Construction, Control Design and Bluetooth Trajectory Control of a Self-Balancing Robot

Giulio Cesare Mastrocinque Santo*, Claudio Garcia**

*Departamento de Engenharia de Telecomunicações e Controle
Escola Politécnica da Universidade de São Paulo, São Paulo, SP (Tel: 11982074101; e-mail: giulio.santo@usp.br)
** Departamento de Engenharia de Telecomunicações e Controle
Escola Politécnica da Universidade de São Paulo, São Paulo, SP (Tel: 1130915648; e-mail: clgarcia@lac.usp.br)

Abstract: This paper is concerned with the multivariable discrete-time control problem of a self-balancing robot, considering the actuator in both wheels independently. A pole placement and a Linear Quadratic Regulator (LQR) controller are designed for the regulation problem and extended to the unidirectional and bidirectional trajectory tracking problems. Sensor noise filtering is discussed, and first order filters are applied. A Kalman Filter is implemented for sensor fusion. Robot construction and implementation aspects are also discussed in order to allow the use of the algorithms presented. The bidirectional control problem is accomplished through a Bluetooth device in such a way that this application could easily be extended to more sophisticated trajectory control problems.

Resumo: Este artigo aborda o problema de controle discreto multivariavel de um robô auto equilibrante, considerando as ações de controle nas duas rodas de forma independente. Um algoritmo de alocamento de polos, bem como um Controlador Linear Quadrático são implementados para o problema de regulação e posteriormente estendidos para o problema de rastreamento de trajetórias unidirecionais e bidirecionais. O problema de filtragem de ruído de medição é discutido e filtros de primeira ordem são aplicados. Um Filtro de Kalman é usado para realizar a fusão de sensores. Aspectos da construção do robô e de implementação dos algoritmos também são discutidos de forma a permitir o uso dos algoritmos apresentados. O controle bidirecional é realizado através de um dispositivo de Bluetooth, de forma que esta aplicação pode ser facilmente estendida para problemas mais sofisticados de controle de trajetória.

Keywords: Pole Allocation; Bluetooth Trajectory Control; Linear Quadratic Regulator; Self-Balancing Robot; Sensor Fusion; Control Systems.

Palavras-chaves: Alocação de Polos; Controle de Trajetória por Bluetooth; Controle Linear Quadrático; Filtro de Kalman; Robô Auto Equilibrante; Fusão de Sensores; Sistemas de Controle.

1. INTRODUCTION

A self-balancing robot can be considered a prototype of the Segway HT personal transporter. Moreover, this two-wheeled robot is a variation of the inverted pendulum, widely studied in control systems theory and, therefore, has great interest in the scientific community.

Quite a few works have been done concerning the control problem of the balancing robot. Juang et al. (2013) present a PID and an LQR-based PI-PD control design. A similar work can be found in (Martins; Nunes, 2017). In (Lam et al. 2001) a two-wheeled mobile robot (WMR) is controlled using a fuzzy model. Kim et al. (2017) develop a nonlinear optimal control for an underactuated inverted pendulum mobile platform. In the work of Ali et al. (2018), $H_2$ and $H_{\infty}$ controllers are designed. Kim et al. (2018) present a position-tracking controller using invariant dynamic surface. A model predictive control is applied to an unstable heavy self-balancing robot by Okulski et al. (2018).

Herein, multivariable discrete-time controllers (pole placement and LQR) are developed and they are applied to unidirectional and bidirectional trajectories tracking. The tracking problem is implemented using a Bluetooth device, so that the bidirectional trajectory can be sent to the robot by a computer or by a portable device like a smartphone in a joystick fashion. Therefore, the problem discussed in this work can be easily extended to more advanced control problems. In addition, practical aspects of implementation are discussed, in order to allow one to use the proposed algorithms.

2. PLANT MODEL

A self-balancing robot was built and construction features are addressed. A Lagrangian model proposed by Goes et al. (2018) is adopted to model the robot in two dimensions. The control actions were taken as independent and the yaw angle was included, allowing to control the robot in the 2D plane of motion. Fig. 1 represents the adopted model: $C.M$ is the Center of Mass of the robot; $L$ is the distance between the $C.M$ and the origin (wheel axle); $\theta_p$ is the pitch angle, $\alpha$ is a generic yaw angle, $R_w$ is the wheel radius, $m_w$ is the wheel mass and $J_w$ is the wheel moment of inertia. Goes et al. (2018) apply two different coordinate systems: system $\{0\}$ is
fixed and represented by vectors (i,j,k) and system {1} that moves along with C.M, represented by (x,y,z).

Fig. 1. Robot free body diagram.

The model was derived based on the Lagrangian method, resulting in Lagrange Equation (1):

\[ L = \frac{1}{2}(I_{w} + m_{w}R_{w}^2)(\omega_{r}^2 + \omega_{l}^2) + \frac{1}{2}(\omega_{p}^2 I_{xx}) \\
+ \frac{1}{2}R_{w}^2 \left( l_{yy} \cos^2(\theta_{p}) + l_{zz} \sin^2(\theta_{p}) \right)(\omega_{r} - \omega_{l})^2 \\
+ \frac{1}{2}m_{p} \left[ \frac{LR_{w}}{D_{w}} (\omega_{r} - \omega_{l})^2 \sin^2(\theta_{p}) + L^2 \omega_{p}^2 \right] \\
+ \frac{1}{2}m_{p} \left[ \frac{1}{4}(\omega_{r} + \omega_{l})^2 R_{w}^2 + LR_{w}\omega_{p}(\omega_{r} + \omega_{l})\cos(\theta_{p}) \right] \\
- m_{p}l_{w}g \cos(\theta_{p}) \]

where \( \omega_{p} \) is the pitch angular velocity of the robot; \( I_{xx}, I_{yy} \) and \( I_{zz} \) are the inertial moments about the centre of mass; \( m_{p} \) is the robot (C.M) mass; \( D_{w} \) is the distance between the two wheels and \( g \) is the gravitational constant.

Variable changes were made in order to control \( y \) and \( \alpha \) positions, as presented in (2) and (3).

\[ y = \frac{R_{w}}{2} (\theta_{r} + \theta_{l}) \]

\[ \alpha = \frac{R_{w}}{D_{w}} (\theta_{r} - \theta_{l}) \]

These transformations can be further extended in (4) and (5).

\[ \omega_{r} = \frac{\dot{y}}{R_{w}} + \frac{D_{w}}{2R_{w}} \dot{\alpha} \quad \omega_{l} = \frac{\dot{y}}{R_{w}} - \frac{D_{w}}{2R_{w}} \dot{\alpha} \]

\[ \omega_{r} = \frac{\dot{y}}{R_{w}} + \frac{D_{w}}{2R_{w}} \dot{\alpha} \quad \omega_{l} = \frac{\dot{y}}{R_{w}} - \frac{D_{w}}{2R_{w}} \dot{\alpha} \]

Notice that \( \omega_{r} \) and \( \omega_{l} \) are the angular velocities of the right and left wheels, respectively.

The equations of motion of the system were obtained solving the Euler-Lagrange Equation (6):

\[ \frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_{l}} \right) - \frac{\partial L}{\partial q_{l}} = Q_{l} \]

where \( q_{l} \) are the generalized coordinates, here represented by \( \theta_{p}, \alpha \) and \( y \); \( Q_{l} \) are the internal torques, estimated by (7), (8) and (9).

\[ Q_{p} = \tau_{p} \]

\[ Q_{\alpha} = \tau_{l} - \tau_{r} \]

\[ Q_{y} = \tau_{l} + \tau_{r} \]

Notice that \( \tau_{r}, \tau_{l} \) and \( \tau_{p} \) are calculated as shown in (10), (11) and (12):

\[ \tau_{r} = \frac{K_{nr}}{R_{r}} (PWM_{r} \cdot V_{max} - K_{er} (w_{r} - w_{p})) \]

\[ \tau_{l} = \frac{K_{ml}}{R_{l}} (PWM_{l} \cdot V_{max} - K_{al} (w_{l} - w_{p})) \]

\[ \tau_{p} = - (\tau_{r} + \tau_{l}) \]

being \( \tau_{r} \) and \( \tau_{l} \) the right and left wheel’s applied torque; \( \tau_{p} \) is the torque applied to the robot; \( PWM_{r} \) and \( PWM_{l} \) are the right and left control input signals (varying from -1 to 1); \( K_{ml} \) is the motor torque constant; \( K_{er} \) is the motor voltage constant, \( R_{l} \) and \( R_{r} \) the motors resistance and \( V_{max} \) the DC voltage feeding each motor (12.1 V). Each motor has its own constants.

To obtain the linear system in the form of (13):

\[ x = Ax + Bu \]

\[ y = Cx \]

with \( x \) being the state vector and \( u \) the control input vector, the Jacobian matrix was solved for the equilibrium point \( (\theta_{p} = 0, w_{p} = 0, \alpha = 0, \dot{\alpha} = 0, y = 0, \dot{y} = 0) \).

Parameters were obtained from a real self-balancing robot (Fig. 2) and its model was implemented in Autodesk Inventor®, using the parameters shown in Table 1.

Table 1. Modelling parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m_{p} )</td>
<td>1.577 kg</td>
<td>Mass of the robot</td>
</tr>
<tr>
<td>( m_{w} )</td>
<td>0.103 kg</td>
<td>Mass of each wheel</td>
</tr>
<tr>
<td>( I_{xx} )</td>
<td>0.01585 kg·m²</td>
<td>Moment of inertia in C.M x direction</td>
</tr>
<tr>
<td>( I_{yy} )</td>
<td>0.008377 kg·m²</td>
<td>Moment of inertia in C.M y direction</td>
</tr>
<tr>
<td>( I_{zz} )</td>
<td>0.023032 kg·m²</td>
<td>Moment of inertia in C.M z direction</td>
</tr>
<tr>
<td>( R_{w} )</td>
<td>0.05 m</td>
<td>Radius of each wheel</td>
</tr>
<tr>
<td>( J_{w} )</td>
<td>0.000168 kg·m²</td>
<td>Moment of inertia of each wheel</td>
</tr>
<tr>
<td>( L )</td>
<td>0.07192 m</td>
<td>Distance of the center of mass from the wheel axle</td>
</tr>
<tr>
<td>( D_{w} )</td>
<td>0.00263 m</td>
<td>Distance between wheels</td>
</tr>
<tr>
<td>( K_{er} )</td>
<td>0.13 V/s/rad</td>
<td>Right motor voltage constant</td>
</tr>
</tbody>
</table>
The resulting dynamic model can be represented by matrices A and B bellow:

\[
A = \begin{bmatrix}
0 & 1.0000 & 0 & 0 & 0 & 0 \\
64.2472 & -0.2299 & 0 & 0 & 0 & 4.5984 \\
0 & 0 & 0 & 1.0000 & 0 & 0 \\
0 & 0 & 0.3415 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1.0000 \\
-3.7996 & 0.0156 & 0 & 0 & 0 & -0.3111
\end{bmatrix}
\]

\[
B = \begin{bmatrix}
0 & 0 \\
-10.6116 & -10.6116 \\
0 & 0 \\
-5.9926 & 5.9926 \\
0 & 0 \\
0.7180 & 0.7180
\end{bmatrix}
\]

being the state vector \( \mathbf{x} = [\theta_p \ w_p \ \alpha \ \dot{\alpha} \ y \ \dot{y}] \). One can easily verify that controllability holds for the system above.

3. ROBOT CONSTRUCTION

The robot was built with the components in Table 2.

### Table 2. Robot components.

<table>
<thead>
<tr>
<th>Component</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>12V DC geared motor</td>
<td></td>
</tr>
<tr>
<td>Dual H bridge DC motor driver module</td>
<td></td>
</tr>
<tr>
<td>Gyroscope and accelerometer sensor module</td>
<td></td>
</tr>
<tr>
<td>Arduino M0 Pro 32-bit board</td>
<td></td>
</tr>
<tr>
<td>Bi-directional logic level converter (3.3V – 5V)</td>
<td></td>
</tr>
<tr>
<td>Bluetooth master slave module</td>
<td></td>
</tr>
<tr>
<td>Set of 3 Li-ion 4.2V batteries</td>
<td></td>
</tr>
</tbody>
</table>

3.1 Pitch Angle Measurement

The pitch angle is measured by the accelerometer. The gyroscope provides the angular velocity in such a way that the angle can be obtained through integration. One way to do that is using the Euler Forward method (14).

\[
\theta_p[n] = \theta_p[n-1] + T_s \omega_p[n-1]
\]

Moreover, both measurements can be combined to obtain a better estimative of the angle, which is called sensor fusion (Paina et al. 2011).

One way to merge both measurements is using the Complementary Filter (Paina et al. 2011), by which the resulting angle is obtained (15):

\[
\theta_p[n] = \alpha \cdot (\theta_p[n-1] + \text{GyroRate}) + (1 - \alpha) \cdot \text{AccAngle}
\]

where \( \alpha = \frac{\pi}{\pi + T_s} \) \( T_s \) is the sampling period adopted, \( \text{GyroRate} \) is the angular velocity measured by the gyroscope and \( \text{AccAngle} \) is the angle measured by the accelerometer.

Another way to obtain a precise angle is using a Kalman Filter for the fusion, which is the approach adopted in this paper. In this case, a measurement model that includes the accelerometer and the gyroscope is required. Paina et al. (2011) suggests the model in (16):

\[
\theta_{M,k} = [1 \ 0] \begin{bmatrix} \theta_{p,k} \\ \omega_{p,b,k} \end{bmatrix} + \begin{bmatrix} T_s \\ 0 \end{bmatrix} \omega_{M,k} + \begin{bmatrix} n_{\alpha,k} \\ n_{\omega,b,k} \end{bmatrix}
\]

where \( \omega_{b,k} \) is the gyro bias, \( \omega_{M} \) is the gyro measurement, \( \theta_{M} \) is the accelerometer measurement and \( n_{\beta}, n_{\omega} \) and \( n_{\alpha} \) are white noise signals with zero mean and Gaussian distribution. Implementation requires knowledge of the process noise \( (R_w) \) and of the measurement noise \( (R_v) \).

\[
R_w = E[w_k w_k^T] = \begin{bmatrix} \sigma_{\omega}^2 & \sigma_{\omega,\alpha} \\ \sigma_{\omega,\alpha} & \sigma_{\alpha}^2 \end{bmatrix}
\]

\[
R_v = E[v_k v_k^T] = \sigma_\alpha^2
\]

In order to estimate the measurement noise, the robot was placed in vertical position with motors in maximum rotation. The histograms obtained are shown in Fig. 3.

![Fig. 3. Gyroscope and accelerometer noise.](image)

Based on the resulting variances, the following values were adopted for the process and the measurement noises:

\[
R_w = \begin{bmatrix} 0.001 & 0 \\ 0 & 0.00004 \end{bmatrix} \quad R_v = 0.01
\]

Notice that the histograms clearly show the offset of the measurements, which has to be corrected.

The ability to use the measurement noise to tune the filter was the main reason why this filter was adopted rather than the complementary one. In addition, the Kalman Filter has been extensively studied in the literature and several implementations are available for microcontrollers.

3.2 Yaw Angle Measurement

The yaw angle was obtained by the variable change mentioned in the modelling section. Consequently, measurements of \( \theta_x \) and \( \theta_y \) are required. These were obtained through the DC motor encoders. The resolution of the encoders are \( 2\pi/(64 \times 18.8) = 0.005222 \) radians per pulse, measured by both encoder channels (A and B).

A linear first-order filter was applied to the angles in order to reduce noise (17):
\[ \theta_{f,1}[n] = \left( \frac{\tau}{T_z + \tau} \right) \theta_{f,1}[n-1] + \left( \frac{\tau}{T_z + \tau} \right) \theta_{r,1}[n] \]  
\[ \text{where } \theta_{f,1}[n] \text{ are the resulting filtered angles.} \]

Finally, to obtain the yaw angle velocity, the Euler Backward method (18) was used with the non-filtered angles and then the resulting derivatives were also filtered (19).

\[ \dot{\theta}_{r,1} = \left( \frac{\theta_{r,1}[n] - \theta_{r,1}[n-1]}{T_z} \right) \]
\[ \dot{\theta}_{f,1}[n] = \left( \frac{\tau}{T_z + \tau} \right) \theta_{f,1}[n-1] + \left( \frac{\tau}{T_z + \tau} \right) \dot{\theta}_{r,1}[n] \]  

The same values of \( \theta_{f,1} \) and \( \dot{\theta}_{f,1} \) were used to calculate the system states \( y \) and \( \dot{y} \). The filter constant considered was \( \tau = 0.05 \text{ s} \) and the sampling period was \( T_z = 0.01 \text{ s} \).

4. CONTROL DESIGN

A MIMO Pole Placement and LQR controllers were applied to the robot for the regulation problem around the equilibrium point. Then, the unidirectional and bidirectional tracking problems are addressed. The control design is based on the discrete-time system, which can be represented by (20):

\[ x[n+1] = \Phi x[n] + \Gamma u[n] \]
\[ y = C x[n] \]  

where \( \Phi = e^{\Delta T z} \) and \( \Gamma = \int_0^{T z} e^{\Delta T z} B \, d\eta \).

4.1 Pole Placement

Since one pole for each state must be chosen, for the Pole Placement Controller only the states \( [\theta_p, w_p, \alpha, \dot{\alpha}, \dot{\gamma}] \) are considered in order to simplify the design. The following specifications were adopted to guide the design:

- Maximum overshoot: 1%
- Settling time for \( \theta_p \) and \( w_p \): 0.5 seconds
- Settling time for \( \alpha \) and \( \dot{\alpha} \): 2.7 seconds
- Settling time for \( y \) and \( \dot{y} \): 1.3 seconds

These values were chosen based on how fast the robot could reach the desired set-points considering its physical dimensions and the DC motor capacity. Moreover, too fast responses can lead to instability. Therefore, different specifications were validated through simulation and in the real robot until satisfactory values were reached.

In this case, \( Y = I_{5 \times 5} \cdot x \) is the system output and the control action that places the poles is \( u = -K x \). Thus, the discrete closed-loop system can be described by (21).

\[ x[n+1] = (\Phi - \Gamma K) x[n] \]  

The regulation problem meets the specifications if \( \text{det}(z I - \Phi + \Gamma K) = (z - p_{12})(z - p_{24})(z - p_3)(z - p_5) \). Only the real part of the poles was taken. The resulting poles are:

\[ p_{12} = 0.9231 \quad p_{24} = 0.9853 \quad p_3 = 0.9697 \]

The obtained matrix gain is:

\[ K = \begin{bmatrix} -4.86 & -0.77 & -0.94 & -0.79 & -3.17 \\ -4.86 & -0.77 & 0.94 & 0.79 & -3.17 \end{bmatrix} \]

Simulation for this control input can be seen in Fig. 4 and results for the real robot are shown in Fig. 5.

4.2 Linear Quadratic Regulator (LQR)

The control action of the LQR has the same structure \( u = -K x \) of the Pole Placement Controller. However, the \( K \) gain is such that the cost function (22) is minimized:

\[ J = \frac{1}{2} \sum_{n=0}^{\infty} (x^T[n] Q x[n] + u^T[n] R u[n]) \]  

where \( Q = Q^T \geq 0 \) is the given state cost and \( R = R^T > 0 \) is the input cost matrix.

![Fig. 4. Pole Placement Controller: simulation.](image)

To obtain the static gain \( K_{ss} \) that minimizes the cost function, the Riccati Equation in stationary form (23) is used.

\[ P_{ss} = \Phi^T (P_{ss} - P_{ss} \Gamma (R + \Gamma^T P_{ss} \Gamma)^{-1} \Gamma^T P_{ss}) \Phi + Q \]  

In this case, the stationary gain is given by (24).

\[ K_{ss} = R^{-1} \Gamma^T P_{ss} \]  

Matrices \( Q \) and \( R \) can be obtained using the Bryson rule (Fadali, 2013).

![Fig. 5. Pole Placement: robot response.](image)

The design was based on the following limits:

\[ Q_u = \frac{1}{\text{Maximum Acceptable Value of } x_i^2} \quad R_{ij} = \frac{1}{\text{Maximum Acceptable Value of } u_j^2} \]
\[ \theta_p \text{ max: } 0.01 \text{ rad } \omega_p \text{ max: } 0.1 \text{ rad/s } \\
\alpha \text{ max: } 0.05 \text{ rad } \dot{\alpha} \text{ max: } 0.1 \text{ rad/s } \\
y \text{ max: } 0.05 \text{ m } \dot{y} \text{ max: } 0.5 \text{ m/s } \\
\]

Notice that all states are considered for the LQR Controller. Moreover, the matrix \( R \) was chosen to be

\[
R = \begin{bmatrix}
100 & 0 \\
0 & 100
\end{bmatrix}
\]

The values were chosen higher than the input saturation limit in order to reduce the control input effort.

The obtained matrix gain is:

\[
K = \begin{bmatrix}
-10.38 & -1.53 & -1.34 & -0.85 & -1.24 & -6.00 \\
-10.38 & -1.53 & 1.34 & 0.85 & -1.24 & -6.00
\end{bmatrix}
\]

Simulation results can be seen in Fig. 6 and results for the real robot are shown in Fig. 7.

Fig. 6. LQR Controller: simulation.

Fig. 7. LQR Controller: robot response.

4.3 Trajectory Control

Tracking a specific variable of the robot can be done augmenting the system with an integrator as shown in Fig. 8.

The integrator states can be described as in (25).

\[
v[n + 1] = v[n] + r[n] - y[n]
\]

The feedback law for the augmented system results in (26):

\[
\begin{bmatrix}
x[n + 1] \\
v[n + 1]
\end{bmatrix} = \begin{bmatrix}
\Phi & 0 \\
-\mathbf{C} & I
\end{bmatrix} \begin{bmatrix}
x[n] \\
v[n]
\end{bmatrix} + \begin{bmatrix}
\mathbf{0} \\
0
\end{bmatrix} r[n]
\]

where \( K_i \) is the integrator gain.

4.3.1 Unidirectional Control

In this case, only the linear position \( y \) is controlled. A square wave of amplitude 0.5 m and 80 s period was used as the set-point. The following gains were adopted using the Brayson rule:

\[
K = \begin{bmatrix}
-6.59 & -1.09 & -0.11 & -0.17 & -2.03 & -5.19 \\
-6.59 & -1.09 & 0.11 & 0.17 & -2.03 & -5.19
\end{bmatrix} \\
K_i = \begin{bmatrix}
0.0041 \\
0.0041
\end{bmatrix}
\]

Simulation results can be seen in Fig. 9 and results for the real robot are shown in Fig. 10.

Fig. 8. System augmented by integrator.

Fig. 9. Square wave tracking: simulation.

Fig. 10. Square wave tracking: robot response.

4.3.2 Bluetooth Bidirectional Control

The 2-dimensional control was performed in a joystick fashion. References for the linear velocity \( \dot{y} \) and for the yaw...
angle $\alpha$ were sent by a serial transmitter that communicates with the Bluetooth HC-05 module. For a forward/backward signal, the $y$ reference signal was increased/decreased by 0.25 m/s. In the same way, the yaw reference was increased/decreased by 0.4 rad for each directional signal received by the Bluetooth. State $y$ was not used in the design. The output signal considered was

$$Y = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \theta_p \\ \omega_p \\ \alpha \\ \dot{\alpha} \\ \dot{y} \end{bmatrix}$$

The following gains were adopted, based on the Brayson rule:

$$K = \begin{bmatrix} -7.00 & -1.05 & -0.93 & -0.42 & -4.10 \\ -7.00 & -1.05 & 0.93 & 0.42 & -4.10 \end{bmatrix}$$

$$K_I = \begin{bmatrix} 0.011 & 0.010 \\ -0.011 & 0.010 \end{bmatrix}$$

The references were transmitted trying to make the robot follow a circular trajectory followed by a linear one with varying velocity. Results can be seen in Fig. 11. A video of the robot with the LQR algorithm for the bidirectional tracking can be found in https://youtu.be/Gu_qGuta_qA.

Fig. 11. Two-dimensional tracking: robot response.

5. CONCLUSIONS

A self-balancing robot was successfully designed and built. Digital multivariable control strategies were embedded in a microcontroller board and were able to keep the robot in vertical position as well as to allow it track complex trajectories using a Bluetooth transmitter. Implementation techniques were discussed and validated, allowing anyone to use the algorithms presented in this work.

From the results presented, one could also conclude that the Linear Quadratic Regulator had a better performance for the regulation problem when compared to the Pole Placement controller. That is because the states were bounded in a smaller range around the equilibrium point with the LQR controller as shows the comparison of Fig. 5 and Fig. 7.

REFERENCES


