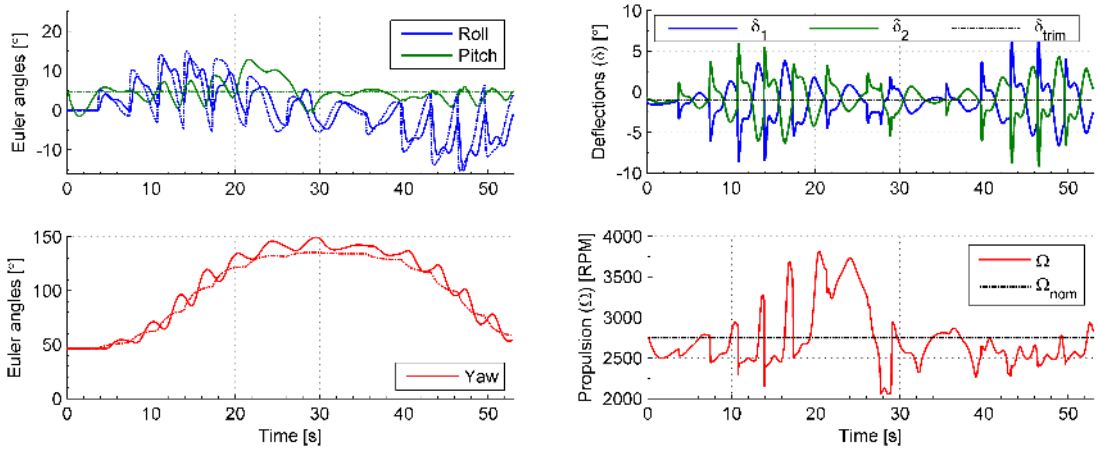

Figure 14. Simulated flight control of the X8 aircraft: Euler angles (left) and control variables (right)

Based on the above results it can be assessed that the proposed nonlinear control approach is applicable also in a rigid wing UAV control system. It should be emphasized that incorporating the aerodynamic force calculation represented by equations (38)–(44) introduces a great complexity in the control equations. The simulation results indicate that the proposed control method is capable of handling this enhanced complexity.

CONCLUSIONS AND OUTLOOK

In this paper a nonlinear control approach based on the state dependent Riccati equation (SDRE) method was introduced. For the generation of the state dependent coefficients (SDC) form of the system matrices an automatic symbolic factorization algorithm was proposed. This algorithm was utilized in different UAV control applications, including academic and real-life examples. It was shown that the suggested nonlinear control approach can overcome the limitations of common linear controllers and can provide an extended domain of stability. Several scenarios were tested during the study, including Matlab modeling and hardware-in-the-loop simulations. It was also demonstrated that the proposed control algorithm can also operate in an embedded hardware environment.

Case	**State space dimensions** n	**Number of terms in** $\mathbf{A}(\mathbf{x},\mathbf{u})$	**Number of terms in** $\mathbf{B}(\mathbf{x},\mathbf{u})$
Hypothetic 2D copter	11	64	2
DJI-F450 quadrocopter, $\mathbf{v}^{(\mathrm{B})}$	18	692	4
DJI-F450 quadrocopter, $\mathbf{v}^{(\mathrm{E})}$	18	1 356	4
X8 rigid wing aircraft	17	2 094	3

Table 3. SDC factorization properties of the UAV systems discussed in the paper

It was seen that the nonlinearity of the dynamic equations introduces a great complexity in the system. The number of factorized terms in the SDC form are summarized in Table 3 for the different UAV systems discussed in the paper. It is seen that the real-life examples involve several hundreds of terms, which would be rather tedious to evaluate manually. Thus, the automatic factorization procedure can be well exploited in real-life UAV control systems. It is worth mentioning that in the case of the DJI-F450 quadrocopter the choice of the coordinate frame for the acceleration equation – see equations (17) and (47) – results in a huge difference regarding the number of factorized terms. The two cases are denoted by $\mathbf{v}^{(\mathrm{E})}$ and $\mathbf{v}^{(\mathrm{B})}$ in the table, respectively.

The detailed mathematical analysis of the proposed factorization algorithm was out of the scope of this paper; however, the rigorous analysis of the resulting stability regions are indeed of great importance for validating the methodology. Another possibility for further studies is the method of constructing the weighting matrices $\mathbf{Q}$ and $\mathbf{R}$. To the best knowledge of the authors there are no general method for this issue, while the choice has a great influence on the performance of the resulting control system. These two particular issues have high priority in the short term research plan of the authors.

ACKNOWLEDGMENTS

The research presented in this paper was supported by the Hungarian government in the form of a KMR_12-1-2012-0121 Research and Development Project named “AMORES” (Autonomous Mobile Remote Sensing). The authors gratefully acknowledge this financial support. The authors also thank the contributions of I. Jankovics (identification of the DJI-F450 and X8 UAVs) and G. Firtha (implementation of the L1 guidance method).

REFERENCES

[1] Z. BELSŐ, B. GÁTI, I. KOLLER, P. RUCZ, A. TURÓCZI. Design of a nonlinear state estimator for navigation of autonomous aerial vehicles. Submitted for publication into the same volume.
[2] A. ISIDORI. Nonlinear control systems, Volumes I–II. Springer-Verlag, London, 1999.
[3] H. K. KHALIL. Nonlinear Systems, Prentice Hall, New Jersey, 2002.
[4] K. D. HAMMET. Control of Nonlinear Systems via State Feedback State-Dependent Riccati Equation Techniques, Dissertation, Air Force Institute of Technology, 1997.
[5] E. B. ERDEM. Analysis and Real-time Implementation of State-dependent Riccati Equation Controlled Systems. PhD thesis, University of Illinois at Urbana-Champaign, 2001.
[6] P. S. MAYBECK. Stochastic models, estimation and control: Volume 1. Academic Press, 1982.
[7] THE MATHWORKS INC. Matlab symbolic math toolbox. (online) url: http://www.mathworks.com/products/symbolic/ (2015.11.24.)
[8] A. J. LAUB. A Schur method for solving algebraic Riccati equations. IEEE Transactions on Automatic Control, 24(6):913–921, 1979.
[9] T. PAPPAS, A. J. LAUB, N. R. SANDELL. On the Numerical Solution of the Discrete-Time Algebraic Riccati Equation. IEEE Transactions on Automatic Control, 25(4):631–641, 1980.
[10] H. SCHAUB, J. L. JUNKINS. Stereographic orientation parameters for attitude dynamics: A generalization of the Rodrigues parameters. Journal of Astronautical Sciences, 44(1):1–19, 1996.
[11] T. GAUSZ, B. GÁTI. Merevszárnyú és többrotoros légi eszközök modellje precíziós repülési feladatok szimulációjához, 2013. AMORES projekt, kutatási jelentés.
[12] S. PARK, J. DEYST, AND J.P. HOW. A new nonlinear guidance logic for trajectory tracking. Proceedings of the AIAA Conference. on Guidance, Navigation, and Control, 2004.

NEMLINEÁRIS SZABÁLYZÓRENDSZEREK TERVEZÉSE ÉS MEGVALÓSÍTÁSA FORGÓ- ÉS MEREVSZÁRNYAS ROBOTREPÜLŐGÉPEKHEZ

Jelen cikk célja egy új módszer bemutatása robotrepülőgépek nemlineáris fedélzeti irányítási rendszereinek tervezéséhez és megvalósításához. A megoldandó feladat legfőbb kihívását a rendszer viselkedését leíró dinamikai egyenletek nemlineáris természete jelenti, mely egyben a megtervezendő szabályzó nagy komplexitásával is jár. Az általunk javasolt módszer egy automatikus állapotfüggő (SDC) faktorizációs eljáráson alapszik, mely képes a rendszeregyenleteket szimbolikus formában is kezelni. Az ennek segítségével létrejövő linearizált rendszerreprezentációt az állapotfüggő Riccati-egyenlet (SDRE) módszerrel oldjuk meg. A javasolt módszer alkalmazásának lehetőségeit a cikkben több példával is szemléltetjük.

***Kulcsszavak**: UAV, nemlineáris szabályzó, automatikus SDC faktorizáció, állapotfüggő Riccati-egyenlet (SDRE)*

http://www.repulestudomany.hu/folyoirat/2015_3/2015-3-20-0240_Rucz_P_et_al.pdf