Model+Learning-based Optimal Control
An Inverted Pendulum Study

Baldi, Simone; Rosa, Muhammad Ridho; Wang, Yuzhang

DOI
10.1109/ICCA51439.2020.9264402

Publication date
2020

Document Version
Accepted author manuscript

Published in
Proceedings of the IEEE 16th International Conference on Control and Automation, ICCA 2020

Model+Learning-based Optimal Control: an Inverted Pendulum Study

Simone Baldi, Muhammad Ridho Rosa and Yuzhang Wang

Abstract— **This work extends and compares some recent model+learning-based methodologies for optimal control with input saturation. We focus on two methodologies: a model-based actor-critic (MBAC) strategy, and a nonlinear policy iteration strategy. To evaluate the performance of the algorithms, these strategies are applied to the swinging up an inverted pendulum. Numerical simulations show that the neural network approximation in the MBAC strategy can be poor, and the algorithm may converge far from the optimum. In the MBAC approach neither stabilization nor monotonic convergence can be guaranteed, and it is observed that the best value function is not always corresponding to the last one. On the other side the nonlinear policy iteration approach guarantees that every new control policy is stabilizing and generally leads to a monotonically decreasing cost.**

I. Introduction

For several decades, inverted pendulum systems have served as excellent test beds for control theory [1]. They were originally used to illustrate ideas in linear control such as stabilization of unstable systems [2]. Because of their nonlinear nature pendulums have maintained their usefulness and they are now used to illustrate many of the ideas emerging in the field of nonlinear control. Typical examples are feedback stabilization, variable structure control [3], passivity based control [4], back-stepping and forwarding [5], nonlinear observers [6], friction compensation [7]. Pendulums have also been used to illustrate task oriented control such as swinging up [8] [9] and are excellently suited to illustrate hybrid [10] and chaotic systems [11].

In this work, the nonlinear pendulum stabilization control from the pointing-down initial position to the unstable upright position is studied. The control performance is quantified in term of a cost function to be minimized. Two strategies are studied here: an actor-critic based strategy and a nonlinear policy iteration strategy. The first strategy is based on Model Based Actor-Critic (MBAC) method, where the controller learns to maximize the cumulative reward received over time (the value function) in order to reach the control goal. For the second strategy, this work revises and extends a policy iteration procedure for the synthesis of optimal control policies for linear input-constrained systems. Both strategies are referred to as *model+learning-based strategies*, because they exploit the knowledge of the system model to learn the optimal control law [12], [13].

The main contributions of this work are the two. First, extending the piecewise policy iteration from linear systems to nonlinear systems, using algorithmic solutions based on Sum-of-Squares (SOS) programs [14]. Second, evaluating and comparing the approximation of value function in both methods: in order to assess convergence to the optimal value function, in this work we compare neural network value function approximation with nonlinear polynomial approximation. In addition, the stability and convergence results, which have been proposed in the linear unsaturated case, will be checked in the nonlinear saturated case.

The paper is organized as follows: Sect. II presents the method of model based actor-critic control; Sect. III recalls the policy iteration method for linear systems with input-saturation; Sect. IV exploits results from sum of squares decomposition for nonlinear systems; Sect. V evaluates and compares the performance of the two model+learning methods in terms of convergence and minimization of cost.

II. Model-Based Actor-Critic Control

Actor-critic techniques which were introduced in [15] are characterized by learning functions for the actor (policy) and the critic (value function). See also [16], [17] for recent advances and applications. In this work, a model-based version of model-learning actor-critic (MLAC) algorithm is introduced. The MLAC algorithm was proposed in [18] to learn a process model in addition to the actor and critic. We slightly modify the algorithm by assuming that the process model is known. We refer to this algorithm as model-based actor-critic (MBAC).

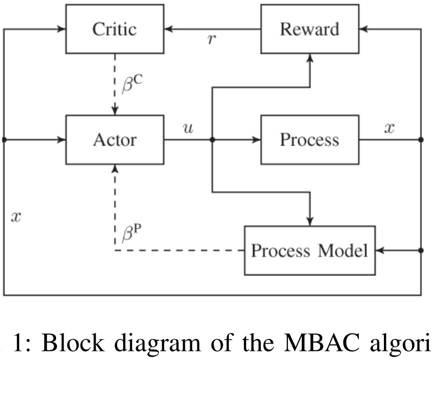


Fig. 1: Block diagram of the MBAC algorithm

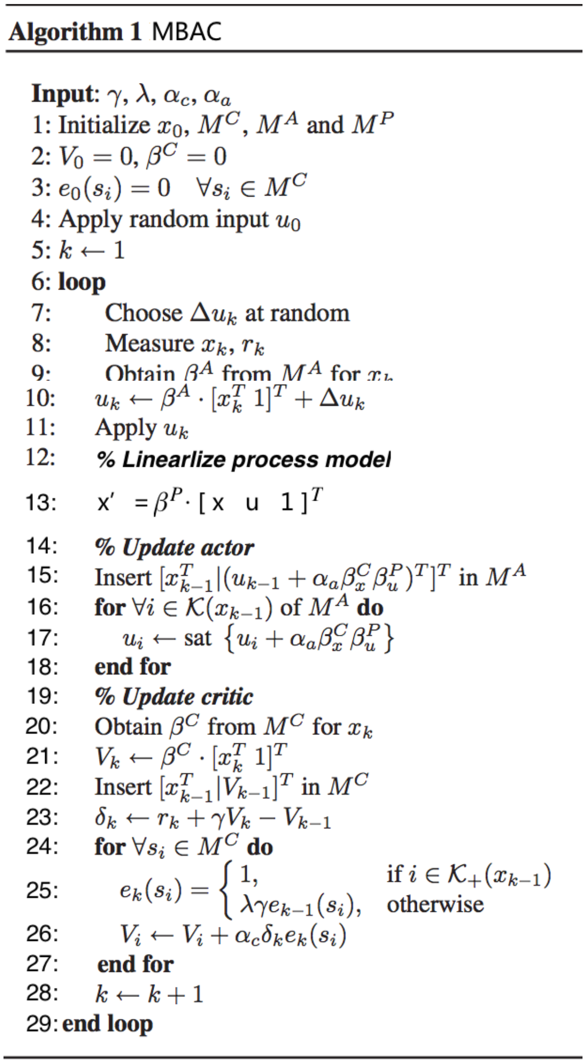
All authors equally contributed as first authors. This work was partially supported by the Fundamental Research Funds for the Central Universities under Grant 3207012004A2, and by the special guiding fund for double first-class under Grants 3307012001A, 6207011901. (Corresponding author: Simone Baldi).

S. Baldi is with School of Mathematics, Southeast University, Nanjing 210096, China, and guest with Delft Center for Systems and Control, TU Delft, 2628 Delft, Netherlands (e-mail: s.baldi@tudelft.nl)
M. R. Rosa is with School of Electrical Engineering, Telkom University, 40257, Bandung, Indonesia, and was with Delft Center for Systems and Control, TU Delft, 2628 Delft, Netherlands (e-mail: mridhorosa@telkomuniversity.ac.id)
Y. Wang is with Center for Precision Engineering, Harbin Institute of Technology, Harbin, China, and was with Delft Center for Systems and Control, TU Delft, 2628 Delft, Netherlands (e-mail: wangyuzhang67@163.com)

The MBAC method uses the model $x' = \hat{f}(x,u)$ to learn the optimal policy. The available process model simplifies the update of the actor, as it allows us to predict the next state x', given some input u. Together with the approximate value function, this allows us to obtain information on the value $V(x')$ of the next state x'. This means that we can choose the input u such that $V(x')$ is optimal. In Figure 1, the scheme of MBAC is shown.

Since we assume that our action space is continuous, we cannot enumerate over all possible inputs u and therefore the input is discretized to a set of discrete values. The actor is updated by multiplying the local gradients of the value function and of the process model to obtain a gradient of the value function with respect to chosen input u (after saturation in the allowed input range $[u_{min},\ u_{max}]$)

$$u_i \leftarrow \text{sat}\left\{u_i + \alpha_a \left.\frac{\partial V}{\partial x}\right|_{x=x'} \frac{\partial x'}{\partial u}\right\} \tag{1}$$



```
Algorithm 1 MBAC

Input: $\gamma$, $\lambda$, $\alpha_c$, $\alpha_a$
Initialize $x_0$, $M^C$, $M^A$ and $M^P$
$V_0 = 0$, $\beta^C = 0$
$e_0(s_i) = 0 \quad \forall s_i \in M^C$
Apply random input $u_0$
$k \leftarrow 1$
loop
    Choose $\Delta u_k$ at random
    Measure $x_k$, $r_k$
    Obtain $\beta^A$ from $M^A$ for $x_k$
    $u_k \leftarrow \beta^A \cdot [x_k^T\ 1]^T + \Delta u_k$
    Apply $u_k$
    % Linearlize process model
    x' $= \beta^P \cdot$ [x u 1]$^T$
    % Update actor
    Insert $[x_{k-1}^T | (u_{k-1} + \alpha_a \beta_x^C \beta_u^P)^T]^T$ in $M^A$
    for $\forall i \in \mathcal{K}(x_{k-1})$ of $M^A$ do
        $u_i \leftarrow$ sat $\{u_i + \alpha_a \beta_x^C \beta_u^P\}$
    end for
    % Update critic
    Obtain $\beta^C$ from $M^C$ for $x_k$
    $V_k \leftarrow \beta^C \cdot [x_k^T\ 1]^T$
    Insert $[x_{k-1}^T | V_{k-1}]^T$ in $M^C$
    $\delta_k \leftarrow r_k + \gamma V_k - V_{k-1}$
    for $\forall s_i \in M^C$ do
        $e_k(s_i) = \begin{cases} 1, & \text{if } i \in \mathcal{K}_+(x_{k-1}) \\ \lambda\gamma e_{k-1}(s_i), & \text{otherwise} \end{cases}$
        $V_i \leftarrow V_i + \alpha_c \delta_k e_k(s_i)$
    end for
    $k \leftarrow k + 1$
end loop
```

Fig. 2: MBAC algorithm

Recall that x' is given by the state transition function $x' = f(x,u)$. The value function is approximated by NNs which estimates a local linear model on the basis of previous observations of $V(x)$. The local linear model is of the form

$$V(x) = \beta^C \cdot \begin{bmatrix} x \\ 1 \end{bmatrix} = \begin{bmatrix} \beta_x^C & \beta_b^C \end{bmatrix} \cdot \begin{bmatrix} x \\ 1 \end{bmatrix} \tag{2}$$

where β_x^C which is the part of β^C, that is the gradient $\frac{\partial V}{\partial x}$ relates the input x to the output V. The gradient $\frac{\partial x'}{\partial u}$ can be found by NNs on previous observations of the process dynamics. The process model is linearized in the form

$$x' = \hat{f}(x,u) = \beta^P \cdot \begin{bmatrix} x \\ u \\ 1 \end{bmatrix} = \begin{bmatrix} \beta_x^P & \beta_u^P & \beta_b^P \end{bmatrix} \cdot \begin{bmatrix} x \\ u \\ 1 \end{bmatrix} \tag{3}$$

where β_u^P which is the part of β^P, that is the gradient $\frac{\partial x'}{\partial u}$ relates u to x'. So we can now use β_x^C, β_x^P and (1) to improve the actor by adapting the nearest neighbor samples with

$$u_i \leftarrow \text{sat}\{u_i + \alpha_a \beta_x^C \beta_u^P\} \tag{4}$$

The pseudocode is found in Algorithm 1. Summarizing, the MBAC uses the model to directly calculate an accurate policy gradient.

III. Piecewise Policy Iterations with Input-Saturation

We first recall the class of input-saturated linear systems without exponentially unstable modes

$$\dot{x} = Ax + Bsat\left(u(x)\right), \quad x(0) = x_0, \tag{5}$$

with $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$, $\Re(\lambda(A)) \leq 0$. The function $sat : \mathbb{R}^m \to \mathcal{U}$ is a vector saturation function, with entries satisfying

$$\left(sat\left(u(x)\right)\right)_j = \begin{cases} \overline{u}_j, & \text{if} \quad u_j > \overline{u}_j \\ u_j, & \text{if} \quad \underline{u}_j \leq u_j \leq \overline{u}_j \\ \underline{u}_j, & \text{if} \quad \underline{u} > u_j \end{cases} \tag{6}$$

and the set of inputs is defined as

$$\mathcal{U} := \left\{u \in \mathbb{R}^m | \, \underline{u}_j \leq u \leq \overline{u}_j, j = 1, \ldots, m\right\}.$$

then we introduce the dead-zone function

$$dz\left(u(x)\right) := u(x) - sat\left(u(x)\right)$$

and rewrite the system (5) as

$$\dot{x} = Ax + Bu(x) - Bdz\left(u(x)\right), \quad x(0) = x_0. \tag{7}$$

In order to have a well-posed problem we make the assumption: there exists a globally stabilizing control policy $\bar{u}$.

We also introduce a discounted cost function to be minimized for the system (7) of the form

$$J = \int_0^\infty e^{-\lambda t} \left[x' Q x + sat'\left(u(x)\right) R sat\left(u(x)\right)\right] dt. \tag{8}$$

We introduce sector condition related to the deadzone function $q(x) = dz\left(u(x)\right)$,

$$q'(x)\left(u(x) - q(x)\right) \geq 0, \quad \forall\, x \in \mathbb{R}^n. \tag{9}$$

Furthermore, define $\phi(x) := \frac{d\, dz(u(x))}{dt}$ satisfying

$$\phi(x) = \left\{ \begin{array}{ll} 0 & \text{if } q(x) = 0 \\ \dot{u}(x) & \text{if } q(x) \neq 0, \end{array} \right. \tag{10}$$

We adopt the well-known result from optimal control theory [19], that states that the optimal control policy $u^o(x)$ that minimizes (8) satisfies

$$u^o = \arg \min_{u(\cdot)\in\mathcal{U}} \left\{ -\lambda V^o + \frac{d\, V^o}{dx}\,(Ax + Bu) + L(x, u) \right\}, \tag{11}$$

where $V^o(x)$ is the value function that solves the Hamilton-Jacobi-Bellman (HJB) equation. The HJB equation is in general hard to solve and the most celebrated method for solving it is via Policy Iteration [20]. The work [21] addresses the policy iteration method for the saturated case, and it is briefly recalled for simplicity.

A. Piecewise policy evaluation and improvement

Consider a piecewise value function of the form $W^c(x, q^c(x))$ instead of $V^c(x)$

$$V^c(x) = W^c(x, dz(u^c(x))) = W^c(x, q^c(x)) \tag{12}$$

Since q is non-differentiable, we consider it as an independent variable, which allow us to describe its time-derivative in terms of a variable $\phi(x)$ as in (10)

$$\begin{aligned} -\lambda W^c + \frac{\partial W^c}{\partial x}(Ax + B(u^c - q^c)) + \frac{\partial W^c}{\partial q^c}\phi^{c/c} \\ +x'Qx + (u^c - q^c)'R(u^c - q^c) = 0 \end{aligned} \tag{13}$$

We consider the following approximate policy improvement,

$$u^{c+1}_{ap}(x) = -\frac{1}{2}R^{-1}B'\left.\frac{\partial W^{c\,\prime}}{\partial x}\right|_{q^c=0} \tag{14}$$

Note that $u^{c+1}_{ap}(x) = u^{c+1}(x)$ (the non approximated policy improvement) in the set

$$\Omega^c(x) = \{x : dz(u^c(x)) = 0\} \tag{15}$$

Furthermore, the policy $u^{c+1}_{ap}(x)$ in (14) defines a different unsaturated region as

$$\Omega^{c+1}(x) = \left\{x : dz(u^{c+1}_{ap}(x)) = 0\right\} \tag{16}$$

B. Modified policy iteration

We now discuss properties of the policy (14). Let us first define the following state-space partition defined by sets Ω^c and Ω^{c+1} in (15), (16)

$$\begin{array}{lcll} \Xi^c_1 & := & \Omega^c \cap \Omega^{c+1} & \text{(Region 1)} \\ \Xi^c_2 & := & \Omega^c \backslash \Omega^{c+1} & \text{(Region 2)} \\ \Xi^c_3 & := & \Omega^{c+1} \backslash \Omega^c & \text{(Region 3)} \\ \Xi^c_4 & := & \mathbb{R}^n \backslash (\Omega^c \cup \Omega^{c+1}) & \text{(Region 4)} \end{array} \tag{17}$$

In the following, we study the stability properties of the policy u^{c+1}_{ap}, given a globally stabilizing policy u^c and a value function W^c that certify global stability. To this purpose, define the piecewise policy

$$sat\left(u^{c+1}_{pw}\right) = \left\{ \begin{array}{ll} sat\left(u^{c+1}_{ap}\right) & \text{in } \Xi^c_1 \cup \Xi^c_2 \cup \Xi^c_3 \\ sat\left(u^c\right) & \text{in } \Xi^c_4 \end{array} \right. \tag{18}$$

and the value function

$$W^c_{pw} := \left\{ \begin{array}{ll} W^c_{un} & \text{in } \Xi^c_1 \cup \Xi^c_2 \cup \Xi^c_3 \\ W^c & \text{in } \Xi^c_4 \end{array} \right. \tag{19}$$

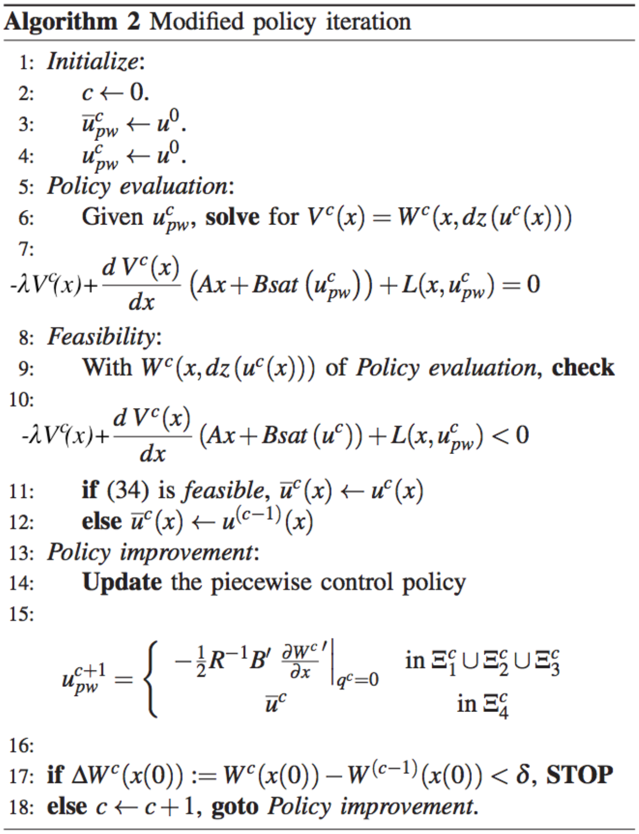
where W^c_{un} is the unsaturated value function defined as

$$W^c_{un}(x) := W^c(x, 0) \tag{20}$$

Then we obtain the following result [21],

Proposition 1: The piecewise value function (19) certifies the global stability of the piecewise control policy (18).

Figure 3 shows the algorithm of modified policy iteration with piecewise value function. In the next section the policy iteration method is extended to polynomial systems and local stability, suited for the pendulum control problem.



Algorithm 2 Modified policy iteration

```
Initialize:
    $c \leftarrow 0$.
    $\overline{u}^c_{pw} \leftarrow u^0$.
    $u^c_{pw} \leftarrow u^0$.
Policy evaluation:
    Given $u^c_{pw}$, solve for $V^c(x) = W^c(x, dz(u^c(x)))$

$-\lambda V^c(x) + \frac{d\, V^c(x)}{dx}\left(Ax + Bsat\left(u^c_{pw}\right)\right) + L(x, u^c_{pw}) = 0$
Feasibility:
    With $W^c(x, dz(u^c(x)))$ of Policy evaluation, check

$-\lambda V^c(x) + \frac{d\, V^c(x)}{dx}\left(Ax + Bsat\left(u^c\right)\right) + L(x, u^c_{pw}) < 0$
    if (34) is feasible, $\overline{u}^c(x) \leftarrow u^c(x)$
    else $\overline{u}^c(x) \leftarrow u^{(c-1)}(x)$
Policy improvement:
    Update the piecewise control policy

    $u^{c+1}_{pw} = \left\{ \begin{array}{ll} -\frac{1}{2}R^{-1}B'\left.\frac{\partial W^{c\,\prime}}{\partial x}\right|_{q^c=0} & \text{in } \Xi^c_1 \cup \Xi^c_2 \cup \Xi^c_3 \\ \overline{u}^c & \text{in } \Xi^c_4 \end{array} \right.$

if $\Delta W^c(x(0)) := W^c(x(0)) - W^{(c-1)}(x(0)) < \delta$, STOP
else $c \leftarrow c + 1$, goto Policy improvement.
```

Fig. 3: Algorithm of modified policy iterations under saturation constraints

IV. Handling Pendulum Nonlinearities with Sum of Squares Decomposition

To evaluate and compare the performance of our algorithms, we apply them to the task of stabilizing an inverted pendulum (from the pointing-down position to the upright position). The control performance is quantified in term of a cost function to be minimized. The actuation signal u is limited with saturation $[-20, 20]$, $[-10, 10]$, $[-5, 5]$ respectively, which make it possible to directly move the pendulum to the upright position. In discrete time, for actor-critic algorithm, we consider the cumulative cost

$$J = \sum_{k=1}^{\infty} x_{k-1}^T Q x_{k-1} + R u_{k-1}^2 \tag{21}$$

In continuous time, for the policy iteration algorithm, we define the cost function

$$J = \int_0^\infty L(x,u)dt = \int_0^\infty x'Qx + sat'(u(x))Rsat(u(x))dt \tag{22}$$

with $Q = \begin{bmatrix} 5 & 0 \\ 0 & 0.1 \end{bmatrix}$, $R = 1$.

The motion equation of this system is

$$J\ddot{\phi} = Mgl\sin(\phi) - \left(b + \frac{K^2}{R}\right)\dot{\phi} + \frac{K}{R}u \tag{23}$$

where ϕ is the angle of the pendulum measured from the upright position. The (fully measurable) state $x = \begin{bmatrix}\phi & \dot{\phi}\end{bmatrix}'$ consists of the angle ϕ of the pendulum and the angular velocity $\dot{\phi}$ of the pendulum. The model parameters are given in Table I.

TABLE I: Inverted Pendulum Model Parameters

Model Parameter	Symbol	Value	Units
Pendulum inertia	J	$1.91 \cdot 10^{-4}$	kgm^2
Pendulum mass	M	$5.50 \cdot 10^{-2}$	kg
Gravity	g	9.81	m/s^2
Pendulum length	l	$4.20 \cdot 10^{-2}$	m
Damping	b	$3 \cdot 10^{-6}$	Nms
Torque constant	K	$5.36 \cdot 10^{-2}$	Nm/A
Rotor resistance	R	9.50	Ω

The MBAC and Nolinear Policy Iteration algorithms were applied in simulation using the parameter settings in Table II. Note that the discounted rate is chosen in such a way that $\gamma = e^{-\lambda T_s}$, so that the discounted cost is the same in continuous and discrete time.

TABLE II: The Parameter Settings for MBAC and Nonlinear Policy Iteration Methods

		MBAC	Nonlinear PI
sampling time(s)	T_s	0.02	-
reward discount rate (discrete)	γ	0.9980	-
discounted cost rate (continuous)	λ	-	0.1
control quantization	Δu	0.2	-
basis function type		triangular	-
number of basis function for x_1	n_1	30	-
number of basis function for x_2	n_2	30	-

Then the system is described as follows,

$$\dot{x}_1 = x_2 \tag{24}$$

$$\dot{x}_2 = \frac{Mgl}{J}sin(x_1) - \left(b + \frac{K^2}{R}\right)\frac{1}{J}x_2 + \frac{K}{JR}u \tag{25}$$

with $x_1 \in [-\pi, \pi], x_2 \in [-20, 20]$. However, we want to normalize the state $x_1 \in [-1,1], x_2 \in [-1,1]$. The advantage of the normalization is that all the monomials will be also between $[-1,1]$ and the P matrix should be better conditioned. For this reason we define a new state which is

$$\bar{x}_1 = \frac{x_1}{\pi}, \quad \bar{x}_2 = \frac{x_2}{2\pi^2} \tag{26}$$

Thus the system (24)-(25) becomes

$$\dot{\bar{x}}_1 = 2\pi\bar{x}_2 \tag{27}$$

$$\dot{\bar{x}}_2 = \frac{Mgl}{2J\pi^2}sin(\pi\bar{x}_1) - \left(b + \frac{K^2}{R}\right)\frac{1}{J}\bar{x}_2 + \frac{K}{2JR\pi^2}u \tag{28}$$

But this system contains the non-polynomial term of $\sin(\pi\bar{x}_1)$, so we transform it to the polynomial system by introducing $\bar{x}_3 = sin(\pi\bar{x}_1)$ that is between $[-1,1]$, so there is no need to normalize it; and $\bar{x}_4 = \frac{cos(\pi\bar{x}_1)-1}{2}$ that is in order to have equilibrium at 0. So the nonlinear system with polynomial term is

$$\dot{\bar{x}}_1 = 2\pi\bar{x}_2 \tag{29}$$

$$\dot{\bar{x}}_2 = \frac{Mgl}{2J\pi^2}\bar{x}_3 - \left(b + \frac{K^2}{R}\right)\frac{1}{J}\bar{x}_2 + \frac{K}{2JR\pi^2}u \tag{30}$$

$$\dot{\bar{x}}_3 = 2\pi^2\bar{x}_2(2\bar{x}_4 + 1) \tag{31}$$

$$\dot{\bar{x}}_4 = -\pi^2\bar{x}_2\bar{x}_3 \tag{32}$$

with equality constraint

$$\bar{x}_3^2 + (2\bar{x}_4 + 1)^2 = 1 \tag{33}$$

Now that we obtained a polynomial vector field, we will be searching for a Value function that is also a polynomial. Then the well-known conditions in Lyapunov's method become polynomial nonnegative conditions, relaxed to SOS decompositions. The Lyapunov's method can then be formulated as SOS program stated in the following proposition [22],

Proposition 2. Suppose that for the system $\dot{z} = f(z)$ there exists a polynomial function $V(z)$ such that $V(0) = 0$ and

$$V(z) - \phi(z) \quad \text{is SOS,} \tag{34}$$

$$-\frac{\partial V(z)}{\partial z}f(z) \quad \text{is SOS.} \tag{35}$$

where $\phi(z) > 0$ for $z \neq 0$. Then the origin is stable.

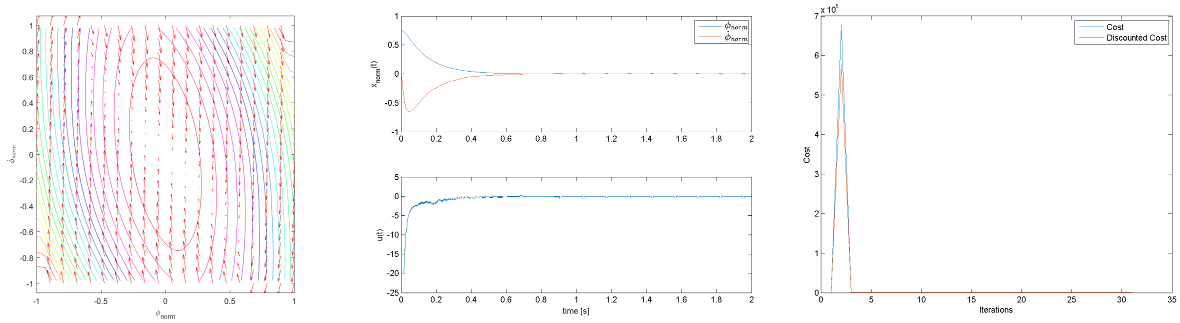
Then, an extension of the Lyapunov theorem in conjunction with the sum of squares decomposition and semidefinite programming can then be used to investigate the stability of the recasted system, the result of which can be used to infer the stability of the original system. Using the results of this chapter, the modified Policy Iteration Algorithm of Figure 3 can be straightforwardly extended to nonlinear systems with input saturation.

V. Numerical results

In the following, we show the simulation results with input saturation $[-20, 20]$, $[-10, 10]$ and $[-5, 5]$ by using model based actor-critic algorithm and nonlinear policy iteration algorithm respectively.

TABLE III: Cost with Input-Saturation $[-20, 20]$

	Final discounted cost	Improvement
MBAC	14.75	
Nonlinear PI	11.25	23.7%

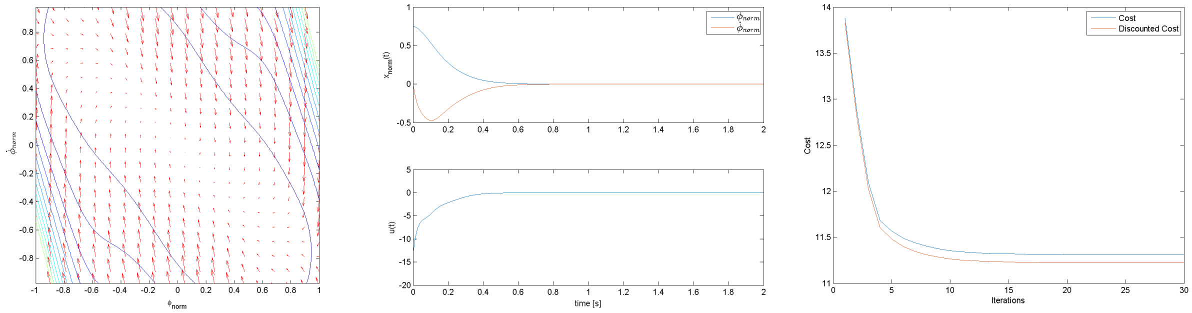


(a) Phase phane

(b) Final trajectory

(c) Cost

Fig. 4: Model-based Actor-Critic, Input-Saturation [-20,20]

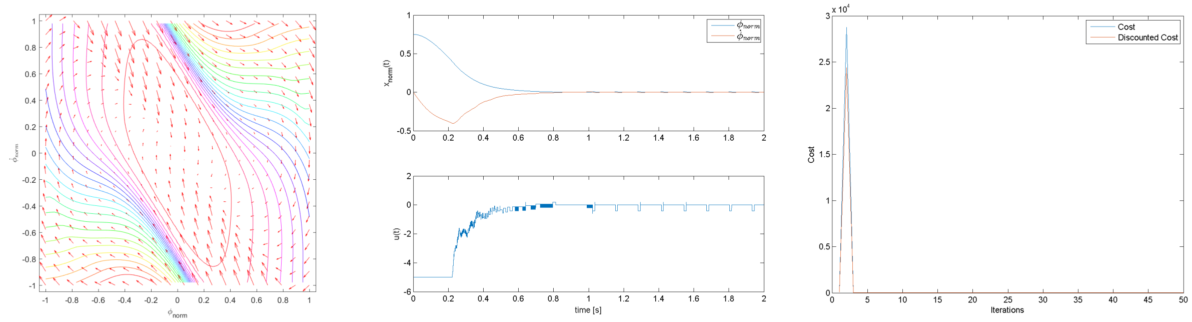


(a) Phase phane

(b) Final trajectory

(c) Cost

Fig. 5: Nonlinear Policy Iteration, Input-Saturation [-20,20]

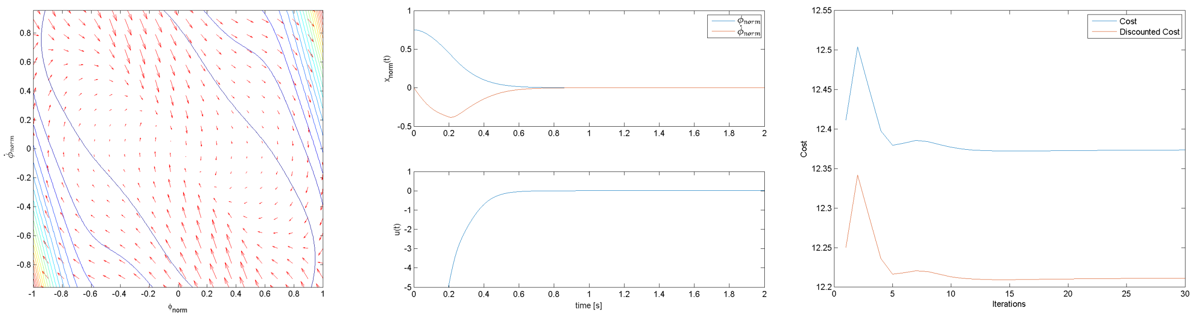


(a) Phase phane

(b) Final trajectory

(c) Cost

Fig. 6: Model-based Actor-Critic, Input-Saturation [-5,5]



(a) Phase phane

(b) Final trajectory

(c) Cost

Fig. 7: Nonlinear Policy Iteration, Input-Saturation [-5,5]

TABLE IV: Cost with Input-Saturation $[-10, 10]$

	Final discounted cost	Improvement
MBAC	11.82	
Nonlinear PI	11.18	5.4%

TABLE V: Cost with Input-Saturation $[-5, 5]$

	Final discounted cost	Improvement
MBAC	12.28	
Nonlinear PI	12.21	0.6%

The following observation can be drawn. For model based actor-critic method:

- The MBAC algorithm is not always able to reach the optimum. This is especially evident with saturation $[-20, 20]$, where the final result is at least 25% far from the optimal.
- For a discretization of 0.2 and adopting 30×30 NNs (respectively in the $x_1 - x_2$ plane), the MBAC approach is computationally less expensive than the Nonlinear Policy Iteration approach. However, with 40×40 NNs, MBAC led to OUT OF MEMORY problems.

For nonlinear policy iteration method:

- In contrast to the MBAC approach, where the value function is stored in a memory, in the nonlinear policy iteration approach the value function is a parameterized function. This leads to a control action which is in general smoother, while in the MBAC approach control action can result "bumpy".
- The nonlinear PI approach will generally lead to a monotonically decreasing cost (at least in the unsaturated region), whereas in the MBAC approach neither stabilization nor monotonic convergence can be guaranteed. In particular, in the MBAC approach, it is observed that the best value function is not always corresponding to the last one.
- The improvement of the nonlinear PI approach is generally decreasing as the saturation becomes tighter and tighter. This can be explained by the fact that when the control authority decreases, the freedom to improve the cost becomes smaller and smaller.

VI. Conclusions and future work

In this work, two model+learning-based methodologies for optimal control of nonlinear systems have been slightly revised and applied to the optimal control of a pendulum: an actor-critic and a nonlinear policy iteration, respectively. Simulation results with different saturation levels show that the NN approximation of the MBAC algorithm may converge far from the optimum and the best value function is not always corresponding to the last one. The nonlinear PI approach guarantees that every new control policy will be stabilizing and generally lead to a monotonically decreasing cost. However, the improvement of the nonlinear policy iteration approach is generally decreasing as the saturation becomes tighter and tighter.

Future work will include further increasing the level of saturation in such a way that a truly swing up action is required (e.g., saturation $[-3, 3]$ or $[-1, 1]$).

References

[1] K. Yoshida. Swing-up control of an inverted pendulum by energy-based methods. In *Proceedings of the American control conference*, volume 6, pages 4045–4047, 1999.

[2] M. Farwig, H. Zu, and H. Unbehauen. Discrete computer control of a triple-inverted pendulum. *Optimal Control Applications and Methods*, 11(2):157–171, 1990.

[3] M. Yamakita, K. Furuta, K. Konohara, J. Hamada, and H. Kusano. Vss adaptive control based on nonlinear model for titech pendulum. In *Industrial Electronics, Control, Instrumentation, and Automation, 1992. Power Electronics and Motion Control., Proceedings of the 1992 International Conference on*, pages 1488–1493. IEEE, 1992.

[4] A. L. Fradkov and A. Y. Pogromsky. Speed gradient control of chaotic continuous-time systems. *IEEE Transactions on Circuits and Systems I Fundamental Theory and Applications*, 43(11):907–913, 1996.

[5] M. Krstić, I. Kanellakopoulos, and P. V. Kokotović. Passivity and parametric robustness of a new class of adaptive systems. *Automatica*, 30(11):1703–1716, 1994.

[6] J. Eker and K. J. Åström. A nonlinear observer for the inverted pendulum. In *Control Applications, 1996., Proceedings of the 1996 IEEE International Conference on*, pages 332–337. IEEE, 1996.

[7] C. F. Abelson. The effect of friction on stabilization of an inverted pendulum. *MSc Theses*, 1996.

[8] K. Furuta, M. Yamakita, and S. Kobayashi. Swing up control of inverted pendulum. In *Industrial Electronics, Control and Instrumentation, 1991. Proceedings. IECON'91., 1991 International Conference on*, pages 2193–2198. IEEE, 1991.

[9] Q. Wei, W. P Dayawansa, and W. S. Levine. Nonlinear controller for an inverted pendulum having restricted travel. *Automatica*, 31(6):841–850, 1995.

[10] J. Guckenheimer. A robust hybrid stabilization strategy for equilibria. *IEEE Transactions on Automatic Control*, 40(2):321–326, 1995.

[11] T. Shinbrot, C. Grebogi, J. Wisdom, and J. A. Yorke. Chaos in a double pendulum. *Am. J. Phys*, 60(6):491–499, 1992.

[12] S. Baldi, I. Michailidis, E. B. Kosmatopoulos, and P. A. Ioannou. A "plug and play" computationally efficient approach for control design of large-scale nonlinear systems using cosimulation: a combination of two "ingredients". *IEEE Control Systems Magazine*, 34(5):56–71, 2014.

[13] S. Baldi, F. Zhang, T. Le Quang, P. Endel, and O. Holub. Passive versus active learning in operation and adaptive maintenance of heating, ventilation, and air conditioning. *Applied Energy*, 252:113478, 2019.

[14] Y. Jiang and Z.-P. Jiang. Global adaptive dynamic programming for continuous-time nonlinear systems. *Automatic Control, IEEE Transactions on*, 60(11):2917–2929, 2015.

[15] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *Systems, Man and Cybernetics, IEEE Transactions on*, (5):834–846, 1983.

[16] C. D. Korkas, S. Baldi, S. Yuan, and E. B. Kosmatopoulos. An adaptive learning-based approach for nearly optimal dynamic charging of electric vehicle fleets. *IEEE Transactions on Intelligent Transportation Systems*, 19(7):2066–2075, 2018.

[17] P. Dai, W. Yu, G. Wen, and S. Baldi. Distributed reinforcement learning algorithm for dynamic economic dispatch with unknown generation cost functions. *IEEE Transactions on Industrial Informatics*, 16(4):2258–2267, 2020.

[18] I. Grondman, M. Vaandrager, L. Busoniu, R. Babuska, and E. Schuitema. Efficient model learning methods for actor–critic control. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 42(3):591–602, 2012.

[19] K. Donald. Optimal control theory: An introduction. *Mineola, NY: Dover Publications, Inc*, 1970.

[20] J. J. Murray, C. J. Cox, G. G. Lendaris, and R. Saeks. Adaptive dynamic programming. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 32(2):140–153, 2002.

[21] S. Baldi, G. Valmorbida, A. Papachristodoulou, and E. B. Kosmatopoulos. Piecewise polynomial policy iterations for synthesis of optimal control laws in input-saturated systems. In *American Control Conference (ACC), 2015*, pages 2850–2855. IEEE, 2015.

[22] A. Papachristodoulou and S. Prajna. Analysis of non-polynomial systems using the sum of squares decomposition. In *Positive polynomials in control*, pages 23–43. Springer, 2005.