Computational Aspects of Optimization-Based Path Following of an Unmanned Helicopter

Johann C. Dauer and Timm Faulwasser and Sven Lorenz

Abstract This paper considers the path following of unmanned helicopters based on dynamic optimization. We assume that the helicopter is equipped with a flight control system which provides an approximation of its closed-loop dynamics. The task at hand is to derive inputs for this flight control system in order to track a geometrically specified path. A concise problem formulation and a discussion of an efficient implementation is presented. This implementation achieves computation times below the flight duration of the path by exploiting differential flatness of components of the dynamics. Finally, we present quantitative results in respect to convergence and required iterations for a challenging nonlinear path. We show that the proposed optimization based approach is capable of tackling nonlinear path following for helicopters in an efficient manner.

1 Introduction

In this contribution the problem of *path following* of small unmanned helicopters such as the one shown in Figure 1 is considered. Path following is defined here as the task to fly along a geometrically specified space curve. The time-wise progress on the path is not a priori known or specified. Rather, we allow mission based requirements such as, for instance, a desired velocity along the path. It is assumed that

Johann C. Dauer
German Aerospace Center (DLR e.V.), Institute of Flight Systems, Braunschweig, Germany, e-mail: johann.dauer@dlr.de

Timm Faulwasser
Laboratoire d'Automatique, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland e-mail: timm.faulwasser@epfl.ch

Sven Lorenz
German Aerospace Center (DLR e.V.), Institute of Flight Systems, Braunschweig, Germany, e-mail: sven.lorenz@dlr.de

a path has been provided, either by the mission specification or by means of a path planner, as presented e. g. in [1]. Furthermore, we assume that there exists a flight control system handling the stabilization of the helicopter, which allows deriving an approximation of the closed-loop dynamics, see e. g. [11].

Fig. 1 Automated helicopter midiARTIS of the German Aerospace Center,maximum-take-off-weight 14 kg, rotor-diameter 1.8 m

Our task is to find suitable inputs for the flight control system offline, which steers the helicopter along the desired path. These inputs have to satisfy dynamic constraints of the vehicle as well as limitations of the control system implementation which would otherwise cause path deviations. A simplified block diagram is presented in Figure 2, where the block considered here is called ”Optimization Based Input Generation”. As shown in [3], a problem formulation like this can be tackled by dynamic optimization using a receding horizon approach. The closed-loop dynamics is not known exactly. However, the control concept presented in [11] is based on a reference model, which can then be considered to be an approximation of the closed-loop. A good starting point for a literature review of alternative approaches can be found in the surveys [2, 9].

In this paper, the results of [3] are extended by details of the numerical implementation of the problem. We present an implementation capable to solve this kind of problem. The implementation is based on the open-source project ACADO Toolkit [8]. A nonlinear representation of the closed-loop behavior of the helicopter is considered as well as nonlinear paths that do not correspond to paths created by trimmed trajectories.

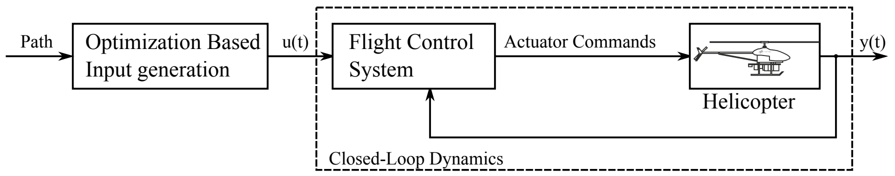

Fig. 2 Simplified block diagram of the optimization based generation of inputs for the flight control system

2 Problem Formulation

This section gives an overview on the problem formulation of path following for an unmanned helicopter. A space curve is defined, which the helicopter is supposed to track. An optimal control problem (OCP) will be formulated.

The closed-loop approximation of the flight control system can be represented in state space by

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \quad \mathbf{x}(0) = \mathbf{x}_0 \tag{1a}$$

$$\mathbf{y}(t) = \mathbf{h}(\mathbf{x}(t)), \tag{1b}$$

where the state vector $\mathbf{x}$ contains position and velocity of the helicopter, as well as rotational and engine states. The inputs $\mathbf{u}$ are the input channels of the flight control system. They contain a velocity command $\mathbf{u}_v$ and a command for a desired scalar azimuth u_ψ. The azimuth is the third component of an Euler representation of the helicopter's attitude corresponding to ISO 1151. The outputs are defined by the position of the helicopter ($\mathbf{r}$) in Cartesian north-east-down (NED) frame and the azimuth (ψ), thus

$$\mathbf{u}(t) = (\mathbf{u}_v(t)^T, u_\psi(t))^T \in \mathbb{R}^4, \tag{2a}$$

$$\mathbf{y}(t) = (\mathbf{r}(t)^T, \psi(t))^T \in \mathbb{R}^4. \tag{2b}$$

This representation results in a consistent problem formulation for missions in obstacle occupied environments for short distance missions. Alternatives can also be considered, for example velocity in respect to wind as well as the side slip angle, which partly defines the attitude of the helicopter with respect to aerodynamic inflow.

In this paper we focus on the structure of the problem. The information needed to reconstruct the complete set of equations can be found in in [3]. For brevity, the following paragraphs limit to the general idea and the needed links to [3] by providing the physical meaning of some of the variables. The state vector can be subdivided into four sub-vectors: the translation dynamics $\mathbf{x}_t \in \mathbb{R}^6$, states of the engine $\mathbf{x}_e \in \mathbb{R}^2$, states for the rotational rates $\mathbf{x}_r \in \mathbb{R}^6$ and states of the quaternion based representation of the attitude $\mathbf{x}_q \in \mathbb{R}^4$. The closed-loop dynamics can be represented under the following structure

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{\mathbf{x}}_e(t) \\ \dot{\mathbf{x}}_r(t) \\ \dot{\mathbf{x}}_q(t) \\ \dot{\mathbf{x}}_t(t) \end{pmatrix} = \begin{pmatrix} \mathbf{A}_e\mathbf{x}_e(t) + \mathbf{g}_e(\mathbf{x}(t), \mathbf{u}_v(t)) \\ \mathbf{A}_r\mathbf{x}_r(t) + \mathbf{g}_r(\mathbf{x}(t), \mathbf{u}_v(t), u_\psi(t)) \\ \mathbf{f}_q(\mathbf{x}_q(t), \mathbf{x}_r(t)) \\ \mathbf{f}_t(\mathbf{x}(t)) \end{pmatrix}, \quad \mathbf{x}(0) = \mathbf{x}_0 \in \mathbb{R}^{18}. \tag{3}$$

Note that there are two components ($\mathbf{x}_e$ and $\mathbf{x}_r$) with linear state map and nonlinear input functions as well as two components ($\mathbf{x}_q$ and $\mathbf{x}_t$) with nonlinear dynamics but not directly influenced by the inputs. The optimization problem can be simplified by exploiting the fact that the engine dynamics $\mathbf{x}_e$ are differentially flat with the thrust of the main rotor as flat output if the remaining states are considered as parameters; see [7] for details on differential flatness. Thus it is possible to calculate the input $\mathbf{u}_v$ if a time-wise evolution of the thrust and a sufficient number of successive derivatives are given. An equivalent argumentation holds for the dynamics of $\mathbf{x}_r$ considering the remaining states and $\mathbf{u}_v$ as parameters. The flat outputs in this case are the body-fixed rotation rates.

These flat subsystems allow to define simple integrator chains of sufficient order with new inputs for the thrust u_T and rotation rates $\mathbf{u}_\omega$, in the following represented using the sparse system matrices $\mathbf{A}_{1,2}$ and input matrices $\mathbf{B}_{1,2}$ containing only a small number of ones. It is thus possible to derive the state trajectories for $\mathbf{x}_e$ and $\mathbf{x}_r$ only by integration of the new inputs and using the algebraic relations of the flat outputs, which results in modifications of $\mathbf{f}_q$ and $\mathbf{f}_t$. The original inputs can be calculated in the same way. The modified structure can then be represented by

$$\dot{\mathbf{x}}_m = \begin{pmatrix} \dot{\mathbf{x}}_{e,m}(t) \\ \dot{\mathbf{x}}_{r,m}(t) \\ \dot{\mathbf{x}}_q(t) \\ \dot{\mathbf{x}}_t(t) \end{pmatrix} = \begin{pmatrix} \mathbf{A}_1\mathbf{x}_{e,m}(t) + \mathbf{B}_1 u_t(t)) \\ \mathbf{A}_2\mathbf{x}_{r,m}(t) + \mathbf{B}_2\mathbf{u}_\omega(t) \\ \mathbf{f}_{q,m}(\mathbf{x}_q(t), \mathbf{x}_{r,m}(t)) \\ \mathbf{f}_{t,m}(\mathbf{x}_m(t)) \end{pmatrix}, \quad \mathbf{x}_m(0) = \mathbf{x}_{m,0} \in \mathbb{R}^{18}. \tag{4}$$

The control system also imposes constraints on the optimization problem. These constraints are defined by the so-called envelope protection. We omit details here and represent the constraint sets of the states as X and of the inputs as U. Details can be found in [3], where constraints for accelerations, velocities, in both body-fixed as well as in NED frame, and actuator deflections are introduced.

The path that has to be tracked with the helicopter is a four dimensional parametric curve depending on a parameter θ. It is defined in the output space of the flight control system

$$\mathscr{P} = \left\{ p(\theta) \in \mathbb{R}^4 \mid \theta \in [\theta_0, \theta_1] \mapsto \left(\mathbf{r}^T(\theta), \psi(\theta)\right)^T \right\}. \tag{5}$$

For the time-wise evolution of the position on the path, we introduce artificial dynamics with an input v which augment the dynamics of the system [5], which was chosen to be a double integrator

$$\dot{\mathbf{z}} = \begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \mathbf{z} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} v, \quad \mathbf{z}(0) = \mathbf{z}_0, \tag{6a}$$

$$\theta = z_1. \tag{6b}$$

Higher degrees of the path-dynamics would increase the smoothness of the evolution along the path. It would, however, increase the computational burden as well. The second order integrator allows us to specify a desired velocity along the path later on.

Now, the optimization problem can be defined: Given the closed-loop approximation according to (1) and a path of the form (5), calculate the time-wise progress $\theta : [\theta_0, \theta_1] \mapsto [t_0, t_1]$ and the inputs $\mathbf{u}$, such that (a) the constraints are satisfied, (b) the progress on the path is positive ($\dot{\theta} > 0$) and (c) the cost function is minimized:

$$\underset{\mathbf{u}_\omega(\cdot),u_t(\cdot),v(\cdot)}{\text{minimize}} \quad \int_0^\tau \underbrace{\left\|(\mathbf{e}^T(t),\dot{\mathbf{e}}^T(t))\right\|^2_{\mathbf{Q}_e}}_{\text{path deviation}} + \underbrace{\|\mathbf{z}(t)-\mathbf{z}_r(t)\|^2_{\mathbf{Q}_z}}_{\text{reference behavior}} + \underbrace{\left\|(\mathbf{u}^T(t),v(t))\right\|^2_{\mathbf{R}}}_{\text{regularization}} dt, \tag{7a}$$

subject to the dynamics and constraints

$$\dot{\mathbf{x}}_m(t) = \mathbf{f}_m(\mathbf{x}_m(t),u_T(t),\mathbf{u}_\omega(t)), \quad \mathbf{x}(0) = \mathbf{x}_0 \in \mathbb{R}^{18} \tag{7b}$$

$$\dot{\mathbf{z}}(t) = \mathbf{l}(\mathbf{z}(t),v(t)), \quad \mathbf{z}(0) = \mathbf{z}_0 \in \mathbb{R}^2 \tag{7c}$$

$$\dot{\mathbf{e}}(t) = \frac{\partial \mathbf{h}}{\partial \mathbf{x}_m}\mathbf{f}_m(\mathbf{x}_m(t),u_T(t),\mathbf{u}_\omega(t)) - \frac{\partial \mathbf{p}}{\partial \theta}\dot{\theta}, \quad \mathbf{e}(0) = \mathbf{e}_0 \in \mathbb{R}^4 \tag{7d}$$

$$\mathbf{x}(t) \in X, \mathbf{u}(t) \in U. \tag{7e}$$

The path error **e** leads to tracking of the reference path and its derivative avoids solutions oscillating around it. The reference behavior term in (7a) allows us to specify dynamic requirements like a desired velocity along the path and finally the regulation enforces certain smoothness on the derived state trajectories.

3 Implementation and Computational Results

The path optimization described in the previous section has been carried out with the help of the open-source project ACADO Toolkit [8]. An advantage of this project is what the authors refer to as code generation. A piece of self-contained code is generated based on the mathematical problem formulation. This code contains an efficient implementation of the optimization problem based on a tailored discretization. This code can then either be used separately, interfacing with MATLAB or directly integrated into piece of software for the desired application.

The optimization is performed using a direct multiple shooting approach [12]. We use the receding horizon technique with a prediction horizon T to reduce the computational burden in contrast to optimize over the complete path. The length of the prediction horizon is chosen taking the computation time and a minimal stopping distance into account. The minimal stopping distance can be transferred into a minimal prediction horizon using the velocity maximally allowed and the deceleration limits which are implemented in the flight control system.

The prediction horizon is subdivided into equal shooting intervals. The solution of the differential equations over these multiple-shooting intervals is normally performed using adaptive step size integration algorithms. Using adaptive integrators has the advantage that the integration grid does not have to be specified a priori. However, it comes at the cost of non-deterministic discretization. This is why in [8] the use of fixed-step size integrators is proposed. The integration algorithm and its step-size has to be determined a priori, e. g. heuristically. By doing so, it is possible to tailor a discretization that is deterministic in calculation time and allows the generation of efficient code exploiting aspects like static memory allocation.

By these means, a nonlinear program (NLP) is formulated for each shift of the prediction horizon that, if feasible, directly solves the optimization problem. The NLP is solved in an iterative process known as sequential quadratic programming (SQP). In each sequential step a quadratic approximation of the cost function is created as well as affine approximations of the constraints thus creating a quadratic program (QP). There exist powerful methods to solve QP. Here, the qpOASES Package [6] is used for that purpose. Using the solution of the QP, the original NLP is approximated again and this process is repeated until a certain residual of the KKT-conditions is sufficiently small. A comprehensive tutorial on this process can be found in [4].

At the beginning of the path an initial guess is used, which corresponds to the hover states of the helicopter. These conditions can be determined by trim calculations [10] or simulation experiments. Using this initial guess, the first NLP is solved over the prediction horizon $T = 2.5\,\mathrm{s}$. Each prediction horizon is subdivided into 25 shooting intervals, of which each is solved using an implicit Runge-Kutta integrator of second order with three discretization steps. The number of SQP formulations that are required to achieve residuals of the KKT-conditions that we define to be less than 10^{-4} and is referred to as SQP iterations in the following. Each SQP iteration requires the solution of a QP that again is solved iteratively and is referred to as QP iterations in the following. After converging the result of the first shooting interval is used for the final solution trajectory. The remaining intervals serve as initial guess after shifting the prediction horizon by one shooting interval thus forming the receding horizon approach.

The path shown in Figure 3 shall serve as an example. The clover leaf with a height profile is generated using

$$\mathbf{r}_p = \begin{pmatrix} \hat{r}\cos\left(3\theta - \frac{3}{2}\pi\right)\cos(\theta) \\ \hat{r}\cos\left(3\theta - \frac{3}{2}\pi\right)\sin(\theta) \\ \hat{h}\sin(4\theta) \end{pmatrix}. \tag{8}$$

Where $\hat{r}$ is the clover leaf's radius of 25 m and $\hat{h}$ the amplitude of the height profile of 3 m. The fourth component of the path, the azimuth, has been defined such that the helicopter is always oriented tangential to the movement resulting in flight without sideslip angle in the wind-free case.

Figure 4 shows the computation characteristics of the optimization. The upper left figure presents the required computation time over the timeline of the solution trajectory. The figure shows how long it takes to compute one NLP that is to optimize once over the prediction horizon. Additionally, the computation time of the preparation is shown, which contains the discretization and the setup of the QP Problem. The feedback time refers to the time needed to solve the QP. The lower left plot depicts how many QP approximations are necessary in total to solve each NLP. The last figure shows the number of iterations necessary to solve a certain QP, which is the only graph not having the trajectory time as abscissa. As for each NLP multiple QP have to be solved the number of iterations for each SQP step are shown.

From the achieved KKT values it is apparent that the formulated problem is feasible as the maximum value allowed is never exceeded. The desired velocity was set to be 8 m/s resulting in a duration of less than 40 s to complete the flight. The overall computation on a standard desktop computer without parallelization takes around 10 s. This alone is a good result, as it means that the optimization over the complete path takes less than a third of the time it takes to fly it.

There are in total five regions of higher computational burden. It can be seen that four of these regions are significantly influenced by the number of SQP iteration. However, the fourth region is created by the QP iteration alone. The discretization of the problem thus gives a lower bound on the needed computation time, which is in this case around 0.02 s for an prediction horizon of 2.5 s. Nevertheless, the solution of the QP problems has significant impact on the overall computational burden as well and lies around the same order of magnitude.

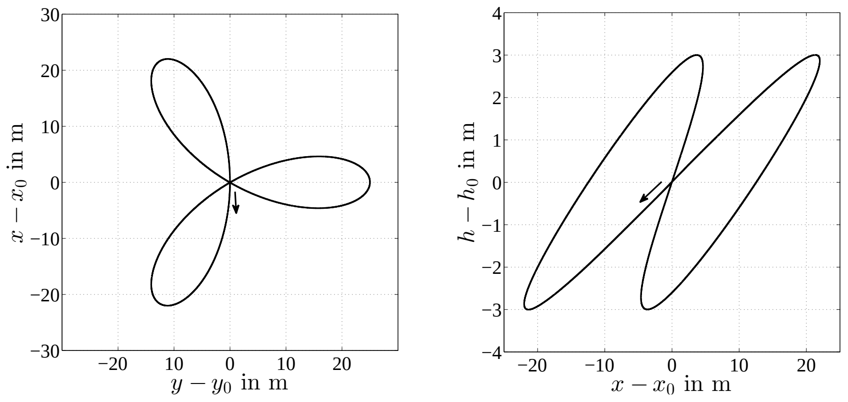

Fig. 3 Example path: clover leaf with sinusoidal height profile. The arrows indicate the start and direction of flight, while the left shows the top-view and right the height over north direction.

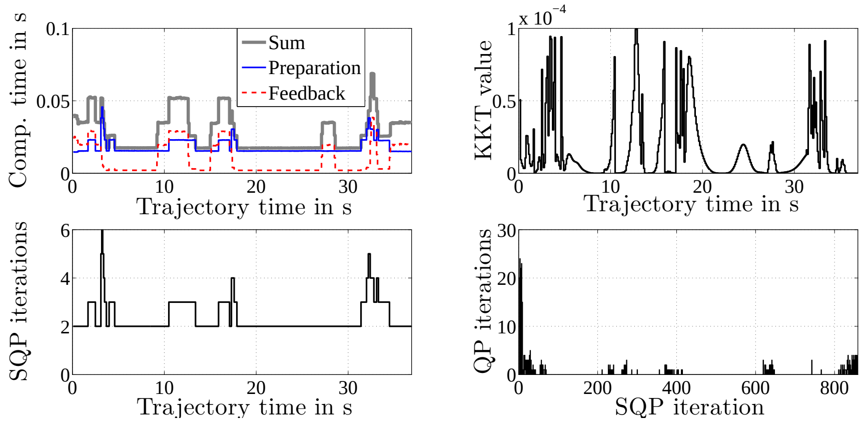

Fig. 4 Calculation characteristics of the example path

4 Conclusion

The proposed approach is well capable to tackle the problem of path following for unmanned helicopters where an approximation of the closed-loop dynamics is available. It is shown that trajectories can be generated faster than it would take to fly them. This fact makes the presented approach very powerful as a great variety of paths and reference variations can be solved in reasonable time. Applications can be mission planning purposes or generation of controller inputs that can be stored in a maneuver database. However, an upper bound for prediction is not guaranteed. Future work will thus investigate possibilities to enable any-time capability which would render the approach also applicable for online purposes. The results of this paper show that an adaption has to consider both, the SQP iterations and QP iterations as both have significant impact on the computation time.

References

1. Adolf, F.M., Andert, F.: Rapid multi-query path planning for a vertical take-off and landing unmanned aerial vehicle. Journal of Aerospace Computing Information and Communication **8**(11), 310–327 (2011)
2. Dadkhah, N., Mettler, B.: Survey of motion planning literature in the presence of uncertainty: Considerations for UAV guidance. Journal of Intelligent & Robotic Systems **65**(1-4), 233–246 (2012). DOI 10.1007/s10846-011-9642-9
3. Dauer, J.C., Faulwasser, T., Lorenz, S., Findeisen, R.: Optimization-based feedforward path following for model reference adaptive control of an unmanned helicopter. In: Proceedings of the AIAA Guidance, Navigation, and Control Conference 2013, AIAA 2013–5002. Boston,MA (2013). DOI 10.2514/6.2013-5002
4. Diehl, M., Bock, H., Diedam, H., Wieber, P.B.: Fast direct multiple shooting algorithms for optimal robot control. In: M. Diehl, K. Mombaur (eds.) Fast Motions in Biomechanics and Robotics, *Lecture Notes in Control and Information Sciences*, vol. 340, pp. 65–93. Springer Berlin Heidelberg (2006)
5. Faulwasser, T.: Optimization-based solutions to constrained trajectory-tracking and path-following problems. Shaker, Aachen, Germany (2013). DOI 10.2370/9783844015942
6. Ferreau, H., Bock, H., Diehl, M.: An online active set strategy to overcome the limitations of explicit mpc. International Journal of Robust and Nonlinear Control **18**(8), 816–830 (2008)
7. Fliess, M., Lévine, J., Martin, P., Rouchon, P.: Flatness and defect of non-linear systems: introductory theory and examples. Int. J. Contr. **61**(6), 1327–1361 (1995)
8. Houska, B., Ferreau, H., Diehl, M.: ACADO toolkit – an open-source framework for automatic control and dynamic optimization. Optimal Control Applications and Methods **32**(3), 298–312 (2011)
9. Huang, G.Q., Lu, Y.P., Nan, Y.: A survey of numerical algorithms for trajectory optimization of flight vehicles. Science China Technological Sciences **55**(9), 2538–2560 (2012). DOI 10.1007/s11431-012-4946-y
10. Leishmann, J.G.: Principles of Helicopter Aerodynamics, second edn. Cambridge University Press, Cambridge (2006)
11. Lorenz, S.: Open-Loop Reference System for Nonlinear Control Applied to Unmanned Helicopters. Journal of Guidance, Control, and Dynamics **35**(1), 259–269 (2012). DOI 10.2514/1.52033
12. Nocedal, J., Wright, S.: Numerical optimization, 2nd edn. Springer series in operations research and financial engineering. Springer, New York (2000)