Article

Improved Forms for Controlling the Acrobot with Motors of Atypical Size Using Artificial Intelligence Techniques

Gonzalo Mier * and Javier de Lope *

Department of Artificial Intelligence, Faculty of Computer Science, Universidad Politécnica de Madrid, 28660 Madrid, Spain

* Correspondence: g.mier@alumnos.upm.es (G.M.); javier.delope@upm.es (J.L.)

Abstract: An acrobot is a planar robot with a passive actuator in its first joint. The control problem of the acrobot tries to make it rise from the rest position to the inverted pendulum position. This control problem can be divided in the swing-up problem, when the robot has to rise itself through swinging up as a human acrobat does, and the balancing problem, when the robot has to maintain itself on the inverted pendulum position. We have developed three controllers for the swing-up problem applied to two types of motors: small and big. For small motors, we used the SARSA controller and the PD with a trajectory generator. For big motors, we propose a new controller to control the acrobot, a PWM controller. All controllers except SARSA are tuned using a Genetic Algorithm.

Keywords: Acrobot; Artificial Intelligence; SARSA; PWM; Genetic Algorithm

---

1. Introduction

A planar robot is a robotic arm that moves on a plane. If a robot has less actuators than grades of freedom, the system is called underactuated. An underactuated planar robot with two rotational joints could be a pendubot ([1]), if the passive actuator is the second joint , or an acrobot ([2–11]), if the passive actuator is the first joint (Figure 1).

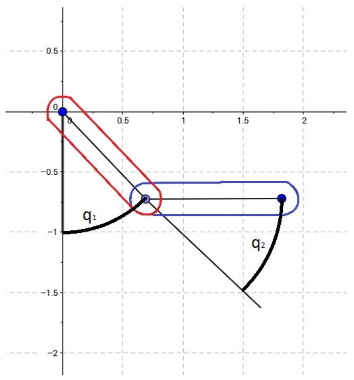


Figure 1. Image of a simulated acrobot. This robot has a passive actuator on the first joint (on red). The second joint is on blue. The angle of the first link is called q_1 and the angle of the second link is q_2.

The acrobot control problem is to make it rise from the rest position to the inverted pendulum position. As the first and second link of an acrobot are coupled, the control problem is split on two ([3,4,7,8]): The swing-up problem, in which the acrobot has to rise itself above one meter as a human acrobat does; The balancing problem, in which it has to maintain itself on the inverted pendulum position. Other researchers had solved the whole problem with only one controller ([5,9,10]).

For the swing-up problem, Duong [4] proposed to use a NeuroController, Spong [8] and Brown [3] designed a PD controller with a trajectory generator, and Mahindrakar [7] used an energy-based controller. The solutions proposed for the balancing problems use an LQR controller [3,4,7,8] or a fuzzy controller [3]. All the parameters of the controllers proposed on these papers were tuned by a Genetic Algorithm.

The whole acrobot problem was solved by Duong [5], as an extension of his previous work [4], with a NeuroController, and by Zhang [12] which used a spiking neural network with a LQR. Both used a Genetic Algorithm to improve the perform of their algorithms. However, Sutton [9] solved the whole problem using Reinforcement Learning. He proposed SARSA, an algorithm that learns by the trial and error method.

Those papers had proposed different methods to solve the acrobot problem avoiding the saturation of the motors. The controllers made by Sutton [9] (SARSA) and Spong [8] (PD controller) can handle with motors with low maximum torque, but none of them had tried to take advantage of the saturation on motors with high maximum torque.

In this paper we propose two variants of the SARSA algorithm [9]: to initialize the algorithm with zeros or non-zeros values and to introduce some error on the sensors of the acrobot. As the PD controller proposed by Spong [8] is not mathematically correct, we have also compared his PD controller with a technically correct PD controller. Finally, we propose a novel PWM controller that uses the saturation of a big motor to control the acrobot in any operation point of the first link.

2. Acrobot model

All the controllers were tested on a simulated acrobot. The parameters of our acrobot are:

Table 1. Parameters of the acrobot

Parameters	Real value	Meaning of the parameter
m_1	$1\ Kg$	Mass of the first link
m_2	$1\ Kg$	Mass of the second link
l_1	$1\ m$	Distance from the beginning to the end of the first link
l_2	$1\ m$	Distance from the beginning to the end of the second link
lc_1	$0.5\ m$	Distance from the beginning to the center of mass of the first link
lc_2	$0.5\ m$	Distance from the beginning to the center of mass of the second link
I_1	$1\ Kg * m^2$	Inertia of the first link
I_2	$1\ Kg * m^2$	Inertia of the second link
g	$9.8\ m/s^2$	Gravity

The equations of motion of the system are the same of a planar robot without the input torque on the first joint:

$$\begin{aligned} \overbrace{d_{11} * \ddot{q}_1 + d_{12} * \ddot{q}_2}^{M(q)\ddot{q}} + \overbrace{h_1 * \dot{q}_1}^{C(q,\dot{q})\dot{q}} + \overbrace{phi_1}^{G(q)} &= \overbrace{0}^{\tau} \\ d_{21} * \ddot{q}_1 + d_{22} * \ddot{q}_2 + h_2 * \dot{q}_2 + phi_2 &= \tau_2 \end{aligned} \tag{1}$$

Where M is the inertia matrix, C is the acceleration of Coriolis, G the gravitation terms, $\ddot{q}_1$ and $\ddot{q}_2$ are the accelerations (rad/s^2) of the first and second joint, $\dot{q}_1$ and $\dot{q}_2$ are the velocities (rad/s) of the first and second joint, q_1 and q_2 are the position (rad) of the first and second joint and τ_2 is the input torque on the second joint. The other terms are:

$$
\begin{aligned}
d_{11} &= m_1 * lc_1^2 + m_2 * (l_1^2 + lc_2^2 + 2 * l_1 * lc_2 * cos(q_2)) + I_1 + I_2 \\
d_{12} &= d_{21} = m_2 * (lc_2^2 + l_1 * lc_2 * cos(q_2)) + I_2 \\
d_{22} &= m_2 * lc_2^2 + I_2 \\
\\
h_1 &= -m_2 * l_1 * lc_2 * \dot{q_2}^2 * sin(q_2) - 2 * m_2 * l_1 * lc_2 * \dot{q_2} * \dot{q_1} * sin(q_2) \\
h_2 &= m_2 * l_1 * lc_2 * \dot{q_1}^2 * sin(q_2) \\
\\
phi_2 &= m_2 * lc_2 * g * cos(q_1 + q_2 - pi/2) \\
phi_1 &= (m_1 * lc_1 + m_2 * l_1) * g * cos(q_1 - pi/2) + phi_2
\end{aligned}
\tag{2}
$$

The control time used for all the controllers is $T = 50ms$. The calculus of q_1 and q_2 is performed 4 times between one control input and the next one. The position of the first and second joint (q_1,q_2) are between $[0, 2\pi)$, the velocity of the first joint ($\dot{q}_1$) is between $[-4\pi, 4\pi]$ and the velocity of the second joint ($\dot{q}_2$) between $[-9\pi, 9\pi]$.

To control an acrobot with this parameters, a motor with a maximum torque of $\pm 10Nm$ can be used. We used smaller motors for the SARSA and the PD controller ($\pm 1Nm$), and bigger for the PWM controller ($\pm 300Nm$). The sensors are simulated, with an added error of the 5% of the range of each dimension (only used for the SARSA controller). The sensors are thus able to read the position and the velocities of the first and the second joint, but not the accelerations.

3. Control methods

3.1. SARSA controller

SARSA [9] is a reinforcement learning algorithm that can learn through experience (on-line learning), which can take the best action at each moment to achieve its goal. Generally it is used a table $Q(s, a)$ that associates states and actions, which stores the weights of how good an action a is at a state s. The SARSA algorithm is shown in Alg. 1

Algorithm 1: SARSA algorithm

```
1  Initialize $Q(s,a)$ arbitrarily.
2  for (each episode:) do
3      Initialize $s$
4      Choose $a$ for $s$ using policy derived from $Q$
5      for each step of episode do
6          Take action $a$, observe $r$, $s'$
7          Choose $a'$ for $s'$ using policy derived from $Q$
8          $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$
9          $s \leftarrow s'$;
10         $a \leftarrow a'$;
11     end
12 end
```

To create $Q(s, a)$ is necessary to discretize the range of each dimension in parts, to make the infinite range of the values a finite range of possible states. The combination of the discretized dimensions is called tiling. It is possible to use more than one tiling, changing the updating rule for:

$$Q_i(s,a) \leftarrow Q_i(s,a) + \alpha[\sum_j (r + \gamma Q_j(s',a') - Q_j(s,a))] \tag{3}$$

Sutton [9] used 48 tiles (12 with 4 dimensions, 12 with 3 dimensions, 12 with 2 dimensions and 12 with 1 dimension). The position and the velocity range are split in 6 intervals, but the dimensions of the velocities are offset by a random fraction of interval, so they have 7 intervals.

As we have used a small motor with a maximum torque of $\pm 1Nm$, the only possible actions are $\{-1, 0, 1\}$. The reward r used is -1 until the end of the acrobot is above 1 meter. For the election of the action a, a greedy policy (with $\epsilon = 0$) is used, because a bad move could end a set of good moves. The other two parameters are $\alpha = 0.2/48$ and $\gamma = 1$.

In section 4.1, we have compared the results (Fig. 2) obtained when $Q(s, a)$ are initialized with zero (Fig. 2a and 2b) or non-zero (Fig. 2c and 2d) values, as two possible ways to initialize $Q(s, a)$ arbitrarily, and when an error of 5% of the range on q_1, q_2, $\dot{q}_1$ and $\dot{q}_2$ (Fig. 2b and 2d) are introduced to the sensors.

3.2. PD controller

Spong [8] proposed a PD controller to solve the swing-up problem:

$$\tau_2 = \overline{d}_{22}\left[K_p\left(\alpha\, arctan(\dot{q}_1) - q_2\right) - K_d\, \dot{q}_2\right] + \overline{h}_2 + \overline{\phi}_2 \tag{4}$$

where the terms $\overline{d}_{22}$, $\overline{h}_2$ y $\overline{\phi}_2$ are:

$$\overline{d}_{22} = d_{22} - d_{12}d_{11}^{-1}d_{12} \tag{5}$$

$$\overline{h}_2 = h_2 - d_{12}d_{11}^{-1}h_1 \tag{6}$$

$$\overline{\phi}_2 = \phi_2 - d_{12}d_{11}^{-1}\phi_1 \tag{7}$$

This equation is not exactly a PD controller. The canonical controller used is:

$$\tau_2 = \overline{d}_{22}\left[K_p\left(q_2^d - q_2\right) + K_d\left(\dot{q}_2^d - \dot{q}_2\right)\right] + \overline{h}_2 + \overline{\phi}_2 \tag{8}$$

If the torque compensation is taken apart:

$$v_2 = K_p\left(q_2^d - q_2\right) + K_d\left(\dot{q}_2^d - \dot{q}_2\right) \tag{9}$$

However, a PD controller have the form:

$$u_k = K_p * e_k + K_d * \frac{e_k - e_{k-1}}{t_k - t_{k-1}} \tag{10}$$

where $e_k = q_2^d - q_2$. When this two formulas are equated to compare if there are differences:

$$u_k = v_2 \tag{11}$$

$$K_{p_B}(e_k) + K_{d_B}\frac{d(e)}{dt} = K_{p_B}(q_2^d - q_2) + K_{d_B}(\dot{q}_2^d - \dot{q}_2) \tag{12}$$

$$K_{d_B}\frac{d(e)}{dt} = K_{d_B}(\dot{q}_2^d - \dot{q}_2) \tag{13}$$

$$\frac{d(e)}{dt} = \dot{q}_2^d - \dot{q}_2 \tag{14}$$

$$\frac{e_k - e_{k-1}}{t_k - t_{k-1}} = \dot{q}_2^d - \frac{q_{2_k} - q_{2_{k-1}}}{t_k - t_{k-1}} \tag{15}$$

$$\frac{(q_{2_k}^d - q_{2_k}) - (q_{2_{k-1}}^d - q_{2_{k-1}})}{t_k - t_{k-1}} = \dot{q}_2^d - \frac{(q_{2_k} - q_{2_{k-1}})}{t_k - t_{k-1}} \tag{16}$$

$$\frac{q_{2_k}^d - q_{2_{k-1}}^d}{t_k - t_{k-1}} = \dot{q}_2^d \tag{17}$$

$$\frac{q_2^d}{dt} = (\frac{q_2}{dt})^d \tag{18}$$

We obtain that the last equation 18 is not always true. In the equation 4, the desired velocity (right part of 18) of the second joint $\dot{q}_2^d$ has a constant value ($= 0$), meanwhile the classic PD controller computes that value as the derivate of the error (left part of 18).

Both controllers are tuned by a genetic algorithm. A genetic algorithm searches which are the best values that reduce a fitness function. The search is made by a population of n individuals. Those individuals are formed by the values of the function we want to optimize. In each iteration (or generation), the best individuals produce more individuals, by mix or mutation operators, and the worst individuals disappear.

We have defined a fitness function in terms of the time needed for the acrobot to get a threshold above its base. As we previously said, the links length is $l_i = 1$ m (i=1,2), so we are using a threshold $t = 1$ m . The best third part of each population is maintained for the next generation. A linear ranking selection selects a third part of the population for reproduction, in order to obtain another third part of the next population. The mix operator used is a BLX-α with $\alpha = 0.5$. The last third part of the population is created randomly. The mutation operator is not being used because the catastrophic and the BLX-α operator gave enough diversity to the algorithm.

The population has 60 individuals, 40 iterations to converge and 60 seconds to achieve the goal of going over one meter. Each individual has 3 values that correspond to [α, K_p, K_d], whose values are between ([0,1],[0,60],[0,5]). These are the initialization values, but the algorithm can search outside these boundaries without restriction.

3.3. PWM controller

The PWM controller is the novel method that we propose to control an acrobot with a big motor in the less time possible. We use the same genetic algorithm as previously. The maximum torque of the big motor needed is, at least, $\pm 200 Nm$, but we have used one with $\pm 300 Nm$. The structure of this controller is a torque compensation with 4 PI controllers, one for each dimension. The formula is:

$$v_2 = \sum_{x=1}^{4} [K_p^x * e_k^x + K_i^x * \sum_{i=0}^{k} e_i^x] \tag{19}$$

$$\tau_2 = \bar{d}_{22}\, v_2 + \bar{h}_2 + \bar{\phi}_2 \tag{20}$$

where x are the dimensions $[q_1, q_2, \dot{q}_1, \dot{q}_2]$, K_p^x and K_i^x are the constant of the controller of the dimension x and e_k^x is the error of the dimension x (equal to the desired value less the real value). The desired values for $[q_2, \dot{q}_1, \dot{q}_2]$ are 0, while the value for q_1 could be any value (in this case, π).

The population of the genetic algorithm is coded in 8 real parameters that are between $[0, 10^5]$. This controller learned how to solve the balancing problem, so the acrobot starts on $q_1 = pi - 0.01$ and $q_2 = 0$. If the end of the acrobot falls below the origin (0 m), it means that the fitness is really high (10^6). To that number, we subtract the time that the acrobot was above that line. An advantage of using a linear ranking selection is that the magnitude order of the fitness has not to be the same. Using this method, the Genetic Algorithm will rank the individuals, choosing as the best individuals those who can stand above the base of the acrobot all of the time, and then choosing those who can stand longer above the base of the acrobot. Those controllers that can solve the balancing problem have a fitness equal to:

$$fitness = [w_1, w_2, w_3, w_4, w_5] * [\sum_{i=0}^{k} x_2^d - x_2^i, \sum_{i=0}^{k} y_2^d - y_2^i, \sum_{i=0}^{k} x_1^d - x_1^i, \sum_{i=0}^{k} y_1^d - y_1^i, \sum_{i=0}^{k} |F_i|]^T \tag{21}$$

Where x_j^i is the value of the axis x of the j^{th} link at the i^th step of time, y_j^i is the value of the axis y of the j^{th} link at the i^th step of time, x_j^d is the desired value of the axis x of the j^{th} link, y_j^d is the desired value of the axis y of the j^{th} link, F_i is the input torque at the i^th step of time and w_j are the weights of preference to reduce. The desired position is $x_1 = 0$, $y_1 = 1$, $x_2 = 0$ and $y_2 = 2$. The weights used are $[1, 10, 0.1, 0.1, 10^{-5}]/(\text{n}^\text{o}$ of steps)

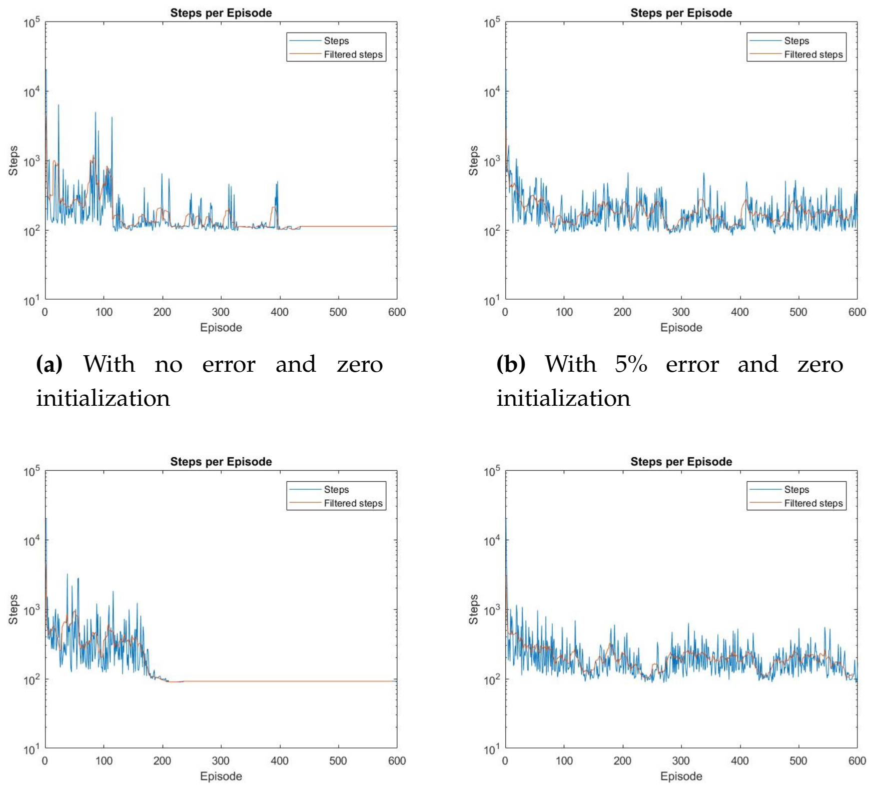
This controller is capable of controlling the acrobot in all the range and to maintain it in that position. Thus, the swing-up problem can be solved without swinging-up the acrobot, only using the half of the range. This paper proposes this controller as an emergency method for the robot if the path for going up is blocked. On the one hand, it could control the acrobot quickly to follow a reference for the first link and maintain it on any point on a dynamic equilibrium. On the other hand, this controller has big oscillations in the output. Using this method to swing-up and then change to a LQR controller for the balancing problem is then recommended.

4. Results

We have made a simulated acrobot to test the controllers. The simulated acrobot was tested with the SARSA, PD and PWM algorithms. On SARSA, we have compared how fast the algorithm learns how to solve the swing-up problem with different variations. The PD proposed by Spong [8] and a technically correct PD controller were compared between themselves. The new controller that we propose, the PWM controller, was tested to observe the performance solving the swing-up problem.

4.1. SARSA

The SARSA algorithm was tested 10 times in each experiment in order to obtain significant results of each one. Four experiments were compared: the SARSA algorithm with no error on sensors and having the matrix $Q(s, a)$ initialized with zero values; with an added error of the 5% to the output of the sensors; having the matrix $Q(s, a)$ initialized with non-zero values; with an added error and having $Q(s, a)$ initialized with non-zero values. The results are depicted in Fig. 2.



(a) With no error and zero initialization

(b) With 5% error and zero initialization

(c) With no error and random initialization

(d) With 5% error and random initialization

Figure 2. SARSA graphs of the learning rates. In each episode (x axis), the acrobot begin on the rest position. The acrobot lasts k steps (y axis) to solve the swing-up problem. If the value of steps is lower on a higher episode, the algorithm is learning. The graphs 2a and 2c have no error on the measured values, while 2b and 2d have an added error of the 5% of the range in all the dimensions. 2a and 2b are initialized with zero values, but 2c and 2d are randomly initialized. The graph on the y axis is on logarithmic scale.

Those experiments which have an added error (Fig 2b and 2d) do not converge, but the algorithm, in this case, is more robust to changes. Those controller that do not have an added error (Fig 2a and 2c) converge without any noise. In the case of the random initialization (Fig 2c and 2d), it converges before, but it needs more steps to achieve the goal. Finally, the initialization with zero values (Fig 2a and 2b) is smoother and it takes less time to go above one meter.

4.2. PD controller

The value of the gains for the PD controller of Spong [8] are $\alpha = 1.93$, $K_p = -15.9289$ and $K_i = -14.7768$. Despite the range of the population didn't have negative values, the BLX-α operator found better values outside this range.

The behaviour of this controller is the following:

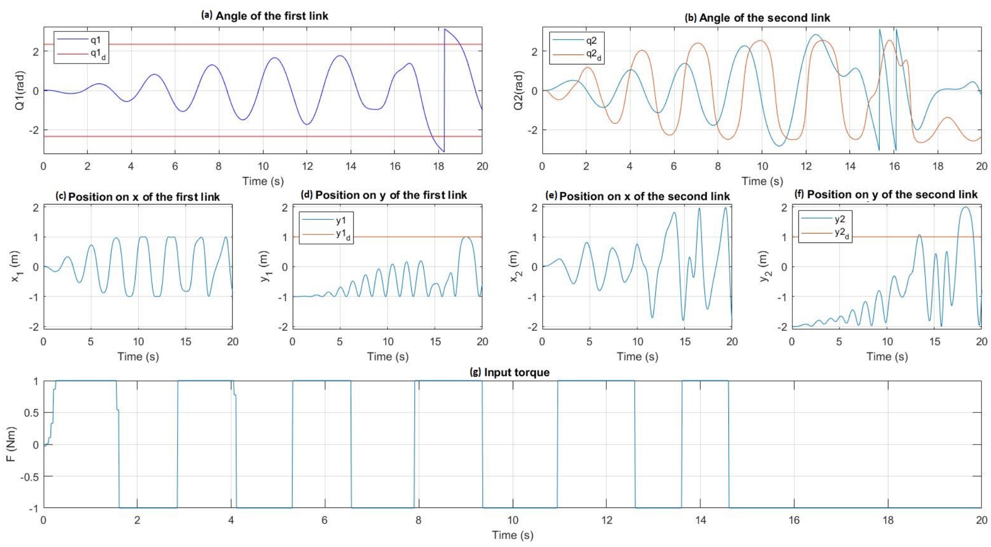


Figure 3. Results of the PD controller by Spong to the swing-up problem. On the first row, it is shown the position of the angle of the first (Fig 3a) and the second (Fig 3b) link on radians. The red line on (Fig 3a) is the moment when the LQR controller starts to control for the balancing problem. The red line on (Fig 3b) is the desired angle of the second link. On the second row, there are the position on x (Fig 3c) and y (Fig 3d) of the first link and on x (Fig 3e) and y (Fig 3f) of the second link. The red line on (Fig 3d) and (Fig 3f) are the goal height (1m) of the swing-up problem. On the last row, the input torque is represented (Fig 3g).

Almost all the time, the torque is saturated (Fig 3g) because of the low maximum torque. The controller lasts 13.3 s (Fig 3f) to achieve the goal, but the controller LQR should begin to control the acrobot at 17.5 s, when the red lines of the first graphic (Fig 3f) are exceeded (Those lines are at $\pi - 0.8$ and $-\pi + 0.8$). Fortunately, at 17.5 s the acrobot is in a good position to be controlled by the LQR controller.

The conventional PD controller has the gains $\alpha = -0.98$, $K_p = 8.65$ and $K_i = -0.31$ and the results are:

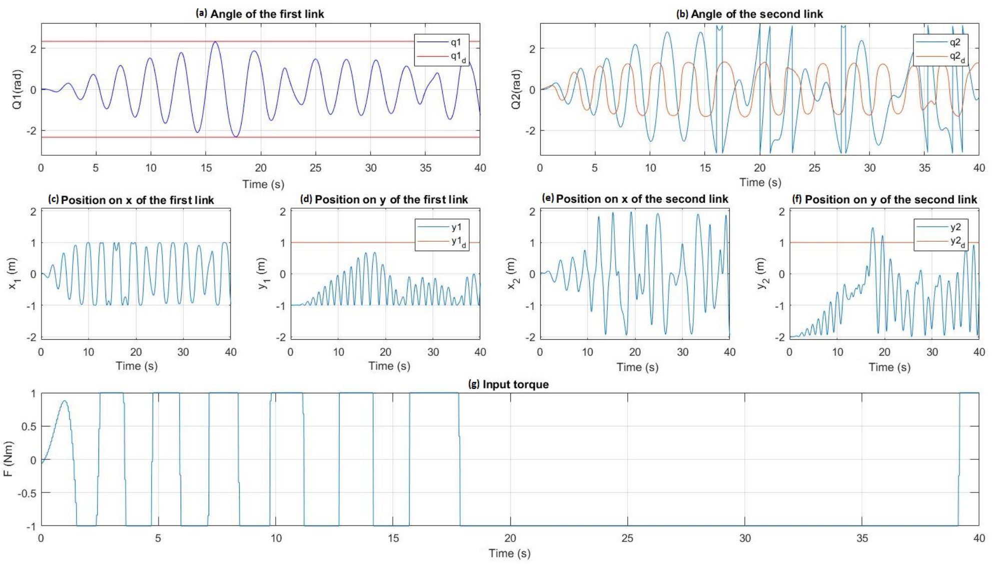


Figure 4. Results of the conventional PD controller to the swing-up problem. On the first row, it is shown the position of the angle of the first (Fig 4a) and the second (Fig 4b) link on radians. The red line on (Fig 4a) is the moment when the LQR controller starts to control for the balancing problem. The red line on (Fig 4b) is the desired angle of the second link. On the second row, there are the position on x (Fig 4c) and y (Fig 4d) of the first link and on x (Fig 4e) and y (Fig 4f) of the second link. The red line on (Fig 4d) and (Fig 4f) are the goal height ($t = 1$ m) of the swing-up problem. On the last row, the input torque is represented (Fig 4g).

The first time the controller achieves the goal is at the 18th second (Fig 4f) , but the LQR can't control on that point because the red line of the first graph (Fig 4a) has not been exceeded. The conventional PD controller can't control the acrobot as well as the PD controller of Spong [8] with such a low maximum torque.

4.3. PWM controller

The gains obtained by the Genetic Algorithm are:

Table 2. Value of the gains of the PWM controller

	K_p	K_i
q_1	$3.913 * 10^4$	$-3.239 * 10^4$
q_2	$1.278 * 10^4$	$-1.644 * 10^4$
dq_1	$1.02 * 10^5$	$-7.094 * 10^4$
dq_2	$1.647 * 10^4$	$4.668 * 10^4$

These gains have the same order of magnitude as the ones we expected. The negative gains K_i are not usual in the classical control, but the Genetic Algorithm found that those are the best values (probably a local optimum) for this problem. The result of this controller is:

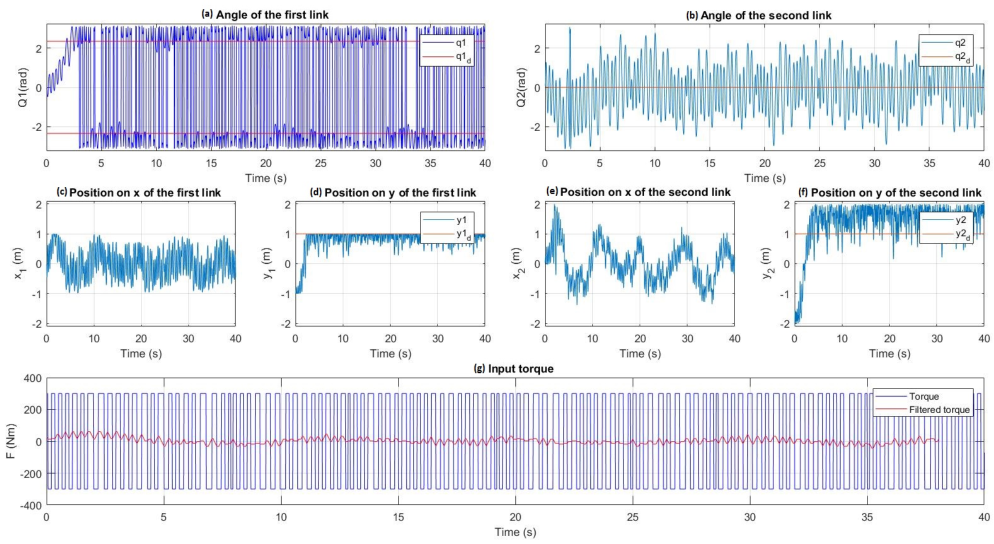


Figure 5. Results of the PWM controller to solve the acrobot problem. On the first row, it is shown the position of the angle of the first (Fig 5a) and the second (Fig 5b) link on radians. The red line on (Fig 5a) is the moment when the LQR controller starts to control for the balancing problem. The red line on (Fig 5b) is the desired angle of the second link. On the second row, there are the position on x (Fig 5c) and y (Fig 5d) of the first link and on x (Fig 5e) and y (Fig 5f) of the second link. The red line on (Fig 5d) and (Fig 5f) are the goal height (1m) of the swing-up problem. On the last row, the graph (Fig 5g) shows in blue the applied torque of the motor, and on red the same torque after a low-pass filter (the average of 10 values). By this method, it is possible to see the effect of the PWM, changing the frequency of the input signal.

The controller takes 3 seconds (Fig 5a) to surpass the red line, where the LQR controller starts to handle the acrobot. Also, the acrobot is at good position for the balancing problem, as the end of the acrobot are close to the two meters on the x axis (Fig 5f) and the angle of the second link is close to 0 (Fig 5b). On the second link's position in the y axis (Fig 5f), most of the time the acrobot is above one meter so this controller is also valid to solve the balancing problem.

Another advantage of this controller is being able to maintain the acrobot on impossible points until now, as the $q_1 = \pi/2$ and $q_2 = 0$, on horizontal position (Fig 6). In this operation point, the acrobot cannot be motionless. The PWM controller can handle with the two joints at the same time and control the acrobot in this point.

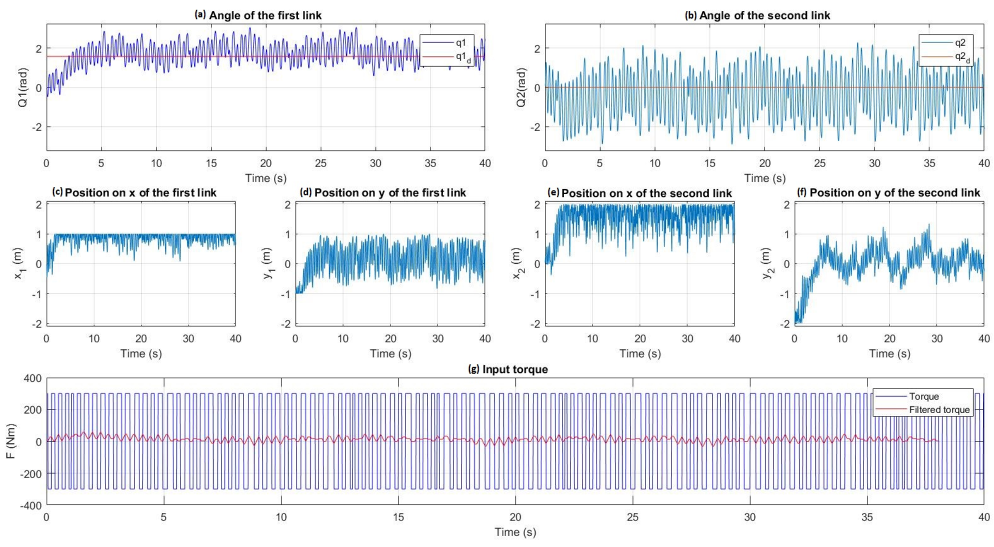


Figure 6. Results of the PWM controller to maintain the acrobot horizontal. On the first row, it is shown the position of the angle of the first (Fig 6a) and the second (Fig 6b) link on radians. The red line on (Fig 6a) is the moment when the LQR controller starts to control for the balancing problem. The red line on (Fig 6b) is the desired angle of the second link. On the second row, there are the position on x (Fig 6c) and y (Fig 6d) of the first link and on x (Fig 6e) and y (Fig 6f) of the second link. The red line on (Fig 6d) and (Fig 6f) are the goal height (1m) of the swing-up problem. On the last row, the input torque is represented (Fig 6g).

The task of maintaining the acrobot horizontal is achieved as it is shown on the graph of the position on the x axis of the first (Fig 6c) and second (Fig 6e) link. The amplitude of the oscillations can be reduced with a lower time of control, but cannot be canceled.

5. Conclusions

In this paper, we prove that the acrobot can be controlled with a very low actuation through SARSA and a modified PD, but not by a conventional PD controller. Also, we have explored two variants of SARSA and we found that the algorithm with a low error is not able to converge, and if the $Q(s,a)$ is initialized randomly the algorithm converges faster but found worst solutions that if the initialization is with zero values.

Furthermore, we have developed a novel PWM controller to control the acrobot with a big motor. This controller has big oscillations, but it lets the acrobot solve the whole problem in the fastest way possible so far. With this new controller it is possible as well to maintain the acrobot in new operation points.

As further work we are exploring new controllers to improve the performance our novel PWM controller, reducing the oscillations and the maximum torque needed.

References

1. Fantoni, I.; Lozano, R.; Spong, M. Energy based control of the Pendubot. *IEEE Transactions on Automatic Control* **2000**, *45*, 725–729.
2. Boone, G. Minimum-time control of the Acrobot. *Proceedings of International Conference on Robotics and Automation* **1997**, *4*, 3281–3287.

3. Brown, S.C.; Passino, K.M. Intelligent Control for an Acrobot. *Journal of Intelligent and Robotic Systems: Theory and Applications* **1997**, *18*, 209–248.
4. Duong, S.C.; Kinjo, H.; Uezato, E. A Switch Controller Design for the Acrobot Using Neural Network and Genetic Algorithm **2008**. pp. 17–20.
5. Duong, S.C.; Kinjo, H.; Uezato, E.; Yamamoto, T. On the continuous control of the acrobot via computational intelligence. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **2009**, *5579 LNAI*, 231–241.
6. Tedrake, R.; Seung, H.S. Improved Dynamic Stability Using Reinforcement Learning. *International Conference on Climbing and Walking Robots (CLAWAR)* **2002**, pp. 341–348.
7. Mahindrakar, A.D.; Banavar, R.N. A swing-up of the Acrobot based on a simple pendulum strategy. *International Journal of Control* **2005**, *78*, 424–429.
8. Spong, M.W. The Swing Up Control Problem For The Acrobot. *IEEE Control Systems* **1995**, *15*, 49–55.
9. Sutton, R.S. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in neural information processing systems* **1996**, pp. 1038–1044.
10. Wiklendt, L.; Chalup, S.; Middleton, R. A small spiking neural network with LQR control applied to the acrobot. *Neural Computing and Applications* **2009**, *18*, 369–375.
11. Yoshimoto, J.; Ishii, S.; Sato, M.a. Application of reinforcement learning to balancing of acrobot 3 Nara Institute of Science and Technology 33 ATR Human Information Processing Research Laboratories. *Science And Technology* **1999**.
12. Zhang, A.; She, J.; Lai, X.; Wu, M. Global Stabilization Control of Acrobot Based on Equivalent-Input-Disturbance Approach. *Control* **2011**, pp. 14596–14601.
13. Barto, A.G.; Sutton, R.S.; Anderson, C.W. Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems. *IEEE Transactions on Systems, Man and Cybernetics* **1983**, *SMC-13*, 834–846.