# Exercise 1: OLCAR

LQR → ILQC

# General Information

- Download from http://www.adrl.ethz.ch/doku.php/adrl:education:lecture:fs2015
- Code/Answers submitted by Wed, 15.4 through website
  - Code (only *_Design.m files)
  - Pdf (max. 1 page): 1-3 sentences per question
- Interviews on Fr, 17.4
  - Group signup for timeslots will be available on course website
  - 10min interview/group
  - Graded pass/fail
  - Grade boost: Ex.1: +0.1, Ex. 2: +0.05, Ex 3. +0.1
- Office hours as usual for questions

# Introduction

- Goal of exercises 1 & 3: design a controller for a quadrotor
  - Exercise 1: Model-based ILQC controller
    - Model accurate and complies with simulation/reality
  - (Exercise 2: Reinforcement learning)
  - Exercise 3: Adapt bias-model based controller
    - Controller model and simulation model differ

# Quadrotor – AscTec Hummingbird

| Technical Data – AscTec Hummingbird | |
|---|---|
| **UAV Type** | **Quadcopter** |
| Onboard computer | Up to Intel® Atom™ Processor Z530 |
| Size | 540 x 540 x 85,5 mm |
| Max. take off weight | 0,71 kg |
| Max. payload | 200 g |
| Flight time incl. payload | 20 min. |
| Range | 4,500 m ASL, 1,000 m AGL |
| Max. airspeed | 15 m/s |
| Max. climb rate | 5 m/s |
| Max. thrust | 20 N |

http://www.asctec.de/en/uav-uas-drone-products/asctec-hummingbird/#pane-0-1

# Quadcopter Model

- State $\mathbf{x} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}]^T \in \mathbb{R}^{12}$
- Input $\mathbf{u} = [F_z, M_x, M_y, M_z]^T \in \mathbb{R}^4$

- System dynamics:
  - underactuated
  - non-linear [Model.f_mode1(x,u)]

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u}) = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u}$$

# Exercise 1: Move quadrotor to goal state

- LQR
- LQR with via point
- ILQC
- ILQC with via point

- main_ex1.m:
  - Task_Design
  - Cost_Design
  - LQR_Design
  - ILQC_Design
  - (Visualization)

Editor - /home/winklera/git/MATLAB/l_olcar_2015_ex1_code/main_ex1.m

main_ex1.m | ILQC_Design.m | LQR_Design.m | Cost_Design.m | Task_Design.m

```
%% OLCAR - Exercise 1 - ILQC
close all; clearvars; clc;

addpath(genpath(pwd)); % adds folders and subfolders to path
% To generate plots of LQR/ILQG rollout
plotvec = {'quad_pos_noLoad','quad_angles_noLoad','control_input_noLoad','rotor_thrust_noLoad'};
create_pdf = [0 0 0 0 ];    % for which plots should a pdf be created
plot_ind   = [1 2 3 4 ];    % which data to plot on screen


%% Task definition
Task = Task_Design();

%Load the dynamic model of the quadcopter
load('Quadrotor_Model.mat','Model'); % save as structure "Model"

% Define cost function
Task.cost = Cost_Design( Model.param.mQ, Task );



%% Problem 1: Initial controller design
[Initial_Controller, Cost_LQR] = LQR_Design(Model, Task);

% Visualization of LQR controller
sim_out_lqr = Quad_Simulator(Model,Task,Initial_Controller);
disp('LQR controller performance:');
fprintf('Cost with LQR controller (metric: LQR cost function!): J* = %.3f \n', Cost_LQR);
fprintf('Start Quadcopter position: x = %.3f, y = %.3f, z = %.3f \n', sim_out_lqr.x(1:3,1));
fprintf('Final Quadcopter position: x = %.3f, y = %.3f, z = %.3f \n\n', sim_out_lqr.x(1:3,end));
Visualize(sim_out_lqr,Model.param,'plot_mode',1);
% Plot_Result(sim_out_lqr,Model.param,'plots',plotvec(plot_ind),'file',create_pdf(plot_ind),'path',pwd)


%% comment out to proceed to Problem 2
return;


%% Problem 2: ILQC controller design
t_cpu = cputime;
[ILQC_Controller, Cost] = ILQC_Design(Model,Task,Initial_Controller,@Quad_Simulator);
t_cpu = cputime - t_cpu;

% Visualization of ILQC controller
sim_out_ilqc = Quad_Simulator(Model,Task,ILQC_Controller);
fprintf('The ILQC algorithm found a solution in %fs \n\n',t_cpu);
fprintf('Final Quadcopter Position: xQ = %.3f, yQ = %.3f, zQ = %.3f \n',sim_out_ilqc.x(1:3,end));
fprintf('Final Quadcopter Velocity: xQ = %.3f, yQ = %.3f, zQ = %.3f \n',sim_out_ilqc.x(7:9,end));
Visualize(sim_out_ilqc,Model.param,'plot_mode',1);
% Plot_Result(sim_out_ilqc,Model.param,'plots',plotvec(plot_ind),'file',create_pdf(plot_ind),'path',pwd)
```

- main_ex1.m:
  - LQR_Design

Editor - /home/winklera/git/MATLAB/l_olcar_2015_ex1_code/main_ex1.m

main_ex1.m | ILQC_Design.m | LQR_Design.m | Cost_Design.m | Task_Design.m

```
%% OLCAR - Exercise 1 - ILQC
close all; clearvars; clc;

addpath(genpath(pwd)); % adds folders and subfolders to path
% To generate plots of LQR/ILQG rollout
plotvec = {'quad_pos_noLoad','quad_angles_noLoad','control_input_noLoad','rotor_thrust_noLoad'};
create_pdf = [0 0 0 0 ];    % for which plots should a pdf be created
plot_ind   = [1 2 3 4 ];    % which data to plot on screen


%% Task definition
Task = Task_Design();

%Load the dynamic model of the quadcopter
load('Quadrotor_Model.mat','Model'); % save as structure "Model"

% Define cost function
Task.cost = Cost_Design( Model.param.mQ, Task );



%% Problem 1: Initial controller design
[Initial_Controller, Cost_LQR] = LQR_Design(Model, Task);

% Visualization of LQR controller
sim_out_lqr = Quad_Simulator(Model,Task,Initial_Controller);
disp('LQR controller performance:');
fprintf('Cost with LQR controller (metric: LQR cost function!): J* = %.3f \n', Cost_LQR);
fprintf('Start Quadcopter position: x = %.3f, y = %.3f, z = %.3f \n', sim_out_lqr.x(1:3,1));
fprintf('Final Quadcopter position: x = %.3f, y = %.3f, z = %.3f \n\n', sim_out_lqr.x(1:3,end));
Visualize(sim_out_lqr,Model.param,'plot_mode',1);
% Plot_Result(sim_out_lqr,Model.param,'plots',plotvec(plot_ind),'file',create_pdf(plot_ind),'path',pwd)


%% comment out to proceed to Problem 2
return;


%% Problem 2: ILQC controller design
t_cpu = cputime;
[ILQC_Controller, Cost] = ILQC_Design(Model,Task,Initial_Controller,@Quad_Simulator);
t_cpu = cputime - t_cpu;

% Visualization of ILQC controller
sim_out_ilqc = Quad_Simulator(Model,Task,ILQC_Controller);
fprintf('The ILQC algorithm found a solution in %fs \n\n',t_cpu);
fprintf('Final Quadcopter Position: xQ = %.3f, yQ = %.3f, zQ = %.3f \n',sim_out_ilqc.x(1:3,end));
fprintf('Final Quadcopter Velocity: xQ = %.3f, yQ = %.3f, zQ = %.3f \n',sim_out_ilqc.x(7:9,end));
Visualize(sim_out_ilqc,Model.param,'plot_mode',1);
% Plot_Result(sim_out_ilqc,Model.param,'plots',plotvec(plot_ind),'file',create_pdf(plot_ind),'path',pwd)
```

# Problem 1.1: Compute LQR feedback gain K

- Linearize system dynamics (A,B):

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u}) = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u}$$

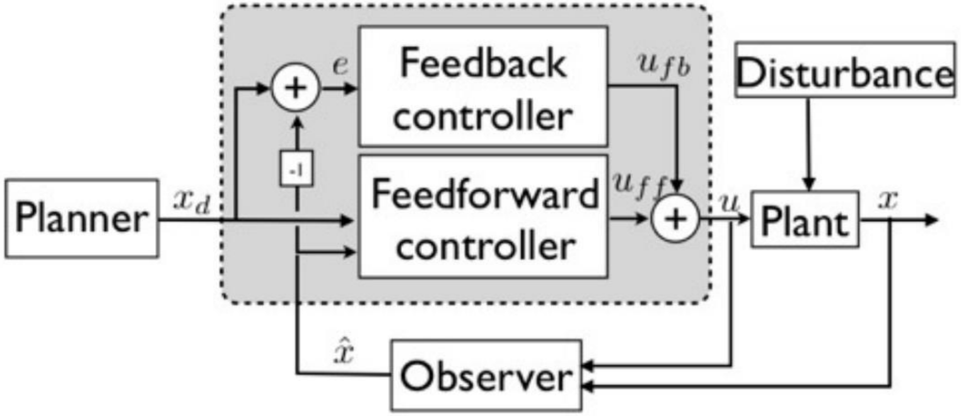
↓ e.g. Model.Alin{1}(x_lin,u_lin, Model.param.syspar_vec),...

$$\dot{\mathbf{x}} = \mathbf{A}_{lin}(\mathbf{x}_{lin})\mathbf{x} + \mathbf{B}_{lin}(\mathbf{u}_{lin})\mathbf{u}$$

- Cost function (Q,R)
  - (given in Cost_Design.m)
- Matlab

# Problem 1.2: Controller structure

- For every time-step n: Feedback and feedforward elements combined in $\theta_n \in \mathbb{R}^{13\times 4}$




$$\begin{aligned}\mathbf{u}_n &= = \mathbf{u}^{ff} + \mathbf{u}^{fb} \\ &= \mathbf{u}^{ff} + K(\mathbf{x} - \mathbf{x}_d) \\ &= \begin{bmatrix}\mathbf{u}^{ff} - K\mathbf{x}_d & K\end{bmatrix}\begin{bmatrix}1 \\ \mathbf{x}\end{bmatrix} \\ &= \theta_n^T \mathbf{x}_{aug}\end{aligned}$$

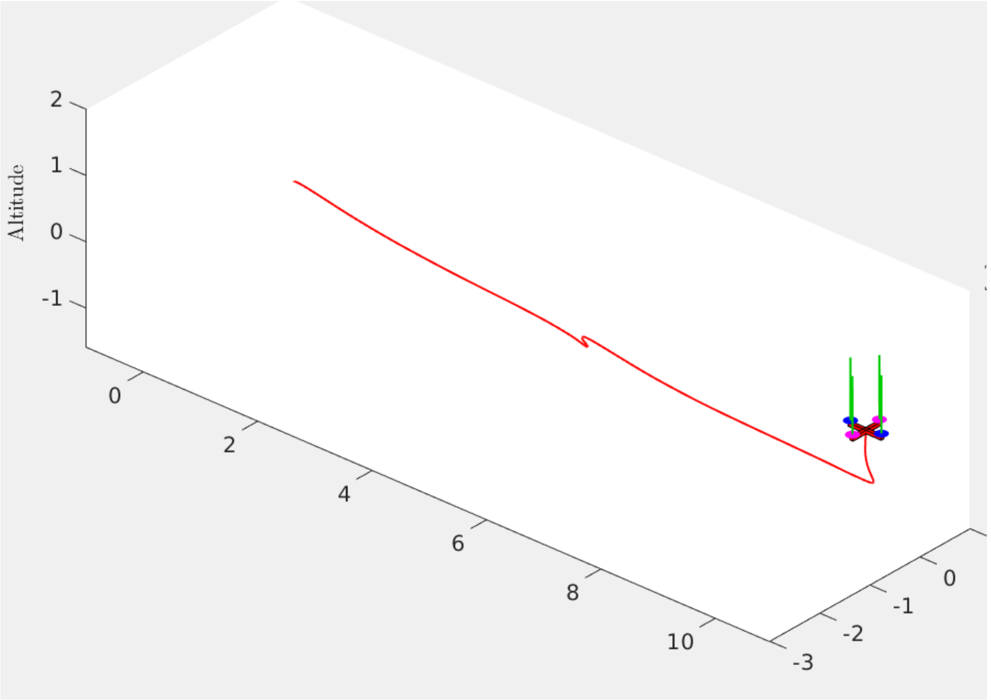
- $\theta \in \mathbb{R}^{13\times 4\times N_t}$

# Problem 1.2: Design controller from K

- Task: Move quadrotor to goal position at $\mathbf{x}_d(1) = 10m$
  - Define metric LQR controller tries to regulate to zero
  - $\mathbf{u} = K(\ldots$
  - Incorporate that in controller structure (design $\theta$)
- Run and observe…
  - (Change cost matrices, goal positions, metric,…)

# Problem 1.3: Include via-points

- Task: Move quadrotor through specific states before reaching the goal state.
- Time-varying control law
  - $\theta_n$ changes over time
- Observe and run



# Problem 1 ✓

- Finished designing LQR controller
  - Protected function “LQR_Design_Solution.p” available
  - → Adapt main_ex1.m.

- LQR Design ✓
- → ILQC Design

Editor - /home/winklera/git/MATLAB/l_olcar_2015_ex1_code/main_ex1.m

main_ex1.m | ILQC_Design.m | LQR_Design.m | Cost_Design.m | Task_Design.m

```matlab
%% OLCAR - Exercise 1 - ILQC
close all; clearvars; clc;

addpath(genpath(pwd)); % adds folders and subfolders to path
% To generate plots of LQR/ILQG rollout
plotvec = {'quad_pos_noLoad','quad_angles_noLoad','control_input_noLoad','rotor_thrust_noLoad'};
create_pdf = [0 0 0 0 ];    % for which plots should a pdf be created
plot_ind   = [1 2 3 4 ];    % which data to plot on screen


%% Task definition
Task = Task_Design();

%Load the dynamic model of the quadcopter
load('Quadrotor_Model.mat','Model'); % save as structure "Model"

% Define cost function
Task.cost = Cost_Design( Model.param.mQ, Task );



%% Problem 1: Initial controller design
[Initial_Controller, Cost_LQR] = LQR_Design(Model, Task);

% Visualization of LQR controller
sim_out_lqr = Quad_Simulator(Model,Task,Initial_Controller);
disp('LQR controller performance:');
fprintf('Cost with LQR controller (metric: LQR cost function!): J* = %.3f \n', Cost_LQR);
fprintf('Start Quadcopter position: x = %.3f, y = %.3f, z = %.3f \n', sim_out_lqr.x(1:3,1));
fprintf('Final Quadcopter position: x = %.3f, y = %.3f, z = %.3f \n\n', sim_out_lqr.x(1:3,end));
Visualize(sim_out_lqr,Model.param,'plot_mode',1);
% Plot_Result(sim_out_lqr,Model.param,'plots',plotvec(plot_ind),'file',create_pdf(plot_ind),'path',pwd)


%% comment out to proceed to Problem 2
return;


%% Problem 2: ILQC controller design
t_cpu = cputime;
[ILQC_Controller, Cost] = ILQC_Design(Model,Task,Initial_Controller,@Quad_Simulator);
t_cpu = cputime - t_cpu;

% Visualization of ILQC controller
sim_out_ilqc = Quad_Simulator(Model,Task,ILQC_Controller);
fprintf('The ILQC algorithm found a solution in %fs \n\n',t_cpu);
fprintf('Final Quadcopter Position: xQ = %.3f, yQ = %.3f, zQ = %.3f \n',sim_out_ilqc.x(1:3,end));
fprintf('Final Quadcopter Velocity: xQ = %.3f, yQ = %.3f, zQ = %.3f \n',sim_out_ilqc.x(7:9,end));
Visualize(sim_out_ilqc,Model.param,'plot_mode',1);
% Plot_Result(sim_out_ilqc,Model.param,'plots',plotvec(plot_ind),'file',create_pdf(plot_ind),'path',pwd)
```

# Problem 2.1: Design ILQC Controller

- LQR: system dynamics linearized around one linearization point
  - → linear time-invariant system
- ILQC: system dynamics linearized around each discretized state (50Hz)
  - → linear time-variant system

ILQC_Design.m

Editor - /home/winklera/git/MATLAB/l_olcar_2015_ex1_code/Design_functions/ILQC_Design.m

main_ex1.m | ILQC_Design.m | LQR_Design.m | Cost_Design.m | Task_Design.m

```matlab

%% Problem 2.1.2: Solve Riccati-like equations backwards in time
% Initialize the value function elements starting at final time step
% (Eq.(1.87)
xf = X0(:,end); % final state when using current controller
% Sm(:,:,n_t) = ...
% Sv(:,n_t)   = ...
% s(n_t)      = ...

% "Backward pass": Calculate the coefficients (s,Sv,Sm) for the value
% functions at earlier times by proceeding backwards in time (DP-approach)
for n = (length(sim_out.t)-1):-1:1

    % state of system at time step n
    x0 = X0(:,n);
    u0 = U0(:,n);

    % Discretize and linearize continuous system dynamics Alin around
    % specific pair (x0,u0). See exercise sheet Eq.(4) for details
    % A = ...;
    % B = ...;


    % 2nd order approximation of cost function at time step n (Eq.(1.78))
    % q   = ...
    % Qv  = ...
    % Qm  = ...
    % Rv  = ...
    % Rm  = ...
    % Pm  = ...

    % control dependent terms of cost function (Eq.(1.81))
    % g = ...                          % linear control dependent
    % G = ...                          % control and state dependent
    % H = ...                          % quadratic control dependent

    % ensuring H remains symmetric
    H = (H+H')/2; % important, do not delete!

    % the optimal change of the input trajectory du = duff + K*dx (Eq.(1.82))
    % duff(:,:,n) = ...
    % K(:,:,n)    = ...

    % Solve Riccati-like equations for current time step n (Eq.(1.84)
    % Sm(:,:,n) = ...
    % Sv(:,n) = ...
    % s(n) = ...

end % of backward pass for solving Riccati equation

% define theta_ff in this function
Controller.theta = Update_Controller(X0,U0,duff,K);

i = i+1;
```

# Problem 2.1: Design ILQC Controller

- (Discretization) :

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \Leftrightarrow \frac{\mathbf{x}_{n+1} - \mathbf{x}_n}{\delta t} &= \mathbf{A}\mathbf{x}_n + \mathbf{B}\mathbf{u}_n \\ \Leftrightarrow \quad \mathbf{x}_{n+1} &= (\mathbf{I} + \mathbf{A}\delta t)\mathbf{x}_n + (\mathbf{B}\delta t)\mathbf{u}_n \\ \Leftrightarrow \quad \mathbf{x}_{n+1} &= \mathbf{A}_n\mathbf{x}_n + \mathbf{B}_n\mathbf{u}_n \end{aligned}$$

(continous system dynamics)

↓

(discrete system dynamics)

- Run and observe…
- Protected function ILQC_Design_Solution.p available

- main_ex1.m:
  - Cost_Design

Editor - /home/winklera/git/MATLAB/l_olcar_2015_ex1_code/main_ex1.m

main_ex1.m | ILQC_Design.m | LQR_Design.m | Cost_Design.m | Task_Design.m

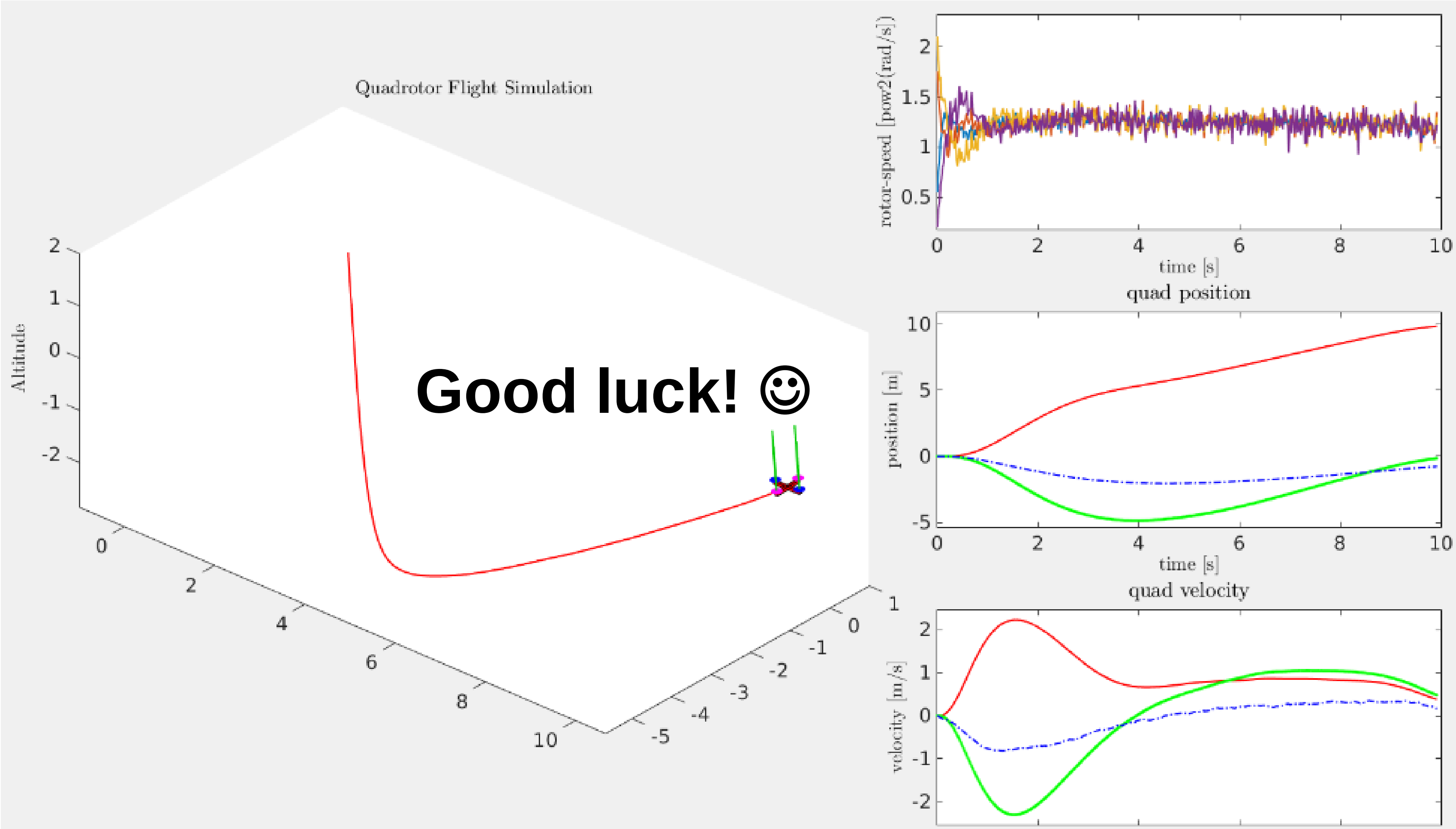
```matlab
%% OLCAR - Exercise 1 - ILQC
close all; clearvars; clc;

addpath(genpath(pwd)); % adds folders and subfolders to path
% To generate plots of LQR/ILQG rollout
plotvec = {'quad_pos_noLoad','quad_angles_noLoad','control_input_noLoad','rotor_thrust_noLoad'};
create_pdf = [0 0 0 0 ];    % for which plots should a pdf be created
plot_ind   = [1 2 3 4 ];    % which data to plot on screen


%% Task definition
Task = Task_Design();

%Load the dynamic model of the quadcopter
load('Quadrotor_Model.mat','Model'); % save as structure "Model"

% Define cost function
Task.cost = Cost_Design( Model.param.mQ, Task );



%% Problem 1: Initial controller design
[Initial_Controller, Cost_LQR] = LQR_Design(Model, Task);

% Visualization of LQR controller
sim_out_lqr = Quad_Simulator(Model,Task,Initial_Controller);
disp('LQR controller performance:');
fprintf('Cost with LQR controller (metric: LQR cost function!): J* = %.3f \n', Cost_LQR);
fprintf('Start Quadcopter position: x = %.3f, y = %.3f, z = %.3f \n', sim_out_lqr.x(1:3,1));
fprintf('Final Quadcopter position: x = %.3f, y = %.3f, z = %.3f \n\n', sim_out_lqr.x(1:3,end));
Visualize(sim_out_lqr,Model.param,'plot_mode',1);
% Plot_Result(sim_out_lqr,Model.param,'plots',plotvec(plot_ind),'file',create_pdf(plot_ind),'path',pwd)


%% comment out to proceed to Problem 2
return;


%% Problem 2: ILQC controller design
t_cpu = cputime;
[ILQC_Controller, Cost] = ILQC_Design(Model,Task,Initial_Controller,@Quad_Simulator);
t_cpu = cputime - t_cpu;

% Visualization of ILQC controller
sim_out_ilqc = Quad_Simulator(Model,Task,ILQC_Controller);
fprintf('The ILQC algorithm found a solution in %fs \n\n',t_cpu);
fprintf('Final Quadcopter Position: xQ = %.3f, yQ = %.3f, zQ = %.3f \n',sim_out_ilqc.x(1:3,end));
fprintf('Final Quadcopter Velocity: xQ = %.3f, yQ = %.3f, zQ = %.3f \n',sim_out_ilqc.x(7:9,end));
Visualize(sim_out_ilqc,Model.param,'plot_mode',1);
% Plot_Result(sim_out_ilqc,Model.param,'plots',plotvec(plot_ind),'file',create_pdf(plot_ind),'path',pwd)
```

# Problem 2.2: Include via-points

- Append ILQC cost function
  - Penalizes deviation from via point $\mathbf{x}_{vp}$ only in proximity of time $t_{vp}$.
  - $$L_{vp}(t) = (\mathbf{x} - \mathbf{x}_{vp})^\top \mathbf{Q}_{vp} \cdot \sqrt{\frac{\rho}{2\pi}} e^{\left(-\frac{\rho}{2}(t-t_{vp})^2\right)} (\mathbf{x} - \mathbf{x}_{vp})$$

- Run and observe
  - Use different via points
  - Compare to LQR

[Cedric de Crousaz, Farbod Farshidian, and Jonas Buchli. Aggressive optimal control for agile flight with a slung load. Workshop, IROS 2014]




Via point (x,y,z) = (5,-5,-5)