Scientific Research Publishing

Applied Mathematics, 2017, 8, 1602-1618
http://www.scirp.org/journal/am
ISSN Online: 2152-7393
ISSN Print: 2152-7385

# Stability Analysis of a Neurocontroller with Kalman Estimator for the Inverted Pendulum Case

**Julián Antonio Pucheta[1], Cristian Rodríguez Rivero[1], Carlos Alberto Salas[2], Martín Herrera[2], Sergio Oscar Laboret[1],**

[1]Departamentos de Electrotecnia y de Electrónica, Laboratorio de Investigación Matemática Aplicada a Control (LIMAC), Facultad de Ciencias Exactas, Físicas y Naturales-Universidad Nacional de Córdoba, Córdoba, Argentina
[2]Departamento de Ingeniería Electrónica, Facultad de Tecnología y Ciencias Aplicadas, Universidad Nacional de Catamarca, Catamarca, Argentina

Email: jpucheta@unc.edu.ar, cristian.rodriguezrivero@gmail.com, slaboret@yahoo.com.ar, calberto.salas@gmail.com, ing_martin_herrera@yahoo.com.ar

**How to cite this paper:** Pucheta, J., Rodríguez Rivero, C., Salas, C., Herrera, M. and Laboret, S. (2017) Stability Analysis of a Neurocontroller with Kalman Estimator for the Inverted Pendulum Case. *Applied Mathematics*, **8**, 1602-1618.
https://doi.org/10.4236/am.2017.811117

**Received:** September 28, 2017
**Accepted:** November 21, 2017
**Published:** November 24, 2017



## Abstract

In this paper, a practical analysis of stability by simulation for the effect of incorporating a Kalman estimator in the control loop of the inverted pendulum with a neurocontroller is presented. The neurocontroller is calculated by approximate optimal control, without considering the Kalman estimator in the loop following the Theorem of the separation. The results are compared with a time-varying linear controller, which in noiseless conditions in the state or in the measurement has an acceptable performance, but when it is under noise conditions its operation closes into a state space range more limited than the one proposed here.

## Keywords

Optimal Control, Neurocontroller, Inverted Pendulum, State Estimation, Stability

## 1. Introduction

The motivation for this case study known as inverted pendulum control arises from the need to obtain robust controller systems to implement in situations where it is desired to maintain equilibrium of an unstable system. A direct related situation is the attitude control of a booster rocket at takeoff for sending a payload to space. It is a well-known problem in the control theory literature [1] [2] and machine learning [3] [4] [5] [6]. However, such problems are challeng-

ing since real systems are difficult to control and this is to some extent due to the fact that redundant feedback systems must be considered by the controller, as an effect similar to that of incorporating an estimator into the variables controller status. This fact causes instability in the closed loop, which must be foreseen and analyzed. The analysis can be done through simulations with the estimator-controller system, in order to establish some stability domain. In this work we opt for the control based on optimization [7] [8] [9], where the optimal control problem is formulated. To solve the problem of optimal control, a very powerful tool is the Dynamic Programming technique [10] [11] implemented with approximations [3] [4] [5] in a machine learning scheme [12], since it allows dealing with constrained, nonlinear processes and non-quadratic performance indexes. However, it is often difficult to achieve a methodology for the implementation of controllers based on machine learning, since they require heuristic and a good knowledge of the involved adaptation mechanisms [3]. In this work, a methodology to determine the conditions that achieve good results to implement in simulation is shown. A controller consisting of a compact function called neurocontroller is achieved.

In this paper, cases with and without estimator of a model that represents an inverted pendulum are studied. When a time varying linear quadratic regulator (TVLQR) with direct state measurement is used, good performance can be achieved. However, it can be improved with a neurocontroller. The obtained performances by using linear and neurocontroller are shown in **Figure 1** and **Figure 2**, respectively. Note that the cumulative cost of the linear controller

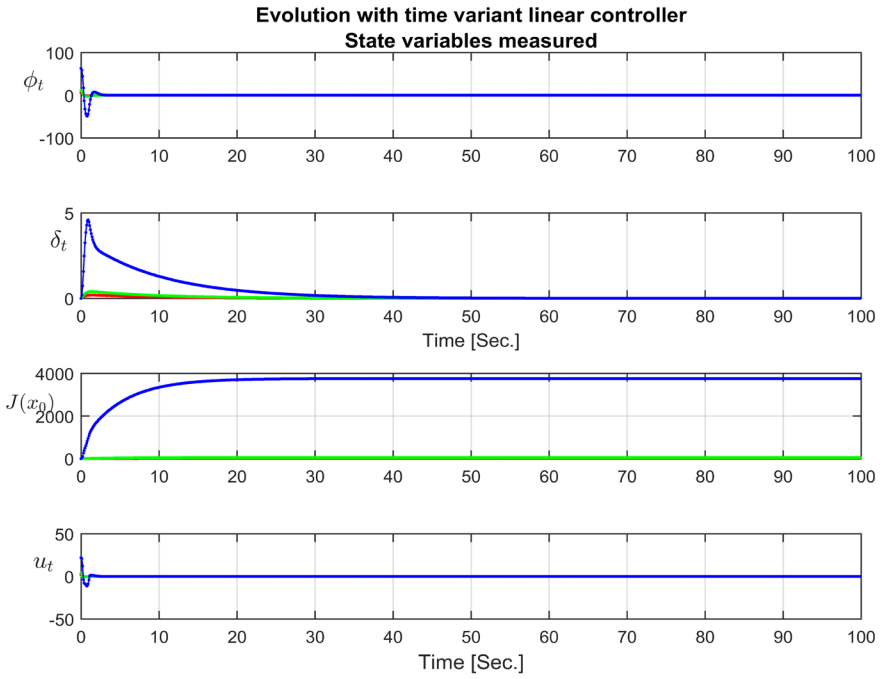


**Figure 1.** Controller TVLQR with direct measurements. The initial conditions are $x_0 = [0\ 0\ \phi_0\ 0]^T$, where $\phi_0$ takes the values 5.7˚, 11.5˚ and 63˚. Accumulated costs from each initial condition are 12.5, 51.8 and 3760.7, respectively.

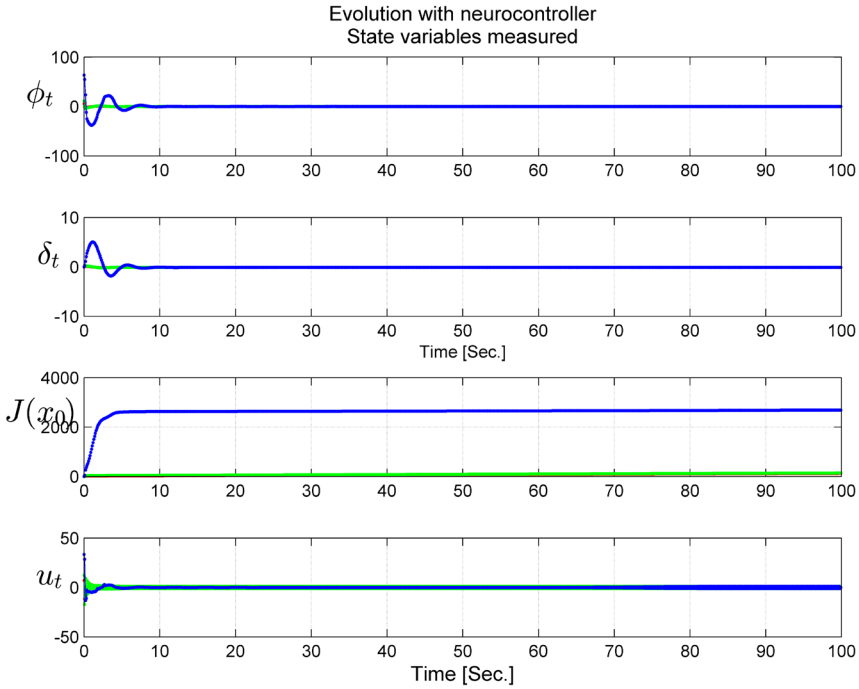


**Figure 2.** Neurocontroller with direct measurements. The initial conditions are $x_0 = [0\ 0\ \phi_0\ 0]^T$, where $\phi_0$ takes the values 5.7˚, 11.5˚ and 63˚. Accumulated costs from each initial condition are 123.8, 140.7 and 2691.3, respectively.

(3760.7) is 28% higher than that of the Neurocontroller (2691.3). However, when the controller is used in more realistic situations using a state estimator and considering noisy conditions in the measurements, the performance of the linear controller deteriorates more than the performance of the neurocontroller even until fails to stabilize the system for the same initial conditions. In this paper, an analysis of the system performance deterioration is shown when it requires a state estimator.

This paper is organized as follows. After this Introduction, the problem is detailed and expressed as mathematical equation in Section 2. In Section 3 is detailed the proposed solution. In Section 4 the implementation of the obtained solution and another one with classical methods for comparison purposes is developed. The obtained results are discussed in Section 5, with its pros and cons.

## 2. Problem Formulation

The dynamic programming approach assumes that the process evolution can be split in stages [10], so take the version of dynamic systems in discrete time [9] is straightforward. The problem formulation puts in formal terms the optimal control elements. These elements are the cost function to minimize, the control law and the dynamic system model with its constraints. If the system model cannot be or is not feasible to express it in closed analytical form through a differential equation, it is useful to generate a black box model [13].

1) ***Notation and Assumptions***

This section introduces the nomenclature used along the article. The symbols

are listed and explained.

i. $k \in \mathbb{N}$ describes the discrete time variable.

ii. $x(k)$, with $\boldsymbol{x}(k) \in \Re^n$ is the time dependent n-dimensional state vector whose components has the system's state variables. These variables describe the process dynamics over discrete time.

iii. $f$ is a nonlinear continuous function, $f : \Re^n \times \Re^m \to \Re^n$ that describes the relation between the state vector for two time instants.

iv. $\boldsymbol{u}(k) \in \Re^m$ is the system input or manipulated vector.

v. $I$ is a convex function, $I : \Re^n \times \Re^m \times \mathbb{N} \to \Re^+$ called the performance index designed by the control engineer.

vi. $J$ is a convex function, $J : \Re^n \times \Re^m \times \mathbb{N} \to \Re^+$ named the cost function designed by the control engineer.

vii. $X$ is a bounded and closed set, $X \subset \Re^n$.

viii. $U$ is a bounded and closed set, $U \subset \Re^m$.

ix. $\mu$ is the control law or decision policy $\mu : \Re^n \to \Re^m$, this function maps each state vector value with a control action $\boldsymbol{u}$.

x. $\boldsymbol{r}$ and $\boldsymbol{v}$ are real value arrays of parameters $\boldsymbol{r}, \boldsymbol{v} \in \Re^{n+h+2}$, where $h \in \mathbb{N}$ is determined by the control engineer.

xi. $\tilde{J}$ is the approximation of the cost function $J$, and its domains includes the parameter vector $\boldsymbol{r}$, $\tilde{J} : \Re^n \times \Re^m \times \Re^{n+h+2} \to \Re^+$.

xii. $\tilde{\mu}$ is a function whose behavior approximates the function $\mu$, includes the parameter vector $\boldsymbol{v}$, $\tilde{\mu} : \Re^n \times \Re^{n+h+2} \to \Re^m$.

xiii. $J^*\left(\boldsymbol{x}(k)\right)$ is the minimum cost to go from the state $\boldsymbol{x}$ at time $k$ up to the terminal state at time $N$.

xiv. $\boldsymbol{u}^o(k) \in \Re^m$ is the optimal control action at time $k$.

xv. $\xi_1, \xi_2, \cdots, \xi_h$ are real scalar values.

xvi. $\tilde{S}$ data set of samples from the process under study.

xvii. $C_{(i)}$ is the cost associate to a control law evolving from the state $i$ up to the terminal process state.

xviii. $J_{(i)}^{\mu}$ is the value of the cost to go function obtained after use the control law $\mu$ starting at state $i$ up to terminal state at time $N$.

xix. $\nabla$ gradient operator.

xx. $Q(i,u)$, real valued function associated at state $i$ and action $u$.

xxi. $\eta_n$ is a function that varies with iteration number $n$, bounded between 0 and 1.

xxii. $\tilde{Q}(i,u)$ is the approximate version of the factor $Q(i,u)$.

xxiii. $\gamma_n$ is the discount factor, variable with iteration $n$ and bounded between 0 and 1.

xxiv. $\overline{\mu}$ control action expressed as look up table from every state.

xxv. $\tilde{J}^{\mu}(\cdot, \boldsymbol{r})$ approximate cost to go function associated with the control law $\mu$.

xxvi. $\boldsymbol{A} \in \Re^{n \times n}$ is the state matrix for the linear dynamic model.

xxvii. $\boldsymbol{B} \in \Re^{n \times 1}$ is the input matrix of the linear dynamic model.

xxviii. $\boldsymbol{F} \in \Re^{n\times n}$ is the additive noise model at the state variables.

xxix. $\boldsymbol{v}(k) \in \Re^{n}$ is the random sequence with Gaussian distribution in each variable, zero mean and unit variance.

xxx. $\boldsymbol{y}(k) \in \Re$ is the linear model output.

xxxi. $\boldsymbol{C} \in \Re^{1\times n}$ is the linear model output matrix.

xxxii. $\boldsymbol{G} \in \Re$ is the noise model at the measured variable.

xxxiii. $\boldsymbol{w}(k) \in \Re$ is a white noise sequence with zero mean and unit variance.

xxxiv. $\delta \in \Re$ is longitudinal displacement of the cart.

xxxv. $\dot{\delta} \in \Re$ is longitudinal velocity of the cart.

xxxvi. $\phi \in \Re$ is the angle of the inverted pendulum bar.

xxxvii. $\dot{\phi} \in \Re$ is the angular velocity of the inverted pendulum bar.

xxxviii. $M_P$ is the cart concentrated mass, whose value here is 0.5 Kgr.

xxxix. $m_P$ is the bar concentrated mass, valued here is 0.1 Kgr.

xl. $F_P$ is the displacement friction constant assigned 0.1 $\mathrm{N\cdot m^{-1}\cdot s}$.

xli. $l_P$ is the size of the pendulum bar, 0.6 m.

xlii. $g_P$ is the standard acceleration due to gravity, 9.81 $\mathrm{m\cdot s^{-2}}$.

xliii. $\boldsymbol{Q} \in \Re^{4\times 4}$ is the weighing matrix for the state vector from $k = 0$ to $k = N - 1$ with $N$ the terminal state time.

xliv. $\boldsymbol{S} \in \Re^{4\times 4}$ is the weighing matrix for the state vector at the terminal state time $N$.

xlv. $\boldsymbol{R} \in \Re$ is the weighing matrix for the control action variable.

**2)** ***The basic problem***

Thus, to formulate the optimal control problem the expressions of the process model in discrete time, the restrictions in the variables and the cost function to be minimized are presented. Next, the problem of minimizing the separable cost function is considered by

$$\boldsymbol{x}(k+1) = f\left(\boldsymbol{x}(k), \boldsymbol{u}(k), k\right), k = 0,1,\cdots,N-1 \tag{1}$$

where $\boldsymbol{x}(0)$ has a fixed value and the constraints must be satisfied together with the system equation,

$$J\left(\boldsymbol{x}(k), \boldsymbol{u}(k)\right) = \sum_{k=0}^{N} I\left(\boldsymbol{x}(k), \boldsymbol{u}(k)\right) \tag{2}$$

where the constraints on state and manipulated variables are

$$\boldsymbol{x} \in X \subset \Re^{n}, \quad \boldsymbol{u} \in U \subset \Re^{m}. \tag{3}$$

The function $I(\cdot)$ is defined by the control engineer which must be convex but not necessarily quadratic, and $f(\cdot)$ is the nonlinear relationship between instants $k$ and $k+1$ of the state and manipulated variables. Moreover, they are bounded and continuous functions of their arguments, and both $\boldsymbol{x}$ and $\boldsymbol{u}$ belong to closed and bounded subsets of $\Re^{n}$ and $\Re^{m}$, respectively. Then, the Weierstrass theorem asserts that there exists a minimization policy also called control law. Therefore, it is desired to find a correspondence relation

$$\mu\left(\boldsymbol{x}(k)\right):\Re^{n}\to\Re^{m} \tag{4}$$

that makes evolve the processes modelled by (2) from any initial condition to the final terminal state $x(N)$ satisfying constraints (3), and minimizing the cost function (1). The implementation is shown in Figure 3, where the flow of information between the controller and the closed-loop system is stated. Note that the behavior of the closed loop system is done by designing the performance index which is added at each stage in the cost function (1).

## 3. Proposed Solution

In order to solve the formulated problem, the proposed solution is by using dynamic programming and then approximations are introduced through functions

$$\tilde{\mu}\left(\boldsymbol{x}(k),\boldsymbol{v}\right):\Re^{n}\to\Re^{m}, \tag{5}$$

$$\tilde{J}\left(\boldsymbol{x},\boldsymbol{u},\boldsymbol{r}\right):\Re^{n\times m\times N}\to\Re^{+} \tag{6}$$

where the parameter vectors $\boldsymbol{v}$ and $\boldsymbol{r}$ must be determined.

1) ***Optimal control for processes modelled as constrained nonlinear systems***

The procedure to solve the optimal control problem for both continuous and discrete time dynamic systems is well known [7] [8] [14], and consists of analytically minimize the proposed cost function (1) and from this minimization achieve an expression for function $\mu$. When the system is linear and the cost function is quadratic, the optimal control problem has unique solution through the Riccati Equation. However, when the system is nonlinear the solution of the Hamilton-Jacobi-Bellman equation [14] must be found, whose solution is restricted to a certain class of nonlinear systems. Here, an optimization principle to solve the same control problem that allows to use any cost function and respecting the constraints in the state variables and in the control variables in a natural way is used.

2) ***Bellman's optimality principle***

The principle of optimality [10] allows solving an optimization problem in

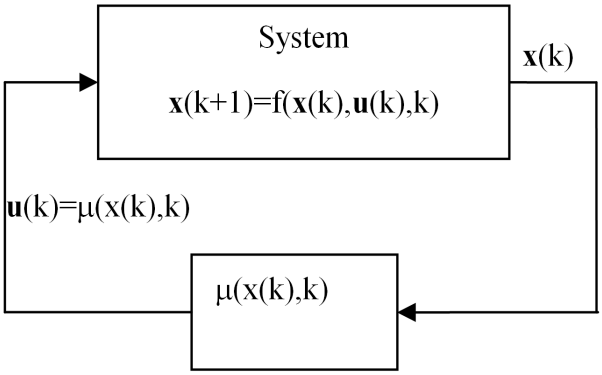


**Figure 3.** Implementation of the controller based on numerical dynamic programming.

which a dynamic process evolves over time through stages. Applying the principle of optimality in (1), we obtain

$$\begin{aligned} J^*\left(\boldsymbol{x}(k)\right) &= \min_{u(k)} J\left(\boldsymbol{x}(k),\boldsymbol{u}(k)\right) \\ &= \min_{u(k)}\left\{I\left(\boldsymbol{x}(k),\boldsymbol{u}(k)\right)+J^*\left(f\left(\boldsymbol{x}(k),\boldsymbol{u}(k)\right)\right)\right\}, \end{aligned} \tag{7}$$

called the Bellman's Equation. Therefore, the optimal control action $u^o$ will be

$$\boldsymbol{u}^o(k) = \arg\min_{u(k)}\left\{I\left(\boldsymbol{x}(k),\boldsymbol{u}(k)\right)+J^*\left(f\left(\boldsymbol{x}(k),\boldsymbol{u}(k)\right)\right)\right\}, \tag{8}$$

which is the optimal policy of decisions or optimal control law. Note that $f^*$ does not depend explicitly on $\boldsymbol{u}(k)$, as shows Equation (8).

**3)** ***Introducing approximations***

To obtain the control law or the decision policy, there exists numerical methods, [3] [4] [10] [11] and approximations [3] [5] [12] which are detailed below. Now, an approximation function for values of Equation (1) in a compact domain is introduced. Thus, a compact representation of the cost associated with each state of the process is obtained.

**4)** ***Design of the approximation function***

The approximation function incorporates a set of vectors of parameters $\boldsymbol{r}$, which is defined as a partitioned vector whose structure defines the function structure,

$$\boldsymbol{r} = \left\{r_1^1, r_1^2, \cdots, r_1^h, r_2\right\} \tag{9}$$

where each vector $\boldsymbol{r}_1$ has the same dimension, which is the number of inputs of the function plus one to consider a static scalar unit parameter. So, $h$ intermediate scalar values $\xi$ are computed as the scalar product between the input vector $\boldsymbol{x}$ and the corresponding parameters as

$$\xi_1 = \left[\boldsymbol{x}^{\mathrm{T}} \quad 1\right]\cdot r_1^1, \tag{10}$$

$$\xi_2 = \left[\boldsymbol{x}^{\mathrm{T}} \quad 1\right]\cdot r_1^2, \tag{11}$$

$$\xi_h = \left[\boldsymbol{x}^{\mathrm{T}} \quad 1\right]\cdot r_1^h, \tag{12}$$

every single value are processed through the hyperbolic tangent function, avoiding large numbers by

$$f(\xi) = \frac{\exp(\xi)-\exp(-\xi)}{\exp(\xi)+\exp(-\xi)} = 1-\frac{2}{1+\exp(2\xi)} \tag{13}$$

where the right side has only one exp(·) computation for improving calculation time. So, with these $h$ values together with the polarization **1** the inner product is implemented with the rest of the $\boldsymbol{r}$ parameter vector which is $\boldsymbol{r}_2$, and must be consistent in its dimension to be able to perform the product

$$\tilde{\mu}\left(\boldsymbol{x}(k),\boldsymbol{r}\right) = \left[f(\xi_1) \quad f(\xi_2) \quad \cdots \quad f(\xi_h) \quad 1\right]^{\mathrm{T}}\cdot \boldsymbol{r}_2. \tag{14}$$

This approximation function has the parameter $h$ as a tuning parameter for the dimension of vector $\boldsymbol{r}$, in terms of the structure of the approximation function.

Finding a suitable value for vector $\boldsymbol{r}$, one have the approximated value of the minimum cost that is incurred to reach the terminal state from the current state $\boldsymbol{x}(k)$, and with the model of the system can be found the control policy using the argument $u(k)$ that minimizes

$$J^*\left(\boldsymbol{x}(k)\right)=\min_{u_k}\left\{I\left(\boldsymbol{x}(k),\boldsymbol{u}(k)\right)+\tilde{J}^*\left(f\left(\boldsymbol{x}(k),\boldsymbol{u}(k)\right),\boldsymbol{r}\right)\right\}. \tag{15}$$

For finding $\boldsymbol{r}$, the search process that finds the policy function is divided into two tasks, as shown in Figure 4. One of them, is the evaluation of a defined control policy or control law, from which the costs for all stages of the process are calculated. The other one, is the policy improvement procedure. Both tasks are done approximately with respect to the original system, because an approximation function is used that tune its behavior.

**5)** ***Implementation***

A set of representative data $\tilde{S}$ in the state space in a domain is available and for each state $i\in\tilde{S}$ the cost values $C(i)$ are calculated. To this end, an initial control law or control policy is proposed, and the system (1) is evolved from the given state $i$ to the terminal stage, evaluating the performance index by expression (2) of the cost function $J_{(i)}^{\mu}$. This procedure is performed for every state $i\in\tilde{S}$. Then the approximated cost function is tuned by minimizing in $\boldsymbol{r}$

$$\min_{r}\sum_{i\in\tilde{S}}\left(\tilde{J}(i,r)-C(i)\right)^2 \tag{16}$$

an approximation function for the cost associated to the evaluated policy is obtained. The parameters vector $\boldsymbol{r}$ is obtained by minimizing expression (16). The incremental gradient iteration is

$$\boldsymbol{r}:=\boldsymbol{r}+\eta_n\nabla\tilde{J}(i,\boldsymbol{r})\left(C(i)-\tilde{J}(i,\boldsymbol{r})\right),\quad \forall i\in\tilde{S} \tag{17}$$

where $\eta$ fulfills the conditions

$$\sum_{n=0}^{\infty}\eta_n(i,u)=\infty,\ \sum_{n=0}^{\infty}\eta_n^2(i,u)<\infty,\quad \forall i,u\in U(i), \tag{18}$$

such that the algorithm converges, where $n$ refers to the tuning iteration $n$. For computing $C(i)$ the performance evaluation is implemented through the system model (1) and the cost function (2).

Then the costs associated with each state-action pair are computed, by using

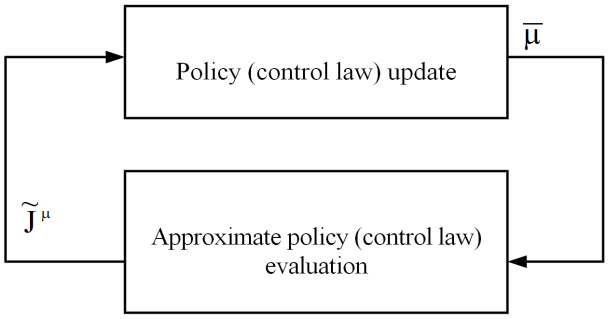


**Figure 4.** Scheme of the optimal control policy search process.

the auxiliary cost function $Q(i, u)$, which in its approximate version is

$$\tilde{Q}(i,u) = I(i,u) + \gamma_n \tilde{J}(j,r) \tag{19}$$

where $\gamma_n$ is a discount factor that can vary from iteration to iteration up to reach unity. Then, the improved policy is obtained by the table

$$\bar{\mu}(i) = \arg\min_{u \in U(i)} \left( I(i,u) + \gamma_n \tilde{J}(j,r) \right), \quad \forall i \in \tilde{S}. \tag{20}$$

Once available $\tilde{J}^{\mu}(\cdot, r)$, it can be obtained $\bar{\mu}(\cdot)$ from Equation (20). Then the costs associated with each state, symbolized by $C(i)$, are evaluated by Equation (17), and the $\boldsymbol{r}$ parameters are tuned by obtaining a new version of the approximation function $\tilde{J}^{\mu}(i,r), \forall i \in \tilde{S}$. Then, the *policy improvement* task is carried out, in which a new tabulated control policy $\bar{\mu}(\cdot)$ expressed as (20) is obtained. After that, the calculation of the costs for each state $i$ starts, and in each iteration the function $\gamma_n$ is updated.

Simultaneously with the described tasks, an approximation for the improved control law $\bar{\mu}(\cdot)$ is introduced, by a function with parameters $\boldsymbol{v}$ as shown Figure 5 following the same structure as that described by Equations (9)-(14).

Thus, since the function $\mu(\cdot)$ is the analytical solution of the optimal control problem, it is intended to obtain an approximation $\tilde{\mu}(\cdot, \boldsymbol{v})$ of the function $\mu(\cdot)$ -which is expressed as table, where $\mathbf{v}$ is the parameter vector.

To find the approximation function $\tilde{\mu}(\cdot, \boldsymbol{v})$, using the data of the improved policy $\bar{\mu}(\cdot)$ defined in (20), it is proposed to minimize the expression

$$\min_{\boldsymbol{v}} \sum_{i \in S} \left( \tilde{\mu}(i, \boldsymbol{v}) - \bar{\mu}(i) \right)^2 \tag{21}$$

within the set $\tilde{S}$, where the control law is represented by $\tilde{\mu}(\cdot, \boldsymbol{v})$ with the tuning parameters vector $\boldsymbol{v}$. A solution for Equation (21) is obtained by the incremental gradient method [13], which is expressed as iterations on $n$ by

$$\boldsymbol{v} := \boldsymbol{v} + \eta_n \nabla \tilde{\mu}(i, \boldsymbol{v}) \left( \bar{\mu}(i) - \tilde{\mu}(i, \boldsymbol{v}) \right), \quad \forall i \in \tilde{S} \tag{22}$$

where $\eta_n$ fulfills the conditions (18). A summary of the algorithm is detailed in Table 1.

Note that two approximation problems are solved at the same time, since given $\bar{\mu}(\cdot)$ it is evaluated to find $\tilde{J}^{\mu}(\cdot, r)$ of $J^{\mu}(\cdot)$ defined by $C(i)$ with $i \in \tilde{S}$.

Then, given $\tilde{J}^{\mu}(\cdot, \boldsymbol{r})$, the improved policy $\bar{\mu}(i)$ is computed for $i \in \tilde{S}$ and then find the new policy $\tilde{\mu}(\cdot, \boldsymbol{v})$. Once the function $\tilde{\mu}(\cdot, \boldsymbol{v})$ is available, the control actions are obtained as shown in Figure 5. The control scheme is shown in Figure 6.

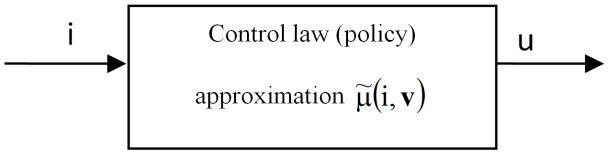


**Figure 5.** Compact expression of the policy or control law.

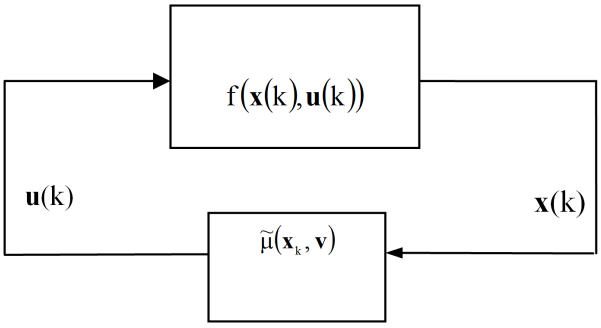


**Figure 6.** System and neurocontroller closed loop scheme.

**Table 1.** Approximate optimal controller calculation algorithm.

| Item | Do |
|---|---|
| 1 | Initialize: $\gamma_n$, Iterations, $h$, parameters $\boldsymbol{r}$ and $\boldsymbol{v}$. |
| 2 | Evaluate the initial policy $\tilde{\mu}(\cdot,\boldsymbol{v})$ through (16). |
| 3 | Update the parameters $\boldsymbol{r}$ via (17) |
| 4 | Compute functions $Q(\cdot)$ by (19) |
| 5 | Update policy $\bar{\mu}(\cdot)$ using (20) |
| 6 | Update the parameters $\mathbf{v}$ of $\tilde{\mu}(\cdot,\boldsymbol{v})$ by (22) |
| 7 | Update function $\gamma_n$. |
| 8 | Go to 2 and repeat items 2, 3, 4, 5, 6 and 7 until the complete the Iterations |

#### 6) *Discussion and comment on the implementation*

The algorithm to solve the optimal control problem for nonlinear processes with non-quadratic cost function and constraints was detailed. Given the employment of approximations, the topic of approximation function in dynamic systems [13] must be well mastered to obtain suitable result in the closed loop system.

As general suggestions, it must be mentioned that as in many nonlinear system, the algorithm is strongly dependent on the initial conditions. Thus, its dependence lies on the initial policy and on the states used to compute $\tilde{\mu}(\cdot,\boldsymbol{v})$, represented by set $\tilde{S}$.

The parameter tune speed with respect of the iterations, is fixed by the function $\gamma$, and the method is sensitive to this parameter. Usually one can make the first attempts setting $\gamma = 1$ constant, with few iterations, and then begin to modify it to converge to 1 with the iterations, always verifying that the performance of the controller improves at the long term. The adjustment parameters amount in each approximation function depends on the data complexity, which generally are conditioned by implementing some normalization or feature extraction techniques.

## 4. Control of the Inverted Pendulum

The inverted pendulum can be represented as shown **Figure 7**. For this cart-bar

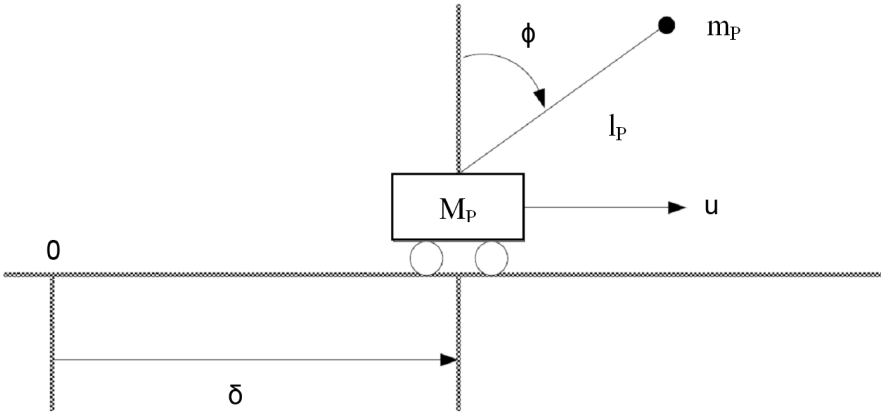


**Figure 7.** Diagram of the inverted pendulum system on a cart.

system, a controller will be designed using the algorithm of **Table 1**. Knowing that the equations that describe the angle and the linear displacement dynamics are,

$$\begin{cases} \left(M_P + m_P\right)\ddot{\delta} + m_P l_P \ddot{\phi}\cos\varphi - m_P l_P \dot{\phi}^2 \sin\phi + F_P\dot{\delta} = u \\ l_P\ddot{\phi} - g_P \sin\phi + \ddot{\delta}\cos\phi = 0 \end{cases} \tag{23}$$

whereas the controller is designed the system trajectories are generated by simulation for initial angle $\phi$ of 0.2 radians. It is considered that the force $u$ must fulfill with the constraint $-30 \le u \le 30$.

The proposed cost function is composed by

$$J\left(\boldsymbol{x},u\right) = \sum_{k=0}^{N} x_k^{\mathrm{T}} \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} x_k + 0.001\cdot\theta_u \tag{24}$$

where $\theta_u$ is defined to constrain the values of $u_k$ by

$$\theta_u = \begin{cases} \left|u_k - 30\right|, \text{ if } u_k > 30, \\ \left|-30 - u_k\right|, \text{ if } u_k < -30. \end{cases} \tag{25}$$

The continuous time model is discretized at a rate of 0.1 Section.

1) ***State estimation***

In order to retrieve the system state vector, a Kalman estimator is used, where the discrete-time linearized version estimate of (23) is given by

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}u_k + \boldsymbol{F}v_k \tag{26}$$

$$\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{G}w_k \tag{27}$$

where $\boldsymbol{x}^{\mathrm{T}} = \left[\delta, \dot{\delta}, \phi, \dot{\phi}\right]$, $u_k$ is the operating force as shows **Figure 7**, and the matrices $\boldsymbol{A} \in \Re^{4\times4}$ and $\boldsymbol{B} \in \Re^{4\times1}$ are obtained by discretizing the continuous time linear version of (23) with a sampling time of 0.1 Section For the case of the pendulum $\boldsymbol{v} \in \Re^4$, $\boldsymbol{w} \in \Re^1$, assuming that Gaussian sequences have zero mean and unit variance. The matrices $F$ and $G$ are defined as

$$F=\begin{bmatrix}\sigma_{11} & 0 & 0 & 0\\ 0 & \sigma_{22} & 0 & 0\\ 0 & 0 & \sigma_{33} & 0\\ 0 & 0 & 0 & \sigma_{44}\end{bmatrix} \tag{28}$$

with $\sigma_{ii}=1\times10^{-3}$ for the state vector $\boldsymbol{x}$, and

$$G=[\varsigma] \tag{29}$$

with $\varsigma=1\times10^{-3}$ for the measure $y(k)$. Furthermore, $\boldsymbol{C}$ = [1 0 0 0].
To find the $\boldsymbol{x}(k)$ estimate, it used a priori estimate of the observed states by means of

$$\hat{\boldsymbol{x}}^{-}_{(k+1)}=\boldsymbol{A}\hat{\boldsymbol{x}}_{(k)}+\boldsymbol{B}\boldsymbol{u}_{(k)} \tag{30}$$

and these states $\hat{\boldsymbol{x}}$ are obtained from measurements of the system output

$$\hat{\boldsymbol{x}}_{(k)}=\hat{\boldsymbol{x}}^{-}_{(k)}+K_O\left(y_{(k)}-C\hat{\boldsymbol{x}}^{-}_{(k)}\right) \tag{31}$$

where $K_O$ is the Kalman gain [15]. This gain is calculated by using a Gaussian noise model in the state $\boldsymbol{x}(k)$ and in the measurement $y(k)$ given by (26) and (27).

**2)** ***Implementing the controller***

The tune of the **Table 1** algorithm was done to achieve the control objective that is to bring the bar to the vertical position, starting from positions smaller than 1 radian. **Figure 10** shows the evolution of the algorithm parameters which are the cost to go function from the initial condition $[0\ 0\ 0.2\ 0]^{\mathrm{T}}$ to the final time, set in 10 sec. Note that the behavior is not stable in first half of the performed iterations, but then stabilizes and the control objective is achieved.

The set $\tilde{S}$ was of 3000 samples in the $\Re^4$ space of the state variables, with the range shown in **Figure 11**. The control law is

$$u(k)=\tilde{\mu}\left(\hat{\boldsymbol{x}}(k),\boldsymbol{v}\right) \tag{32}$$

where $\hat{\boldsymbol{x}}(k)$ is obtained from Equation (31), and $\boldsymbol{v}$ contains the parameters corresponding to a set as (9), where 7 hidden nodes were used, which gives 7 vectors $\boldsymbol{r}_1^i\in\Re^5$, and vector $\boldsymbol{r}_2\in\Re^8$, implementing the Equation (14). For the approximation of the function $f(\cdot)$ defined in Equation (24) it is used the same structure of the approximation function as for the control law. The parameters tuning was performed by the Levenberg Marquardt algorithm [5].

In order to perform the comparison of the neurocontroller performance, a time variant-discrete linear quadratic regulator controller with the classical LQR theory in discrete time [16] (TVDLQR) was implemented. Here, the design matrices were

$$\boldsymbol{Q}=\begin{bmatrix}10 & 0 & 0 & 0\\ 0 & 10^3 & 0 & 0\\ 0 & 0 & 10 & 0\\ 0 & 0 & 0 & 10^3\end{bmatrix},\quad \boldsymbol{S}=\begin{bmatrix}10 & 0 & 0 & 0\\ 0 & 10 & 0 & 0\\ 0 & 0 & 10 & 0\\ 0 & 0 & 0 & 10^2\end{bmatrix}\ \text{and}\ R=1. \tag{33}$$

3) *System simulation results under noise conditions*

Figure 8 and Figure 12 show both performance results of the neurocontroller and the TVDLQR, with the same Kalman estimator. Under no-noise conditions, the performance of the system in both cases are quite similar, as shown in Figure 1 and Figure 2. Nevertheless, under noise conditions the TVDLQR does not achieve the same performance as that of the neurocontroller since the last allows to increase the range of initial conditions from 0.19 to 0.47 rad as seen in Figure 13 and Figure 12. Furthermore, estimated variables used by the LQR controller are shown in Figure 9.

4) *Discussion*

Since the control objective of the system with estimator is that the pendulum does not fall, a qualitative analysis of the performance of the TVLQR and NC controllers can be inferred. As can be seen in the examples shown, in Figure 8 can be seen that the linear controller meets the control objective for initial conditions of 0.19 rad or less. In contrast, for the case of the NC in Figure 13 and Figure 12 shows that with initial conditions of 0.19 radians the linear controller does not meet the control objective, but the NC does. In Figure 10 can be see that the tuning parameter's procedure is erratic and difficult to adjust since the value of the costs associated with the control policies are not necessarily monotonically decreasing. This means that is hard to tune the algorithm of Table 1, which must be tuned by trial-test and error for each particular system. Also, in Figure 11 the range of the system samples used in the calculation of the NC to

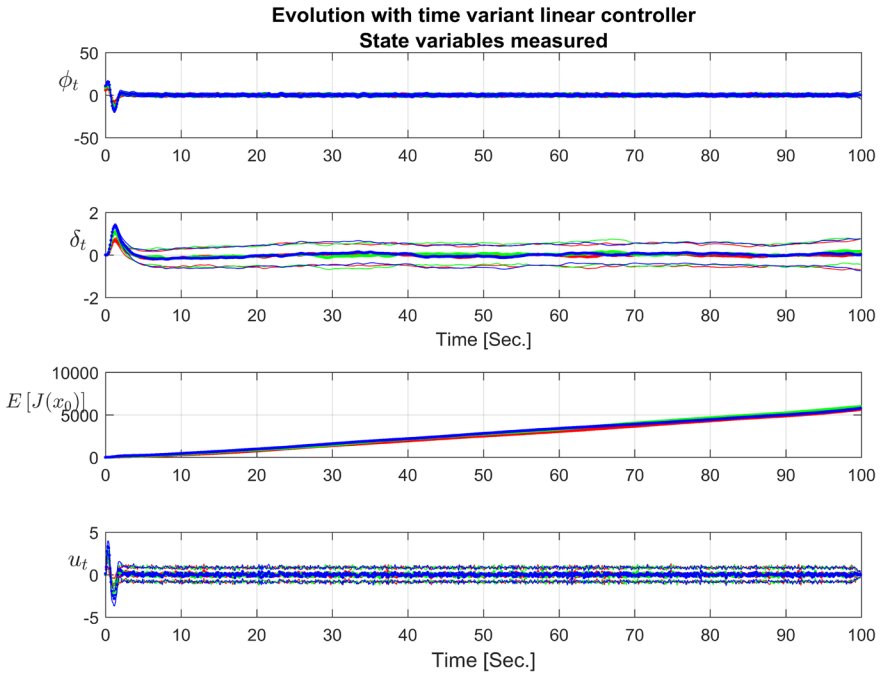


Figure 8. Output evolution of the inverted pendulum system controlled by the time variant DLQR with estimator and with noise in the state and in the measurement. The initial conditions are $\boldsymbol{x}_0 = [0\ 0\ \phi_0\ 0]^{\mathrm{T}}$, with $\phi_0$ taking values 0.1, 0.15 and 0.19 rad or 5.7296˚, 8.5944˚ and 10.8862˚.

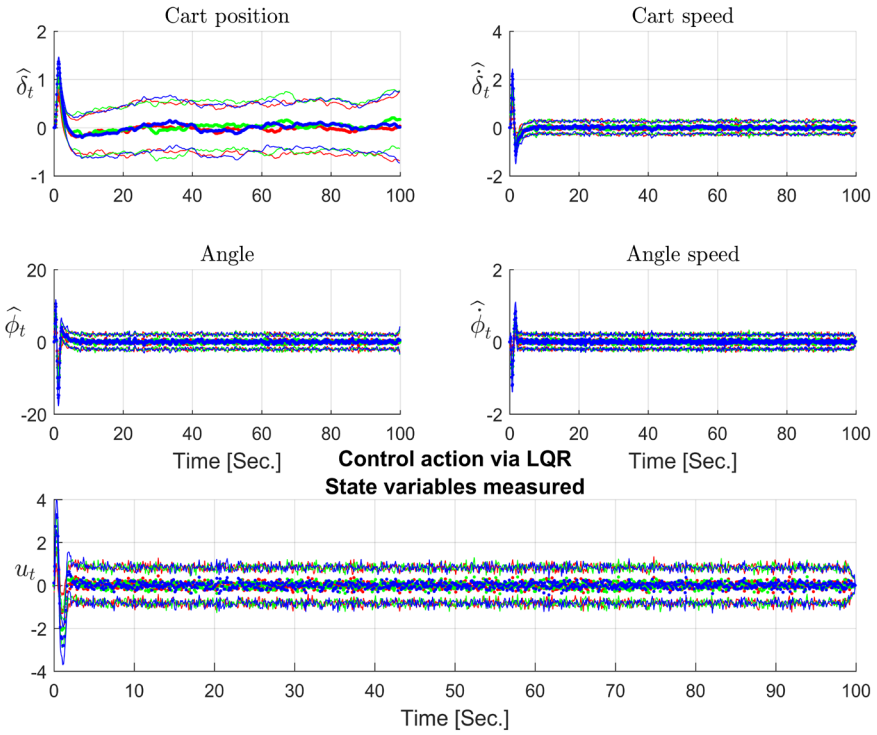


**Figure 9.** Estimated variables evolution for three initial conditions of the inverted pendulum system when it has noise in the states and in the measurement, with initial conditions of $\boldsymbol{x}_0 = [0\ 0\ \phi_0\ 0]^{\mathrm{T}}$, where $\phi_0$ takes the values 0.1, 0.15 , and 0.19 rad, *i.e.* 5.7296˚, 8.5944˚ and 10.8862˚.

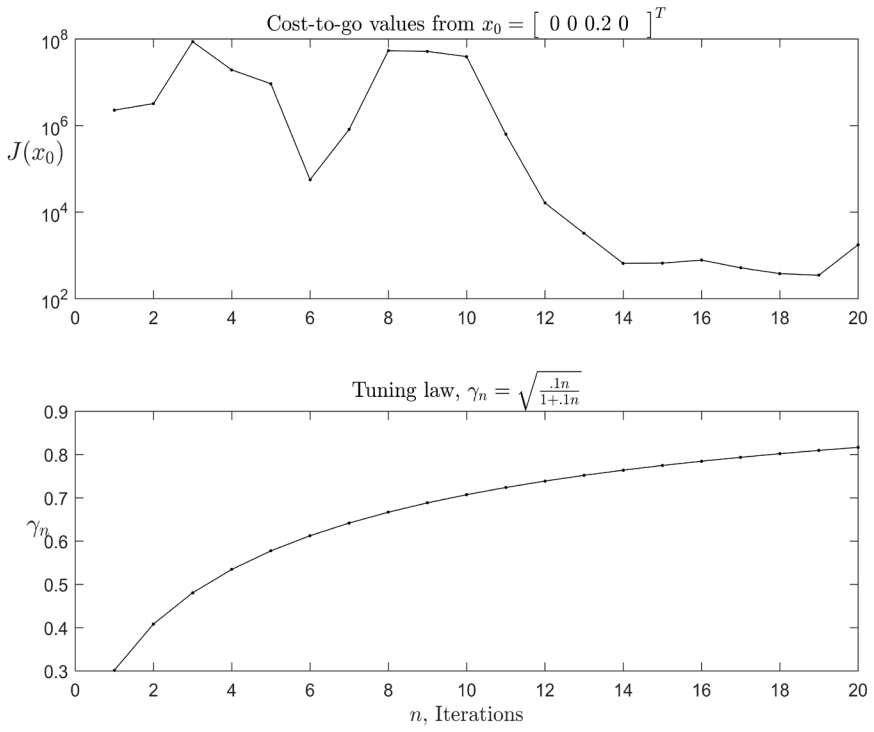


**Figure 10.** Cost-to-go function evolution associated with the initial condition $\boldsymbol{x}_0 = [0\ 0\ 0.2\ 0]^{\mathrm{T}}$ and tuning law evolution used for the neurocontroller calculation.

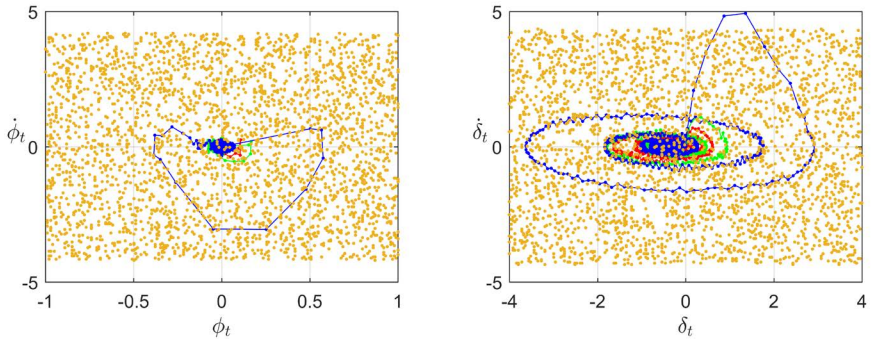


**Figure 11.** Phase planes for the initial conditions $\boldsymbol{x}_0 = [0\ 0\ \phi_0\ 0]^{\mathrm{T}}$ where $\phi_0$ takes the values 0.1, 0.15, and 0.47 rad *i.e.* 5.7296˚, 8.5944˚ and 26.9290˚. Continuous lines show the evolution of system trajectories under noise conditions in the state and in the measurement.

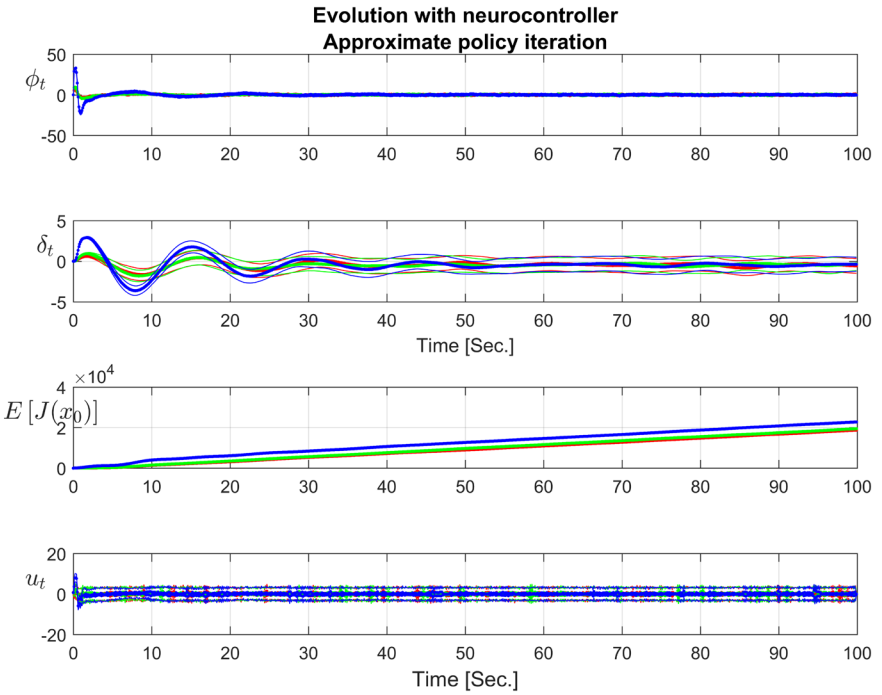


**Figure 12.** System evolution of the inverted pendulum with Kalman estimator neurocontroller. The initial conditions are $\boldsymbol{x}_0 = [0\ 0\ \phi_0\ 0]^{\mathrm{T}}$. $\phi_0 = \{0.1, 0.15, 0.47\}$ rad, *i.e.* {5.7296˚, 8.5944˚, 26.9290˚}.

implement the algorithm of Table 1 is detailed. Note that the system evolution can be out of range and the policy function must be able to give a response that stabilizes the system. This was achieved because of the suitable relation between the data set $\tilde{S}$ and the problem complexity. Therefore, the results are encouraging since in the examples the control objective is met, which is to obtain zero error in the output with respect to the desired value that is the origin. In a comparison of the performances from both control systems can be seen in Table 2. It is important to state that the linear controller is unable to keep the pendulum vertical when the angle initial condition exceeds 0.2 rad whereas the NC achieves it even up to 0.47 rad.

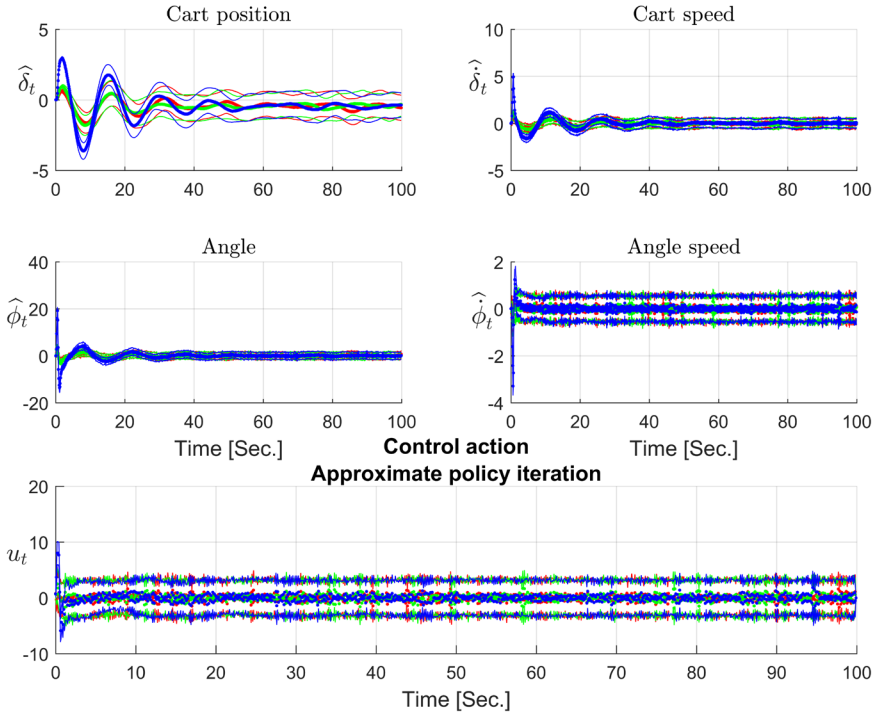


**Figure 13.** Evolution of the estimated variables of the inverted pendulum system with the Kalman estimator and neurocontroller. The initial conditions are $\mathbf{x}_0 = [0\ 0\ \phi_o\ 0]^T$. $\phi_0 = \{.1,.15,.47\}$ rad, *i.e.* {5.7296˚, 8.5944˚, 26.9290˚).

**Table 2.** Comparative figures of performance results obtained by both controller estimator systems. The Monte Carlo simulation was realized with 150 trays, along 1000 time steps of 0.1sec. **Figure 8** and **Figure 12** show the temporal evolution of these indices.

| | LQR | TVLQR | NC |
|---|---|---|---|
| $\phi_o$ [rad] | | Expected Cost to go | Expected Cost to go |
| 0.1 | 4282.3523 | 4346.1949 | 45574.606 |
| 0.15 | 4330.45 | 4394.2924 | 45595.4206 |
| 0.19 | 4396.1917 | 4460.034 | 45605.0514 |
| 0.47 | Too large | Too large | 45760.5241 |

The temporal evolution of these indices is shown in **Figure 8** and **Figure 12**, where the mean value and the 66% quota from the 150 trajectories of the Monte Carlo simulation are highlighted. In **Table 2** are resumed the performance achieved by each controller. Note that even with higher cost to go figures, the NC gives robust behavior with regards to the initial conditions and noise.

## 5. Conclusions

In this paper, a stability analysis of a neurocontroller with Kalman estimator for the inverted pendulum case was presented. The NC performance was compared against the TVLQR controller with the same Kalman estimator.

The obtained results can be stated that the use of approximate optimal control

implemented through Table 1, is a very powerful and promising tool to deal with controllers for nonlinear processes with Kalman estimator.

The initial angle range was possible to extend from 0.19 rad for the case of linear controller with estimator, to 0.47 rad in the present scheme simulating a Monte Carlo with 150 trajectories.

It is important to highlight that the technique requires a good mastery for functions approximation for dynamic processes, and simulation of natural process through numerical methods.

## References


[1] Sontag, E. (1998) Mathematical Control Theory, Deterministic Finite Dimensional Systems. Springer, USA, 104-115.

[2] Almobaied, M., Eksin, I. and Guzelkaya, M. (2016) Design of LQR Controller with Big Bang-Big Crunch Optimization Algorithm Based on Time Domain Criteria. *Control and Automation* (*Med*) 2016, 24*th Mediterranean Conference On*, Athens, Greece, June 21-24 2016, 1192-1197.

[3] Zhang, H.G., Liu, D.R., Luo, Y.H. and Wang, D. (2013) Adaptive Dynamic Programming for Control. Algorithms and Stability. Springer-Verlag, London. https://doi.org/10.1007/978-1-4471-4757-2

[4] Liu, D.R., Wei, Q.L., Wang, D., Yang, X. and Li, H.L. (2017) Adaptive Dynamic Programming with Applications in Optimal Control. Springer International Publishing AG, London. https://doi.org/10.1007/978-3-319-50815-3

[5] Bertsekas, D. and Tsitsiklis, J. (1996) Neuro-Dynamic Programming. Athena Scientific. The MIT Press, Cambridge, MA.

[6] Sutton, R.S. and Barto, A.G. (1998) Reinforcement Learning: An Introduction. In: Adaptive Computation and Machine Learning. The MIT Press, Cambridge, MA. https://doi.org/10.1109/TNN.1998.712192

[7] Anderson, B. and Moore, J. (1971) Linear Optimal Control. Prentice-Hall International Inc., London. https://doi.org/10.1115/1.3426525

[8] Ogata, K. (1997) Modern Control Engineering. Fifth Edition, Prentice Hall Inc., Englewood Cliffs, NJ, ISBN 0-13-615673-8.

[9] Ogata, K. (1995) Discrete-Time Control Systems. 2nd Ed., Prentice Hall, Englewood Cliffs, NJ.

[10] Bellman, R. and Dreyfus, S. (1962) Applied Dynamic Programming. Princeton University Press, Princeton, NJ. https://doi.org/10.1515/9781400874651

[11] Luus, R. (2000) Iterative Dynamic Programming. CRC Press, Inc., Boca Raton, FL, USA. https://doi.org/10.1201/9781420036022

[12] Bertsekas, D. and Tsitsiklis, J. (2006) Slides MIT. http://web.mit.edu/dimitrib/www/NDP_Review.pdf

[13] Kirk, D.E. (2004) Optimal Control Theory: An Introduction. Dover Publications, USA.

[14] Ljung, L. (1998) System Identification: Theory for the User. Prentice Hall PTR, USA. https://doi.org/10.1007/978-1-4612-1768-8_11

[15] Naidu, D.S. (2003) Optimal Control Systems. Electrical Engineering Handbook, CRC Press, Boca Raton, Florida, 275.

[16] Kalman, R.E. (1960) A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, **82**, 35-45. https://doi.org/10.1115/1.3662552