Fitted Q-function Control Methodology based on Takagi-Sugeno Systems

Henry Díaz*, Leopoldo Armesto†, Antonio Sala‡

***Abstract*—This paper presents a combined identification/Q-function fitting methodology, which involves identification of a Takagi-Sugeno model, computation of (sub)optimal controllers from Linear Matrix Inequalities, and subsequent data-based fitting of the Q-function via monotonic optimisation. The LMI-based initialisation provides a conservative solution but it is a sensible starting point to avoid convergence/local-minima issues in raw data-based fitted Q-iteration or Bellman residual minimisation. An inverted-pendulum experimental case study illustrates the approach.**

***Index Terms*—Reinforcement learning, adaptive dynamic programming, fitted Q-function, Takagi-Sugeno, LMI.**

I. Introduction

Optimal controllers minimising an infinite-time cost index are of interest in many fields. Dynamic programming (DP) [1], [2] and reinforcement learning (RL) [3] are powerful paradigms to obtain them, used in many applications [4], [5]. DP and RL pursue computing value functions $V(x)$, or action-value functions $Q(x,u)$ [3] in order to yield optimal controllers (also denoted as policies[1]).

Policy iteration (PI) and value iteration (VI) are widely-used techniques to iteratively compute such optimal value functions and associated policies [7], [8], [9]. Based on the principles of fixed-point theorem, PI and VI converge to the optimal value and policy under mild conditions ensuring a contraction mapping [10], [11]. However, these mild conditions are actually so only in systems with a finite number of states and control actions. In continuous-valued settings, some approximation is needed to map a maybe complex controller/value function; theb exact Bellman equation gets now converted to a Bellman *residual* [12] (also called Bellman *error* in [13]) *minimisation* problem. Such approximation may spoil the contractive nature of the iteration steps and, hence, convergence may be lost [14]. Only in some quite restrictive settings such as fuzzy Q-iteration [11] convergence can be guaranteed.

A way to avoid PI/VI divergence is trying to minimise the Bellman error via gradient descent [15], [16], [17] or other monotonic optimisation methods [18]. However, such methods may get caught on local minima if not properly initialised.

A. Sala and H. Díaz are with the Instituto Universitario de Automática e Informática Industrial, Universitat Politècnica de València,Valencia 46022, Spain (e-mail: asala@isa.upv.es; hendia@posgrado.upv.es)

L. Armesto is with the Instituto de Diseño y Fabricación, Universitat Politècnica de València,Valencia 46022, Spain (e-mail: larmesto@idf.upv.es)

[1]Another option to look for optimal controllers is the so-called "policy search" techniques, in which value functions are avoided and only its gradient is estimated with respect to some controller parameters. These techniques are out of the scope of this work, see [6] and references therein for ample detail.

In a model-based approach to optimal control, many nonlinear systems can be modelled as the so-called fuzzy Takagi-Sugeno (TS) systems [19], via the well-known sector-nonlinearity approach [20]. These TS models express the nonlinearity as a convex combination of linear systems. Some convex conditions on the vertex linear models can easily obtain quadratic value function bounds by solving the so-called linear matrix inequalities (LMIs). The LMIs were introduced by the seminal book [21] and exploited for (sub)optimal nonlinear control in many works, for instance [20], [22], [23], [24] and references therein. Nonlinear optimal control using linear-like techniques can be also approached via Jacobian linearisation at several points [25]; nevertheless, these developments are related to predictive control, and out of the scope of our proposal.

The objective of this paper is to propose a methodology for nonlinear optimal control applications bridging the DP/RL and the model-based LMI approaches: given a nonlinear system in Takagi-Sugeno form, the LMI solution and the fuzzy controller structures associated to them inspire a particular parametrisation of the Q-function so that fitting algorithms can be initialised with the LMI solution. Monotonic/gradient-descent setups from such an initial solution provide controllers with lower Bellman residual than the parameters they were initialised at, without the risk of divergence of traditional PI/VI (PI/VI can also be tested under the proposed parametrisation). Preliminary versions of some ideas in this respect appear in [26], [27]. In a somehow similar philosophy, the paper [28] also bridges control-theoretic virtual-reference tuning and Q-learning.

The structure of the paper is as follows: Section II introduces necessary preliminaries of the paper and states the problem. Section III describes our methodology proposal. Section IV discusses the proposals on Takagi-Sugeno model identification and guaranteed-cost LMI controllers. Section V proposes a fuzzy Q-function parametrisation arising from the LMIs and an improved one-step controller. The Q-function parametrisation is generalised in section VI. Section VII discusses the generic optimisation approach to temporal-difference error minimisation and presents a summary of previous ideas onto a methodology proposal. Section VIII evaluates such methodology both in a simulation case study and an experimental inverted pendulum setup. Some conclusions are given in Section IX.

II. Preliminaries and problem Statement

This paper will consider nonlinear discrete-time systems

$$x_{t+1} = f(x_t, u_t), \tag{1}$$

with $x_t \in \mathbb{X} \subset \mathbb{R}^{n_x}$ and $u_t \in \mathbb{U} \subset \mathbb{R}^{n_u}$ being the model validity and input constraint region, where n_x and n_u are the number of states and inputs, respectively. Without loss of generality, we will assume that $(x, u) = (0, 0)$ is an equilibrium point, i.e., $f(0, 0) = 0$. By assumption, the above system will be controlled by a state-feedback policy $u = \pi(x)$.

Let us define the value of a policy $\pi(x)$ as:

$$V^\pi(x) := \sum_{t=0}^{\infty} \gamma^t r(x_t, \pi(x_t)) \tag{2}$$

where $x_0, x_1, \ldots,$ is the trajectory of (1) under $u_t = \pi(x_t)$ with initial condition $x_0 = x$. In (2), $0 \leq \gamma \leq 1$ is a discount factor, and $r(\cdot, \cdot)$ is a function $r : \mathbb{R}^{n_x+n_u} \mapsto \mathbb{R}^+$ of state and input known as *immediate cost*.

The control objective will be driving the system towards the origin so, by, assumption, $r(0, 0) = 0$ and $r(x, u) > 0$ for any $(x, u) \neq (0, 0)$. $V^\pi(x)$ can be seen as the expected return (cost) if policy $\pi(x)$ were used at all times from initial condition x_0.

Optimal control problems for the above system are usually stated as obtaining an optimal policy $u = \pi^*(x)$ such that, given x, the following cost index is minimised:

$$\pi(x)^* := \arg\min_{\pi \in \Pi} V^\pi(x) \tag{3}$$

In general, the above problem is solved by searching for the optimal π in a suitable set of functions Π, usually a parameterised function approximator $\Pi := \{\phi(x, \theta) : \theta \in \Theta\}$ so the optimal θ^* is sought in a given parameter set Θ.

A. Dynamic programming and Q-function fitting

From (1) and (2), we can write the well-known Bellman equation,

$$V^\pi(x) = r(x, \pi(x)) + \gamma V^\pi(f(x, \pi(x))). \tag{4}$$

In addition to the value of a policy, the so-called *action-value* function $Q^\pi(x, u)$, also known as Q-function, is defined in [29] as the return for a given state and action if policy π where applied from the next instant onwards:

$$Q^\pi(x, u) := r(x, u) + \gamma V^\pi(x_+), \tag{5}$$

being $x_+ := f(x, u)$.

For a given policy $\pi(x)$, holds $V^\pi(x) = Q^\pi(x, \pi(x))$; thus, the Q-function fulfills the Bellman's equation:

$$Q^\pi(x, u) - (r(x, u) + \gamma Q^\pi(x_+, \pi(x_+))) = 0, \tag{6}$$

and the optimal Q-function fulfills:

$$Q^{\pi^*}(x, u) - \left(r(x, u) + \gamma \min_{u_+} Q^{\pi^*}(x_+, u_+) \right) = 0 \tag{7}$$

In the literature there are quite a few iterative algorithms to estimate the optimal policy, based on dynamic programming [30] and reinforcement learning [3], such as value iteration, policy iteration, actor-critic setups, etc. Most of them reduce to Riccati equations (or iterations converging to the Riccati solution) for the linear case in model (1), see [8]. Let us outline the basic ideas of some of these algorithms which will be later used and compared.

a) Policy iteration (PI): It can be proved that, given a suboptimal policy $\pi(x)$ and its action-value function $Q^\pi(x, u)$, we can define an improved policy given by

$$\hat{\pi}(x) := \arg\min_u Q^\pi(x, u), \tag{8}$$

Also we can prove that [31]:

$$Q^{\hat{\pi}}(x, \hat{\pi}(x)) \leq Q^\pi(x, \pi(x)) \tag{9}$$

The next step in the referred algorithms is obtaining an expression $Q^{\hat{\pi}}(x, \hat{\pi}(x))$, denoted as *policy evaluation*, understanding (6) as a system of equations that the Q-function must fulfil for all (state, input, successor state) triplets. The interleaved policy evaluation and policy improvement steps are denoted as *policy iteration* [8] algorithm.

If the set of states and control actions are finite, then a mere lookup-table implementation of $Q^\pi(x, u)$ can be used [32], [10]. In continuous-valued state and input spaces, function approximators $Q^\pi(x, u) \equiv Q(x, u, \theta^\pi)$ are needed. Usually, with $Q(x, u, \theta^\pi)$, equation (6) cannot be fulfilled and minimising the norm (integral over state+action space) of its (squared) left-hand side is denoted as Bellman residual minimisation (BRM) [12], [13]. As the parametrised version[2] of policy improvement (8) ends up being:

$$\hat{\pi}(x, \theta^\pi) := \arg\min_u Q(x, u, \theta^\pi) \tag{10}$$

the so-called *fitted* policy iteration is described in Algorithm 1: from an initial stabilising policy $\hat{\pi}(x, \theta_0)$, its BRM policy-evaluation (11) yields θ_1, so $\hat{\pi}(x, \theta_1)$ is obtained, and so on. The norm notation $\| \cdot \|$ indicates the integral of the square over $\mathbb{X} \times \mathbb{U}$.

b) Value iteration (VI): Another alternative to find an approximately optimal Q-function is iterating (12). The difference with (11) is that both Q-function and policy are using the parameter from previous iteration.

Fitted PI/VI are well-developed techniques; the reader is referred to [11], [33], [34], [12], [13] for further detail. However, it is well known, too, that these iterative approaches do not converge in a general case [14], and only very restrictive lookup-table-like parametrisations guarantee convergence [11].

When converged, both algorithms would end up in the iteration fixed point given by the following Bellman residual minimisation:

$$\theta^* := \arg\min_\theta \|Q(x, u, \theta) - r(x, u) - \gamma Q(x_+, \hat{\pi}(x_+, \theta), \theta)\|. \tag{13}$$

So, conceivably, it might be carried out with any technique, not necessarily fitted PI/VI. This option is not without problems, as

[2]Note that the policy in (10) has been intentionally expressed as a parametrised expression depending on θ^π. Actually, in many cases, such parametrisation is implicit, i.e., $\hat{\pi}$ is obtained as a numerical solution of the optimization problem at the right-hand side, instead of a closed-form expression in x and θ^π. The parametrised notation is of interest for later developments in this work.

the chosen algorithm and initialisation may influence convergence, or getting stuck in spurious local minima. These issues may arise even in the case when the ground-truth optimal value function can be parametrised by $Q(x,u,\theta)$, see section VIII-A for an example of such behaviour with a linear system.

If $Q(x,u,\theta) := \Phi(x,u)\theta$, i.e., the value function approximator is linear in parameters θ, then problems (11) and (12) can be easily stated as standard linear least squares optimisation. However, this will not hold, in general, with (13).

B. Problem statement

As above discussed, fitted PI/VI or generic minimisation (13) need a careful choice of approximator structure (regressors) and initial parameters; otherwise, the learning algorithm may not converge or, even if it does, a suboptimal or even non-stabilising controller can be obtained.

The objective of this paper is to propose a methodology, based on fuzzy optimal control for TS systems, to address the above issues in applications of dynamic programing or reinforcement learning.

In addition to this, it would be of interest in applications to choose a parametrisation of $Q(x,u,\theta)$ such that an explicit closed-form solution of $\hat{\pi}$ can be derived, to avoid the need of real-time optimisation or large memory requirements and, too, speeding up the computations.

Our proposal will use data-based Takagi-Sugeno identification, fuzzy LMI-based control, and BRM to achieve the mentioned goals.

III. Methodology Proposal

This paper will present a series of ingredients to build up a methodology proposal mixing model-based and data-based Q-function fitting approaches, in the line expressed in the above problem statement.

The said ingredients will be as follows:

1) Fuzzy TS modelling and identification, with the objective of building a model so that (sub)optimal fuzzy controllers can be build upon it.
2) The referred fuzzy controllers will be computed with the so-called *guaranteed cost* solutions in literature which, using Linear Matrix Inequalities (LMI) can provide an *upper* bound on the value function Q.
3) The above LMIs can only optimise over Q-function parametrisations in quadratic form. An improved controller will be found by solving one step of the Bellman equation.
4) A more general fuzzy parametrisation of the Q-function will be proposed in Section VI, and its optimisation via a generic (monotonic descent) optimization algorithm will be discussed. In this way, conservatism over LMI results can be reduced.

With all these tools, we can craft a methodology in which an initial model-based approach based on fuzzy-TS models can be a very good starting point for data-based fitted Q-function approaches: with good LMI-based initialisation, the second stage can avoid getting caught at spurious local minima and, also, if monotonic optimisation is used, the LMI result can be improved even in the case PI/VI were not convergent.

Remark: Instead of the actual norm in (11)–(13), minimisation of these Bellman residuals will be carried out using the finite sum over a set of points in a dataset $\mathcal{D}$ composed of triplets (x,u,x_+), being x_+ the successor state. The dataset can be obtained either by simulation or by experimentation. So we need exciting the system with an input sequence $\{u_0,u_1,\ldots,u_N\}$ and collect $\{x_0,\ldots,x_{N+1}\}$. The dataset will be arranged as :

$$\mathcal{D} := \{(x_0,u_0,x_1),(x_1,u_1,x_2),\ldots,(x_N,u_N,x_{N+1})\} \quad (14)$$

Experiment design in such situation may be an important issue, as the relevant region of the state and action space should be well explored. Nevertheless, these issues are out of scope of the present work. Note that the collected data do not need to be generated with any of the policies involved in Algorithm 1, i.e., the proposal is an *off*-policy learner [3].

Algorithm 1 Approximate (fitted) policy/value iteration

1: Set an initial parameter θ_0.
2: Using the policy-improvement definition (10), solve one of the optimisation problems*:
[policy iteration]
$$\theta_{i+1} := \arg\min_{\tilde{\theta}} \|Q(x,u,\tilde{\theta}) - r(x,u) - \gamma Q(x_+,\hat{\pi}(x_+,\theta_i),\tilde{\theta})\| \quad (11)$$
The initial parameter in the policy iteration case must yield a stabilising policy $\hat{\pi}(x,\theta_0)$, ensuring that $Q(x,u,\theta_1) \geq 0$ for all $(x,u) \in \mathbb{D}$, see [31].
[value iteration]
$$\theta_{i+1} := \arg\min_{\tilde{\theta}} \|Q(x,u,\tilde{\theta}) - r(x,u) - \gamma Q(x_+,\hat{\pi}(x_+,\theta_i),\theta_i)\| \quad (12)$$
3: If $\|\theta_{i+1} - \theta_i\| \geq \varepsilon$, set $i = i+1$ and go to step 2; otherwise STOP.

*Note: in exact policy/value iteration algorithms [3], [11] the minimum achieved norm would be zero. Note also, that the policy has been expressed as the parametrised expression (10), even if such parametrisation may be implicit, see footnote 2.

IV. TS models and guaranteed cost controllers

This section will review prior results which will be part of our proposed methodology for engineering applications of fitted Q-function algorithms.

A. Takagi-Sugeno modelling and identification

If $f(x,u)$ in (1) can be expressed as $f(x,u) = h(x) + g(x)u$, and $h(x)$ has continuous first derivatives, then the nonlinear system can be **exactly** modelled using sector nonlinearity based on Takagi-Sugeno fuzzy models [20], [35]:

$$x_{t+1} = \sum_{i=1}^{\rho} \mu_i(x_t)(A_i x_t + B_i u_t) = A_{[\mu]} x_t + B_{[\mu]} u_t \quad (15)$$

being $\mu(x_t) := \{\mu_1(x_t),\ldots,\mu_\rho(x_t)\}$ a set of $\rho = 2^p$ membership functions with p nonlinearities, where

$$\sum_{i=1}^{\rho} \mu_i(x_t) = 1, \quad 0 \leq \mu_i(x_t) \leq 1 \quad (16)$$

and the notation $A_{[\mu]} := \sum_{i=1}^{\rho} \mu_i(x_t) A_i$, and similarly for $B_{[\mu]}$ has been introduced for compactness.

The above technique is a model-based one, however in the learning approach considered in this manuscript, we pursue a data-based approach. Hence, the TS models will be identified from experiments.

We will assume that a preliminary theoretical model exists for the process under control, so that the membership functions can be extracted from it. If such memberships are known, then the dataset $\mathcal{D}$ can be used to identify the vertices of the TS model, because (15) can be written as:

$$x_{t+1} = \begin{pmatrix} A_1 & B_1 & \dots & A_\rho & B_\rho \end{pmatrix} \cdot \begin{pmatrix} \mu_1(x_t)x_t \\ \mu_1(x_t)u_t \\ \vdots \\ \mu_\rho(x_t)x_t \\ \mu_\rho(x_t)u_t \end{pmatrix} \tag{17}$$

Now, if we form matrices:

$$\mathcal{X} := \begin{pmatrix} x_1 & x_2 \dots & x_{N+1} \end{pmatrix}_{n_x \times N} \tag{18}$$

$$\Gamma := \begin{pmatrix} \mu_1(x_0)x_0 & \dots & \mu_1(x_N)x_N \\ \mu_1(x_0)u_0 & \dots & \mu_1(x_N)u_N \\ & \vdots & \\ \mu_\rho(x_0)x_0 & \dots & \mu_\rho(x_N)x_N \\ \mu_\rho(x_0)u_0 & \dots & \mu_\rho(x_N)u_N \end{pmatrix}_{(n_x+n_u)\rho \times N} \tag{19}$$

the least-squares estimate of the model matrices is:

$$\begin{pmatrix} \hat{A}_1 & \hat{B}_1 & \dots & \hat{A}_\rho & \hat{B}_\rho \end{pmatrix} = \Gamma^\dagger \mathcal{X} \tag{20}$$

where $\Gamma^\dagger$ denotes the Moore-Penrose pseudo-inverse.

In the case where no preliminary model exists and there is no prior insight on which membership functions are the involved, we have a pure black-box identification problem that might need clustering, non-linear model fitting, etc. These issues are, intentionally, out of the scope of this work; the reader is referred to [36], [37], [38], [39] and references therein for details.

B. Guaranteed-cost control

It is well known that a suboptimal control policy for a TS model and an upper bound of its value function can be numerically found in a very efficient way (convex optimisation) using LMIs [20]. The fact that these LMIs provide only an upper bound of the cost motivates that these algorithms are known in literature as *guaranteed-cost* design methods.

These methods, derived from the LQR techniques, require a quadratic cost index:

$$r(x,u) := x^T H_x x + u^T H_u u \tag{21}$$

thus, this immediate cost structure will be assumed in the sequel.

The aim of LMI approaches is to find a positive-definite matrix X that overbounds the optimal *value* function, with guaranteed-cost:

$$\bar{V}^{\pi_{LMI}}(x) := x^T X^{-1} x \geq V^{\pi_{LMI}}(x) \geq V^{\pi^*}(x). \tag{22}$$

where the LMI policy $\pi_{LMI}(x_t)$ is fixed to the following structure, known as parallel-distributed compensator (PDC):

$$\pi_{LMI}(x) := F_{[\mu]} X^{-1} \cdot x := -\sum_{i=1}^{\rho} \mu_i(x) F_i X^{-1} x, \tag{23}$$

F_i, X being the decision-variable matrices to be found by solving a set of LMIs:

$$\mathcal{L}(i,j) \geq 0 \quad \forall j = i \tag{24}$$

$$\frac{2\mathcal{L}(i,i)}{\rho - 1} + \mathcal{L}(i,j) + \mathcal{L}(j,i) \geq 0 \quad \forall j > i \tag{25}$$

with,

$$\mathcal{L}(i,j) = \begin{pmatrix} X & X & F_j & (A_i P - B_i F_j) \\ X & H_x^{-1} & 0 & 0 \\ F_j & 0 & H_u^{-1} & 0 \\ (P A_i^T - F_j^T B_i^T) & 0 & 0 & \gamma^{-1} X \end{pmatrix}$$

where (25) is a relaxation to avoid double summation of the Lyapunov equations [40].

The proof of the above assertion appears in the Appendix. In fact, it is a straightforward adaptation of the well-known undiscounted $\gamma = 1$ expressions in literature. Furthermore, generalisations to non-quadratic cost bounds $\bar{V}(x) = x^T X_{[\mu]}^{-1} x$ can be also thought of [41], [42] but, for brevity, they are left to the reader. The goal of the TS framework will be providing a reasonable initialisation for further data-based learning improvements; actually, the final result will have membership-dependent value functions and non-PDC controllers even if the initial LMIs have membership-independent Lyapunov functions.

V. Improved TS guaranteed-cost controllers

Now we will present a so-called one-step controller which will improve the performance bound from LMIs based on the evaluation of the Q-function arising from the LMI solution.

Let us first discuss how the action-value function relates to Lyapunov functions in a linear discrete-time case. Consider $f(x_t, u_t)$ in (1) be:

$$x_{t+1} = A x_t + B u_t,$$

under a linear state feedback law $u_t = \pi(x_t) = -K^\pi x_t$. It is well known that the value function V^π is quadratic, in the form $V^\pi(x) = x^T P^\pi x$, where P^π is the solution to the Lyapunov equation associated to the feedback gain:

$$P^\pi - H_x - (K^\pi)^T H_u K^\pi - \gamma (A - BK^\pi)^T P^\pi (A - BK^\pi) = 0 \tag{26}$$

coming from the Bellman equation (4). From V^π, it can be proved that the Q-function in a linear case is, too, quadratic, because replacing the model and value functions in (5) we get:

$$\begin{aligned} Q^\pi(x,u) &= \begin{bmatrix} x \\ u \end{bmatrix}^T \left[\begin{pmatrix} H_x & 0 \\ 0 & H_u \end{pmatrix} + \gamma \begin{pmatrix} A^T \\ B^T \end{pmatrix} P^\pi \begin{pmatrix} A & B \end{pmatrix} \right] \begin{bmatrix} x \\ u \end{bmatrix} \\ &= \begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} H_x + \gamma A^T P^\pi A & \gamma A^T P^\pi B \\ \gamma B^T P^\pi A & \gamma B^T P^\pi B + H_u \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \\ &:= \begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} S_{xx} & S_{ux}^T \\ S_{ux} & S_{uu} \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \end{aligned} \tag{27}$$

As (27) is quadratic in u, straightforward differentiation obtains the explicit solution for the improved policy $\hat{\pi}$ in (8):

$$\hat{\pi}(x) = -S_{uu}^{-1} S_{ux} \cdot x := -K^{\hat{\pi}} x. \tag{28}$$

Alternatively, a matrix-based proof of the fact that $\hat{\pi}(x)$ improves over $\pi(x)$ appears in the Appendix.

Obviously, if $K^{\pi} = K^{\hat{\pi}}$ we have converged to the optimal controller. Thus, policy iteration evaluates repeatedly (26), (27) and (28) until convergence.

A. Fuzzy Q-function

Based on (27), we can extend to a TS case the above argumentation. Instead of a Lyapunov equation, in a TS case (22) provides an *upper* bound $\bar{V}^{\pi_{LMI}}(x)$ of the cost function of a fuzzy controller $\pi_{LMI}(x)$ given in (23) obtained from LMIs. Analogously to the linear case, we can assert an upper bound of the Q-function given by:

$$\bar{Q}^{\pi_{LMI}}(x,u) = \begin{bmatrix} x \\ u \end{bmatrix}^T S_{[\mu^2]} \begin{bmatrix} x \\ u \end{bmatrix} \tag{29}$$

being $S_{[\mu^2]}$ a matrix, depending on membership functions, which is partitioned analogously to (27) with:

$$S_{[\mu^2],xx} := H_x + \gamma \sum_{i=1}^{\rho} \mu_i(x) A_i^T \cdot X^{-1} \cdot \sum_{i=1}^{\rho} \mu_i(x) A_i \tag{30}$$

$$S_{[\mu^2],ux} := \gamma \sum_{i=0}^{\rho} \mu_i(x) B_i^T \cdot X^{-1} \cdot \sum_{i=1}^{\rho} \mu_i(x) A_i \tag{31}$$

$$S_{[\mu^2],uu} := \gamma \sum_{i=1}^{\rho} \mu_i(x) B_i^T \cdot X^{-1} \cdot \sum_{i=1}^{\rho} \mu_i(x) B_i + H_u \tag{32}$$

Note that elements of matrix $S_{[\mu^2]}$ are quadratic polynomials in the membership functions, which motivates the introduced subscript $_{[\mu^2]}$ notation.

B. One-step controller

Now, using (28) and replacing the $S_{[\mu^2]}$ matrix from (29), the resulting controller is a rational function in the membership functions (with degree-2 numerator and denominator terms):

$$\hat{\pi}_1(x) := -(S_{[\mu^2],uu})^{-1} S_{[\mu^2],ux} \cdot x. \tag{33}$$

Invertibility of the required matrices is guaranteed if $H_u > 0$.

We can prove that the worst-case performance (upper bound) of $\hat{\pi}_1(x)$ improves over $\pi_{LMI}(x)$. Indeed, an upper bound of the value function of $\hat{\pi}_1(x)$ can be computed as:

$$\bar{V}^{\hat{\pi}_1}(x) := x^T \cdot \left[S_{[\mu^2],xx} - S_{[\mu^2],ux}^T S_{[\mu^2],uu}^{-1} S_{[\mu^2],ux} \right] \cdot x \tag{34}$$

The proof of the above is straightforward, by substitution of (33) in (29). From (34), a proof that $\bar{V}^{\hat{\pi}_1}(x) \leq \bar{V}^{\pi_{LMI}}(x)$ can be found in the Appendix.

The feedback law (33) will be denoted as *one-step controller*. Thus, we have proof that the performance bound $\bar{V}^{\hat{\pi}_1}(x)$ is better than that of the PDC fuzzy controller (23), and it is directly obtained from the LMI decision variables and vertex model matrices.

Note that both the classic PDC and the one-step controllers are *shape-independent* [35], in the sense that, at design-time, no specific knowledge about the shape of the $\mu_i(x)$ functions is needed. Thus, our first proposal is to actually use (33) in applications, instead of (23) as its implementation is straightforward and it provides an improved performance bound.

VI. Q-function parametrisation for Takagi-Sugeno systems

Inspired in the polynomial-in-memberships expression (29) for the Q-function of the one-step controller, we will propose a generalisation of it in order to apply fitted Q-iteration and related algorithms to achieve further improvements.

In our proposal, we will set Q to be a polynomial-in-membership expression with arbitrary degree d (homogeneous, with no loss of generality). A monomial of degree d in variables $\mu \in \mathbb{R}^{\rho}$ will be represented as:

$$\mu^{\mathbf{a}} := \prod_{i=1}^{\rho} \mu_i^{a_i} \tag{35}$$

for any multi-index vector $\mathbf{a} := (a_1, \ldots, a_\rho)$ such that $|\mathbf{a}| := \sum_{i=1}^{\rho} a_i = d$ and each a_i is a non-negative integer. Dependence on x of the elements of μ has been omitted for notation compactness.

An homogeneous polynomial of degree d, denoted as $\Xi_{[\mu^d]}$, will be the sum of all these monomials multiplied by a real coefficient, i.e.:

$$\Xi_{[\mu^d]} := \sum_{|\mathbf{a}|=d} \mu^{\mathbf{a}} \xi_{\mathbf{a}} \tag{36}$$

where $\xi_{\mathbf{a}}$ conforms a d-dimensional coefficient tensor. Using a similar notation for monomials in x and u, we can express the value function bound (29) as:

$$\bar{Q}^{\pi_{LMI}}(x,u) = \Upsilon_{[\mu^2]}(x,u,\bar{q}_{LMI}) := \sum_{\substack{|\mathbf{a}|=2 \\ |\mathbf{b}+\mathbf{c}|=2}} \mu^{\mathbf{a}} x^{\mathbf{b}} u^{\mathbf{c}} q_{\mathbf{abc}} \tag{37}$$

for some values of the coefficients $q_{\mathbf{abc}}$ which can be obtained in a straightforward way from (30)–(32). Notation $\bar{q}_{LMI}$ denotes the vector containing all $q_{\mathbf{abc}}$ in any prescribed order.

Note that (37) is linear in the coefficients $q_{\mathbf{abc}}$, thus we can think on $\bar{q}_{LMI}$ as a parameter vector and Υ a linear-in-parameter approximator. Thus, the proposal at this stage is identifying the learnable parameter vector $\theta_0 \equiv \bar{q}_{LMI}$ thus initializing it at the LMI solution. With this initialisation, we may start an iterative Q-function optimisation procedure with the parametrisation $Q(x,u,\theta) \equiv \Upsilon_{[\mu^2]}(x,u,\theta)$, with a total of $\frac{1}{4}\rho(\rho+1)(n_x+n_u)(n_x+n_u+1)$ adjustable parameters.

In fact, the value function approximator can be generalised to a higher-degree homogeneous polynomial in the memberships:

$$\Upsilon_{[\mu^d]}(x,u,\theta) := \sum_{\substack{|\mathbf{a}|=d \\ |\mathbf{b}+\mathbf{c}|=2}} \mu^{\mathbf{a}} x^{\mathbf{b}} u^{\mathbf{c}} q_{\mathbf{abc}} \tag{38}$$

and the initial parameter value can be obtained from the coefficients resulting from multiplying the terms in (30)–(32) by $(\sum_i \mu_i)^{d-2}$ (or $(\sum_i \mu_i)^d$ in the case H_x and H_u).

As an example, in a system with $d = 3$, $\rho = 2$, $\Upsilon_{[\mu^d]}(x,u,\theta)$ would yield the following expression:

$$\Upsilon_{[\mu^3]}(x,u,\theta) = \begin{pmatrix} x^T & u^T \end{pmatrix} \left(\mu_1^3 S_{[\mu^3],1} + \mu_1^2\mu_2 S_{[\mu^3],2} + \mu_1\mu_2^2 S_{[\mu^3],3} + \mu_2^3 S_{[\mu^3],4}\right) \begin{pmatrix} x \\ u \end{pmatrix} \quad (39)$$

Partial derivatives of the memberships may be added to the parametrisation, too, [27]. Of course, a last generalisation would be adding to $\Upsilon_{[\mu^d]}$ any approximator (a neural network $NN(x,u,\theta_{NN})$, for instance), with its parameters initialised to zero output. In this way, LMI-based initialisation would still be possible. Even if, in theory, such option would offer greater flexibility, this NN approach will not be further pursued because it would, in a generic case, hinder obtaining explicit expressions of the controller (10) in the policy-improvement step. The lack of an explicit controller would slow the learning algorithms due to nested numerical optimisation and possibly cause convergence issues. Also, given that the scope of this paper is exploiting the partial knowledge of the nonlinearities (embedded in memberships) and LMI information for TS systems, we are, intentionally, leaving neural function approximators (or any other unstructured generic option) out of the scope of this work. For neural-network optimal-control applications of the concepts in this paper, the reader is referred to [32], [2], for instance. Kernel-based function approximators and Gaussian processes might also be used in reinforcement learning [43], [44].

VII. A GENERIC-OPTIMISATION APPROACH

An alternative interpretation of the fitted Q-function objective would be, as discussed in Section II, directly solving (13).

The above can be conceived as a generic optimisation problem: actually, Algorithm 1 can be interpreted as an iterative way of solving it, but any other iterative numerical optimisation algorithm in literature (gradient, Levenberg-Marquardt, Nelder-Mead, ...) may be equally used. Thus, under this interpretation, the relevant result of learning is a parameter value achieving good accuracy in the cost index in (13), independently of the the actual algorithm used to compute it.

The Q-function parametrisation proposed in the previous section allows to obtain an explicit expression for $\hat{\pi}$, henceforth of the Bellman residual, that avoids nested optimisation and eases the optimisation steps.

For instance, the gradient with respect to θ of the Bellman residual inside the norm in (13), denoted as $e(x,u,\theta)$, can be computed as follows:

$$\frac{\partial e}{\partial \theta} = \frac{\partial Q}{\partial \theta}(x,u,\theta) - \gamma \frac{\partial Q}{\partial \theta}(x_+,\hat{\pi},\theta) - \gamma \underbrace{\frac{\partial Q}{\partial u}(x_+,\hat{\pi},\theta)\frac{\partial \hat{\pi}}{\partial \theta}}_{\Omega} \quad (40)$$

On the other hand, if no explicit expression for $\hat{\pi}$ were available, as $\hat{\pi}$ is optimal, the value function must fulfil $\frac{\partial Q}{\partial u}(x_+,\hat{\pi},\theta) = 0$. Using a quadratic Taylor-series approximation of $Q(x_+,\hat{\pi}+\delta\hat{\pi},\theta+\delta\theta)$ and taking derivatives of it with respect to $\delta\hat{\pi}$, and equating to zero (optimality of $\delta\hat{\pi}$), after some manipulations, the derivative of the optimal policy required in (40) is:

$$\frac{\partial \hat{\pi}}{\partial \theta} = -\left(\frac{\partial^2 Q}{\partial u^2}\right)^{-1} \frac{\partial^2 Q}{\partial u \partial \theta} \quad (41)$$

In this way, we have completed the computation of the gradient of the Bellman Residual.

Note that, in policy-evaluation BRM approaches [12], i.e., the gradient involved in (11), we have $\Omega \equiv 0$; thus, the policy-evaluation gradient does not move the parameters in the actual gradient-descent direction implied in (40).

In addition to this, it is well-known that convergence (numerical stability) of generic optimisation tools may be a problem; indeed, this is often the case with PI and VI which require restrictive contraction-related conditions recalled in Section II-A. Contrarily, state-of-the-art optimisation techniques incorporate variable step-sizes, intermediate line search stages, etc. greatly improving its numerical stability [45]. In the examples in Section VIII, Matlab has been used to carry out the optimization, using the default quasi-Newton plus line-search algorithm of `fminunc`, as well as the simplex Nelder-Mead in `fminsearch`.

Nevertheless, when generic nonlinear optimisation is considered, it is well known that many of the algorithms in literature can easily get trapped in spurious solutions (local minima). A good initialisation is essential for succeeding in finding a good optimiser θ^* in this case; this is why our methodology proposes LMI-based initialisation[3].

A. Summary

Given the above issues, our proposal outlined in Section VII-A and developed in sections IV onwards can be summarised as:

1) From preliminary theoretical insight, identify the most relevant nonlinearities and build the associated membership functions.
2) Identify a TS model, as proposed in Section IV-A.
3) Obtain a guaranteed-cost LMI controller, Section IV-B.
4) Build the fuzzy Q-function (29) based on the LMI solution.
5) Initialise a monotonic optimisation algorithm with the above Q-function decision variables, and perform the optimisation of (13).

The reason of proposing the last monotonic optimisation step is to ensure that the result of our proposal will, at least, improve over the conservative LMI solution in the cost index of (13) without convergence problems. Note that, however, PI/VI (Algorithm 1) using the initial LMI solution might also be a sensible option. Furthermore, greedy search

[3]Obviously, global optimisers using, say, evolving algorithms or other population-based ideas [46], [37] might avoid the need of a good initialisation. However, their computational efficiency with large parameter sets is usually worse than the Quasi-Newton or Nelder-Mead options, so these options will not be further discussed.

TABLE I
INVERTED PENDULUM MODEL PARAMETERS

Model parameter	Symbol	Value	Units
Pendulum mass	M	2.40	kg
Center of gravity length	L	0.30	m
Friction Coefficient	β	0.35	$\text{kg} \cdot \text{m}^2 \cdot \text{s}^{-1}$
Pendulum inertia	I	0.272	$\text{kg} \cdot \text{m}^2$
Gravity	g	9.81	$\text{m} \cdot \text{s}^{-2}$

or population-based directed random search (genetic, swarm, etc.) techniques may be used instead of the suggested monotonic optimisation. However, these algorithms are much slower than derivative-based options, so they are not proposed as a first-choice option.

VIII. Case study: inverted pendulum

This section will present a case study using an inverted pendulum where the previous proposals will be illustrated. First, simulation studies with a theoretical model will be addressed. Later on, experimental identification and learning on an actual pendulum will be tested.

The objective of these examples will be showing the advantages of our proposed methodology (LMI initialisation + monotonic optimization) with respect to the other alternatives reviewed in previous sections. In particular, a simulation of an ideal linearised setup will show that VI/PI (Algorithm 1) may fail even if the function approximator includes the true value function (known to be quadratic in this case). In fact, even the BRM may fail if wrongly initialised, as shown there. Examples show, too, that with LMI initialisation, in case PI/VI are convergent they converge to basically the same solution as the monotonic BRM optimization.

A. Simulation examples

Consider an inverted pendulum model with 1 degree of freedom discretised with forward Euler approximation:

$$\alpha_{t+1} = \alpha_t + \delta\dot{\alpha}_t \tag{42}$$

$$\dot{\alpha}_{t+1} = \dot{\alpha}_t + \delta\frac{u_t - \beta\dot{\alpha}_t + MgL\sin(\alpha_t)}{I} \tag{43}$$

with $x_t = [\alpha_t \ \dot{\alpha}_t]^T$ where α_t is the beam angle and $\dot{\alpha}_t$ its velocity, M the mass, L length of the bar (and the position of the centre of gravity), β the friction coefficient, I the inertia and $\delta = 0.01$ s the sampling time; see Table I for numerical values of physical parameters.

Linearised case: Now, the linearisation of the inverted pendulum (42)-(43) system is considered. Specifically, we will linearise the equations around the vertically upward equilibrium position, α_t = 0. So, the resulting linearised system is

$$x_{t+1} = Ax_t + Bu_t \tag{44}$$

with:

$$A = \begin{bmatrix} 1 & 0.0100 \\ 0.2598 & 0.9871 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0.0368 \end{bmatrix} \tag{45}$$

The control problem will be stated using the quadratic cost (21) with:

$$H_x = \text{diag}([100,\ 10]),\ H_u = 1$$

and the discount factor will be set to $\gamma = 1$, so it is a standard LQR setup.

The optimal solution for the above-stated quadratic cost problem is well-known, as well as the fact that the true Q-function is quadratic in states and control inputs. The actual LQR solution is:

$$\begin{aligned} Q^*(x,u) &= 6350.8x_1^2 + 140.16x_2^2 \\ &+ 1272.8x_1x_2 + 42.88x_1u + 9.25x_2u + 1.165u^2 \end{aligned} \tag{46}$$

Thus, if we define the following linear-in-parameters approximator:

$$Q(x,u,\theta) := [\varphi(x) \quad \varphi(x)u \quad \varphi(x)u^2] \cdot \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} \tag{47}$$

for some unknown parameter vector $\theta := [\theta_1^T \ \theta_2^T \ \theta_3^T]$ and regressor vector $\varphi(x)$

$$\varphi(x) := [x_1^2 \quad x_1x_2 \quad x_2^2 \quad x_1 \quad x_2 \quad 1] \tag{48}$$

then, this choice of $Q(x,u,\theta)$ with 18 adjustable parameters can exactly fit the optimal LQR Q-function if the 6 parameters multiplying x_1^2, x_1x_2, x_2^2, x_1u, x_2u, u^2 are non-zero and the remaining 12 are zero.

The objective of this subsection is showing that even in this idealistic situation (linear process, true model of Q in the model set, i.e., the function approximator for Q includes the LQR solution), the PI/VI (Algorithm 1) or local optimisation approaches (Section VII) might fail.

In order to show these drawbacks, we have initialised the parameter vector θ to the following values:

$$\begin{aligned} \theta_1 &= \begin{bmatrix} 6342.3 & 163.7 & 1329.4 & -0.216 & 0 & 0 \end{bmatrix}^T \\ \theta_2 &= \begin{bmatrix} 0 & 0 & 0 & 42.3 & 9.27 & 0 \end{bmatrix}^T \\ \theta_3 &= \begin{bmatrix} 0.051 & 0.002 & -0.061 & 0 & 0 & 0.359 \end{bmatrix}^T \end{aligned}$$

With these parameters, the resulting controller from (10) was slightly nonlinear but, nevertheless, it was shown to be stabilising when starting in several simulations with random initial conditions with $x_1 \in [-\pi, \pi]$, $x_2 \in [-15, 15]$.

A dataset (14) was generated from an equally-spaced grid in the above position/speed region, as well as a similar grid in the interval for control action $u \in [-120, 120]$.

Applying Algorithm 1 with the above dataset, approximator and initial (stabilising) parameters, it can be shown that neither VI nor PI versions of the algorithm converge to a stabilising controller (details omitted for brevity).

Additionally, using `fminsearch` or `fminunc` functions from Matlab Optimisation Toolbox Version 7.6 (R2017a) to solve problem (13), the objective function gets stuck on a local minimum and the resulting controller is not stabilizing, even if the original one was.

In summary, PI and VI algorithms cannot provide a valid controller despite of having a linear system with an initial

stabilising controller and a Q-function approximator able to represent the true LQR solution. Of course, such divergence can occur in many other situations [14]. Also, generic optimisation may give useless results if not properly initialised.

Proposed fitted Q-function methodology: In the model (43), the nonlinearity is given by the sinusoidal function and the membership functions for the TS model are:

$$\mu_1(x) := \begin{cases} \frac{\sin\alpha/\alpha - \sin(\alpha_{max})/\alpha_{max}}{1-\sin(\alpha_{max})/\alpha_{max}} & \text{if } \alpha \neq 0 \\ 1 & \text{if } \alpha = 0 \end{cases}$$
$$\mu_2(x) := 1 - \mu_1(x)$$

with $\alpha_t \in [-\alpha_{max}, \alpha_{\max}]$ rad. and $\alpha_{max} = \pi$. Model vertex matrices are:

$$A_1 = \begin{bmatrix} 1 & \delta \\ \frac{\delta MgL}{I} & 1 - \frac{\beta\delta}{I} \end{bmatrix}, \qquad B_1 = \begin{bmatrix} 0 \\ \frac{\delta}{I} \end{bmatrix},$$
$$A_2 = \begin{bmatrix} 1 & \delta \\ 0 & 1 - \frac{\beta\delta}{I} \end{bmatrix}, \qquad B_2 = \begin{bmatrix} 0 \\ \frac{\delta}{I} \end{bmatrix}.$$

The LMI controller for the same problem as in the linearised case provides the following bound for the *value* function:

$$\bar{V}^{\pi_{LMI}}(x) = x^T \begin{bmatrix} 7598.37 & 588.28 \\ 588.28 & 130.05 \end{bmatrix} x \tag{49}$$

and a policy[4] from (23):

$$\hat{\pi}_{LMI}(x) = -\left[19.19972 \;\; 4.25971\right] x. \tag{50}$$

From the LMI output in (50), the resulting matrices for the one-step controller (33) are:

$$S_{[\mu^2],xx}(x) = \begin{bmatrix} 7698.37 + 305.65\mu_1(x) + & 656.69 + \\ +8.777\mu_1^2(x) & +34.88\mu_1(x) \\ 656.69 + 34.88\mu_1(x) & 149.1 \end{bmatrix}$$
$$S_{[\mu^2],ux}(x) = \begin{bmatrix} 21.63 + 1.24\mu_1(x) & 4.93 \end{bmatrix}$$
$$S_{[\mu^2],uu}(x) = \begin{bmatrix} 1.17 \end{bmatrix}$$

where μ_2 has been replaced by $1 - \mu_1$ so only μ_1 appears.

This provides an improved one-step controller yielding a fuzzy policy:

$$\hat{\pi}_1(x) = -\left[18.39977 + 1.05675\mu_1(x) \quad 4.19946\right] x$$

Figure 1 represents the ratio between the *actual* performance of the one-step and LMI controllers, evaluated by a simulation from a grid of different initial conditions. Note that the figure plots actual 300-step "measured" performance (2), and not the performance bounds $\bar{V}^{\pi_{LMI}}(x)$ or $\bar{V}^{\hat{\pi}_1}(x)$ in (34). Thus, the proposed one-step controller slightly improves performance up to 2% in some states over the LMI solution (50).

One-step controller will be further improved via fitted Q-function algorithms using a dataset $\mathcal{D}$, with $N = 1000$ samples equally spaced on the interval $\alpha_t \in [-\pi, \pi]$ rad, $\dot{\alpha}_t \in [-15, 15]$ rad/s and $u_t \in [-120, 120]$ N·m using the nonlinear model in (42)-(43).The fuzzy Q-function approximator (38) was set to $d = 2$, and initialised with (30)–(32).

[4]In this case, the particular model and cost index parametrisation results in a linear state-feedback controller, because the LMI solver outputs $F_1 = F_2$.

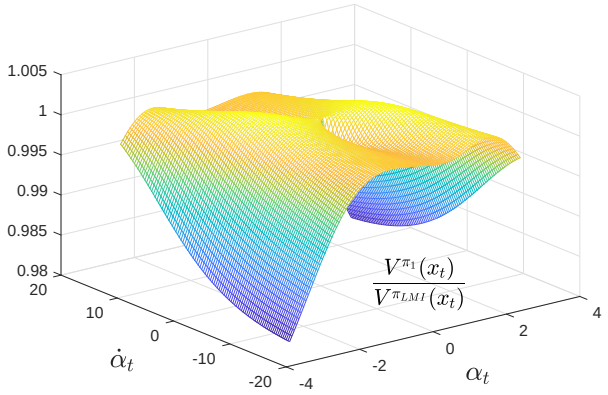

Fig. 1. Ratio of measured performance of one-step controller with respect to the LMI controller.

Then, using `fminunc`[5]optimisation of (13) the Q-function approximation converges to:

$$Q(x,u) \approx \begin{pmatrix} x \\ u \end{pmatrix}^T \left(S_{[\mu^2]}\right) \begin{pmatrix} x \\ u \end{pmatrix} \tag{51}$$

with:

$$S_{[\mu^2],xx} = \begin{bmatrix} 4736.7 - 6887.6\mu_1(x) + & 319.4 - 586.6\mu_1(x) + \\ 2089.5\mu_1(x)^2 & 208.2\mu_1(x)^2 \\ 319.4 - 586.6\mu_1(x) + & 104.8 - 121.6\mu_1(x) + \\ 208.2\mu_1(x)^2 & 81.6\mu_1(x)^2 \end{bmatrix}$$
$$S_{[\mu^2],ux} = \begin{bmatrix} 10.6 - 17.2\mu_1(x) + & 3.5 - 3.6\mu_1(x) + \\ 5.2\mu_1^2(x) & 2.3\mu_1(x)^2 \end{bmatrix}$$
$$S_{[\mu^2],uu} = \begin{bmatrix} 1.1 - 0.4\mu_1(x) + 0.3\mu_1(x)^2 \end{bmatrix}$$

and, as a result, the controller (8) yields the following rational-in-memberships controller:

$$\hat{\pi}_{\mu^2}(x_t) = -\begin{bmatrix} \frac{5.2657\mu_1^2(x_t) + 6.6635\mu_1(x_t) + 10.643}{0.09026\mu_1^2(x_t) - 0.048143\mu_1(x_t) + 1.12789} \\ \frac{2.6398\mu_1^2(x_t) - 1.3338\mu_1(x_t) + 3.46478}{0.09026\mu_1^2(x_t) - 0.048143\mu_1(x_t) + 1.12789} \end{bmatrix}^T \cdot x_t.$$

If we compare this learnt controller against the initial one, we can see that there is a performance improvement of around 15% in some states, as shown in Figure 2.

If the procedure is done with a Q-function approximator of degree $d = 3$, the result is practically identical in this example as the one with $d = 2$, so a degree increase does not seem worthwhile for this particular application.

As an alternative, using policy iteration Algorithm 1 from the same initial parameters (i.e., obtained with LMI and one-step controller) yielded basically the same solution, as discussed in our previous work [26]: LMI initialisation is a convenient option *per se*, whatever the later tuning steps are. However, our proposal has a *guaranteed* Bellman residual descent from an already sensible LMI solution, whereas PI does *not*.

Summarising, the proposed methodology for Q-function parametrisation and initialisation is able to achieve improved controllers with respect to the LMI solution in this process.

[5]According to Matlab's documentation, `fminunc` implements a quasi-Newton plus line-search monotonic descent algorithm.

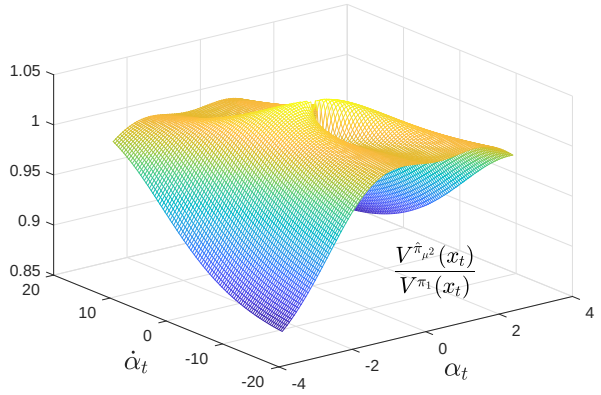

Fig. 2. *Value* function ratio of shape-dependent fitted Q-function controller with $d = 2$ (with respect to the one-step controller).

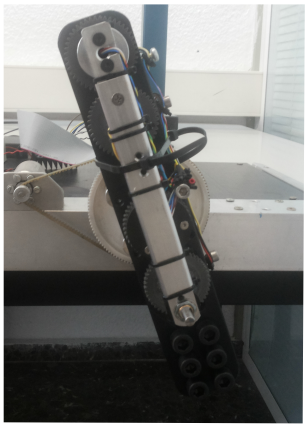

Fig. 3. Inverted Pendulum for experiments

B. Experimental evaluation

Now, we will apply our proposed methodology in Section VII-A, from scratch, to a dataset obtained from the 1 DoF pendulum in Figure 3, including identification steps.

The hardware is an inverted pendulum mechanism consisting of an electric motor, which actuates a link with a bar. A power drive is able to operate the motor in the range ± 12 V. A position sensor based on an Hall effect encoder is used for data acquisition. Control sampling period was set to $\delta = 10$ ms and the position sensor was operated at 1 ms sampling period. The controller was fed with filtered position/speed data (every 10 samples) coming from a Kalman filter for a double-integrator model. Dataset acquisition is performed in real-time using a NI myRIO-1900 embedded device by National Instruments and LabVIEW 2015. The learning algorithm phase and data processing were implemented in Matlab R2017a.

Also, as there was a significant actuator dead-zone due to Coulomb friction, such dead-zone was compensated as follows

$$u_a = u + u_{comp}$$
$$u_{comp} = \begin{cases} 0, & \text{if } |u| < k_1, \\ k_2 \cdot sign(u), & \text{otherwise} \end{cases}$$

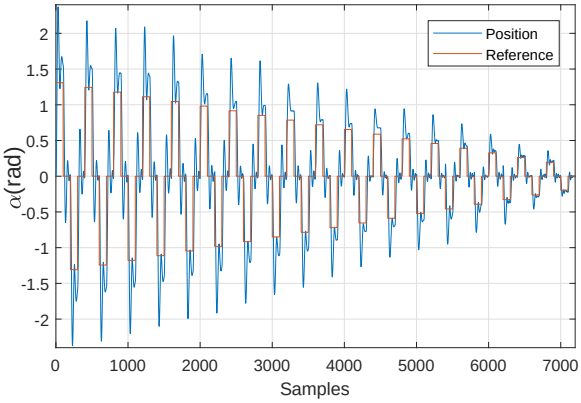

Fig. 4. Position dataset for identification and learning.

where k_1 and k_2 are 1% and 15% of input range respectively, and u is the output of the controllers to be designed whereas u_a is the actual voltage applied to the motor. Note that, as intuitively expected, there will be some chattering in control input when the state is close to the equilibrium. The chattering appears due to the combined action of disturbances on an unstable system and the sign function in the above compensation.

As the pendulum is unstable, we need an initial stabilising controller to be able to collect any significant amount of data. Thus, the PD controller $u = -60 * (\alpha - \alpha_{ref}) - 1 * \dot{\alpha}$ was implemented to gather the dataset.

A dataset of 7200 data points (72 seconds of experiment) has been collected using the above PD controller and a varying set-point signal. In addition to the control action of this PD, we added a Gaussian noise with standard deviation 5% of the actuator range to generate some extra excitation, as typically required in identification and learning techniques. The sample time T_s is 10 ms. The set-point and actual pendulum angle is shown in Figure 4.

Thus, according to the proposed methodology, we will obtain first an identified TS model, second and an initial model-based stabilizing control law using LMIs, for a later third stage of data based tuning.

c) Identification and model-based LMI controller: Applying the identification method proposed in Section IV-A the estimated matrices are:

$$A_1 = \begin{bmatrix} 1.0019 & 0.0097 \\ 0.3312 & 0.9718 \end{bmatrix} \qquad B_1 = \begin{bmatrix} 0.0001 \\ 0.0220 \end{bmatrix}$$
$$A_2 = \begin{bmatrix} 1.0002 & 0.0097 \\ -0.0156 & 0.9529 \end{bmatrix} \qquad B_2 = \begin{bmatrix} 0.0001 \\ 0.0131 \end{bmatrix}$$

The quadratic cost index was set with $H_x = \text{diag}([100,\ 2.5])$, $H_u = 0.15$ and $\gamma = 1$ for the discount factor. The LMIs provide the following bound for the *value* function and its associated state feedback policy:

$$\bar{V}^{\pi_{LMI}}(x) \leq x^T \begin{bmatrix} 4050.4 & 360.1 \\ 360.1 & 60.4 \end{bmatrix} x \qquad (52)$$
$$\pi_{LMI}(x) = -\begin{bmatrix} 41.1 & 5.8 \end{bmatrix} x$$

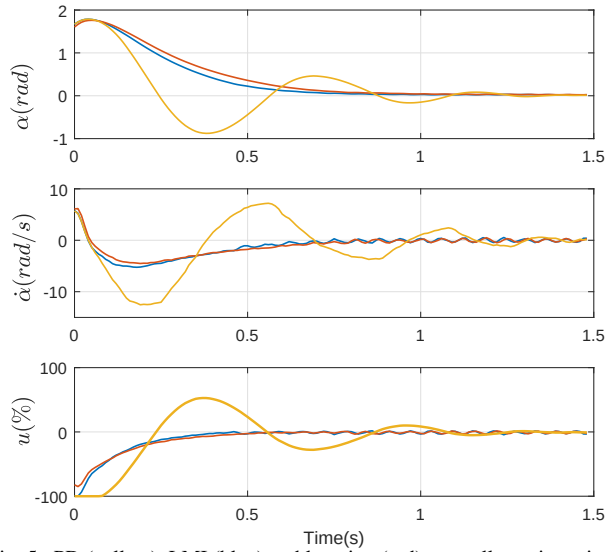

Fig. 5. PD (yellow), LMI (blue) and learning (red) controller trajectories from $x_0 = [1.6, 5.2]^T$. Recall that position and speed figures have been obtained from a 10x oversampling Kalman filter which averages measurement noise.

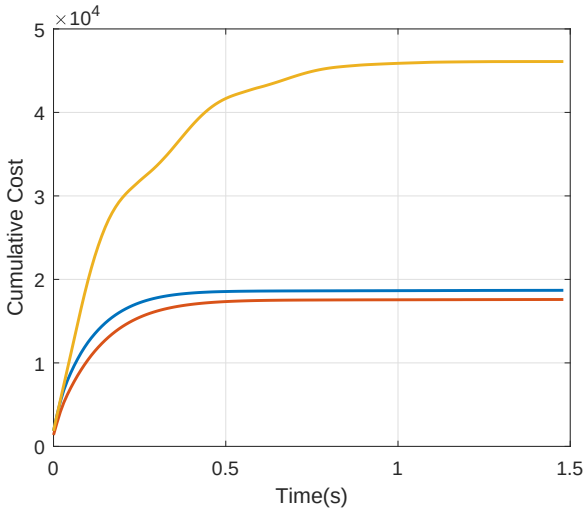

Fig. 6. Cost performance of PD (yellow), LMI (blue) and learning (red) controllers.

These computations with the identified TS model will be only used to initialise the function approximation for data-based fitted Q-function, to be discussed next.

d) Data-based fitted Q-function tuning: After these model-based initial stages, finally, a data-based optimisation with `fminsearch` (which implements variations of simplex Nelder-Mead monotonic descent algoritm [18]) is carried out (obviously `fminunc` may also be used, as done in the simulation examples, but we chose `fminsearch` to illustrate another option).

The said optimisation minimises the sum of the squares of the Bellman residual, i.e., (13), over the same experimental dataset used for identification (Fig. 4) arranged as in (14), so the second phase of the Q-function fitting is model-free. The linear-in-parameter fuzzy Q-function approximator was (37), initialised with the LMI solution.

The resulting controller using the Fuzzy fitted Q-function algorithm converges to the following controller (rational in the membership functions):

$$\hat{\pi}_{[\mu^2]}(x) = -\begin{bmatrix} \frac{-9.17\mu_1^2(x)+6.19\mu_1(x)+7.91}{-0.36\mu_1^2(x)+0.33\mu_1(x)+0.18} \\ \frac{-1.12\mu_1^2(x)+0.94\mu_1(x)+0.98}{-0.36\mu_1^2(x)+0.33\mu_1(x)+0.18} \end{bmatrix}^T \cdot x \qquad (53)$$

From the initial state $x_0 = [1.6, 5.2]^T$, Figure 5 show the position, speed trajectories and control input. For the sake of comparison, the trajectories for the initial PD, the LMI controller and the final learnt one (53) are all plotted.

In order to show the actual experimental cost index performance, Figure 6 depicts the accumulated quadratic cost for the three compared controllers.

IX. Conclusions

In this paper we have presented a fuzzy fitted Q-function methodology to obtain approximately optimal controllers based on Takagi-Sugeno systems.

The method first proposes to identify a fuzzy model based on collected data, incorporating partial model information via the memberships arising from sector-nonlinearity TS theoretical models. Second, a guaranteed cost controller based on LMIs can be designed for the identified system. Last, based on this controller, we can use it to initialize a fuzzy function approximator of the Q-function and apply Q-function fitting algorithms from the same dataset used in the identification phase. The Q-function fitting step can be carried out with standard policy/value iteration or, alternatively, if there are convergence problems, via generic (monotonic) optimisation algorithms. Examples illustrate that the methodology proposal (LMI initialisation plus monotonic optimization) is advantageous with respect to standard PI/VI (which may have convergence issues) or BRM (which may converge to a local minima if not properly initialised).

This methodology combines model-based optimal control (via identification) with data-based learning. We have shown that the proposed initialization (guaranteed-cost controller for the identified TS system) can be considered a good starting condition for most fitted Q-function algorithms. The paper presents results both in simulation and with real experimental data from an inverted pendulum.

Acknowledgements

The authors are grateful to the financial support of Spanish Ministry of Economy and European Union, grant DPI2016-81002-R (AEI/FEDER, UE) and PhD. grant SENESCYT from the Government of Ecuador.

References

[1] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA, USA: Athena Scientific, 2005.

[2] H. Zhang, D. Liu, Y. Luo, and D. Wang, *Adaptive dynamic programming for control: algorithms and stability*. Springer Science & Business Media, 2012.

[3] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1.

[4] C. Wei, Z. Zhang, W. Qiao, and L. Qu, "Reinforcement-learning-based intelligent maximum power point tracking control for wind energy conversion systems," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 10, pp. 6360–6370, Oct 2015.

[5] A. Pomprapa, S. Leonhardt, and B. J. Misgeld, "Optimal learning control of oxygen saturation using a policy iteration algorithm and a proof-of-concept in an interconnecting three-tank system," *Control Engineering Practice*, vol. 59, pp. 194–203, 2017.

[6] M. P. Deisenroth, G. Neumann, J. Peters *et al.*, "A survey on policy search for robotics," *Foundations and Trends® in Robotics*, vol. 2, no. 1–2, pp. 1–142, 2013.

[7] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *Circuits and Systems Magazine, IEEE*, vol. 9, no. 3, pp. 32–50, 2009.

[8] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, "Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers," *Control Systems, IEEE*, vol. 32, no. 6, pp. 76–105, 2012.

[9] F. L. Lewis and D. Liu, *Reinforcement learning and approximate dynamic programming for feedback control*. Hoboken, NJ, USA: Wiley, 2013.

[10] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. Hoboken, NJ, USA: Wiley, 2011.

[11] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst, *Reinforcement learning and dynamic programming using function approximators*. Boca Raton, FL, USA: CRC press, 2010.

[12] A. Antos, C. Szepesvári, and R. Munos, "Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path," *Machine Learning*, vol. 71, no. 1, pp. 89–129, 2008.

[13] R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora, "Fast gradient-descent methods for temporal-difference learning with linear function approximation," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 993–1000.

[14] M. Fairbank and E. Alonso, "The divergence of reinforcement learning algorithms with value-iteration and function approximation," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, June 2012, pp. 1–8.

[15] R. Munos, L. C. Baird, and A. W. Moore, "Gradient descent approaches to neural-net-based solutions of the hamilton-jacobi-bellman equation," in *Neural Networks, 1999. IJCNN'99. International Joint Conference on*, vol. 3. IEEE, 1999, pp. 2152–2157.

[16] L. C. Baird and A. W. Moore, "Gradient descent for general reinforcement learning," in *Advances in neural information processing systems*, 1999, pp. 968–974.

[17] M. Geist and O. Pietquin, "Algorithmic survey of parametric value function approximation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 6, pp. 845–867, 2013.

[18] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the nelder–mead simplex method in low dimensions," *SIAM Journal on optimization*, vol. 9, no. 1, pp. 112–147, 1998.

[19] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE transactions on systems, man, and cybernetics*, no. 1, pp. 116–132, 1985.

[20] K. Tanaka and H. O. Wang, *Fuzzy control systems design and analysis: a linear matrix inequality approach*. John Wiley & Sons, 2001.

[21] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear matrix inequalities in system and control theory*. SIAM, 1994.

[22] H.-N. Wu and K.-Y. Cai, "H 2 guaranteed cost fuzzy control for uncertain nonlinear systems via linear matrix inequalities," *Fuzzy sets and systems*, vol. 148, no. 3, pp. 411–429, 2004.

[23] C. Ariño, E. Pérez, and A. Sala, "Guaranteed cost control analysis and iterative design for constrained takagi–sugeno systems," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 8, pp. 1420–1427, 2010.

[24] T. M. Guerra, A. Sala, and K. Tanaka, "Fuzzy control turns 50: 10 years later," *Fuzzy sets and systems*, vol. 281, pp. 168–182, 2015.

[25] L. Armesto, V. Girbés, A. Sala, M. Zima, and V. Šmídl, "Duality-based nonlinear quadratic control: Application to mobile robot trajectory-following," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 4, pp. 1494–1504, 2015.

[26] H. Díaz, L. Armesto, and A. Sala, "Improvement of LMI controllers of takagi-sugeno models via Q-learning," *IFAC-PapersOnLine*, vol. 49, no. 5, pp. 67–72, 2016.

[27] ——, "Improving LMI controller for discrete nonlinear systems using policy iteration," in *System Theory, Control and Computing (ICSTCC), 2016 20th International Conference on*. IEEE, 2017, pp. 833–838.

[28] M.-B. Radac, R.-E. Precup, and R.-C. Roman, "Model-free control performance improvement using virtual reference feedback tuning and reinforcement q-learning," *International Journal of Systems Science*, vol. 48, no. 5, pp. 1071–1083, 2017.

[29] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, University of Cambridge England, 1989.

[30] R. Bellman, *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[31] D. Liu, Q. Wei, D. Wang, X. Yang, and H. Li, *Adaptive dynamic programming with applications in optimal control*. Springer, 2017.

[32] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Athena Scientific, Belmont,MA, 1996.

[33] M. Riedmiller, "Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method," in *European Conference on Machine Learning*. Springer, 2005, pp. 317–328.

[34] R. Munos and C. Szepesvári, "Finite-time bounds for fitted value iteration," *Journal of Machine Learning Research*, vol. 9, no. May, pp. 815–857, 2008.

[35] A. Sala, "On the conservativeness of fuzzy and fuzzy-polynomial control of nonlinear systems," *Annual Reviews in Control*, vol. 33, no. 1, pp. 48–58, 2009.

[36] H. Hellendoorn and D. Driankov, *Fuzzy model identification: selected approaches*. Springer, 1997.

[37] E. Lughofer, *Evolving fuzzy systems-methodologies, advanced concepts and applications*. Springer, 2011, vol. 53.

[38] M. Lovera, C. Novara, P. L. Dos Santos, and D. Rivera, "Guest editorial special issue on applied lpv modeling and identification," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 1, pp. 1–4, 2011.

[39] I. Škrjanc, "Evolving fuzzy-model-based design of experiments with supervised hierarchical clustering," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 4, pp. 861–871, 2015.

[40] H. D. Tuan, P. Apkarian, T. Narikiyo, and Y. Yamamoto, "Parameterized linear matrix inequality techniques in fuzzy control system design," *IEEE Transactions on fuzzy systems*, vol. 9, no. 2, pp. 324–332, 2001.

[41] R. Márquez, T. M. Guerra, A. Kruszewski, and M. Bernal, "Improvements on non-quadratic stabilization of takagi-sugeno models via line-integral lyapunov functions," *IFAC Proceedings Volumes*, vol. 46, no. 20, pp. 473–478, 2013.

[42] K. Tanaka, H. Ohtake, and H. O. Wang, "Guaranteed cost control of polynomial fuzzy systems via a sum of squares approach," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 561–567, 2009.

[43] X. Xu, C. Lian, L. Zuo, and H. He, "Kernel-based approximate dynamic programming for real-time online learning control: An experimental study," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 1, pp. 146–156, 2014.

[44] M. De Paula, L. O. Avila, C. Sanchez Reinoso, and G. G. Acosta, "Multimodal control in uncertain environments using reinforcement learning and gaussian processes," *Rev. Ib. Automatica Informatica Industrial*, vol. 12, no. 4, pp. 385–396, 2015.

[45] E. K. Chong and S. H. Zak, *An introduction to optimization*. John Wiley & Sons, 2013, vol. 76.

[46] M. Pelikan, D. E. Goldberg, and F. G. Lobo, "A survey of optimization by building and using probabilistic models," *Computational optimization and applications*, vol. 21, no. 1, pp. 5–20, 2002.

APPENDIX

The LMIs (24) and (25) come from replacing the Bellman equation (4) by an inequality

$$\bar{V}^{\pi}(x) \geq r(x, \pi(x)) + \gamma \bar{V}^{\pi}(x_+) \tag{54}$$

so that the value functions $\bar{V}$ that fulfil it are only upper bounds of the optimal cost, i.e., $\bar{V}(x) \geq V^{\pi}(x)$.

With the proposed expression for $\bar{V}$ and π_{LMI}, the inequality gets:

$$\begin{aligned} &x^T X^{-1} x - x^T H_x x - x^T X^{-1} F_{[\mu]}^T H_u F_{[\mu]} X^{-1} x \\ &-\gamma x^T (A_{[\mu]} + B_{[\mu]} F_{[\mu]} X^{-1})^T X^{-1} (A_{[\mu]} + B_{[\mu]} F_{[\mu]} X^{-1}) x \geq 0 \end{aligned} \tag{55}$$

The change of variable $X^{-1}x = \psi$ transforms the above to:

$$\begin{aligned}&\psi^T X\psi - \psi^T X H_x X\psi - \psi^T F_{[\mu]}^T H_u F_{[\mu]}\psi\\ &-\gamma\psi^T(A_{[\mu]}X + B_{[\mu]}F_{[\mu]})^T X^{-1}(A_{[\mu]}X + B_{[\mu]}F_{[\mu]})\psi \geq 0\end{aligned} \tag{56}$$

So, application of Schur complement and double-sum relaxation yields the stated LMIs conditions.

Consider a matrix

$$S := \begin{bmatrix} S_{xx} & S_{ux}^T \\ S_{ux} & S_{uu}\end{bmatrix} \tag{57}$$

and consider now the expression:

$$\Xi(S,\hat{K}) := \begin{pmatrix} I & \hat{K}^T\end{pmatrix}\begin{bmatrix} S_{xx} & S_{ux}^T \\ S_{ux} & S_{uu}\end{bmatrix}\begin{pmatrix} I \\ \hat{K}\end{pmatrix} \tag{58}$$

If we set $\hat{K} = -S_{uu}^{-1}S_{ux} + \Delta$, elementary algebraic manipulations yield:

$$\Xi(S, -S_{uu}^{-1}S_{ux} + \Delta) = S_{xx} - S_{ux}^T S_{uu}^{-1} S_{xu} + \Delta^T S_{uu}\Delta \tag{59}$$

thus, if S_{uu} is positive definite, we can ensure that

$$\begin{aligned}\Xi(S,\hat{K}) &= \begin{pmatrix} I & \hat{K}^T\end{pmatrix}\begin{bmatrix} S_{xx} & S_{ux}^T \\ S_{ux} & S_{uu}\end{bmatrix}\begin{pmatrix} I \\ \hat{K}\end{pmatrix}\\ &\geq \begin{pmatrix} I & -S_{ux}^T S_{uu}^{-1}\end{pmatrix}\begin{bmatrix} S_{xx} & S_{ux}^T \\ S_{ux} & S_{uu}\end{bmatrix}\begin{pmatrix} I \\ -S_{uu}^{-1}S_{ux}\end{pmatrix}\\ &= S_{xx} - S_{ux}^T S_{uu}^{-1} S_{xu} = \Xi(S, -S_{uu}^{-1}S_{ux})\end{aligned} \tag{60}$$

for any matrix $\hat{K}$ of compatible size.

a) Linear case: In a linear case with controller $\pi(x) = -K^\pi x$, the Lyapunov equation (26) can be equivalently understood as $\Xi(S, K^\pi) = P^\pi$, thus from (60) we can assert $x^T P x \geq x^T(S_{xx} - S_{ux}^T S_{uu}^{-1} S_{xu})x = Q^\pi(x, \hat{\pi}(x))$, and from standard dynamic-programming (9) argumentations, we can quantify a bound for the policy improvement $x^T P x \geq x^T(S_{uu} - S_{ux}^T S_{uu}^{-1} S_{xu})x \geq Q^{\hat{\pi}}(x, \hat{\pi}(x))$.

b) TS case: The guaranteed-cost LMIs (24) and (25) imply (56), and the latter equation implies $X^{-1} \geq \Xi(X^{-1}, F_{[\mu]})$, understanding the generic S in (57) to be $S_{[\mu^2]}$ in (29). Thus, we have proven that

$$\begin{aligned}&\bar{V}^{\pi_{LMI}}(x) = x^T X^{-1} x \geq x^T \Xi(X^{-1}, F_{[\mu]})x\\ &\geq x^T \Xi(X^{-1}, -(S_{[\mu^2],uu})^{-1} S_{[\mu^2],ux})x = \bar{V}^{\hat{\pi}_1}(x) \geq V^{\hat{\pi}_1}(x)\end{aligned} \tag{61}$$

thus the upper bound $\bar{V}^{\hat{\pi}_1}(x)$ is better than the LMI-proven bound $\bar{V}^{\pi_{LMI}}(x)$.