Time Optimal Flight of a Hexacopter

Johan Vertens and Fabian Fischer

***Abstract*— This project presents a way to calculate a time optimal control path of a hexacopter using direct multiple shooting in Matlab and the code generated integrators from the ACADO toolkit. The path is along some waypoints and within constraints. A physical model of the hexacopter and some example trajectories are also presented. The trajectory was simulated in Matlab and visualized with RViz from ROS.**

I. INTRODUCTION

In this project we calculated an optimal control of a hexacopter using direct Multiple Shooting. The goal was to let a multi-rotor system with 6 rotors fly along some waypoints and within some given constraints, e.g. on angle and speed. We used a physical model of the hexacopter and Matlab's `fmincon` in combination with a runge kutta integrator of the order four for the control and trajectory calculation. The trajectory was then visualized using a Matlab-ROS bridge and the ROS tool RViz.
The hexacopter model is presented in Section II, while the control calculation is presented in Section III. In the final Section IV we present two example trajectories.

II. HEXACOPTER MODEL

For simulating the hexacopter within Matlab we implemented a model of the system and pd-regulators consisting of ordinary differential equations (ODE). Movement of the hexacopter can be generated by changing the thrusts of the different motors ($T_1 ... T_6$, Fig. 1) which leads to a change of the torques and the velocities. In order to compensate the rotational moment of the motors there are three motors spinning right and three motors spinning left.
The torques can be described with the following equation. The motor thrusts are negative here because the z axis is pointing downwards.

$$\begin{pmatrix} T \\ M_x \\ M_y \\ M_z \end{pmatrix} = \begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 \\ 0 & -L\frac{\sqrt[2]{3}}{2} & -L\frac{\sqrt[2]{3}}{2} & 0 & L\frac{\sqrt[2]{3}}{2} & L\frac{\sqrt[2]{3}}{2} \\ L & \frac{L}{2} & -\frac{L}{2} & -L & -\frac{L}{2} & \frac{L}{2} \\ -M_c & M_c & -M_c & M_c & -M_c & M_c \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \end{pmatrix} \quad (1)$$

T : total thrust
$T_1 ... T_6$: thrust of each motor
M_x, M_y, M_z : torques of the x, y and z axis
L : distance between the motor axis and the center of the hexacopter

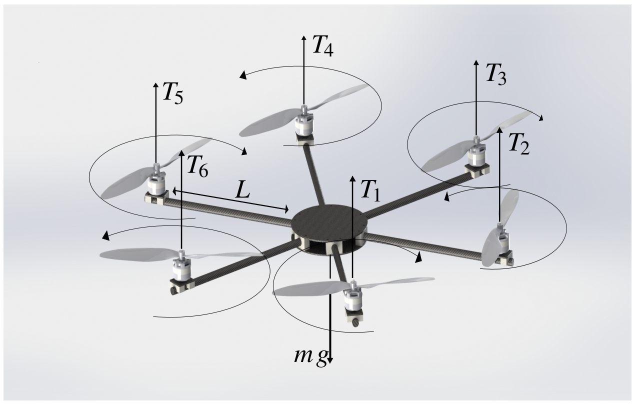

Fig. 1. Thrusts of the motors [1]

With the calculated torques it is easily possible to compute also the velocities and the accelerations of the system. Here the translation-velocities are in the coordinate system of the world, while the angular velocities and angular accelerations are in the coordinate system of the hexacopter.

We summarize all quantities of interest:
$\dot{v}_x, \dot{v}_y, \dot{v}_z$: Accelerations in x-, y- and z
$\omega_x, \omega_y, \omega_z$: Angular velocities of x- ,y- and z
$\dot{x}, \dot{y}, \dot{z}$: Velocities of x-, y- und z.
Θ : Pitch angle
Φ : Roll angle
Ψ : Yaw angle
K_y : Propeller-yaw ratio
M_c : Yaw moment of the motors
J_{mp} : Inertia of a propeller
J_x, J_y, J_z : Inertia of the hexacopter

For the model equation part we used the model from Baranek et. al presented in [6]. The accelerations in the local coordinate system of the hexacopter can be modeled in the following way:

$$\begin{aligned} \dot{v}_x &= -v_z\omega_y + v_y\omega_z - g\sin(\Theta) \\ \dot{v}_y &= -v_x\omega_z + v_z\omega_x + g\cos(\Theta)\sin(\Phi) \\ \dot{v}_z &= -v_y\omega_x + v_x\omega_y + g\cos(\Theta)\cos(\Phi) - \frac{T}{m} \end{aligned} \quad (2)$$

Optimal Control and Estimation Course, Final Project
Albert-Ludwigs University of Freiburg, Faculty of Engineering

The angular accelerations are modeled with:

$$\begin{aligned}
\dot{\omega}_x &= \frac{1}{J_x}(-\omega_y\omega_z(J_z-J_y)+M_x+K_y\cdot J_{mp}\cdot M_z\cdot\omega_y)\\
\dot{\omega}_y &= \frac{1}{J_y}(-\omega_x\omega_z(J_x-J_z)+M_y+K_y\cdot J_{mp}\cdot M_z\cdot\omega_x)\\
\dot{\omega}_z &= \frac{M_z}{J_z}
\end{aligned} \tag{3}$$

The angular velocities are modeled with:

$$\begin{aligned}
\dot{\Theta} &= \omega_y\cos(\Phi)-\omega_z\sin(\Phi)\\
\dot{\Phi} &= \omega_x+\omega_y\sin(\Phi)\tan(\Theta)+\omega_z\cos(\Phi)\tan(\Theta)\\
\dot{\Psi} &= \omega_y\frac{\sin(\Phi)}{\cos(\Theta)}+\omega_z\frac{\cos(\Phi)}{\cos(\Theta)}
\end{aligned} \tag{4}$$

The velocities of the hexacopter in the world coordinate system are modeled with:

$$\begin{aligned}
\dot{x} &= \cos(\Phi)\cos(\Theta)v_x+(-\sin(\Psi)\cos(\Phi)\\
&\quad+\cos(\Psi)\sin(\Theta)\sin(\Phi))v_y\\
&\quad+(\sin(\Psi)\sin(\Phi)+\cos(\Psi)\sin(\Theta)\cos(\Phi))v_z\\
\dot{y} &= \sin(\Phi)\cos(\Theta)v_x+(-\cos(\Psi)\cos(\Phi)\\
&\quad+\sin(\Psi)\sin(\Theta)\sin(\Phi))v_y\\
&\quad+(-\cos(\Psi)\sin(\Phi)+\sin(\Psi)\sin(\Theta)\cos(\Phi))v_z\\
\dot{z} &= -\sin(\Theta)v_x+\cos(\Theta)\sin(\Phi)v_y+\cos(\Theta)\cos(\Phi)v_z
\end{aligned} \tag{5}$$

A. PD Regulators

The PD-regulators are controlling the motor thrusts with respect to angle inputs in order to be able to control roll, pitch and yaw (Fig. 2), while the total thrust is controlled directly without any regulator. The resulting controls from the PD's ($R_{\text{pitch}}, R_{\text{roll}}, R_{\text{yaw}}$) are combined in a motor mixer function Eq.9 to calculate the desired thrusts of each motor. This function is part of the microcopter driver [2].
In conclusion the control inputs of the model are then $u_{\text{pitch}}, u_{\text{roll}}$, u_{yaw} and u_{thrust}.
The parameters given in Tab. I were obtained from Solidworks simulations and experiments. A overview of the inputs is represented in Fig. 3.

$$\begin{aligned}
R_{\text{pitch}} &= P_1\, e_{\text{pitch}}(t)+D_1\,\frac{d}{dt}e_{\text{pitch}}(t)\\
R_{\text{roll}} &= P_2\, e_{\text{roll}}(t)+D_2\,\frac{d}{dt}e_{\text{roll}}(t)\\
R_{\text{yaw}} &= P_3\, e_{\text{yaw}}(t)+D_3\,\frac{d}{dt}e_{\text{yaw}}(t)
\end{aligned} \tag{6}$$

with the equations in (7,8):

$$\begin{aligned}
e_{\text{pitch}}(t) &= \Theta(t)-u_{\text{pitch}}(t)\\
e_{\text{roll}}(t) &= \Phi(t)-u_{\text{roll}}(t)\\
e_{\text{yaw}}(t) &= \Psi(t)-u_{\text{yaw}}(t)
\end{aligned} \tag{7}$$

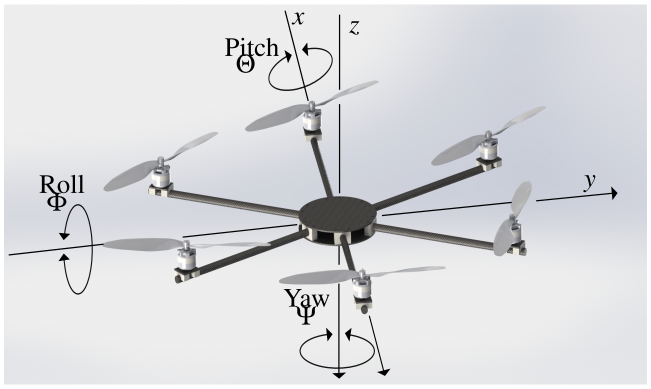

Fig. 2. Local coordinate system of the hexacopter [1]

TABLE I
PARAMETERS

Parameter	Value	Parameter	Value
K_y	0.001	P_1	8
J_{mp}	0.00102638 $\frac{\text{kgm}}{\text{s}^2}$	P_1	0.83
J_x	0.006503 $\frac{\text{kgm}}{\text{s}^2}$	P_3	9
J_y	0.006503 $\frac{\text{kgm}}{\text{s}^2}$	D_1	0.83
J_z	0.012983 $\frac{\text{kgm}}{\text{s}^2}$	D_2	0.3
M_c	0.1 Nm	D_3	0.2

$$\begin{aligned}
\frac{d}{dt}e_{\text{pitch}}(t) &\approx \omega_x\\
\frac{d}{dt}e_{\text{roll}}(t) &\approx \omega_y\\
\frac{d}{dt}e_{\text{yaw}}(t) &\approx \omega_z
\end{aligned} \tag{8}$$

$$\begin{aligned}
T_1 &= u_{\text{thrust}}+\delta_{\text{pitch}}+\delta_{\text{yaw}}\\
T_2 &= u_{\text{thrust}}+\delta_{\text{roll}}+\frac{\delta_{\text{pitch}}}{2}-\delta_{\text{yaw}}\\
T_3 &= u_{\text{thrust}}+\delta_{\text{roll}}-\frac{\delta_{\text{pitch}}}{2}+\delta_{\text{yaw}}\\
T_4 &= u_{\text{thrust}}-\delta_{\text{pitch}}-\delta_{\text{yaw}}\\
T_5 &= u_{\text{thrust}}-\delta_{\text{roll}}-\frac{\delta_{\text{pitch}}}{2}+\delta_{\text{yaw}}\\
T_6 &= u_{\text{thrust}}-\delta_{\text{roll}}+\frac{\delta_{\text{pitch}}}{2}-\delta_{\text{yaw}}
\end{aligned} \tag{9}$$

with Eq. 10:

$$\begin{aligned}
\delta_{\text{pitch}} &= \frac{pd_{\text{pitch}}}{3}\\
\delta_{\text{roll}} &= \frac{4pd_{\text{roll}}}{7}\\
\delta_{\text{yaw}} &= pd_{\text{yaw}}
\end{aligned} \tag{10}$$

$$\tag{11}$$

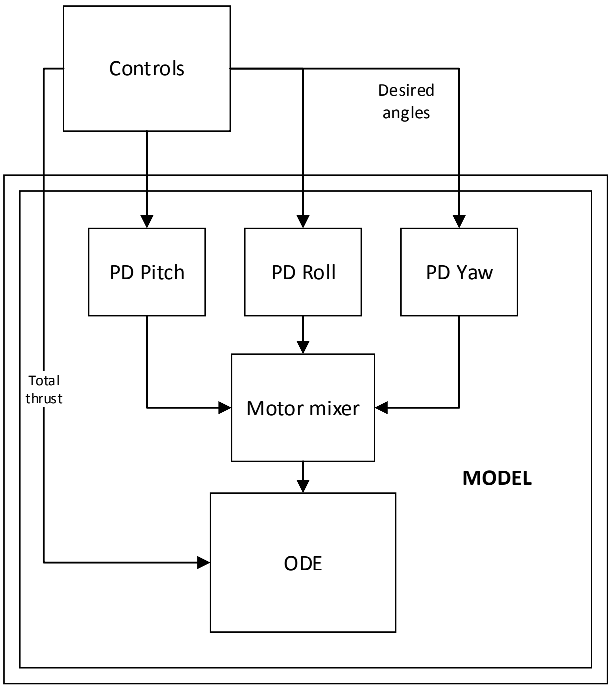

Fig. 3. Model structure

III. Optimization

The problem is to find a time optimal trajectory from a starting point $\bar{x}_0$ to a terminal point $\bar{x}_m$ via some $k = 1\dots m-1$ waypoints $\bar{x}_k$.

A. Problem description

Because this is a time optimal problem we used the method in [3] by adding a free variable T which is constant between two waypoints to the state x of the system. The scaling factor T can be changed, but only at a waypoint state $s_{i(k)}$. This way it is possible to find a time optimal flight path between two waypoints, which then also leads to an overall time optimal flight path. The augmented state s and system function $f(s,q)$ can be seen in (12). The set $I(k) := \{i(1),\dots,i(m-1)\}$ denotes the set of all indices where a new waypoint starts, with $i(0) = 0$ and $i(m) = N$.

$$\begin{aligned} s_i &= \begin{bmatrix} x_i \\ T_i \end{bmatrix} \\ f(s_i,q_i) &= \begin{bmatrix} T_i \cdot f_{\text{model}}(x_i,q_i) \\ T_i \end{bmatrix} \end{aligned} \tag{12}$$

The continous time optimal problem can be described as:

$$\min_{s,q} \frac{1}{m} \sum_{k=0}^{m-1} T_{i(k)} \tag{13}$$

subject to:

$$\begin{aligned} x_0 - \bar{x}_0 &= 0 \\ f(s_i,q_i) - s_{i+1} &= 0, i \in \{0,\dots,N-1\} \setminus \{i(1),\dots,i(m-1)\} \\ P \cdot f(s_i,q_i) - x_{i+1} &= 0, i \in \{i(1),i(2),\dots,i(m-1)\} \\ x_{\min} \leq \quad x_i &\leq x_{\max}, i \in \{0,\dots,N\} \\ u_{\min} \leq \quad u_i &\leq u_{\max}, i \in \{0,\dots,N-1\} \\ T_{\min} \leq \quad T_i &\leq T_{\max}, i \in \{0,\dots,N\} \end{aligned}$$

$$P = \left[\begin{array}{cccc|c} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right]_{12\times 13} \tag{14}$$

where P is a projection matrix from s to x. In the objective function we sum up all the different $T_{i(k)}$ in the states $s_{i(k)}$ at the waypoints and add the $T_{i(0)}$ in the initial State $s_{i(0)}$. This sum divided by m is the overall flight time of the hexacopter.

B. Solving with fmincon

For solving the above problem we used Direct Multiple Shooting [4] in combination with the Matlab `fmincon` solver. By choosing a fixed number of controls N we made a design variables vector D which contains all the states and controls.

$$D = \begin{bmatrix} s_0 \\ q_0 \\ s_1 \\ q_1 \\ \vdots \\ s_{N-1} \\ s_{N-1} \\ s_N \end{bmatrix} \tag{15}$$

As an inital guess x_0 and x_N were set to $\bar{x}_0$ and $\bar{x}_m$ and all T state variables to some initial flight time $\bar{T}_0$. In `fmincon` we used an upper- and lowerbound vector to constrain D in the following way:

- control constraints: By constraining the controls like in Table II we avoided negative or excessive thrust and oversteering with too high pitch and roll angle controls.
- state constraints: By constraining all states to the minimal and maximal values in Table III we avoided behaviour which is not supported by the hexacopter model. Also we excluded flying through the ground plane and a negative time-scaling factor T.
- initial state, terminal state and waypoint constraints: By constraining the minimal and maximal values of the initial state s_0 and terminal state s_N to the initial and terminal points $\bar{x}_0$ and $\bar{x}_N$ respectivly, we determined the start and end point. The k waypoints were determined in the same manner at every $\left\lfloor \frac{j \cdot N}{k+1} \right\rfloor$ state for $j = 1\dots k$. This way every path between two points had the same number of controls. For the waypoints and the end point

TABLE II
LOWER AND UPPER BOUND OF THE CONTROLS

Control	min	max
Thrust	0 N	5.5 N
Pitch	$-50°$	$50°$
Roll	$-50°$	$50°$
Yaw	$-180°$	$180°$

TABLE III
LOWER AND UPPER BOUND OF THE STATES

State	min	max
Velocity in X-direction	-16 m/s	16 m/s
Velocity in Y-direction	-16 m/s	16 m/s
Velocity in Z-direction	-6 m/s	6 m/s
Pitch	$-50°$	$50°$
Roll	$-50°$	$50°$
Yaw	$-180°$	$180°$

we also added, respectivly subtracted, some small delta values to the states, which helps finding a solution.

The objective function used by `fmincon` returned the overall flight time calculated in Eq. (13).

We modeled the system dynamics using the `nonlnconstr` function provided to `fmincon`. In this function we calculated the vector c_{eq} which can be seen in (16). The integrator function $f(s_i, q_i)$ is explained in detail in Section III-C. Since the integrator function provided the sensitivity with respect to s_i and q_i it was possible to build the Jacobian G_{ceq} of the `nonlnconstr` function, which can be seen in (17). By switching the `GradConstr` option on, `fmincon` uses the Jacobian for finding a solution faster.

$$c_{\text{eq}} = \left\{ \begin{matrix} f(s_0, q_0) - s_1 \\ f(s_1, q_1) - s_2 \\ \vdots \\ f(s_{N-1}, q_{N-1}) - s_N \end{matrix} \right\} \tag{16}$$

$$\begin{aligned} G_{\text{ceq}} &= \begin{bmatrix} A_0 & B_0 & -I & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & A_1 & B_1 & -I & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & A_{N-1} & B_{N-1} & -I \end{bmatrix} \\ A_i &= \frac{\partial f}{\partial s}(s, q) \\ B_i &= \frac{\partial f}{\partial q}(s, q) \end{aligned} \tag{17}$$

We then deleted the continuity condition for the T state variable at every waypoint-state $s_{i(k)}$ with $k = 1 \dots m-1$ in c_{eq} , so the optimizer is allowed to change the time scaling factor T in this state. We also deleted the respective line in the Jacobian G_{ceq} of the `nonlinconstr` function.
The solver `fmincon` then calculated an optimal solution using the `interior-point` algorithm.

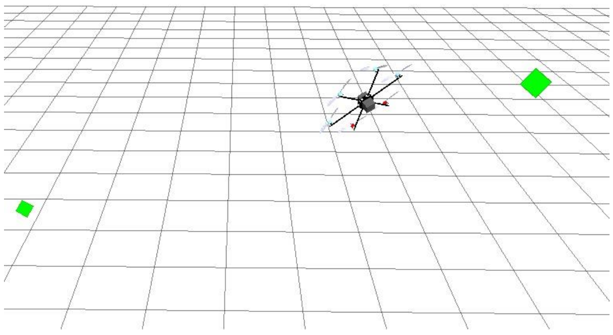
Fig. 4. Flight visualization in RViz.

C. Acado Integrator

To speed up `fmincon` we used the ACADO Toolkit and its built-in Matlab interface [5]. ACADO converted the hexacopter model from Section II to C code, which was then compiled with GCC to a binary MEX-file, callable from Matlab. We then used an RK4 integrator with 3 built-in steps provided by ACADO. The integrator step duration was set to $\frac{1}{N}$, because our augmented problem formulation was scaled from 0 to 1. This integrator function $f(s_i, q_i)$, which also provided the sensitivity in s and q direction, was used to compute the nonlinear equality constraints and their Jacobian in Section III-B.

IV. RESULTS

A. Visualization

For visualizing the whole trajectory we implemented a Matlab-ROS bridge, which sends the real time motion of the hexacopter to a ROS node. The ROS node computes the rotation and translation and transfers the data directly to RViz. On the RViz side we used a CAD Model of the hexacopter for showing the trajectory in a 3D space (Fig. 4).

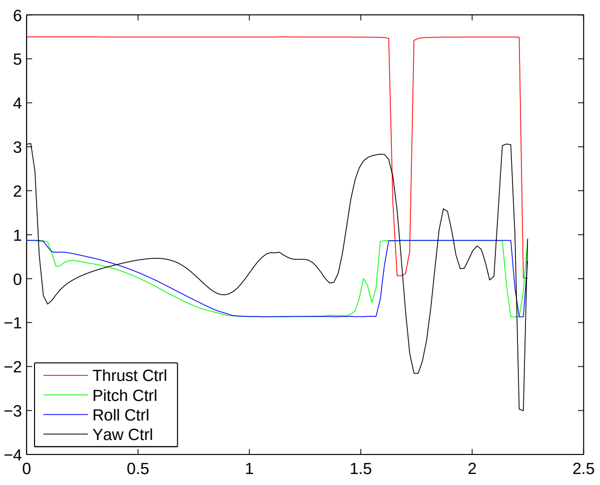

Fig. 5. Simple flight controls

B. *Simple flight*

In Fig. 5 and Fig.6 we present a simple hexacopter flight from position (0,0,0) to (2,-2,2) via (-2,0,4). In the end point the hexacopter should stop with nearly zero velocity. The duration of the flight was 2.14 s.

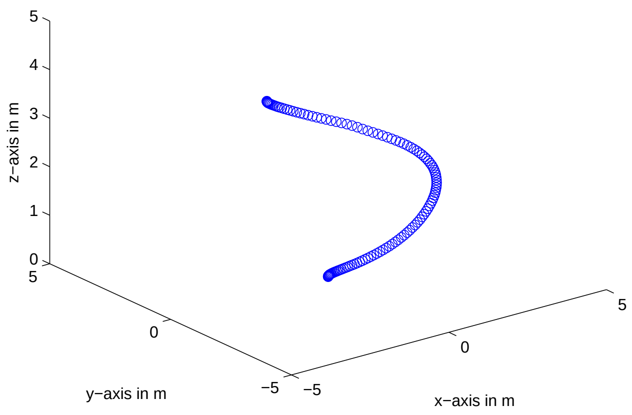

Fig. 6. Simple flight trajectory

C. *Angular-constrained flight*

In Fig. 7 and Fig. 8 we present a hexacopter flight along the waypoints given in Tab. IV. On every waypoints the velocities should be nearly zero. The duration of the flight was 5.99 s.

TABLE IV
ANGULAR-CONSTRAINED WAYPOINTS

Waypoint	X	Y	Z	Pitch	Roll	Yaw
S_{init}	0	0	0	0	0	0
S_1	3	-3	3	-45°	0	45°
S_2	3	-5	3	-45°	0	135°
S_{term}	0	0	0	0	0	0

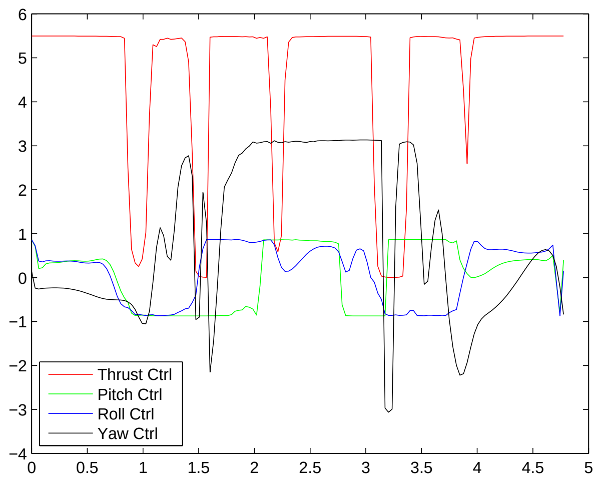

Fig. 7. Angular-constrained controls

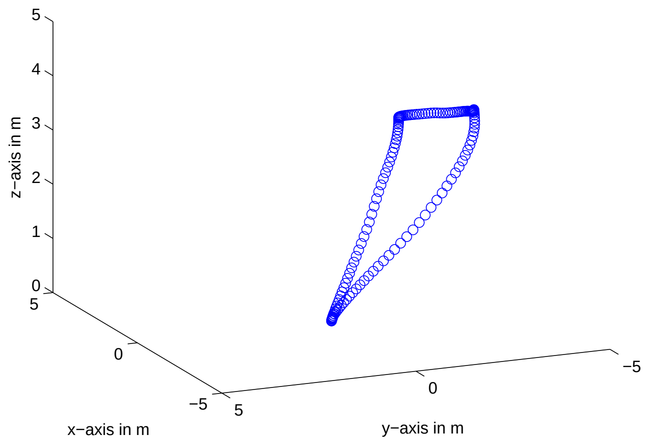

Fig. 8. Angular-constrained trajectory

REFERENCES

[1] Bachelor Thesis: 3D-Objektkartierung mit Multi-Rotor-Systemen, Johan Vertens, 2013.
[2] www.mikrokopter.de, July 27, 2014, flight-ctrl
[3] M. Diehl, Lecture Notes on Optimal Control and Estimation, July 23, 2014 , p 60.
[4] M. Diehl, Lecture Notes on Optimal Control and Estimation, July 23, 2014 , pp 64-65.
[5] www.acadotoolkit.org, July 27, 2014
[6] Baranek, R. and Solc, F. , Modelling and control of a hexa-copter, May 13, 2012