Dynamic Programming for Trajectory through Doors of a UAV

Xin Ye 16.04.2019

"uav_time_opt.m" implements a dynamic optimization for UAV trajectory with the open source library CASADI. To run this program, you need to import the library first. You might have to modify the following lines.

```
addpath('../../../casadi')
import casadi.*;
```

The input variables are the thrust forces at four rotors. The state variables are position (x, y, z), velocity (x, y, z), angles (roll, pitch, yaw), and angular rates (roll, pitch, yaw).

$$x = [p_x\ p_y\ p_z\ v_x\ v_y\ v_z\ \phi\ \psi\ \theta\ \omega_x\ \omega_y\ \omega_z]^T$$

The objective is to minimize the weighted cost of consumed energy by rotors, under constraints of UAV dynamics, bounds of states and inputs, starting and final conditions. The constraints also take the way points into consideration. In this example, there are two way points. The UAV flies pass these way points at a given state (given position, velocity etc. accordingly). These way points separate the overall time into time intervals. The objective functional is accumulated in each of the intervals.

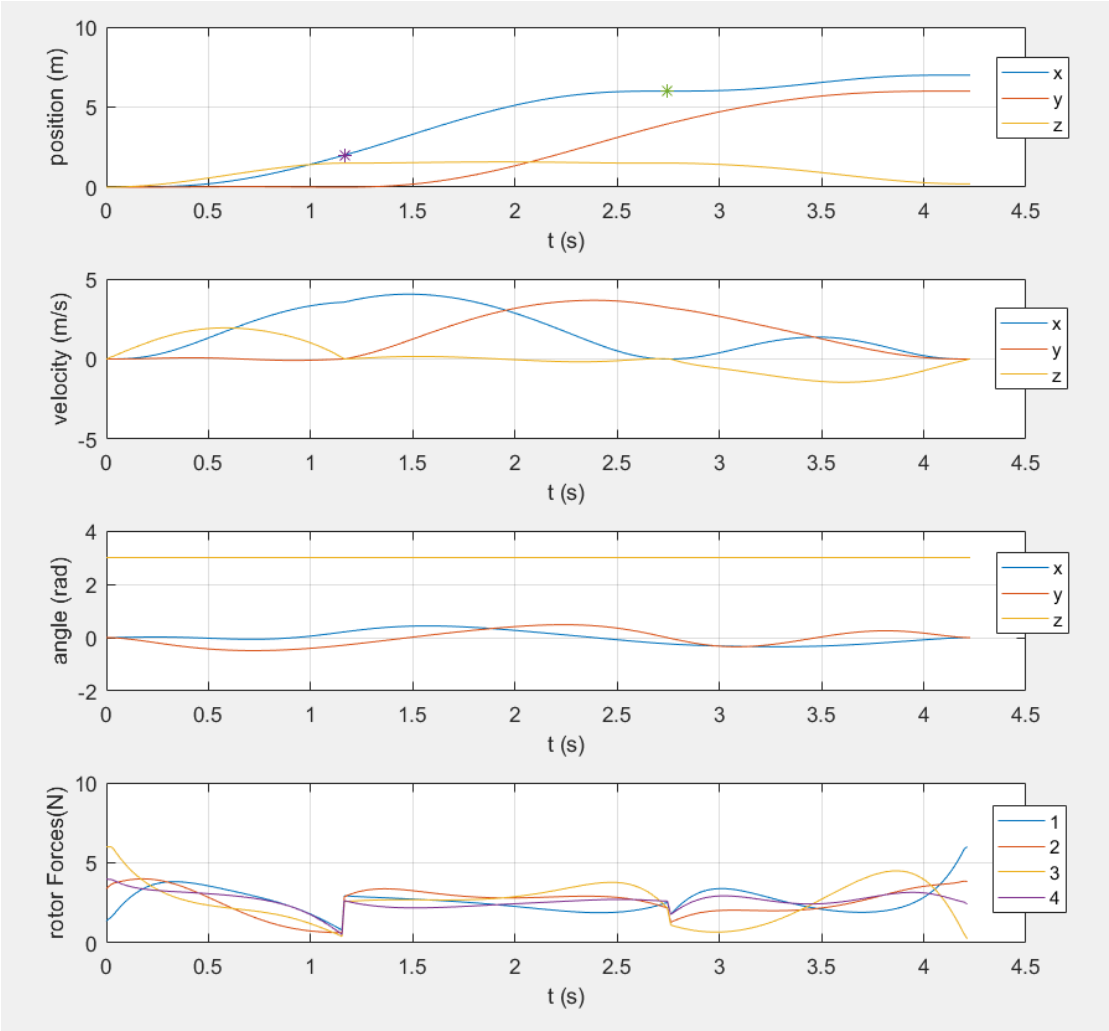
$$J = \int_{t_0}^{t_1} u^T(t)Ru(t)\mathrm{d}t + \int_{t_1}^{t_2} u^T(t)Ru(t)\mathrm{d}t + \int_{t_2}^{t_3} u^T(t)Ru(t)\mathrm{d}t$$

s.t.

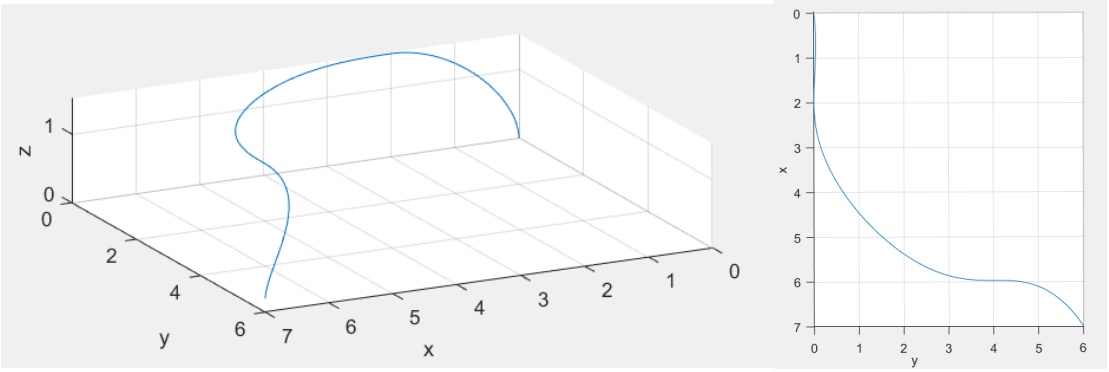
$$\frac{dx(t)}{dt} = f(t, x(t), u(t))$$
$$u(\cdot) \in [0, u_{max}]$$
$$v_{x,y,z}(\cdot) \in [-v_{max}, v_{max}]$$
$$\phi(\cdot) \in [-\phi_{max}, \phi_{max}]$$
$$\psi(\cdot) \in [-\psi_{max}, \psi_{max}]$$
$$x(t_0) = [p_{x0}\ p_{y0}\ p_{z0}\ 0\ 0\ 0\ 0\ 0\ \theta_0\ 0\ 0\ 0]^T$$
$$p(t_1) = p_1$$
$$v(t_1) = v_1$$
$$p(t_2) = p_2$$
$$v(t_2) = v_2$$
$$x(t_3) = [p_{x3}\ p_{y3}\ p_{z3}\ 0\ 0\ 0\ 0\ 0\ \theta_3\ 0\ 0\ 0]^T$$

For this free end time problem, time transformation is applied for $i = 1,2,3$.

$$t_i = \tau t_{fi}$$
$$\tau \in [0, 1]$$
$$\frac{dx(\tau)}{d\tau} = t_{fi} f(\tau, x(\tau), u(\tau))$$



The two way points are marked once in the position plot with "*". To be noticed is that the output changes rapidly at these two points, which might cause instability!



The figures above are the 3D view and top-down view of the trajectory with given starting states and end states, passing through way points [2;0;1.5] and [7;6;0.2] with desired velocity directions

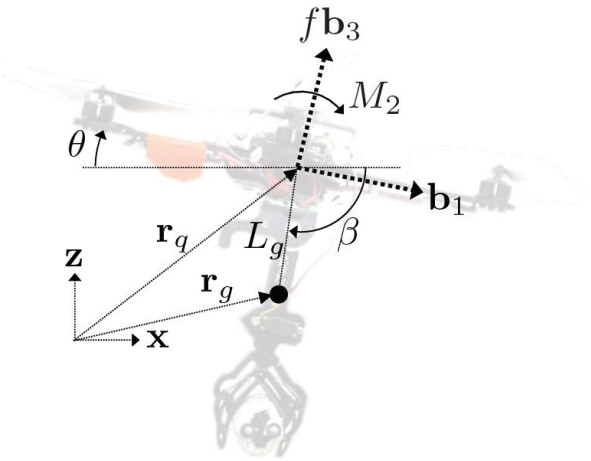
All above have also been implemented in python, which is compatible in ROS system.The program

is able to publish control set points including position, velocity, acceleration (attitude), and angular velocity. It can then be simulated at real time in GAZEBO.
https://github.com/waltYeh/rotors_simulator

Dynamic Programming for Grasping of a UAV

"uav_grasp_opt_var_mass.m" implements a dynamic optimization for UAV grasping with the open source library CASADI.

The model is built in a plane, in which the front and rear rotor of the UAV lie. The arm can rotate only in this plane. Another simplification is that the joint between arm and UAV is at the center of mass of this UAV. (Justin R. Thomas, Grasping, perching and visual servoing for micro aerial vehicles, 2017)



$$\ddot{\mathbf{q}} = D^{-1}\left(\mathbf{F} - C\dot{\mathbf{q}} - G\right)$$

$$D = \begin{bmatrix} m_g + m_q & 0 & 0 & -L_g m_g \sin(\beta) \\ 0 & m_g + m_q & 0 & -L_g m_g \cos(\beta) \\ 0 & 0 & \mathcal{I}_q & 0 \\ -L_g m_g \sin(\beta) & -L_g m_g \cos(\beta) & 0 & \mathcal{I}_g + L_g^2 m_g \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 0 & -L_g m_g \cos(\beta)\dot{\beta} \\ 0 & 0 & 0 & L_g m_g \sin(\beta)\dot{\beta} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$G = \begin{bmatrix} 0 \\ g(m_g + m_q) \\ 0 \\ -g L_g m_g \cos(\beta) \end{bmatrix}.$$

The input variables are the total thrust forces of four rotors T, the torque generated by the front and rear rotor M, and the torque of the servo at the joint τ.

$$u = [T, M, \tau]^T$$

The state variables are horizontal position of UAV, vertical position of UAV, pitch angle of UAV, angle of arm with respect to horizon, and also their derivatives.

$$x = [x_q \;\; z_q \;\; \theta \; \beta \; \dot{x}_q \; \dot{z}_q \dot{\theta} \; \dot{\beta}]^T$$

The objective is to minimize the weighted cost of consumed energy of inputs, and the angle between the UAV and arm should not be far away from 90 degrees. The dynamics of the system can be obtained from Lagrange mechanics. Bear in mind that the payload $m_P(t)$ is added only in the second interval. Other constraints include bounds of states and inputs, starting and final conditions. The constraints also take the state of grasping into consideration. When the UAV flies pass the object, the end point of arm should meet the object at the absolute velocity of zero. The objective functional is accumulated in the intervals before and after grasping.

$$J = \int_{t_0}^{t_1} u^T(t)Ru(t) + (\beta - \theta - \frac{\pi}{2})^T \mathrm{Q}(\beta - \theta - \frac{\pi}{2})\mathrm{dt} + \int_{t_1}^{t_2} u^T(t)Ru(t) + (\beta - \theta - \frac{\pi}{2})^T \mathrm{Q}(\beta - \theta - \frac{\pi}{2})\mathrm{dt}$$

s.t.

$$\frac{dx(t)}{dt} = f(t, x(t), u(t), m_P(t))$$

$$T(\cdot) \in [0, T_{max}]$$

$$M(\cdot) \in [-M_{max}, M_{max}]$$

$$\tau(\cdot) \in [-\tau_{max}, \tau_{max}]$$

$$\gamma = \beta - \theta \in [0, \pi]$$

$$\dot{x}_g(\cdot) \in [-v_{max}, v_{max}]$$

$$\dot{z}_g(\cdot) \in [-v_{max}, v_{max}]$$

$$\theta(\cdot) \in [-\theta_{max}, \theta_{max}]$$

$$x(t_0) = [x_{q0} \;\; z_{q0} \;\; 0 \;\; \frac{\pi}{2} \;\; 0 \; 0 \; 0 \; 0]^T$$

$$x(t_2) = [x_{q2} \;\; z_{q2} \;\; 0 \;\; \frac{\pi}{2} \;\; 0 \; 0 \; 0 \; 0]^T$$

$$x_g(t_1) = x_q(t_1) + L_{arm}\cos\big(\beta(t_1)\big) = x_{obj}$$

$$z_g(t_1) = z_q(t_1) - L_{arm}\sin\big(\beta(t_1)\big) = z_{obj}$$

$$\dot{x}_g(t_1) = \dot{x}_q(t_1) - \dot{\beta}(t_1)L_{arm}\sin\big(\beta(t_1)\big) = 0$$

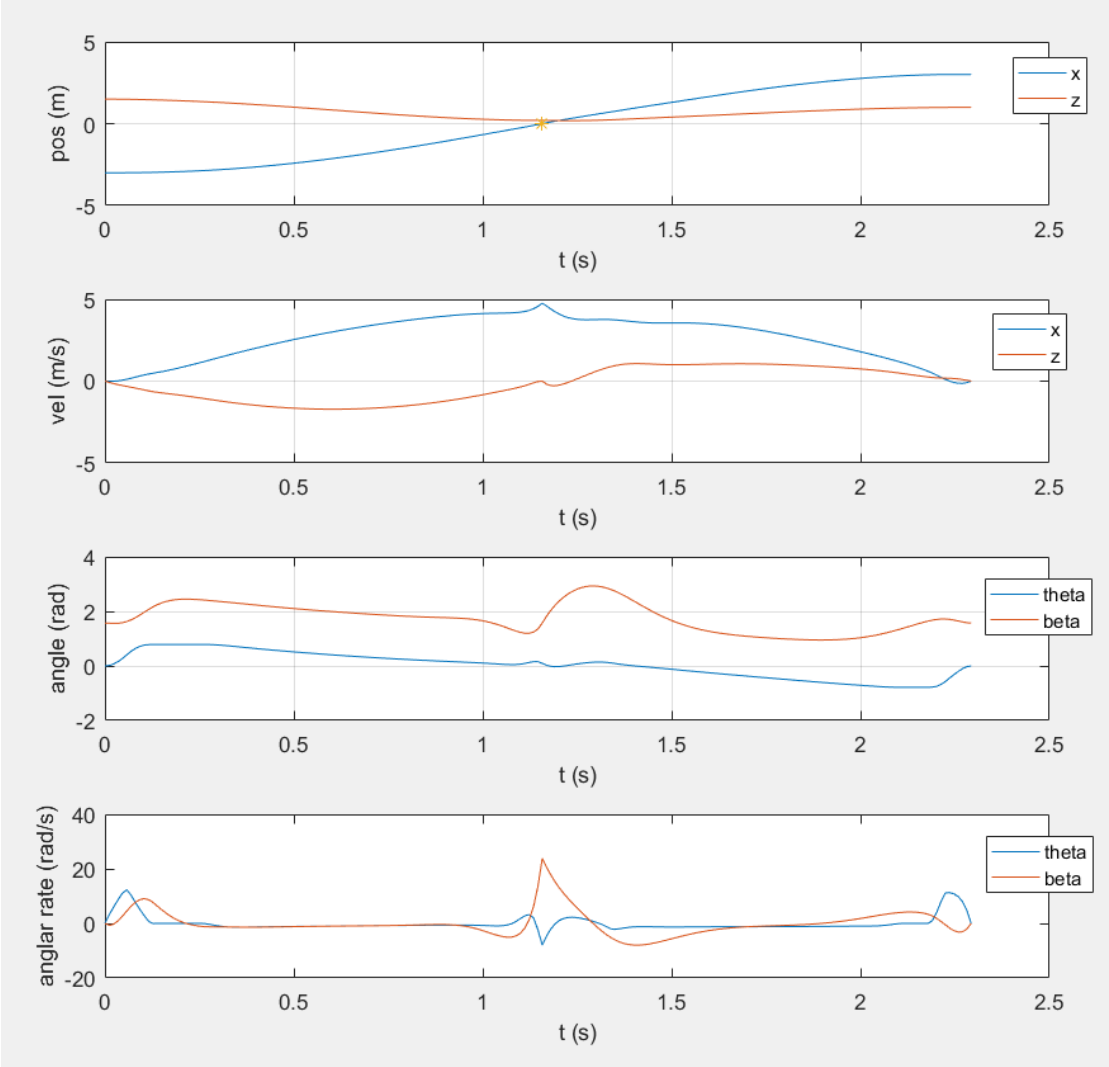
$$\dot{z}_g(t_1) = \dot{z}_q(t_1) - \dot{\beta}(t_1)L_{arm}\cos\big(\beta(t_1)\big) = 0$$

For this free end time problem, time transformation is applied for $i = 1,2$.

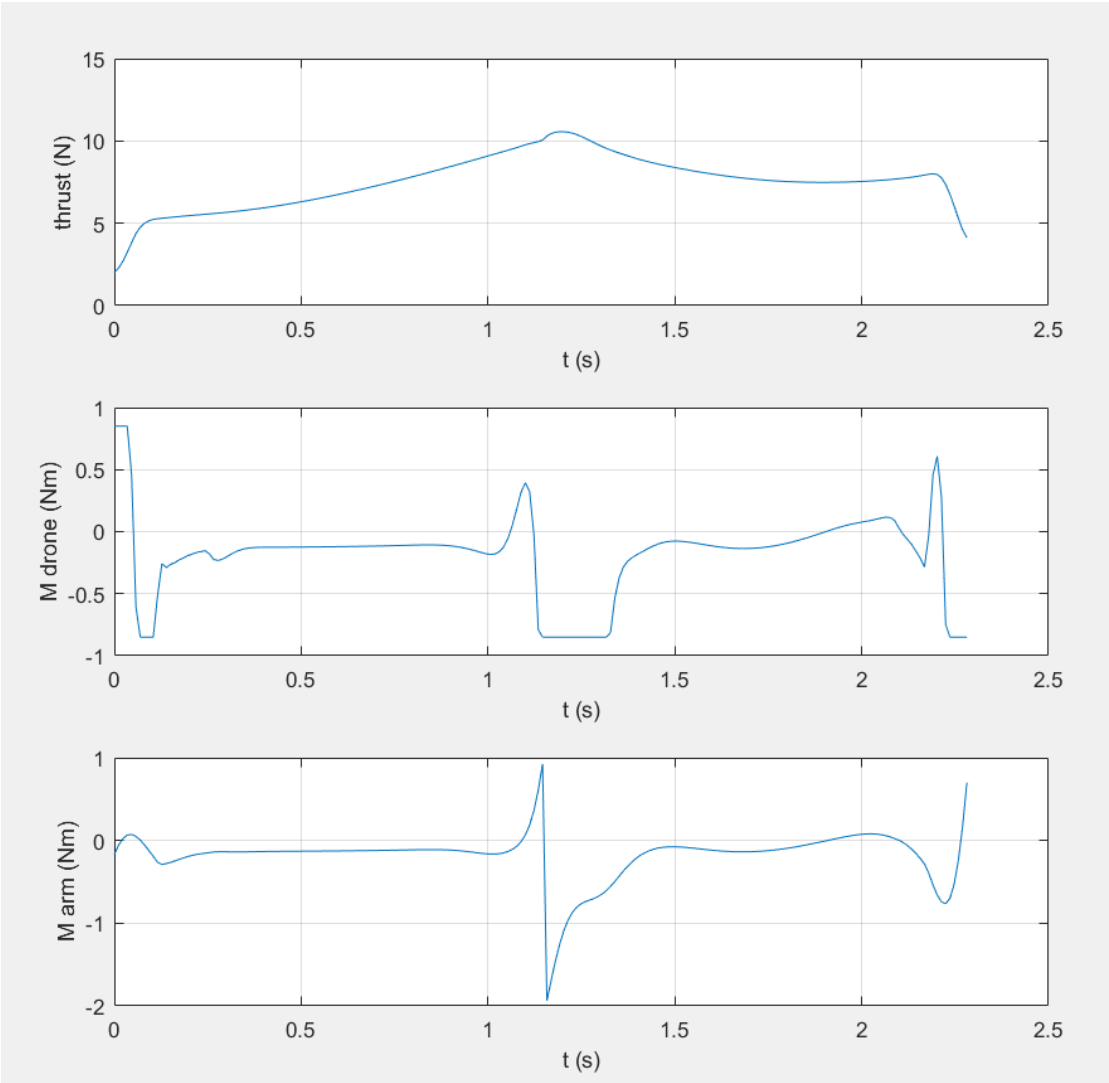
$$t_i = \tau t_{fi}$$

$$\tau \in [0, 1]$$

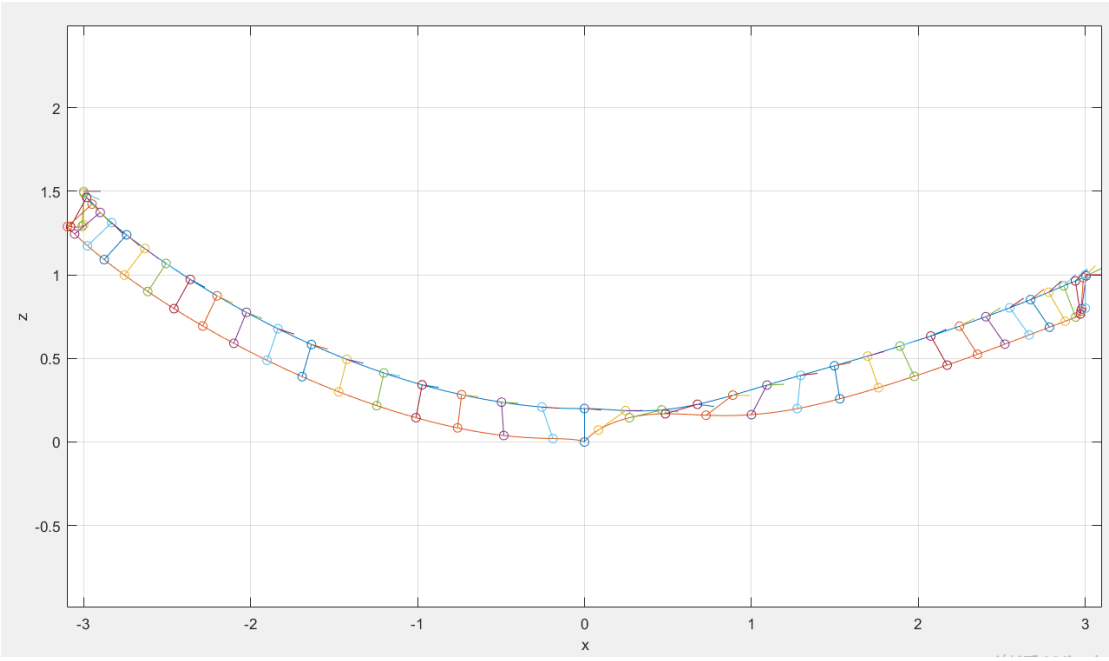
$$\frac{dx(\tau)}{d\tau} = t_{fi} f(\tau, x(\tau), u(\tau))$$



The states with respect to time. The grasping state is marked once in the position plot with “*”.



The plotting of system inputs.



The figure above is the trajectory of UAV and the trajectory of the end point of arm. The quivers show the incline angles of the UAV at some time points, and the line connecting two trajectories shows the poses of arm at some time points.

The next move can be implementing these in python, which is compatible in ROS system. Also important is to build a model in GAZEBO. It can then be simulated at real time.

Controller Synthesis for the Elimination of Disturbance

"disturbance_compensation.m" implements a controller, which on the one hand eliminates the disturbance of arm upon position of quad rotor, on the other hand eliminates the disturbance of drone movement upon the attitude of arm. The controller is designed under linearized model. It is compared to another controller with same design parameters in "riccati_controller_linear.m"

Take the working point as follows: the drone stays at a near hovering attitude, and the arm is vertically pointing downwards. When the angles and the angular rates are small, the plant is linearized.

$$\ddot{\mathbf{q}} = D^{-1}\left(\mathbf{F} - C\dot{\mathbf{q}} - G\right)$$

$$\begin{bmatrix}\dot{x}\\ \dot{z}\\ \dot{\theta}\\ \dot{\beta}\\ \ddot{x}\\ \ddot{z}\\ \ddot{\theta}\\ \ddot{\beta}\end{bmatrix} = \begin{bmatrix}0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1\\ 0&0&A_{53}&A_{54}&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&A_{83}&A_{84}&0&0&0&0\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta z\\ \theta\\ \Delta\beta\\ \dot{x}\\ \dot{z}\\ \dot{\theta}\\ \dot{\beta}\end{bmatrix} + \begin{bmatrix}0&0&0\\ 0&0&0\\ 0&0&0\\ 0&0&0\\ 0&0&B_{53}\\ 1/(m_g+m_q)&0&0\\ 0&1/I_q&-1/I_q\\ 0&0&B_{83}\end{bmatrix}\begin{bmatrix}\Delta f\\ M\\ \tau\end{bmatrix}$$

If the input variables is rewritten as $[\Delta f_1, \Delta f_3, \tau]^T$:

$$\begin{bmatrix}\dot{x}\\ \dot{z}\\ \dot{\theta}\\ \dot{\beta}\\ \ddot{x}\\ \ddot{z}\\ \ddot{\theta}\\ \ddot{\beta}\end{bmatrix} = \begin{bmatrix}0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1\\ 0&0&A_{53}&A_{54}&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&A_{83}&A_{84}&0&0&0&0\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta z\\ \theta\\ \Delta\beta\\ \dot{x}\\ \dot{z}\\ \dot{\theta}\\ \dot{\beta}\end{bmatrix} + \begin{bmatrix}0&0&0\\ 0&0&0\\ 0&0&0\\ 0&0&0\\ 0&0&B_{53}\\ 2/(m_g+m_q)&2/(m_g+m_q)&0\\ -1/I_q&1/I_q&-1/I_q\\ 0&0&B_{83}\end{bmatrix}\begin{bmatrix}\Delta f_1\\ \Delta f_3\\ \tau\end{bmatrix}$$

Now consider the arm as disturbance. Rewrite the differential equation without state variables $\Delta\beta$ and $\dot{\beta}$:

$$\begin{bmatrix}\dot{x}\\ \dot{z}\\ \dot{\theta}\\ \ddot{x}\\ \ddot{z}\\ \ddot{\theta}\end{bmatrix} = \begin{bmatrix}0&0&0&1&0&0\\ 0&0&0&0&1&0\\ 0&0&0&0&0&1\\ 0&0&A_{53}&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta z\\ \theta\\ \dot{x}\\ \dot{z}\\ \dot{\theta}\end{bmatrix} + \begin{bmatrix}0&0\\ 0&0\\ 0&0\\ 0&0\\ 2/(m_g+m_q)&2/(m_g+m_q)\\ -1/I_q&1/I_q\end{bmatrix}\begin{bmatrix}\Delta f_1\\ \Delta f_3\end{bmatrix}$$

$$+\begin{bmatrix}0&0\\ 0&0\\ 0&0\\ A_{54}&B_{53}\\ 0&0\\ 0&-1/I_q\end{bmatrix}\begin{bmatrix}\Delta\beta\\ \tau\end{bmatrix}$$

$$\dot{x}_d = A_d x_d + B_d u_d + E_d z_d$$

Now consider the drone as disturbance:

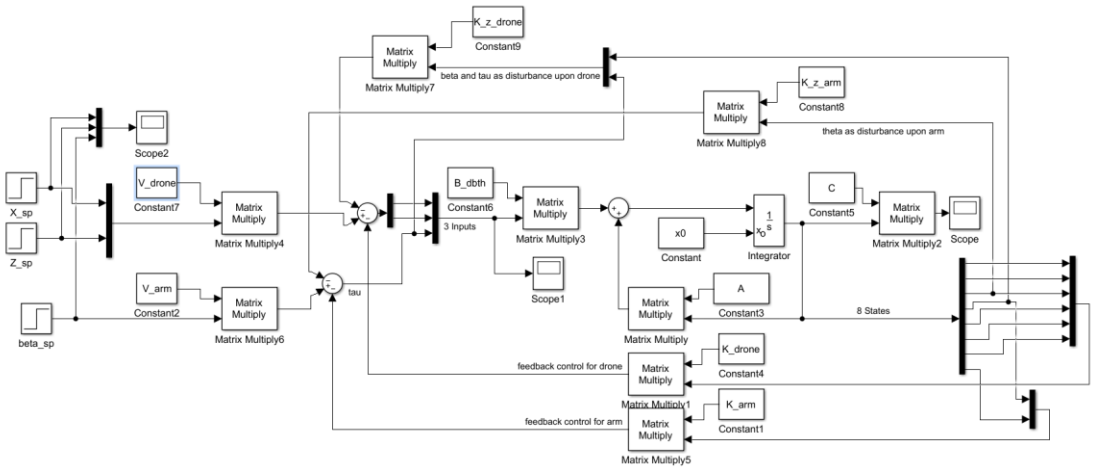
$$\begin{bmatrix}\dot{\beta}\\ \ddot{\beta}\end{bmatrix}=\begin{bmatrix}0 & 1\\ A_{84} & 0\end{bmatrix}\begin{bmatrix}\Delta\beta\\ \dot{\beta}\end{bmatrix}+\begin{bmatrix}0\\ B_{83}\end{bmatrix}\tau+\begin{bmatrix}0\\ A_{83}\end{bmatrix}\theta$$

$$\dot{x}_a = A_a x_a + B_a \tau + E_a \theta$$

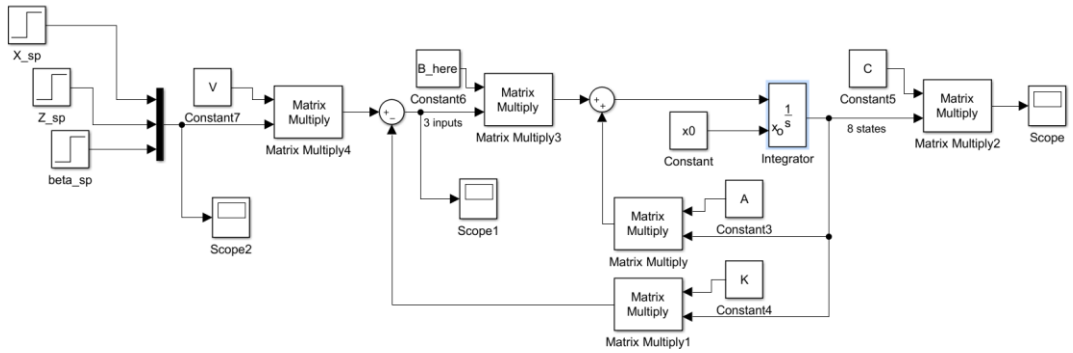
Assume that the disturbance can be measured, especially we need a servo with current feedback. Then the compensated input for drone and for arm is respectively:

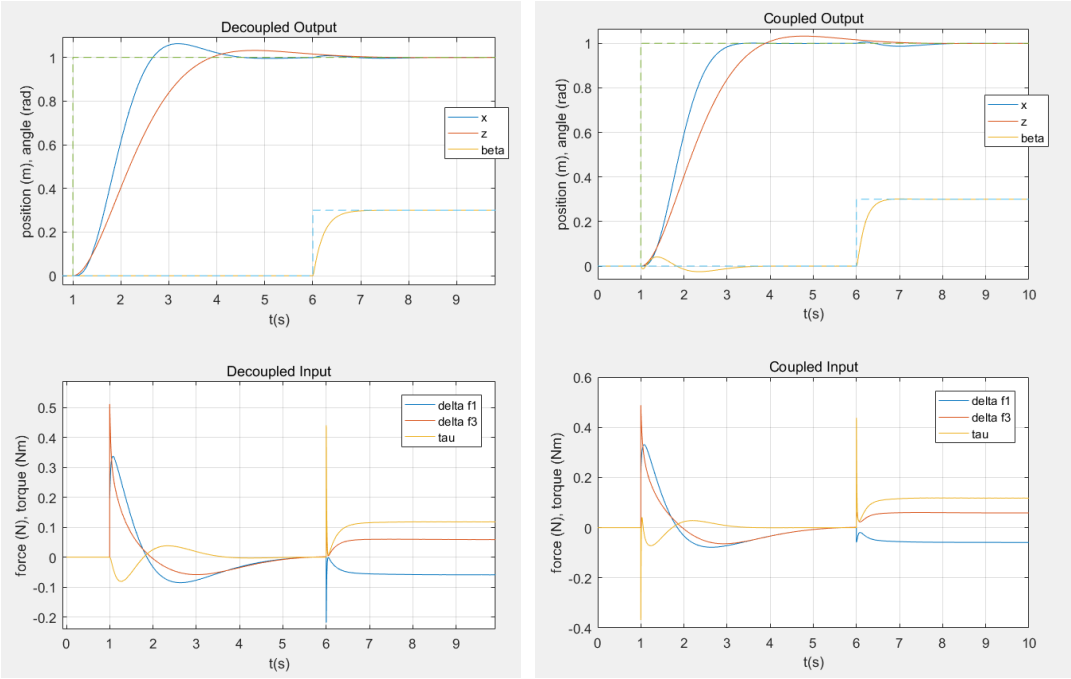
$$u_d = u_{dR} + u_{dz}$$
$$u_a = u_{aR} + u_{az}$$
$$u_{dz} = -(B_d^T B_d)^{-1} B_d^T E_d z_d$$
$$u_{az} = -(B_a^T B_a)^{-1} B_a^T E_a \theta$$

u_{dR} and u_{aR} are designed separately by Riccati equations. A pre-filter (Vorfilter) is then designed to ensure stationery accuracy.



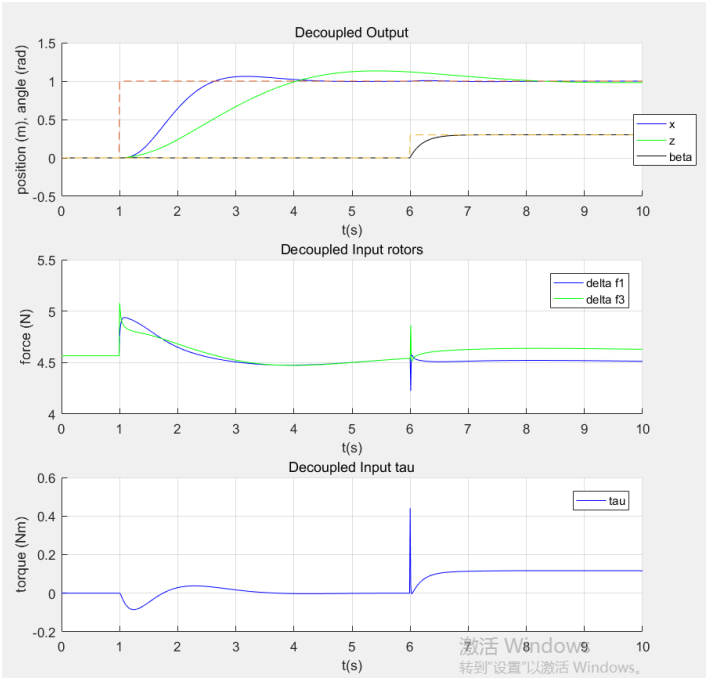
In comparison, another controller considering the system as a whole without elimination of disturbance is designed by Riccati equations. The cost coefficients for Riccati equations Q and R are identical for the comparison.





Figures show the step responses (at 1 sec. a step signal for x and z, and at 6 sec. a step for the angle of arm beta) of disturbance eliminating controller (left) and an ordinary controller (right) with the same design parameters. However, the disturbance eliminating controller takes longer time to settle x.

The controller is designed under linearized model. When it is directly applied to the original non-linear model, the effectiveness of disturbance elimination is not weakened too much, when the angle of arm is not too far away from vertically downwards (smaller than 0.6 rad). However, the reaction of output z is much slower. This is due to the thrust reduction in z direction when the drone is tilt for horizontal movement.



Difficulties applying Falb-Wolovich Decoupling

The problem of disturbance elimination introduced above is mainly due to the torque of the servo, which in many cases is not measurable. Therefore it is natural to use a decoupling controller.

According to the formulary of the course RLM:

Entkopplung nach Falb-Wolovich:

- Ausgangsgröße y_i hat Differenzordnung $\delta_i \Leftrightarrow \underline{c}_i^T \underline{A}^k \underline{B} \begin{cases} = \underline{0}^T \text{ für } 0 \le k \le \delta_i - 2 \\ \neq \underline{0}^T \text{ für } k = \delta_i - 1 \end{cases}$
- Entkopplungsrückführung: $\underline{R} = \underline{D}^{*-1} \underline{F}$ Entkopplungsvorfilter: $\underline{M} = \underline{D}^{*-1} \underline{K}$

  mit $\underline{D}^* = \begin{bmatrix} \underline{c}_1^T \underline{A}^{\delta_1 - 1} \underline{B} \\ \vdots \\ \underline{c}_p^T \underline{A}^{\delta_p - 1} \underline{B} \end{bmatrix}$, $\underline{F} = \begin{bmatrix} \underline{c}_1^T \underline{A}^{\delta_1} + \sum_{\nu=0}^{\delta_1 - 1} q_{1\nu} \underline{c}_1^T \underline{A}^{\nu} \\ \vdots \\ \underline{c}_p^T \underline{A}^{\delta_p} + \sum_{\nu=0}^{\delta_p - 1} q_{p\nu} \underline{c}_p^T \underline{A}^{\nu} \end{bmatrix}$, $\underline{K} = \begin{bmatrix} k_1 & & \underline{0} \\ & \ddots & \\ \underline{0} & & k_p \end{bmatrix}$
- Entkoppelte Teilsysteme:

$$Y_i(s) = \frac{k_i}{s^{\delta_i} + q_{i(\delta_i - 1)} s^{\delta_i - 1} + \cdots + q_{i1} s + q_{i0}} W_i(s) \qquad (i = 1, \ldots, p)$$

The first step of Falb Wolovich Decoupling is to determine the differential order of each output x, z, and beta. According to common sense and without considering the coupling, x has a differential order of 4 w.r.t. the input torque generated by rotors (torque (angular acceleration) – angular rate – angle (translational acceleration) – velocity - position), z has an order of 2 w.r.t. the rotor thrust, and beta has an order of 2 w.r.t. the torque of the servo. However, the differential order of x in coupled model is 2 due to the direct influence of servo torque upon the translational acceleration of the drone, which is described by the element B_{53} in the linearized model. In this case, the matrix D^* is singular. Falb-Wolovich decoupling cannot be applied.

If we nevertheless set the differential order as 4, the decoupling cannot be achieved, because the designed controller neglects the influence of B_{53}, which actually plays an important role in the dynamic.