OLCAR Exercise 3

Path Integral Policy Improvement

Markus Giftthaler

Assigned: 12.05.2015

Due: 26.05.2015

Motivation

- In **Exercise 1**: model-based optimal control (ILQC)
  - Designs feedforward & feedback simultaneously
  - Very good performance in simulation

- **Reality**: imperfect model knowledge
  → performance of model-based controllers on real robots often "sub-optimal"

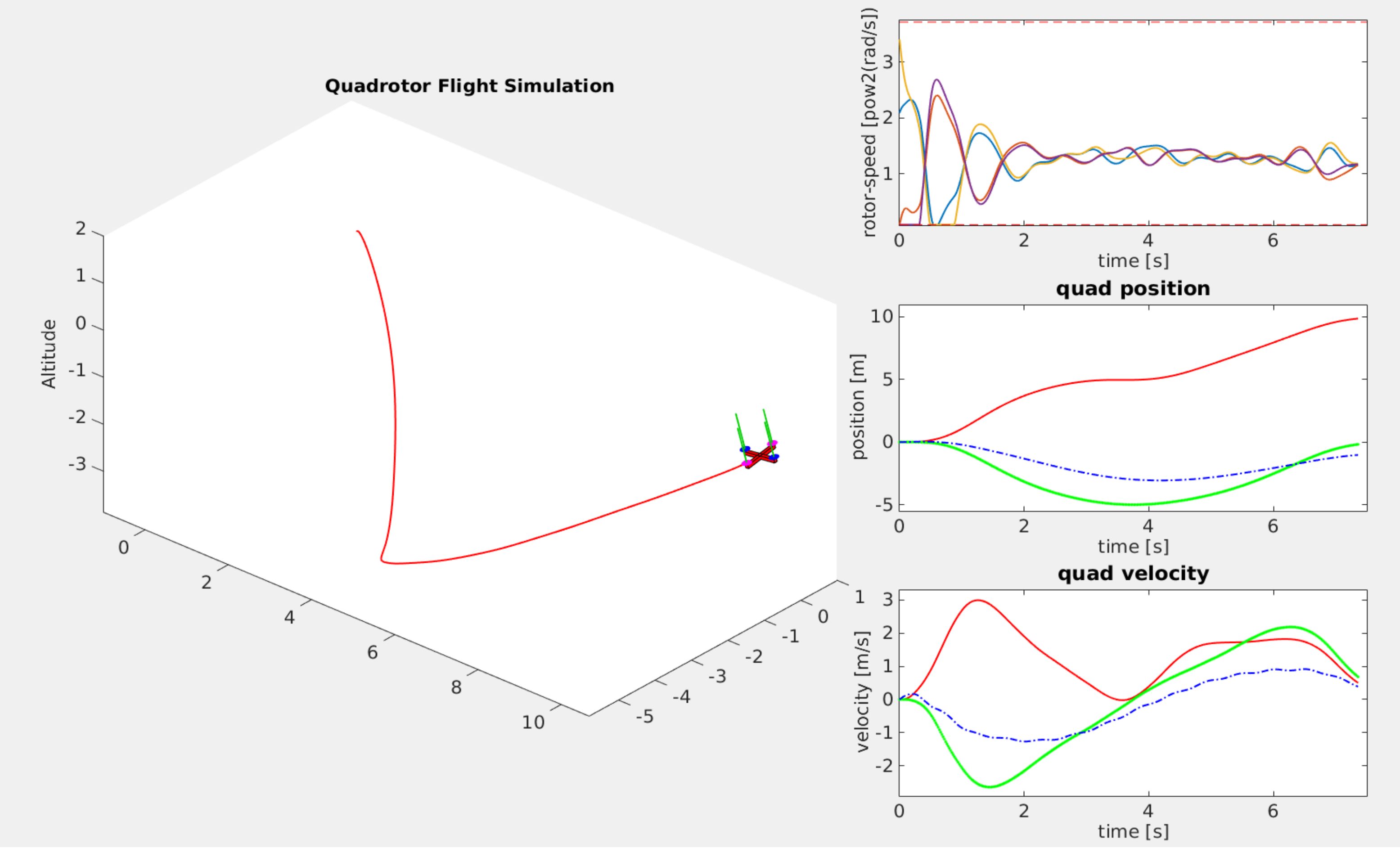
Motivation

- Now: Model-free Learning, e.g. Path Integral Policy Improvement (**PI2**)
  - *Learns* reference trajectory and controller
  - Does not exploit *domain knowledge* of the system designer
  - Performance highly depends on *quality of the initial guess*

- Idea:
  - Initialize PI2 with a trajectory and controller obtained from a model-based approach

Exercise 3

- PI2 on the Quadrotor (MATLAB)
- Only 1 type of task: via-point



A D R L

ETH Zürich

Exercise 3

```
%% OLCAR - Exercise 3: PI2 Learning

close all; 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 nominal model of the quadcopter
load('Quadrotor_Model.mat','Model');

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


%% Initial controller design - fill in your ILQC_Design here
[Initial_Controller, Cost_LQR] = LQR_Design(Model, Task);
[ILQC_Controller, Cost] = ...
clear Model;
```

Use your own ILQC_Design() function to compute the model-based optimal controller

Exercise 3

```matlab
%% Load the perturbed "real" quadrotor model
load('Quadrotor_Model_perturbed.mat','Model_perturbed');

%% Visualization of initial guess with policy in base function representation on perturbed system
Task.noise_insertion_method = '';
Test_sim_out = Sample_Rollout(Model_perturbed,Task,ReducedController);
fprintf('Final Quadcopter Position: xQ = %.3f, yQ = %.3f, zQ = %.3f \n',Test_sim_out.x(1:3,end));
fprintf('Final Quadcopter Velocity: xQ = % .3f, yQ = %.3f, zQ = %.3f \n',Test_sim_out.x(7:9,end));
Visualize(Test_sim_out,Model_perturbed.param,'plot_mode',1);
%Plot_Result(Test_sim_out,Model_perturbed.param,'plots',plotvec(plot_ind),'file',create_pdf(plot_ind),'path',pwd)

%% Start PI Learning
t_cpu = cputime;
[LearnedController,AllCost,AllController] = PIs_Learning(Model_perturbed,Task,ReducedController);
t_cpu = cputime - t_cpu;
fprintf('CPU time: %f \n',t_cpu);
fprintf('The PI² algorithm took %fs to converge \n\n',t_cpu);
```

Implements Algorithm 10 of Chapter 3
(General PI2 Learning)

Exercise 3

Algorithm 10 General PI2 Algorithm

given

The cost function:

$J = \Phi(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} \left(q(t,\mathbf{x}) + \frac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u} \right) dt$

A Linear Model for function approximation: $\mathbf{u}(t,\mathbf{x}) = [u_i(t,\mathbf{x})] = \left[\text{grand sum}\left[\bar{\boldsymbol{\Upsilon}}(t,\mathbf{x}) \circ \boldsymbol{\theta}_i\right]\right]$

Initialize $\{\boldsymbol{\theta}_i\}$ with a sophisticated guess

Initialize exploration noise standard deviation: c

repeat

Create K rollouts of the system with the perturbated parameter $\{\boldsymbol{\theta}_i\} + \{\boldsymbol{\epsilon}_i\}$, $\{\boldsymbol{\epsilon}_{i,j}\} \sim \mathcal{N}(\mathbf{0}, c^2\mathbf{I})$

for the ith control input **do**

for each time, s **do**

Calculate the Return from starting time s for the kth rollout:

$R(\tau^k(s)) = \Phi(\mathbf{x}(t_f)) + \int_{s}^{t_f} \left(q(t,\mathbf{x}) + \frac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u} \right) dt$

Calculate α from starting time s for the kth rollout:

$\alpha^k(s) = \exp(-\frac{1}{\lambda}R(\tau^k(s))) / \sum_{k=1}^{K} \exp(-\frac{1}{\lambda}R(\tau^k(s)))$

Calculate the time varying parameter increment $\Delta\boldsymbol{\theta}_i(s)$:

$\Delta\boldsymbol{\theta}_i(s) = \sum_{k=1}^{K} \alpha^k(s) \frac{\boldsymbol{\Upsilon}(s)\boldsymbol{\Upsilon}^T(s)}{\boldsymbol{\Upsilon}^T(s)\boldsymbol{\Upsilon}(s)} \boldsymbol{\epsilon}_i^k(s)$

end for

for the jth column of $\Delta\boldsymbol{\theta}_i$ matrix, $\Delta\boldsymbol{\theta}_{i,j}$ **do**

Time-averaging the parameter vector

$$\Delta\boldsymbol{\theta}_{i,j} = \left(\int_{t_0}^{t_f} \Delta\boldsymbol{\theta}_{i,j}(s) \circ \boldsymbol{\Upsilon}(s) ds \right) \cdot \bigg/ \int_{t_0}^{t_f} \boldsymbol{\Upsilon}(s) ds$$

end for

Update parameter vector for control input i, $\boldsymbol{\theta}_i$:

$\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i + \omega\Delta\boldsymbol{\theta}_i$

end for

- Decrease c for noise annealing

until maximum number of iterations

Task: implement this in PI2_Update()

Given in PIs_Learning()

Exercise 3

- delta_theta = PI2_Update(Task, batch_sim_out, batch_cost)

- Important: in PI2_Update, we provide you two additional functions:

```
%% additional functions vec2mat and mat2vec that are provided
function mat = vec2mat(vec)...


function vec = mat2vec(mat)...
```

Why?

Simulator accepts 'theta' only in vector form, but in Algorithm 10, 'theta' is in matrix form.

More details are given in the handout!

- Your task: <u>complete PI2_Update + answer 5 additional Questions</u>

ADRL

ETH Zürich

General Info

- Files and code: http://www.adrl.ethz.ch/doku.php/adrl:education:lecture:fs2015
- Submit solutions by Tue, **26.05.2015** at Midnight
  - Deliverables: PI2_Update.m + .pdf with answers to questions
  - Answers to questions in .pdf format – Max. 5 sentences per question
- Interviews on **Thu 28.5 08.00-13.00** and **Fri 29.5 08.00-13.00**
  - Sign-up doodle: https://ethz.doodle.com/bsi7gvkycvrmht6t
  - Room **CLA D 11.1**
  - 10 minute interview for each group
  - Graded pass/fail
  - Bonus credit Ex 3. +0.1

Good Luck!

Organisation:

Markus Giftthaler mgiftthaler@ethz.ch

Please note:

No Office hours this Thursday, 14.05 (public holiday)