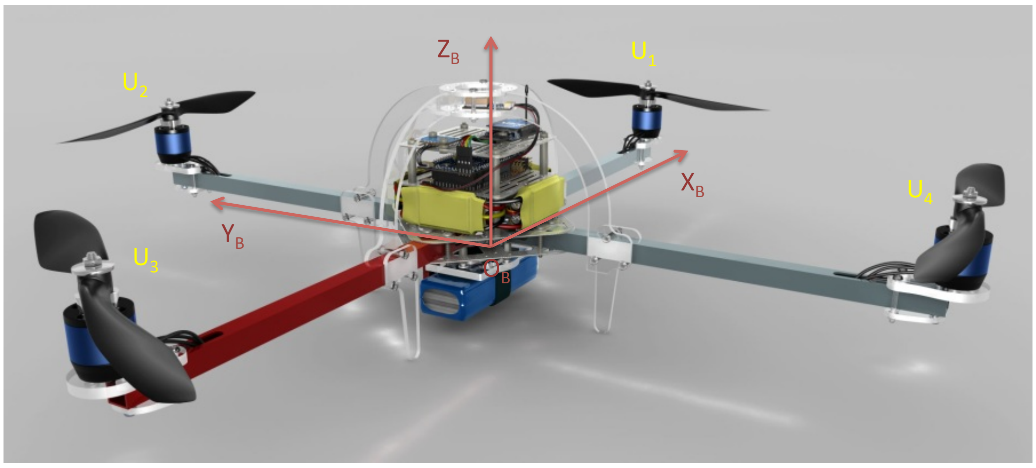
Model Predictive Control
Prof. Colin Jones

ME-425

Mini-Project
Quadrotor control

In this final project you will implement an MPC controller for a quadrotor, which is an aircraft driven by four independently controlled rotors; have a look at the figure to get an idea of what it looks like, or even better watch one of the videos from the GRASP lab[1] to get a feel for what these toys can do.



What is expected

At the end of the project you should hand in

- working code
- a small report with requested plots and answers to questions posed along the way

Files

Please go to moodle, download and unpack the files `qpc.zip`, `yalmip.zip` and `quadrotor.zip`. Add the unpacked folders `qpc` and `yalmip` to the Matlab path and then run the function `main.m` from the folder `quadrotor`; if it finishes successfully, you are ready to start.

Getting acquainted - nonlinear model and linearization

In order to derive a nonlinear model consider two reference frames. The first one is the body frame (subscript B) with the origin O_B attached to the center of mass of the quadrotor as drawn in the picture. The second one is the world frame which remains fixed. We are going to derive a 12-state description of the robot with the state vector

$$\mathbf{x} = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \alpha & \beta & \gamma & \dot{\alpha} & \dot{\beta} & \dot{\gamma} \end{bmatrix},$$

where $(x, y, z) = O_B$ is the position of the center of mass and (α, β, γ) are the roll, pitch and yaw angles. These angles represent, in order, the rotation around the x_B axis followed by the rotation around the y_B axis followed by rotation around the z_B axis.

The input of the model

$$\mathbf{u} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}$$

[1] e.g., `http://www.youtube.com/watch?v=E7X0_6o9J10`

is the thrusts of the four rotors. These thrusts are proportional to the square of the rotor angular velocity and hence are nonnegative. Consequently, each rotor produces a force F_i and moment M_i according to

$$F_i = k_F u_i, \quad M_i = k_M u_i.$$

The net body force F and body moments $(M_\alpha, M_\beta, M_\gamma)$ can be expressed in terms of $\mathbf{u}$ as

$$\begin{bmatrix} F \\ M_\alpha \\ M_\beta \\ M_\gamma \end{bmatrix} = \begin{bmatrix} k_F & k_F & k_F & k_F \\ 0 & k_F L & 0 & -k_F L \\ -k_F L & 0 & k_F L & 0 \\ k_M & -k_M & k_M & -k_M \end{bmatrix} \mathbf{u}. \tag{1}$$

where L is the distance from the center of mass to the center of the rotor.

The acceleration of the center of mass is then given by

$$\ddot{O}_B = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + F\mathbf{z}_B. \tag{2}$$

where g is the acceleration due to gravity and $\mathbf{z}_B$ is the unit vector in the z direction of the body coordinate frame expressed in the world coordinates.

The angular dynamics are given by

$$\dot{\omega} = \mathcal{I}^{-1} \left(-\omega \times \mathcal{I}\omega + \begin{bmatrix} M_\alpha \\ M_\beta \\ M_\gamma \end{bmatrix} \right), \tag{3}$$

where $\omega = \dot{\alpha}\mathbf{x}_B + \dot{\beta}\mathbf{y}_B + \dot{\gamma}\mathbf{z}_B$ is the angular velocity of body coordinate frame with respect to the world coordinate frame and $\mathcal{I}$ is the inertia matrix of the quadrotor.

Now take a look at Equation 1, which is the key to the understanding of how a quadrotor works. From the first line we can see that the net body force F (in the direction $\mathbf{z}_\mathrm{B}$) is simply the sum of the four rotor thrusts (times a constant k_F). From the second line we see that the moment around the $\mathbf{x}_\mathrm{B}$ axis (which then gives rise to a change in the roll angle α) is given by the imbalance in the thrusts of the rotors 2 and 4. Similarly the pitch angle β is controlled by the imbalance in the thrusts of the rotors 1 and 3. Finally, from the fourth line we can see that the yaw angle γ is controlled by the imbalance in the net thrusts of the opposite rotor pairs (1,3) and (2,4). This is because the two pairs rotate in the opposite direction (and therefore generate no net moment around $\mathbf{z}_B$ when the net thrusts of the two pairs are equal; otherwise there is a moment imbalance and the quad rotates).

From Equation 2 we see that in order to fly the quadrotor around we need to control the body direction $\mathbf{z}_B$ (which is a function of α and β) and simultaneously apply the net body force F in that direction, all of this while counteracting gravity.

Finally, Equation 3 says that the body angles (α, β, γ) are controlled by the moments $(M_\alpha, M_\beta, M_\gamma)$, which are directly related to the rotor thrusts through Equation 1.

Now its time that you have a look at the linearized model that we have prepared for you. The state-space description is given by the matrices A_c and B_c which has been loaded by running the function `main.m`. The model has been linearized in a hovering equilibrium $(\mathbf{x}_s, \mathbf{u}_s)$. This is an equilibrium given by zero angles (α, β, γ), some (x, y, z) position, and a steady state control input $\mathbf{u}_s$ that exactly counteracts the gravity.

Consider the structure of the matrices A_c and B_c. Does it make intuitive sense in relation to the physical model?

Deliverables *5 %*

1. interpretation of the structure of matrices A_c and B_c; explain in particular the nonzero numbers in rows 4 and 5 of A_c and the nonzero rows of B_c in connection to the nonlinear dynamics described above.

Control Structure

In this exercise we will use a cascade controller, where the inner loop controller controls

$$\mathbf{x}_1 = \begin{bmatrix} \dot{z} & \alpha & \beta & \gamma & \dot{\alpha} & \dot{\beta} & \dot{\gamma} \end{bmatrix},$$

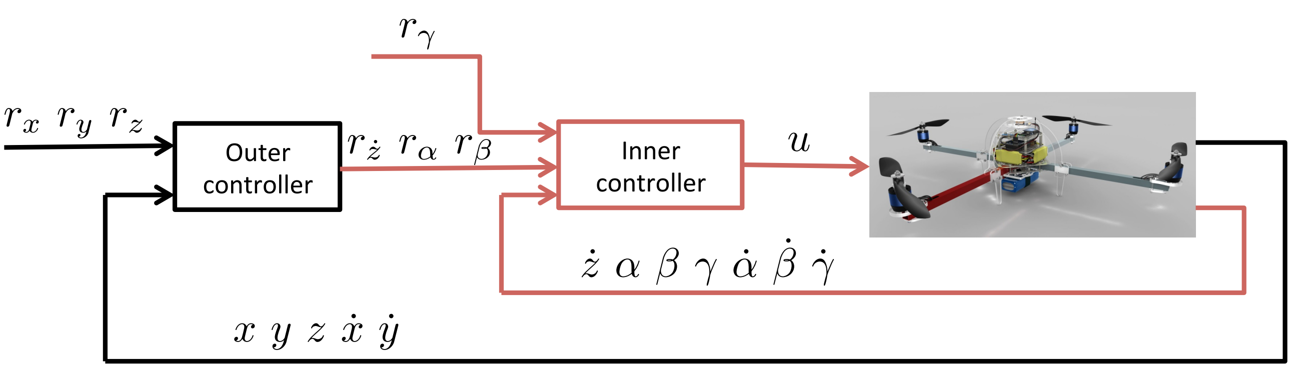
and the outer loop controller is in charge of

$$\mathbf{x}_2 = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \end{bmatrix}.$$

The inner-loop reference signal is

$$\mathbf{r}_1 = (r_{\dot{z}}, r_\alpha, r_\beta, r_\gamma),$$

where the first three components $(r_{\dot{z}}, r_\alpha, r_\beta)$ are governed by the outer loop controller, whereas r_γ (the yaw angle) comes from outside. The outer-loop reference signal $\mathbf{r}_2$ is the space position (r_x, r_y, r_z). You can have a look at the control structure in the figure.



r_γ
$r_x\ r_y\ r_z$
Outer controller
$r_{\dot{z}}\ r_\alpha\ r_\beta$
Inner controller
u
$\dot{z}\ \alpha\ \beta\ \gamma\ \dot{\alpha}\ \dot{\beta}\ \dot{\gamma}$
$x\ y\ z\ \dot{x}\ \dot{y}$

You don't need to worry about the outer-loop controller and the interconnection of the two controllers – we have designed the outer-loop controller for you. Your task is therefore to design the inner loop controller shown in red.

From us you have been given an inner-loop model discretized with sampling period $T_s = 0.1\,\text{s}$. In sum, all you need to focus from now on is the inner loop model with the fully observed state $\mathbf{x}_1$, the input $\mathbf{u}$ and the reference signal $\mathbf{r}_1$.

The discrete time model matrices A, B and the sampling period T_s are contained in the structure `sys` that has been loaded by running `main.m`.

Constraint specification

The inner-loop quadrotor model is subject to constraints on the maximum angle, maximum angle velocity as well as maximum velocity in the z direction – these constraints mainly ensure a validity of the linearized model and have been specified for you in the following list

- $|\dot{z}| \leq 1$ m/s,
- $|\alpha| \leq 10°$, $\quad |\beta| \leq 10°$,
- $|\dot{\alpha}| \leq 15°/\text{s}$, $\quad |\dot{\beta}| \leq 15°/\text{s}$,
- $|\dot{\gamma}| \leq 60°/\text{s}$.

The nonlinear model is also subject to the input constraints

$$0 \leq \mathbf{u} \leq 1.$$

You are advised to exploit Yalmip's modeling capabilities when specifying the constraints. For instance, the constraint on α and β along the prediction horizon N (excluding the initial state) can be written as

```
x1 = sdpvar(7, N);
constraints = [];
for i = 2:N
  constraints = constraints + set(-angleMax <= x1(2:3,i) <= angleMax);
end
```

where `angleMax` is a vector containing the constraints on α and β.

Warning: You will need to convert the degrees to radians and also bear in mind that the input constraint is on the *nonlinear* model whereas you are specifying constraints for the model linearized in the hovering equilibrium $(\mathbf{x}_s, \mathbf{u}_s)$ for which $\mathbf{u}_s \neq 0$.

Note that there are also constraints on the outer loop variables, that is, on the spatial coordinates and velocities of the quadrotor. You don't need to take them into account for the design of the inner loop controller, but they will constrain the movement of the quad during simulations on the full nonlinear model. These constraints are

- $|x| \leq 1$ m,
- $|y| \leq 1$ m,
- $-1\,\mathrm{m} \leq z \leq 4$ m.

First MPC controller

Now you are ready to design and test an inner loop MPC controller. You don't need to use invariant sets if you don't want to – this is a practical exercise. Therefore design any MPC controller that will regulate the state from a nonzero initial condition, say

$$\mathbf{x}_0 = \begin{bmatrix} -1 & 10 & -10 & 120 & 0 & 0 & 0 \end{bmatrix},$$

to the origin. (*warning*: you will need to translate the three angles to radians). The response you are aiming for is settling time of about 2 s for roll and pitch angles α and β and similarly for the vertical velocity $\dot{z}$ – these are the signals controlled by the outer controller and hence have to be fast. The yaw angle γ can be slower. Therefore the standard initial guess $Q = I$ and $R = I$ may not work for you in this case. You will also need to choose the prediction horizon length; the requirement of 2 s settling time may be a good starting point.

To simulate the system you can use the function `simQuad.m`. It takes the Yalmip optimizer instance as its second argument. Take a look at the documentation to the function. The syntax you end up using may look like this

```
options = sdpsettings('solver','qpip');
innerController = optimizer(constraint, objective, options, x(:,1), u(:,1));
simQuad( sys, innerController, x0, T);
```

where `x(:,1)` and `u(:,1)` are sdpvar instances representing the initial state x_0 and the first control input u_0, and T is the simulation length in seconds. The remaining arguments of `simQuad.m` will be needed later on for reference tracking and disturbance rejection.

Deliverables

1. choice of tuning parameters and motivation for them *5 %*
2. plots of the response to an appropriate initial condition *15 %*

Reference tracking

Now you will add reference tracking. The reference signals to be tracked are $(\dot{z}, \alpha, \beta, \gamma)$ commands from the outer loop controller. For design purposes you can regard them as any constant signals.

First you should find the set of equilibrium states and inputs $(\mathbf{x}_r, \mathbf{u}_r)$ corresponding to the reference signal

$$\mathbf{r}_1 = \begin{bmatrix}\dot{z} & \alpha & \beta & \gamma\end{bmatrix}^T.$$

You can do so by solving the system of linear equations

$$\mathbf{x}_r = A\mathbf{x}_r + B\mathbf{u}_r, \quad \mathbf{r}_1 = C\mathbf{x}_r, \tag{4}$$

where $C = [I_4 \;\; 0_{4\times 3}]$. The solution for $\mathbf{u}_\mathrm{r}$ may be surprising.

After computing the steady state solution, you can implement any reference tracking you know from the lectures. When you are done, you should check that you can also track (perhaps with some error) slowly varying reference signals.

The parameter vector for your controller optimizer instance should be $[\mathbf{x}_1{}^T \; \mathbf{r}_1{}^T]^T$; then you can use the same function for simulations as before.

Deliverables

1. Interpretation of the solution $(\mathbf{x}_\mathrm{r}, \mathbf{u}_\mathrm{r})$ to system (4) for arbitrary $\mathbf{r}_1$ *5 %*
2. plots of the response to a constant reference signal *10 %*
3. plots of the response to a slowly varying reference signal *5 %*

First simulation of the nonlinear model

Now you will try out your controller on the nonlinear model. Open the file `simulation1.mdl`; if your optimizer instance is named `innerController` and the order of variables is $[\mathbf{x}_1{}^T \; \mathbf{r}_1{}^T]^T$, then you should be able to run the simulation without any changes, just by pressing CTRL+T.

Run the simulation. You can use either the prespecified reference signal or create your own, but keep in mind that there are also constraints on the spatial position and velocity of the quadrotor as described above.

Deliverables

1. plots of a reference tracking response of the nonlinear model. *10 %*

Offset free MPC

The nonlinear model is subject various disturbances. First of all there is the wind – random disturbance with a nonzero mean value. There is also multiplicative and additive input disturbance and a gravity error (disturbance affecting the $\dot{z}$ component). All these disturbances can be packed into a single disturbance (random) variable $\mathbf{d} \in \mathbb{R}^7$ with mean $\bar{\mathbf{d}}$ and zero-mean component $\tilde{\mathbf{d}}$. The discrete-dynamics can then be written as

$$\mathbf{x}_{\mathrm{k}+1} = A\mathbf{x}_\mathrm{k} + B\mathbf{u}_\mathrm{k} + \bar{\mathbf{d}} + \tilde{\mathbf{d}}_\mathrm{k}.$$

Your goal is to estimate the time-invariant mean $\bar{\mathbf{d}}$ while rejecting the unpredictable $\tilde{\mathbf{d}}_\mathrm{k}$. Therefore, design a disturbance estimator (not really a Kalman filter since there is no measurement noise) to estimate $\bar{\mathbf{d}}$. You can model the disturbance as the dynamical system

$$\bar{\mathbf{d}}_{k+1} = \bar{\mathbf{d}}_\mathrm{k},$$

and design a full-state estimator for the augmented system

$$\begin{bmatrix}\mathbf{x}_1^+ \\ \bar{\mathbf{d}}^+\end{bmatrix} = \underbrace{\begin{bmatrix}A_1 & I \\ 0 & I\end{bmatrix}}_{\hat{A}} \begin{bmatrix}\mathbf{x}_1 \\ \bar{\mathbf{d}}\end{bmatrix} + \underbrace{\begin{bmatrix}B \\ 0\end{bmatrix}}_{\hat{B}} \mathbf{u}, \quad y = \underbrace{\begin{bmatrix}I_{7\times 7} & 0_{7\times 7}\end{bmatrix}}_{\hat{C}} \underbrace{\begin{bmatrix}\mathbf{x}_1 \\ \bar{\mathbf{d}}\end{bmatrix}}_{\hat{x}}.$$

The estimator should be of the form

$$\mathbf{x}_f^+ = A_f\mathbf{x}_f + B_f\begin{bmatrix}\mathbf{u}_1 \\ \mathbf{x}_1\end{bmatrix},$$

where x_f is the estimate of $\hat{x}$, that is, the last seven components of $\mathbf{x}_f$ are the estimate of $\bar{\mathbf{d}}$, whereas the first seven components represent the estimate of $\mathbf{x}_1$ (which is not needed since it is measured). The matrices A_f and B_f are of the form

$$A_f = \hat{A} - L\hat{C}, \quad B_f = \begin{bmatrix} \hat{B} & L \end{bmatrix},$$

where $L \in \mathbb{R}^{14\times 7}$ is the output injection matrix that you need to design. *Warning:* Make sure that the sign of L is right such that A_f is stable.

Again, you can use `simQuad.m` to perform your simulations with one of the arguments being a structure `filter` that has to contain the matrices (A_f, B_f) (as fields `filter.Af` and `filter.Bf`) of the state-space representation of the disturbance estimator. If you want to use your own disturbance instead of a pre-specified one, you can do so through the last argument of `simQuad.m`.

Hints: If you have troubles (e.g., running into infeasibility) when tuning the estimator, you may consider dropping the constraints in your MPC problem during the tuning stage or otherwise separate the controller and estimator design. In addition, if you use the disturbance generated by the function `simQuad.m`, you can adjust the magnitudes of both the mean and the random part of the disturbance by editing line 29 of that function.

Deliverables

1. motivation for the choice of the estimation error dynamics (noise covariances if you use Kalman filter design procedure, pole locations if you use pole placement etc.) *5 %*
2. step reference tracking plots in the presence of disturbance *15 %*
3. slowly-varying reference tracking plots in the presence of disturbance *10 %*

Simulations on the nonlinear model

Finally you can try out your MPC controller on the nonlinear model. Open the Simulink model `Simulation2.mdl` and type in the Inner MPC block the name of your controller optimizer instance. The parameter vector for your controller optimizer instance should be $[\mathbf{x}_1{}^T \ \mathbf{r}_1{}^T \ \hat{\mathbf{d}}^T]^T$, where $\hat{\mathbf{d}}$ is the estimate of $\bar{\mathbf{d}}$.

Simulate first the model without disturbances (you can turn them on and off via a checkbox on the quadrotor block). If the response is satisfactory, simulate the system with disturbances. If you can perform this for the two reference signals we have prepared, you are done!

Deliverables

All of these need to be performed in the presence of disturbance.

1. plots of the reference tracking of a step signal *10 %*
2. plots of the reference tracking of the hexagon signal *5 %*
3. bonus: plots of the reference tracking of an “eight” (i.e., a lemniscate or ∞ shape) – you need to create the signal on your own *10 %*
4. bonus: add slew rate constraints on the control inputs, that is, constraints of the form $|u_{k+1} - u_k| \leq \Delta$. How small a Δ can you find without significant performance deterioration / feasibility problems? Provide plots of all state variables and control inputs for the step response. Plots for the linear model are enough. The function `simQuad.m` (with 1 as its last argument) can handle a controller optimizer instance which accepts the last control move applied to the system as its *third* input and the estimated disturbance as its *fourth* input (read the description of the function carefully). What would you do to improve feasibility properties in practice? 20 %