Computergestuurde Regeltechniek
exercise session
Case study : lightsoccer

Oscar Mauricio Agudelo, Bart De Moor
(mauricio.agudelo@esat.kuleuven.be, bart.demoor@esat.kuleuven.be)

June 8, 2016

Problem description

Consider the soccer situation depicted in Figure 1. An attacker is running towards the goal (just outside of the penalty area). The goalkeeper has to defend his goal in the best possible way and therefore has to choose the most optimal position and orientation (see Figure 2) within the penalty area.

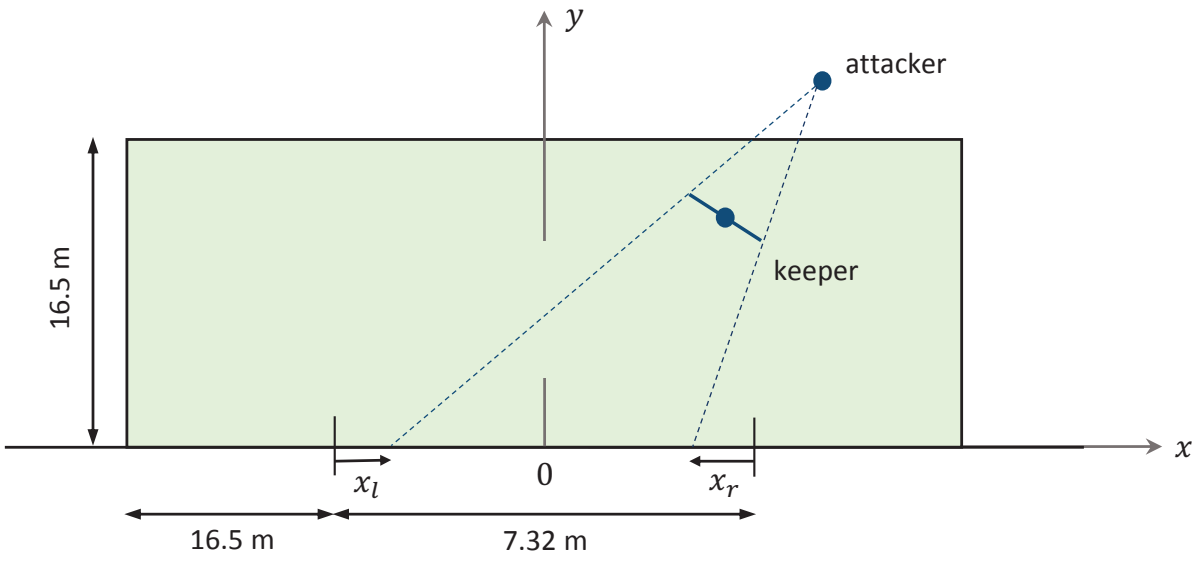

Figure 1: A typical soccer situation. An attacker tries to kick the ball into the goal, the goalkeeper tries to prevent this.

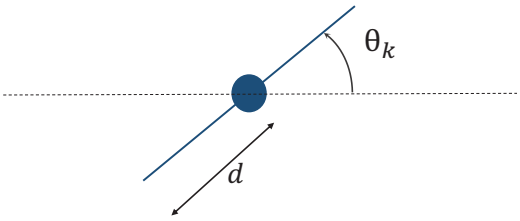

Figure 2: Goalkeeper orientation

In this exercise the following assumptions are made:

- The effect of the finite velocity of the ball is neglected, so it is assumed that the ball is moving at the speed of light ($\rightarrow$ *lightsoccer*).
- Curved trajectories of the ball are not possible, the path of the ball is always perfectly straight.
- The ball is considered to be a point mass.
- The attacker can aim the ball perfectly. The ball always arrives where the attacker wants it to. The attacker is always outside the penalty area.
- A flat, 2D-world is assumed!

1 Assignment

1.1 Formulation of the optimization problem for the static (no dynamics) goalkeeper case

The aim of this section is to formulate the optimization problem that has to be solved in order to find the best position (i.e., $x_k \rightarrow x$-coordinate, $y_k \rightarrow y$-coordinate) and orientation ($\theta_k \rightarrow$ angle of the arms with respect to the x-axis) of the goalkeeper for a given attacker position ($x_a \rightarrow x$-coordinate, $y_a \rightarrow y$-coordinate). The numerical values of x_a, y_a and d, the length of each arm of the goalkeeper, can be found in Table 1.

- In Figure 1, x_l and x_r are respectively the left and right uncovered parts of the goal-line of the goalkeeper. Find an expression for x_l and x_r in terms of x_k, y_k, θ_k, x_a and y_a.
- Write an m-file `cover.m` that implements the function

  `[xl xr] = cover(goalkeeper,attacker)`,

  and run this function for some test values (keep in mind that the goalkeeper is not allowed to leave the penalty area) and check the results. In addition, make a plot that shows the goalkeeper and the attacker on the soccer field, as well as the lines that delimit the area covered by the goalkeeper.

To find the best position and orientation of the goalkeeper (for a known position of the attacker), we look for the minimum of the cost function

$$J = x_l^2 + x_r^2.$$

- Do you think this cost function will give good results? Why/why not? Discuss!
- Write an m-file `costfunction.m` that implements the function

  `J = costfunction(goalkeeper,attacker).`

  In order to "get an idea" on how the cost function looks like (bear in mind that the cost function depends on three independent variables, x_k, y_k, θ_k and therefore a 4D plot would be necessary to properly visualize it), make a 3D plot of

  $$\log_{10}\left(J(x_k, y_k)\right)$$

  for $x_k \in [-30,\ 30]$ and $y_k \in [-10,\ 50]$. Here θ_k is set to be equal to 0 and the numerical values of x_a and y_a are given in Table 1.
- Finally, formulate the optimization problem that has to be solved in order to find the optimal position and orientation (x_k^*, y_k^* and θ^*) of the keeper, bearing in mind that the goalkeeper cannot leave the penalty area.

1.2 Dynamic keeper model

In the Section 1.1, we considered the static keeper positioning problem. In the **rest of this exercise** we will treat **the goalkeeper as a dynamical system**, and we will focus on the design of LQR and MPC controllers for tracking purposes. The control goal is to make sure that the goalkeeper follows as close as possible a set of desired positions and orientations, while satisfying the goalkeeper constraints (e.g, the keeper has a maximum torque to make turns, the keeper can not leave the penalty area, etc.).

We are still thinking of a 2D world. The keeper has a mass m, a rotational inertia I, has a certain maximal force $F_{\max}$ (component-wise) in order to accelerate, and a certain maximum torque $M_{\max}$ (to make turns). We also assume that the keeper, while running, is subject to a 'damping force' from the grass, proportional to its speed, with a factor ρ. Also, the rotation speed is slightly damped. You can find the numerical values of the physical parameters in Table 1.

The resulting continuous-time differential equations are as follows:

$$\begin{aligned} \dot{x} &= v_x & \dot{v}_x &= \frac{1}{m}(F_x - \rho\, v_x) \\ \dot{y} &= v_y & \dot{v}_y &= \frac{1}{m}(F_y - \rho\, v_y) \\ \dot{\theta} &= \omega & \dot{\omega} &= \frac{1}{I}(M - 0.01\omega) \end{aligned}$$

where x and y are the coordinates of the keeper position and θ is the angle of the goalkeeper arms with respect to the x-axis. x, y and θ are the outputs of the system.

- Derive a discrete-time state-space model by applying Euler's rule. The sampling time T_s is given in Table 1. Provide the symbolic expressions of $\boldsymbol{A}_d$, $\boldsymbol{B}_d$, $\boldsymbol{C}_d$ and $\boldsymbol{D}_d$ as well as their numerical values.

- Is the discrete-time model obtained with the Euler's rule stable?, what are the poles?, is it controllable?, is it observable?, is it stabilisable?, is it detectable?, is it minimal?, are there transmission zeros?

- Try out other discretization rules, such as the bilinear transformation (also called tustins rule), the backward rectangular rule and the zero-order hold method. Plot the step responses of the linear continuous-time system and the different discrete-time models. Discuss. Choose one of your discrete-time models to use in the rest of the exercise. **Remark:** You can compute the discrete-time models numerically using the Matlab command `c2dm` and/or by applying the formulas that appear in the course notes (see page 53).

1.3 Setpoints

A set of M desired keeper positions and orientations, $\boldsymbol{y}_{ref}(k) = \begin{bmatrix} x_{ref}(k) & y_{ref}(k) & \theta_{ref}(k) \end{bmatrix}^T$ is given in the *.mat file `trajectory_gk.mat`. This file can be found in the website of the course.

- Calculate a sequence of reference inputs $\boldsymbol{u}_{ref}(k)$ and states $\boldsymbol{x}_{ref}(k)$ from the given sequence of keeper positions and orientations (dynamic reference insertion).

  - You can find the input and state sequences in two steps. In the first step, you can compute the input references by solving an optimization problem (see pages 125 -131 of the course notes) and in the second step, you can carry out a simulation with the input references you found in order to calculate the state references. You can assume that the initial reference state for the goalkeeper is $\boldsymbol{x}_{\text{ref}}(k=0) = \boldsymbol{0}$ (Notice that the initial output reference $\boldsymbol{y}_{ref}(k=0)$ given to you is also equal to $\boldsymbol{0}$). **For the computation of the setpoints, you should not add input constraints, or the goal constraint to the optimization problem. Also you should not clip the values of $\boldsymbol{u}_{\text{ref}}(k)$.** (Hint: Make sure that your optimization (least-squares) problem is well-posed).

- Plot the references for the states ($\boldsymbol{x}_{ref}(k)$) and the inputs ($\boldsymbol{u}_{ref}(k)$). Are they what you would expect them to be? Discuss.

1.4 LQR control

The control goal is to make sure that the goalkeeper can properly follow the set of desired positions and orientations given to you. You can assume that the entire state vector is available for the rest of this exercise.

- By using a state-feedback law that takes into account the setpoints computed previously, that is, $\boldsymbol{u}(k) = \boldsymbol{u}_{ref}(k) - \boldsymbol{K}(\boldsymbol{x}(k) - \boldsymbol{x}_{ref}(k))$, design two LQR controllers for the keeper:
  - LQR_1: Use $\boldsymbol{R} = 10^{-4} \cdot \boldsymbol{I}_{3\times3}$ as the weight matrix for the inputs, and $\boldsymbol{Q} = \boldsymbol{I}_{6\times6}$ as the weight matrix for the state vector.
  - LQR_2: Use $\boldsymbol{R} = 10^{-4} \cdot \boldsymbol{I}_{3\times3}$ as the weight matrix for the inputs, and $\boldsymbol{Q} = diag(\boldsymbol{I}_{3\times3}, \boldsymbol{0}_{3\times3})$ as the weight matrix for the state vector.

  Provide the numerical values of the matrix gains of both LQR controllers. Note that the state-feedback law used in this exercise, is not equivalent to the one presented in the course notes (see page 219), since in the course notes the system under control has reached steady-state, and this is not the case here (dynamic reference insertion).
- Simulate both controllers by using the discrete-time model of the keeper and the references previously found (i.e., $\boldsymbol{x}_{ref}(k)$ and $\boldsymbol{u}_{ref}(k)$). **Make sure that you clip the control actions** when they exceed the input constraints.
  - Plot the trajectory followed by the keeper with each controller together with the desired trajectory. Plot the control actions and the state variables. Discuss the results.
  - Calculate the total-simulation cost for each controller (hint: think of the LQR objective function). Discuss.

1.5 MPC control

- Formulate an MPC controller with horizon N to control the keeper. Use $\boldsymbol{R} = 10^{-4} \cdot \boldsymbol{I}_{3\times3}$ and $\boldsymbol{Q} = diag(\boldsymbol{I}_{3\times3}, \boldsymbol{0}_{3\times3})$ as input and state weight matrices respectively. **Remark:** Here you should only consider input constraints.
- Simulate the controlled system (hint: use `quadprog` to solve the optimization problem).
  - Plot the trajectory followed by the keeper and the desired trajectory. Plot the control actions and the state variables. Calculate the total-simulation cost in a similar fashion as you did in Section 1.4. Compare the results with those obtained with LQR_2. What can you conclude?
  - Investigate the computational complexity at each iteration (hint : use `tic, toc`) and discuss.
  - Repeat the simulations with horizon $N/2$ and $2N$. What are the implications on performance and computational complexity?

1.6 MPC control with state constraints

- Modify your MPC formulation such that the constraint that the keeper cannot leave the goal area is incorporated.

- Simulate the controlled system and evaluate its performance with horizons N, $N/2$ and $2N$ as you did in Section 1.5. Compare the results with those obtained in Section 1.5. Do you observe infeasibilities?
  - Increase the maximal force in this way: $F_{\max}^{\text{new}} = 10F_{\max}$, and repeat the simulations. Discuss the results.

1.7 MPC control with terminal cost

For this part of the exercise, the **state constraints** (i.e., the goal area constraint) **are not considered** anymore, so you can forget them.

- Add a terminal cost to your MPC formulation (i.e., the one of Section 1.5) based on an LQR controller with the same $\boldsymbol{Q}$ and $\boldsymbol{R}$ as the MPC controller.
- Simulate the controlled system and evaluate its performance with horizons N, $N/2$ and $2N$ as you did in Section 1.5. Compare the results with those obtained in Section 1.5 and discuss.

Table 1: Numerical values

Group No.	Attacker position		Keeper parameters				Input constraints		Sampling time	Prediction horizon
	x_a	y_a	d	m	I	ρ	$F_{\max}$	$M_{\max}$	T_s	N
1, 11, 21, 31	20	17	0.80	80	1.6	20	200	10	0.15	10
2, 12, 22, 32	-20	18	0.90	70	1.5	20	150	10	0.15	10
3, 13, 23, 33	10	17	1.00	90	1.8	20	180	10	0.12	12
4, 14, 24, 34	-10	17.5	1.10	90	1.8	20	200	10	0.12	10
5, 15, 25, 35	15	17	1.00	75	1.5	20	200	10	0.12	12
6, 16, 26, 36	-15	18	0.90	85	1.7	20	180	10	0.15	10
7, 17, 27, 37	-5	17.5	0.90	70	1.4	20	190	10	0.15	12
8, 18, 28, 38	5	17	0.85	85	1.8	20	200	10	0.15	10
9, 19, 29, 39	8	18	0.95	80	1.6	20	180	10	0.15	12
10, 20, 30, 40	-12	18	0.90	80	1.5	20	170	10	0.15	10

Remark: all values are expressed in SI-units and positions are given with respect to the middle of the goal line.

2 Deliverables

We expect from you:

- A written report (a hard copy and a digital copy).
- All the Matlab files that allow us to reproduce your results. Also provide a document where you briefly explain the purpose of each of them.

Remarks about the report: The report should be written preferably **in English**, and it should address every single point of Section 1. Not only the technical correctness of your report is evaluated but also the quality and presentation. So, make sure that everything is clear and well explained. For example, make sure that you label every axis of every plot, that your figures include legends if necessary, that in the text you discuss or address what is presented in every figure of the manuscript, that the text is free of typos and well redacted, etc. In addition, your report must be self-contained. Do not refer the reader to your Matlab files. Keep in mind that we only ask you for your Matlab files for verification purposes.

3 Examination

At the end of the semester you will have an oral exam. This oral exam will include some questions about your report (e.g., technical details, results, etc.) as well as some general questions. Further information (dates of examination, how the points are distributed between exercises and practicum, etc.) can be found in the webpage of the Computergestuurde Regeltechniek course in Toledo.