Model-free and model-based time-optimal control of a badminton robot

Liu, M; Depraetere, B; Pinte, G; Grondman, I; Babuska, R

DOI
10.1109/ASCC.2013.6606242

Publication date
2013

Document Version
Final published version

Published in
Proceedings 2013 9th Asian Control Conference

Model-free and model-based time-optimal control of a badminton robot

M. Liu*, B. Depraetere†‡, G. Pinte†, I. Grondman* and R. Babuška*
*Delft Center for Systems and Control, Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands
†Flanders' Mechatronics Technology Centre (FMTC), Celestijnenlaan 300 D, 3001 Heverlee-Leuven, Belgium
‡ Email: bruno.depraetere@fmtc.be

***Abstract*—In this research, time optimal control is considered for the hit motion of a badminton robot during a serve operation. For this task the racket always starts at rest in a given position and has to move to a target state, defined by a target position and a non-zero target velocity. The goal is to complete this motion in as little time as possible, yet without violating bounds on the actuator. To find controllers satisfying these requirements, a reinforcement learning approach is implemented, using a Natural Actor-Critic (NAC) reinforcement learning algorithm. This approach is experimentally shown to yield the desired robot motions after about 200 trials. Next to this model-free learning approach, the control signals obtained with a model-based optimization are also applied to the robot. The results achieved with both approaches are compared, and a thorough analysis is presented, highlighting the properties of each approach, as well as their advantages and drawbacks.**

I. Introduction

In this paper, time-optimal motion is considered for the serve operation of a badminton robot. For this operation, the racket has to move from an initial stance to a desired hit position with a prescribed hit velocity, and this has to be done as quickly as possible taking into account the actuator limitations, so that the opponents preparation time is low. In general, similar time-optimal motion control problems are relevant for many mechatronic applications, since being able to generate faster motions typically means more units can be produced or more output can be generated in a given time span. Research into time-optimal motion has therefore already received attention [1], [2], [3], [4], [5], but mostly only a model-based approach has been considered. An optimal control problem is then typically solved numerically, explicitly minimizing the motion time or using approximate costs yielding simpler optimization problems. While these model-based approaches yield good results, they do require an accurate model of the system to be controlled. In many cases however, such models are not available or are difficult to obtain, so that model-free approaches would be preferred. These can operate purely by interacting with the environment and by learning from these interactions, so that they can be employed with only very little prior information. In this research, the goal is therefore to implement a model-free approach as well as a classical approach from the field of model-based control, and to compare their performance for the specific task of the badminton robot's serve operation.

This work has been carried out within the framework of projects IWT-SBO 80032 (LeCoPro) of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

Learning approaches have already been developed for time-optimal motion [4], [5], but they usually rely on explicit models of the controlled system or implicitly identify a model for the dynamic behaviour. In this research, the goal is to apply a model-free reinforcement learning (RL) algorithm, which directly learns a control policy for the considered task, without the knowledge of a model of the system. Among the model-free methods, RL is a framework in which an agent (the controller) learns an optimal policy to control its environment (the considered system) by using experience obtained from interacting with it. The interaction is characterized by two relevant aspects, being perception and action [6]; Each time an action is taken (driving actuators), a state transition can occur and a scalar reward is calculated using the observed results (evaluation of the quality of the action). The agent then adapts the actions it can take, aiming to maximize the reward received in the future. It is thus essential to select a reward function corresponding to the control objectives.

A popular class of RL algorithms are the actor-critic methods [7], where the actor is equivalent to the control policy and the critic is the value function evaluating it. Actor-critic methods can deal with continuous state and action spaces and, in general, have good convergence properties and performance if a gradient-based policy improvement is used [7]. The critic provides a low-variance value function estimate based on which the policy gradient is computed, and the actor updates the policy parameters in the direction of performance improvement indicated by the gradient. For a better learning performance, it has also been suggested to use the natural policy gradient, which gives the steepest ascent direction with respect to the Fisher information matrix, instead of the standard gradient [8]. The Natural Actor-Critic (NAC) algorithm [9] is therefore employed in this research, which is an algorithm employing the natural policy gradient. It is widely used in robotics and often yields a good learning performance [10], [11], [12]. For this application however, some modifications are made to the normal NAC algorithm to make it work off-line.

The remainder of the paper is organized as follows. First, the badminton robot is introduced in Section II, and the task to be performed is defined. Next, the model-based and model-free approaches are developed, in Sections III and IV, respectively. The experimentally obtained results for both controllers are then presented in Section V, including a detailed comparison and discussion. Finally, conclusions are drawn in Section VI.

II. Badminton robot

A. *Overview of badminton robot*

The setup used for the experimental validation is the badminton robot developed by FMTC. The overall system, illustrated in Figure 1, consists of a mobile platform on which the racket is mounted, and a complex controller which detects shuttles using a 3-dimensional camera system and moves the robot to intercept and hit them. A more detailed description of the badminton robot can be found in [13], and a movie of the robot in action at http://www.fmtc.be.

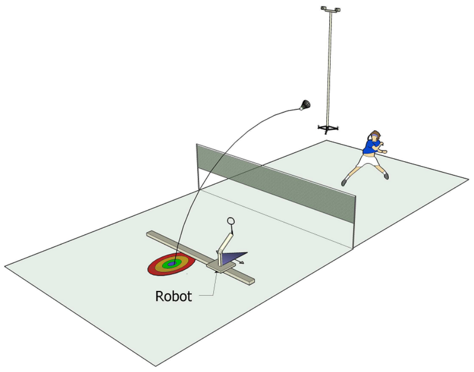

Figure 1. Overview of the badminton robot system, consisting of a mobile robot platform with a mounted racket, and a complex controller that collects data from a 3-dimensional camera system and drives the robot's actuators.

The mobile platform is essentially a serial robot with three degrees of freedom, as shown in Figure 2. The first two are driven by the linear and rotational motor (Rot. motor in the figure), and they respectively allow the robot to move along the linear guide and to rotate in a plane perpendicular to a typical shuttle trajectory. Since no or only small motions of these motors are needed to serve, their control is not considered in this paper. For the serve, the important degree of freedom allows the racket to rotate backward and forward along the joint near its handle, making it possible to perform hit motions. This motion is controlled by the hit motor indicated in the figure, and will be studied in the remainder of the paper.

The setup is controlled using a triphase rapid prototyping platform [14], onto which a Simulink model is compiled. This model interacts with EtherCAT modules to collect sensor data and control the motors, and sends data to and from a Matlab environment where all calculations are performed.

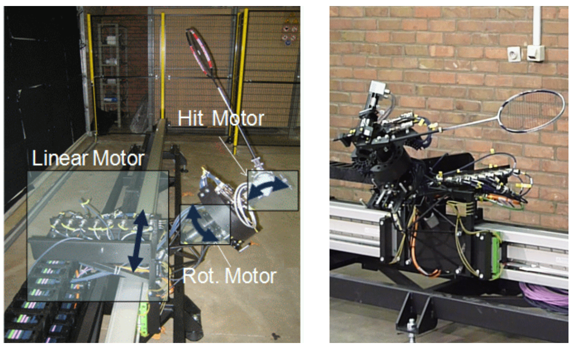

Figure 2. Close-up of the badminton robot and its three degrees of freedom, of which only the hit motion is considered in this research.

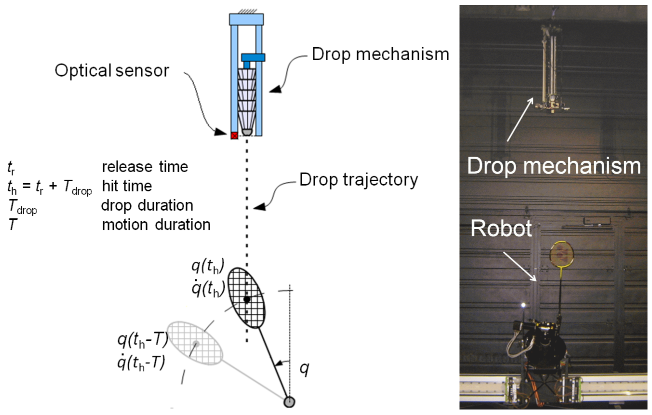

Figure 3. Schematic overview of the serve operation, where a shuttle is dropped from a mechanism above the robot, and the robot has to perform a hit motion to intercept the shuttle at a predetermined position and time.

B. *Serve operation of badminton robot*

In this paper only the serve motion of the robot is considered, since this motion is always performed in a similar manner. As a result, learning becomes possible, and it is straightforward to compare the results obtained during different hits and using different strategies. In order to perform a serve, a shuttle is dropped from a mechanism placed at a fixed point above the robot, and the instant $t = t_r$ it starts to fall is detected using an optical sensor, as shown in Figure 3. Since the drop mechanism always releases the shuttles in the same manner, the point where the robot has to hit the shuttle is always approximately the same, as is the period of time T_{drop} between the shuttle's release t_r and the hit $t_h = t_r + T_{\text{drop}}$. To obtain a time-optimal motion, the goal is then to start moving as late as possible, at $t = t_h - T$, minimizing the motion's duration T. For the ease of notation, however, it is assumed in the remainder of the paper that the hit motion starts at $t = 0$ and is completed at $t = T$.

Summarizing the specifications for the serve motion, the hit motor has to start at an angle q and angular velocity $\dot{q}$

$$\{q(0)\,,\ \dot{q}(0)\} = \{-\pi/4\,,\ 0\}\ \text{rad}, \tag{1}$$

and has to move to the desired hit point with

$$\{q(T)\,,\ \dot{q}(T)\} = \{0\,,\ 3\}\ \text{rad/s}, \tag{2}$$

while minimizing T so that the motor reacts as late as possible, and without violating bounds on the motor. In this case, the controller sends a voltage signal to a motor drive operating in torque mode. The allowable range for these controller voltages and hence also the controller outputs is $[-0.2,\ 0.2]$ V.

III. Model-based control

For the model-based approach, a parametric model is needed to predict the behaviour of the racket. To obtain this model, a frequency response function (FRF) between the actuator current u and the racket's angle q is first estimated, as shown in Figure 4. The racket generally behaves like a double integrator, as would be expected for an inertia driven by a motor. Other dynamics are observable as well however, at low frequencies due to friction in the bearing, and around 10 Hz, where the resonance frequency for the racket is found. In a next

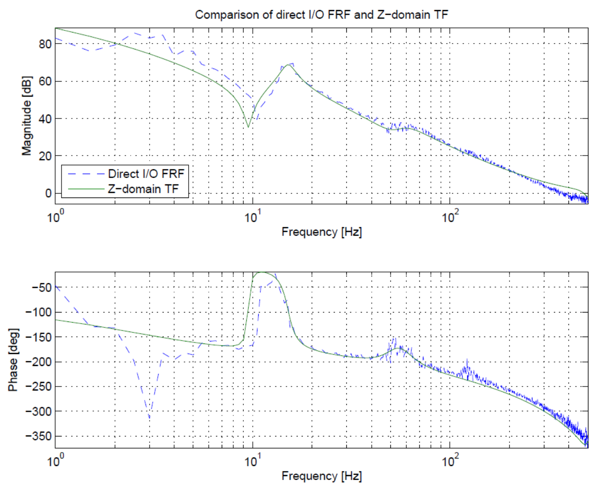

Figure 4. FRF's from input current to angular displacement of the hit axis, for the measurements (dashed line) and for the estimated model (solid line).

step, a parametric model is then fitted onto this FRF, using a typical least-squares model-fitting procedure [15]. In this case, a linear state space model is found, of which the matrices A, B, C and D will be used in the optimization procedure.

Using the estimated prediction model and the specifications, an optimal control problem can be formulated, searching for the optimal motion profiles and the corresponding control signals. The following problem is then obtained:

$$\min_{x,\, q,\, \dot{q},\, u,\, T} \quad T, \tag{3a}$$

s.t.

$$q(0) = -\pi/4, \quad \dot{q}(0) = 0, \tag{3b}$$

$$q(T) = 0, \quad \dot{q}(T) = 3, \tag{3c}$$

$$\dot{x} = Ax + Bu, \tag{3d}$$

$$\begin{pmatrix} q \\ \dot{q} \end{pmatrix} = Cx + Du, \tag{3e}$$

$$-0.2 \leq u \leq 0.2. \tag{3f}$$

Constraints (3b) and (3c) are included to impose the initial and final conditions specified by (1) and (2), constraints (3d) and (3e) describe the linear system dynamics given by the state space model indicated in Figure 4, and constraint (3f) finally bounds the allowable motor current, ensuring that no currents are demanded that the motor drive cannot supply.

In problem (3), the duration of the motion T is an optimization variable, and is not known beforehand, making this a non-convex problem that is difficult to solve. In this paper, we will instead solve a series of simpler problems, in which the duration T_i is always fixed, as described in [1]. The idea is to check for which values of T_i a feasible solution can be found satisfying the constraints, and for which values of T_i this is not the case. To do so, T_i is fixed and the cost function omitted in problem (3), after which it suffices to check the feasibility of the resulting convex linear optimization problem, which takes significantly less time than if a cost was included. Since the problem is feasible for values $T_i \geq T^*$, and infeasible for values $T_i < T^*$, a bisection algorithm can be used to find the minimal time T^* needed to perform the desired motion. Before applying the control signals to the robot, they have to be discretized, which means that an accuracy of T^* up to the sampling time is sufficient[1]. As a result, the bisection algorithm always converges in a limited number of steps, yielding a solution to the original, non-convex, problem (3).

IV. Reinforcement learning

Finding time-optimal control signals using RL cannot be tackled directly as for the model-based techniques, since the environment is unknown to the agent. A common approach [16], [17] is to use negative rewards. The agent is then urged to learn a policy that minimizes the motion time in order to avoid accumulating negative rewards over the period of interaction. Based on this concept, a modified Natural Actor-Critic algorithm and its implementation to the serve operation are presented in this section.

A. Modified Natural Actor-Critic algorithm

In general, an RL problem can be represented by states $x \in \mathbf{X}$ of the environment, actions $u \in \mathbf{U}$ of the agent, and rewards $r \in \mathbb{R}$. At each time step k, the agent draws an action u_k from a stochastic policy $\pi(x_k) = p(u_k|x_k)$, and applies it to its environment to cause a transition from x_k to x_{k+1}. A reward $r_k = \mathcal{R}(x_k, u_k)$ is also calculated, evaluating the quality of applying u_k in x_k, and the goal will be to use these measurements to find the best policy π, which maximizes the accumulated future rewards

$$J(\pi, x_0) = \mathrm{E}\left\{ \sum_{k=0}^{\infty} \gamma^k r_k \middle| x_0, \pi \right\}. \tag{4}$$

In this equation, x_0 is a designated initial state and $\gamma \in (0, 1)$ the discounting factor. To run the learning algorithm, the state-value function

$$V^{\pi}(x) = \mathrm{E}\left\{ \sum_{k=0}^{\infty} \gamma^k r_k \middle| x_0 = x, \pi \right\} \tag{5}$$

is also needed, as is the state-action value function [18]

$$Q^{\pi}(x, u) = \mathrm{E}\left\{ \sum_{k=0}^{\infty} \gamma^k r_k \middle| x_0 = x, u_0 = u, \pi \right\}. \tag{6}$$

These value functions are used to evaluate the current policy π and to assist in improving the policy. Since there is no model, they have to be estimated using the measured samples. Batches of transitions (x_k, u_k, r_k, x_{k+1}) collected during an episode of exploration can for example be used to estimate the state-action value function $Q^{\pi}(x_k, u_k)$. To do so, the recursive form

$$Q^{\pi}(x_k, u_k) = \mathrm{E}\left\{ \mathcal{R}(x_k, u_k) + \gamma V^{\pi}(x'_k) \middle| \pi \right\} \tag{7}$$

is used to yield a set of linear equations from which Q^{π} is estimated by the least-squares temporal difference for Q-functions (LSTD-Q).

[1]Due to the discretization with sampling time T_s, many solutions typically exist for the lowest n satisfying $nT_s \geq T^*$, since nT_s can be up to T_s larger than T^*. Once nT_s is found, the optimization is then calculated one last time for $T = nT_s$, and a cost function promoting a smooth solution is added again. As long as this cost is convex, the resulting problem then remains convex, and finding a solution remains straightforward.

In this research, the NAC method [9] is used, which is an actor-critic method that employs the natural policy gradient to update the actor $\pi_\vartheta(x,u) = p(u|x,\vartheta)$, parameterized by ϑ. The natural policy gradient w is estimated by the critic, who also estimates a linear approximation $\phi(x)^T v$ for the state value function V^π, with basis function $\phi(x)$ and parameter vector v. By exploiting the compatible approximation [19] and (7), both w and v can be obtained by LSTD-Q(λ), which is a modified version of LSTD-Q using eligibility trace $z(x,u)$ with a trace discount factor $\lambda \in [0,1]$. Updates to the actor's parameters are performed once the gradient estimate is accurate enough, which happens once the angle between two consecutive estimates is below a threshold ϵ. Then parts of the accumulated experience are forgotten using a forgetting factor $\beta \in [0,1]$, and the entire procedure is repeated.

Since the robot only allows episodic interaction, an off-line version of NAC is needed. Episodic NAC (eNAC) as proposed in [9] was implemented, but no satisfactory learning performance was obtained. Our modified NAC (mNAC) algorithm is therefore implemented, combining aspects of NAC and eNAC [20], and a summary of this algorithm is given in Algorithm 1.

Algorithm 1 Modified Natural Actor-Critic (mNAC)

```
Input: Parameterized policy $\pi_\vartheta$ with initial parameters $\vartheta_0$, policy derivative $\nabla_\vartheta \log \pi_\vartheta$, basis function $\phi(x)$ for the parameterized $V^\pi$, and learning parameters $\alpha$, $\gamma$, $\lambda$, $\beta$
Initialize: initial state $x_0$, statistics $A_0 = \delta I$, $b_0 = \mathbf{0}$, and eligibility trace $z_0 = \mathbf{0}$
for Episode $e = 1, 2, 3, \dots$ do
    Collect samples
    $(x_k, u_k, r_k, x_{k+1})$ for $k = 1, 2, \dots, N$
    Critic Evaluation with LSTD-Q($\lambda$):
        for $k = 0, 1, 2, 3, \dots, N-1$ do
            Update basis functions
            $\hat{\phi}_k = [\nabla_\vartheta \pi_\vartheta(x_k, u_k)^T \ \phi(x_k)^T]^T$, $\tilde{\phi}_k = [0^T \ \phi(x_{k+1})^T]^T$
            Update eligibility
            $z_e \leftarrow \lambda z_e + \hat{\phi}_k$
            Update statistics
            $A_e \leftarrow A_e + z_e(\hat{\phi}_k - \gamma\tilde{\phi}_k)^T$, $b_e \leftarrow b_e + z_e r_k$
        end for
        Update critic parameters
        $[w_e^T \ v_e^T]^T = A_e^{-1} b_e$
    EndCritic
    Actor Update: If $\angle(w_e, w_{e-1}) \leq \epsilon$
        Update policy parameter
        $\vartheta_{e+1} = \vartheta_e + \alpha w_e$
        Forget statistics
        $z_{e+1} = \beta z_e$, $A_{e+1} = \beta A_e + (1-\beta) A_0$, $b_{e+1} = \beta b_e$
    EndActor
end for
```

B. mNAC applied to serve operation of badminton robot

To apply mNAC on the badminton robot, the states of the system, the allowable actions, and the reward function for the serve operation need to be defined.

The state vector is defined by $x(k) = (q(k) \ \dot{q}(k))^T$, where $q \in \left[\frac{-\pi}{2}, \frac{\pi}{4}\right]$ rad and is considered limited to $\dot{q} \in [-5\pi, 5\pi]$ rad/s, based on the geometry of the robot. Please note that the state x defined here is the state perceived by the agent, not the state vector for the state space model in (3d) and (3e). As described in Section II, the initial state is $x_0 = (-\pi/4 \ \ 0)^T$ and the target state $\chi = x(T) = (q^\chi \ \dot{q}^\chi)^T = (0 \ \ 3)^T$. During the tests, the robot is always reset to its initial position x_0 once the target state is reached. For a practical and safe implementation, this also happens if the robot enters either of two buffering regions defined to restrict the state space and prevent mechanical damage. The set of absorbing states $\mathbf{\Omega}$ is thus defined as

$$\mathbf{\Omega} = \left\{ x = \chi \text{ or } q \in \left[-\frac{\pi}{2}, -\frac{3\pi}{8}\right] \text{ or } q \in \left[\frac{\pi}{8}, \frac{\pi}{4}\right] \right\}. \quad (8)$$

Due to the resetting, each episode then starts from x_0 and ends when either an absorbing state is reached or when the maximum allowed time of 3 s has passed.

The goal for the agent is to learn a policy that moves the robot from x_0 to χ while minimizing T. To do so, it is allowed to take actions $u \in \mathbb{R}$. No bounds are thus imposed on the current demanded by the motor, as these are considered part of the environment and should be learned through interaction.

Now that the states and actions are defined, a reward function has to be chosen that allows the agent to find the motions with minimal T while reaching χ. We use a quadratic function of the states and actions

$$r_k = -(\chi - x_k)^T Q (\chi - x_k) - R u_k^2, \quad (9)$$

where $Q = \operatorname{diag}([1 \ \ 10])$ and $R = 0.2$. Penalizing the differences between the current state and the target, sufficient information is given to the agent about the desired state transitions to reach the target. At the same time, the negative rewards implicitly specify the objective of minimizing T.

In this paper, a parameterized, stochastic policy in the form of a normal Gaussian distribution $\pi(u|x) = \mathcal{N}(\psi(x)^T\vartheta, \sigma)$ is used, with $\sigma \in [0.02, 0.0667]$ selected by the updating rule suggested in [9]. The basis functions of the policy $\psi(x)$ and the value function $\phi(x)$ are both radial basis functions (RBF). The sampling time used for RL is $h = 0.01$ s, the learning rate $\alpha = 0.002$, the discount factor $\gamma = 0.95$, the updating threshold $\epsilon = 0.15$, the forgetting factor $\beta = 0.65$, and the eligibility discount factor $\lambda = 0.85$.

V. Experimental validation

A. Model-based results

For the model-based controller, the signals found by solving optimization problem (3) can directly be applied to the hit motor. The resulting motion is shown in Figure 5, where the demanded current is shown, as well as the racket's angle and angular velocity. As can be seen from the figure, a bang-bang like current is demanded, with the current first equal to the maximum allowable value, with the racket accelerating rapidly to a velocity higher than the desired hit velocity. Afterwards, the current equals the minimum allowable value to rapidly decelerate the racket, such that it arrives at the target with the demanded velocity after a period of T =0.21 s.

B. Model-free results

For the reinforcement learning algorithm, the method described in Section IV is used to learn while interacting with the robot. The evolution of the reward per episode over the course of this learning process is shown in Figure 6.

As seen, convergence is obtained after about 10 min, which is equivalent to 200 episodes, after which the performance does

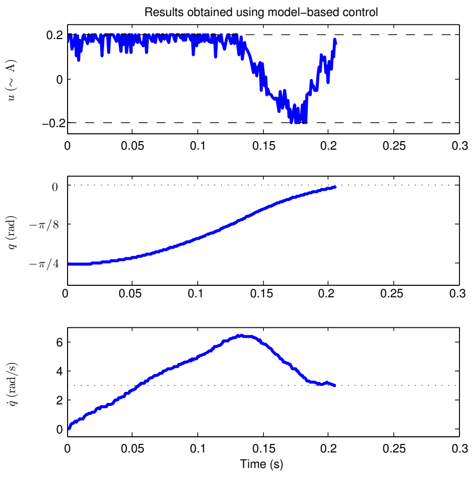

Figure 5. The hit motion obtained with the model-based controller.

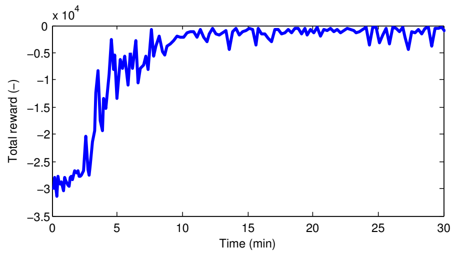

Figure 6. Evolution of the total reward per episode during learning process.

not alter significantly. The resulting hit motion obtained with the final learnt policy is shown in Figure 7. As can be seen, the velocity reaches the target velocity quickly, and then stays more or less constant while the racket moves towards its target position. As a result, the racket reaches the desired hit position with the desired velocity, after a period of $T = 0.28$ s.

C. Discussion

Comparing the results, it can be seen that the velocity increases to values higher than the desired end velocity for the model-based controller, but not for the policy learnt with reinforcement learning. To analyse this, Figure 8 shows the rewards (9) for both, and it can be seen that the reward for the model-based results is significantly worse than for those obtained using reinforcement learning, with summed total rewards of -800 and -300 respectively. This is due to the large weighting on the velocity differences, causing a large negative reward from $t = 0.1$ s to 0.2 s for the model-based controller, due to the velocity being higher than the desired hit velocity. From the point of view of maximizing the reward, the reinforcement learning algorithm has thus achieved a good result. Since the required motion time is longer than for the model-based controller however, the maximum of the reward function does not correspond to the true time-optimal motion.

Another difference is the level of actuation: bang-bang like control signals are found with the model-based controller, while less aggressive results are found using reinforcement learning. This is also illustrated in Figures 9 and 10, where the control policies are shown. Only values of $+0.2$ or -0.2 are selected by the model-based controller, and the overall map is very simple, corresponding to a bang-bang profile with only a single switch [2]. The reinforcement learning policy however generates near maximal actuation only in the beginning of the motions, while intermediate values occur afterwards. This is partially due to the earlier observed phenomenon, where the velocity does not increase beyond the target velocity. Another reason for this however is the term in the reward function penalizing values of $u \neq 0$. This term tries to reduce the overall control effort, which is undesirable for time-optimality, but it is observed from experiments that it is necessary in order to achieve successful learning.

D. Next steps

To enforce time-optimality with reinforcement learning, several approaches can be taken:

- A modified reward function can be used, specifying a strict time optimality by equally penalizing all states except χ, which is essentially the same as applying an $\mathcal{L}_0$ norm instead of the current $\mathcal{L}_2$ norm. However, slow learning might result, due to the less informative rewards compared to the current setting.
- An alternative could be to look for heuristic terms that can be added to the reward function, yielding a performance similar to using the $\mathcal{L}_0$ norm, but avoiding the issues with a slow convergence speed.
- A two-step learning strategy may be helpful if the reward is altered, avoiding issues with slow learning by starting from an initial policy as found in this paper.

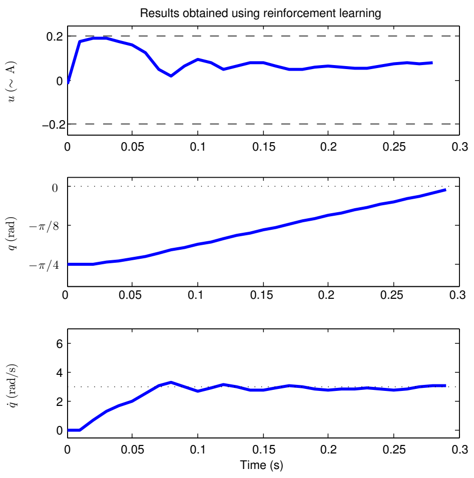

Figure 7. The hit motion obtained with the final policy of the RL algorithm.

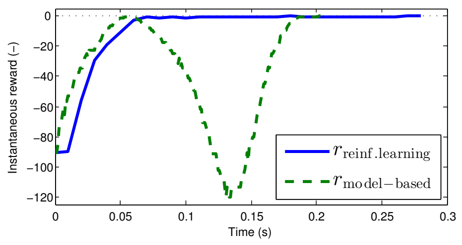

Figure 8. The immediate rewards r_k obtained with both controllers.

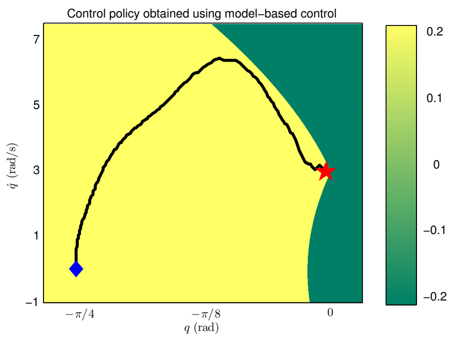

Figure 9. The equivalent policy of the model-based controller, shown for simplicity for a friction- and resonance-free double integrator. Also indicated as a solid line is the state trajectory shown in Figure 5.

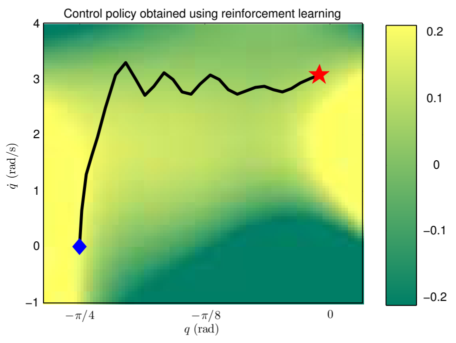

Figure 10. The final policy obtained with the RL algorithm. Also indicated as a solid line is the state trajectory shown in Figure 7.

- A different exploration strategy can be considered. Currently only the states near x_0 and near the trajectory in Figure 10 are explored. If states with larger velocities can be explored more thoroughly, a policy similar to the bang-bang-like control may be learned.
- Finally, since currently the control actions can assume any value in $(-0.2 \quad 0.2)$, the search space could be reduced by enforcing saturated actions, or actively promoting them. The difficulties observed without a penalty on u would then need be investigated however.

VI. Conclusions and Outlook

In this paper, time-optimal motion control is considered, and a model-free reinforcement learning technique is compared to a classical model-based technique. The model-free approach requires less information a priori, but does require a period of interaction during which the controller is exploring and learning the appropriate actions. When reliable models are available, it is therefore generally better to use them and avoid this learning period, but when this is not the case, model-free methods can be used to automatically learn a good controller.

For the task of controlling the hit motion of a badminton robot, a slightly longer motion time is obtained with the model-free approach than with the model-based one. When the total rewards are compared however, much better values are realized by the model-free controller. It can thus be concluded that these model-free controllers are converging to solutions different from the time-optimal ones, and that the reward function penalizing state differences, although a common choice in the literature, does not fully correspond to time-optimal motions.

Future work will focus on adapting the model-free techniques to more closely yield time-optimal motions. Several topics will be investigated, such as alterations to the reward function, different exploration strategies, promoting saturated controls and multi-step learning techniques.

References

[1] L. Zadeh and B. Whalen, “On optimal control and linear programming,” *Automatic Control, IRE Transactions on*, vol. 7, no. 4, pp. 45 – 46, jul 1962.

[2] M. L. Workman and G. F. Franklin, “Implementation of adaptive proximate time-optimal controllers,” in *American Control Conference, 1988*, june 1988, pp. 1629 –1635.

[3] P. Grieder and M. Morari, “Complexity reduction of receding horizon control,” in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 3, dec. 2003, pp. 3179 – 3190 Vol.3.

[4] P. Janssens, G. Pipeleers, and J. Swevers, “Model-free iterative learning of time-optimal point-to-point motions for lti systems,” in *50th IEEE Conference on Decision and Control and European Control Conference*, Orlando, Florida, December 2011.

[5] N. Sakagami and S. Kawamura, “Time optimal control for underwater robot manipulators based on iterative learning control and time-scale transformation,” in *OCEANS 2003. Proceedings*, vol. 3, sept. 2003, pp. 1180 – 1186 Vol.3.

[6] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[7] V. R. Konda and J. N. Tsitsiklis, “Actor-critic algorithms,” *Siam Journal on Control and Optimization*, pp. 1008–1014, 2001.

[8] S. Amari and S. C. Douglas, “Why natural gradient?” in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, vol. 2, 1998, pp. 1213–1216 vol.2.

[9] J. Peters and S. Schaal, “Natural actor-critic,” *Neurocomputing*, vol. 71, no. 7-9, pp. 1180–1190, 2008.

[10] A. El-Fakdi, M. Carreras, and E. Galceran, “Two steps natural actor critic learning for underwater cable tracking,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 2267–2272.

[11] J. Peters, S. Vijayakumar, and S. Schaal, “Reinforcement learning for humanoid robotics,” in *Proceedings of the Third IEEE-RAS International Conference on Humanoid Robots*, 2003.

[12] B. Kim, J. Park, S. Park, and S. Kang, “Impedance learning for robotic contact tasks using natural actor-critic algorithm,” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 40, no. 2, pp. 433–443, 2010.

[13] J. Stoev, S. Gillijns, A. Bartic, and W. Symens, “Badminton playing robot - a multidisciplinary test case in mechatronics,” in *5th IFAC Symposium on Mechatronic Systems*, Cambridge, Massachusetts, Sept. 2010.

[14] Triphase rapid-prototyping-platform: http://www.triphase.be/3pexpress/rapid-prototyping-platform.

[15] R. Pintelon and J. Schoukens, *System Identification: A Frequency Domain Approach.* John Wiley & Sons, 2012.

[16] I. Grondman, M. Vaandrager, L. Buşoniu, R. Babuška, and E. Schuitema, “Efficient model learning methods for actor-critic control,” *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 2011.

[17] X. Xu, H.-g. He, and D. Hu, “Efficient reinforcement learning using recursive least-squares methods,” 2002.

[18] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. MIT Press, 1998.

[19] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in *NIPS’99*, 1999, pp. 1057–1063.

[20] M.-H. Liu, “Service control of a badminton robot using actor-critic reinforcement learning,” Master’s thesis, Delft University of Technology, 2013.