Project in Adaptive Control and Real Time Systems

March 2, 2016

Todo: Much remains to be defined, but in general, I have used the same definitions as Lukkonen [1]. Let

$$\boldsymbol{\xi} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \boldsymbol{\eta} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}, \quad \boldsymbol{\omega} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} \tag{1}$$

Where $\boldsymbol{\xi}$ denotes the position of the center of mass in the global frame, $\boldsymbol{\eta}$ is the euler anfles in the body frame and ω_i is the angular speed of the rotor i. The non-linear dynamics of the quadcopter are then goverened by

$$\ddot{\boldsymbol{\xi}} = \mathbf{G} + \frac{1}{m}\mathbf{R}\mathbf{T} - \frac{1}{m}\mathbf{D}\dot{\boldsymbol{\xi}} \tag{2}$$

$$\mathbf{R} = \begin{bmatrix} \cos(\psi)\cos(\phi) & \cos(\psi)\sin(\theta)\sin(\phi) - \sin(\psi)\cos(\phi) & \cos(\psi)\sin(\theta)\cos(\phi) + \sin(\psi)\sin(\phi) \\ \sin(\psi)\cos(\phi) & \sin(\psi)\sin(\theta)\sin(\phi) + \cos(\psi)\cos(\phi) & \sin(\psi)\sin(\theta)\cos(\phi) - \cos(\psi)\sin(\phi) \\ -\sin(\theta) & \cos(\theta)\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix} \tag{3}$$

$$\ddot{\boldsymbol{\eta}} = \mathbf{J}^{-1}(\boldsymbol{\tau}_B - \mathbf{C}\dot{\boldsymbol{\eta}}) \tag{4}$$

By defining the states and control signals as

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\xi} \\ \dot{\boldsymbol{\xi}} \\ \boldsymbol{\eta} \\ \dot{\boldsymbol{\eta}} \end{bmatrix} \in \mathbb{R}^{12\times 1}, \quad \text{and} \quad \mathbf{u} = \begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \in \mathbb{R}^{4\times 1} \tag{5}$$

respectively, the non-linear system can be written

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbb{I}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & -\frac{1}{m}\mathbf{D} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbb{I}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & -\mathbf{J}^{-1}\mathbf{C} \end{bmatrix}\mathbf{x} + \begin{bmatrix} \mathbf{0}_{1\times 3} & \mathbf{0}_{3\times 3} \\ \frac{1}{m}\mathbf{R}\hat{\mathbf{z}} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{1\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{1\times 3} & \mathbf{J}^{-1} \end{bmatrix}\mathbf{u} + \begin{bmatrix} \mathbf{0}_{1\times 3} \\ \mathbf{G} \\ \mathbf{0}_{1\times 3} \\ \mathbf{0}_{1\times 3} \end{bmatrix} = \mathbf{A}^c\mathbf{x} + \mathbf{B}^c\mathbf{u} + \mathbf{G}^c \tag{6}$$

$$\mathbf{y} = \begin{bmatrix} z & \phi & \theta & \psi \end{bmatrix} = \mathbf{C}^c\mathbf{x} \tag{7}$$

We note that the only non-linear block in the system matrix is the $(-\mathbf{J}^{-1}\mathbf{C})$-block which is only dependent on $\boldsymbol{\eta}$ and $\dot{\boldsymbol{\eta}}$, which means that we only need to linearise this block with regards to the angular states. Furthermore, the $\mathbf{G}^c$-matrix vanishes in the linearised system, which can be written

0.1 Full model

(see `quadcopter_model.m`) (see `quadcopter_init.m`)

TODO

1. Define a complete nonlinear model in Simulink, which is currently buggy when in terms of z and ψ. I fixed two sign errors, and there is bound to be a couple more... [1].
2. Derive linearised state-space model.

1 MPC control

The dynamics of the quadcopter defined in [2] defines the pitch as θ_1 and yaw as θ_2. The quadcopter is then linearised round $(\theta_1, \theta_2) = \mathbf{0}$ with the control signals $u_1 = \Delta\theta_1$, $u_2 = \Delta\theta_2$ and u_3 as the commanded thrust (see `quadcopter_MPC_init.m`). Discretisation is here done through ZOH but other methods could be used as well. The model was replicated in Simulink, and the system responses look reasonable (see `quadcopter_MPC_simulate.m`)

TODO

1. ~~Create simplified linearised system ss-model for use in MPC~~ (see eg. [2]).
2. ~~Validate by comparison to the results in~~ [2].
3. Set up QP-MPC controller with Simulink MPC-library (see eg. [2]).
4. Validate by comparison to the results in [2].
5. Set up QP-MPC controller with CVXGEN m-code (see eg. [2] [3]).
6. Validate by comparison to the results in Simulink.
7. System identification.
8. Simulate system with proper parameters.

2 $\mathcal{L}_1$-control

The $\mathbf{\Gamma}$-projection operator for two vectors $\theta, y \in \mathbb{R}^k$ is defined as

$$\mathrm{Proj}_{\mathbf{\Gamma}}(\theta, y, f) = \begin{cases} \mathbf{\Gamma} y - \mathbf{\Gamma}\dfrac{\nabla f(\theta)(\nabla f(\theta))^T}{||\nabla f(\theta)||_2}\mathbf{\Gamma} y f(\theta) & \text{if } f(\theta) > 0 \text{ and } y^T \nabla f(\theta) > 0 \\ \mathbf{\Gamma} y & \text{otherwise.} \end{cases} \tag{8}$$

where $\mathbf{\Gamma} = \mathbb{I}_{k\times k}\Gamma$ for some scalar $\Gamma > 0$ (typically $\Gamma \approx 10^5$) and $f(\theta)$ is a convex function [4]. By solving the Lyapunov equation

$$\mathbf{A}_m\mathbf{X} + \mathbf{X}\mathbf{A}_m^T + \mathbf{Q} = 0, \tag{9}$$

for $\mathbf{P} = \mathbf{P}^T$, with some arbitrary $\mathbf{Q} > 0$, the feedback controller

$$\begin{cases} u(t) = \hat{\theta}^T x(t) + k_g r(t) \\ \dot{\hat{\theta}}(t) = \text{Proj}_{\mathbf{\Gamma}}(\hat{\theta}^T(t), x(t)\tilde{x}^T(t)\mathbf{X}b) \end{cases} \tag{10}$$

can be constructed, where $\tilde{x} = \hat{x} - x$ is the state estimation error, k_g is a gain and $r(t)$ is the reference signal. By designing the companion system

$$\begin{cases} \dot{x}(t) = \mathbf{A}_m \hat{x}(t) + b(u(t) - \hat{\theta}^T(t)x(t)) \\ y(t) = c^T \hat{x}(t) \end{cases} \tag{11}$$

it can be shown (by Theorem 2 [5]) that the state estimation error,

$$\lim_{t\to\infty} \tilde{x} = 0. \tag{12}$$

By a corollary of the theorem, choosing

$$k_g = -\frac{1}{c^T \mathbf{A}_m^{-1} b} \Rightarrow \lim_{t\to\infty} y(t) = r \tag{13}$$

if $r \equiv$ constant.

TODO

1. ~~Define general control structure.~~
2. Create Simulink projection operator (see eg. [6])
3. Validate projection operator against benchmark Simulink models (eg. [5]).
4. Define robustness metrics (see eg. [6] [7])
5. Create script for computing the $\mathcal{L}_1$-gain (see eg. [6]).
6. Validate script against benchmark Simulink models (eg. [5]).
7. Simulate control.

3 General TODOs

1. Found that one of the copters were broken - sent to Bitcraze for repairs, expected to be done in early march.
2. Tried installing IRIS in Ubuntu but ran into issues using the PODS "make" command required to get everything up and running. TODO: contact Claes or Anders to get help in finding someone experienced with PODS.

References

[1] T. Luukkonen, “Modelling and control of quadcopter,” *Independent research project in applied mathematics, Espoo*, 2011.

[2] P. Bouffard, “On-board model predictive control of a quadrotor helicopter: Design, implementation, and experiments,” Master’s thesis, EECS Department, University of California, Berkeley, Dec 2012. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-241.html

[3] J. Mattingley and S. Boyd, “Cvxgen: A code generator for embedded convex optimization,” *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.

[4] E. Lavretsky, T. E. Gibson, and A. M. Annaswamy, “Projection operator in adaptive systems,” 2011.

[5] C. Cao and N. Hovakimyan, “Design and analysis of a novel l1 adaptive controller, part i: Control signal and asymptotic stability,” in *American Control Conference, 2006*. IEEE, 2006, pp. 3397–3402.

[6] N. Hovakimyan, “L1 tutorial.” [Online]. Available: http://naira-hovakimyan.mechse.illinois.edu/l1-adaptive-control-tutorials/

[7] M. Q. Huynh, W. Zhao, and L. Xie, “L 1 adaptive control for quadcopter: Design and implementation,” in *Control Automation Robotics & Vision (ICARCV), 2014 13th International Conference on*. IEEE, 2014, pp. 1496–1501.