Detailed description of a two joint robot arm

Nagy Zoltán

Technical University of Cluj Napoca, Cluj Napoca, Romania e-mail: Zoltan.Nagy@aut.utcluj.ro

Abstract: This documentation describes the necessary information needed to control the two joint robot arm. The documentation starts from the design and the creation of the robot arm. The description of the new firmware uploaded on the servo motors, and finally the estimated parameters and the connection to the Matlab are presented.

Keywords: robot arm, firmware, Matlab, serial communication, near real-time

1. HARDWARE AND FIRMWARE

The basic element of this project is the Dynamixel AX-12A servo motor. A detailed description of this motor can be found in (http://www.trossenrobotics.com/images/productdownloads/AX-12(English).pdf). Usually a servo motor is composed of a DC motor, gears to reduce the velocity, a sensor to measure the position, and a small microcontroller which synchronizes all of these components and communicates with the master, which in this case is an intermediary controller with USB compatibilities. Unfortunately, with the basic settings the AX-12A servo motor can be controlled only in position or velocity. The original firmware is not open source, so it is not modifiable. However there is an open source version available online. This new firmware makes possible to control the motors directly in PWM, i. e. in torque. In our system the firmware on the motors was already updated.

Detailed description of the firmware and the necessary steps to install it, can be found in:

https://actuated.wordpress.com/ax12firmware/

The AX-12A motor can be seen in Figure 1. An inside

Fig. 1. Dynamixel AX-12A servo motor

view of the servo motor can be seen in Figure 2. In order to work with this, three main components are needed: the

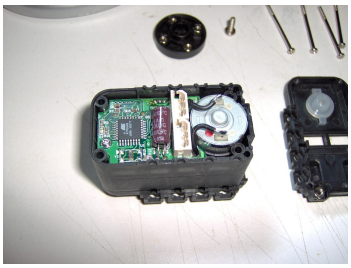

Fig. 2. Dynamixel AX-12A servo motor inside

servo motor, USB2Dynamixel converter and a power safety circuit.

2. ROBOT ARM DESIGN

We wanted to make a simple robot arm which can be used for our experiments. For this reason we put two joints in the position seen in Figure 3.

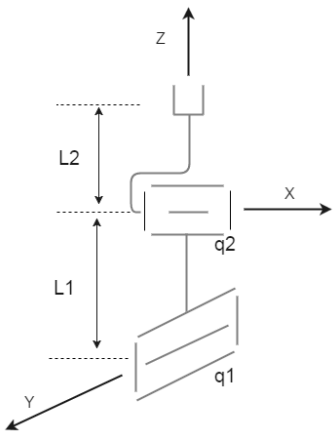

Fig. 3. Schematic representation of a 2 joint robot arm

The assembled robot arm can be seen in Figure 4.

There are two motors: the motor which is fixed to the table lets the arm move on the x axis, and the second one is responsible for the motion on the y axis of the upper

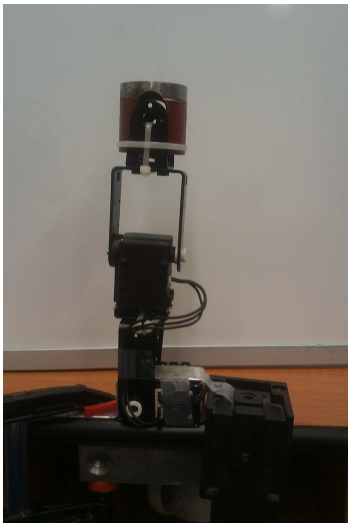

Fig. 4. Two joint robot arm

part. At the upper part we put an extra mass, to have a nonlinear effect.

3. SYSTEM IDENTIFICATION

In order to use model based control, first we have to identify the model. Based on the geometric and kinematic model the equations for the robot arm motion can be achieved. We have the following general equation for the model:

$$\ddot{q} = M(q)^{-1}(-(Co(q,\dot{q}) + F)\dot{q} - G(q) + \tau) \tag{1}$$

where $q = [q_1 \ q_2]^T$, q_1 and q_2 are the angles for the two joints, $\dot{q}$ the angular velocity vector and τ is the input vector. F is the friction matrix; for the sake of simplicity we used a simple viscous friction, so $F = \begin{bmatrix} b_1 & 0 \\ 0 & b_2 \end{bmatrix}$. G is the gravity matrix, having the form:

$$G = \begin{bmatrix} \frac{-g\sin(q_1)(L_1M_1 + 2L_1M_2 + L_2M_2\cos(q_2))}{2} \\ \frac{-(L_2M_2g\cos(q_1)\sin(q_2))}{2} \end{bmatrix} \tag{2}$$

As it can be seen the gravity matrix is dependent only on the q vector. M is the mass matrix, also dependent on the q vector:

$$\begin{aligned} M(q) &= \begin{bmatrix} M(1,1) & 0 \\ 0 & \frac{M_2L_2^2}{4} + I_{2y} \end{bmatrix} \\ M(1,1) &= I_{1x} + I_{2z} + \cos(q_2)^2(I_{2x} - I_{2z}) \\ &\quad + M_2(L_1 + \frac{L_2\cos(q_2)}{2})^2 \end{aligned} \tag{3}$$

The B matrix is the Coriolis matrix, this can be selected in several ways, because it contains multiplications of two states. The following matrix was chosen in our case:

$$\begin{aligned} B(q) &= \begin{bmatrix} B(1,1) & 0 \\ B(2,1) & 0 \end{bmatrix} \\ B(1,1) &= (\dot{q}_2(\sin(2q_2)(\frac{M_2L_2^2}{4} + I_{2x} - I_{2z}) \\ &\quad + L_1L_2M_2\sin(q_2))) \\ B(2,1) &= -\dot{q}_1(\frac{\sin(2q_2)}{2}(I_{2x} - I_{2z} + \frac{L_2}{4}) \\ &\quad + \frac{L_2M_2L_1\sin(q_2)}{2}) \end{aligned} \tag{4}$$

The values of the parameters can be found in Table 1. Some of them were obtained by measuring, the case of

Table 1. Margin settings

$L_1[m]$	0.095	length first-second joint
$L_2[m]$	0.1	length second joint end-effector
$M_1[kg]$	0.095	mass first joint
$M_2[kg]$	0.37	mass second joint
$g[m/s^2]$	9.81	gravitational acceleration
$I_{1x}[kgm^2]$	$0.227 \cdot 10^{-1}$	moment of inertia
$I_{1y}[kgm^2]$	$0.833 \cdot 10^{-4}$	moment of inertia
$I_{1z}[kgm^2]$	$0.181 \cdot 10^{-4}$	moment of inertia
$I_{2x}[kgm^2]$	$0.833 \cdot 10^{-4}$	moment of inertia
$I_{2y}[kgm^2]$	$0.227 \cdot 10^{-1}$	moment of inertia
$I_{2z}[kgm^2]$	$0.707 \cdot 10^{-4}$	moment of inertia
$b_1[-]$	0.24	friction coefficient, first joint
$b_2[-]$	0.16	friction coefficient, second joint

M_1, M_2, L_1 and L_2. Some other parameters, such as friction coefficient, cannot be measured, because the lack of expensive equipments, so they need to be estimated. In order to estimate them we created some test cases. For instance to estimate the friction coefficient we used a free-fall test. We did this for both joints.

Based on the model the robot arm has 4 equilibrium points, but two of them are unattainable on the real system. The first equilibrium point is when the robot arm is pointing up, all the angles are 0. This position can be seen on Figure 4. The other equilibrium point is when the first joint angle is 180^o and the second joint angle is 0; this is the pointing down position. As it can be seen in Figure 4 the system is fixed on the table, in order to achieve the second equilibrium point we have to put the fixers and the robot arm in a different position.

The first equilibrium point is an unstable one, while the second one is stable. For the free fall test, we started the system from an initial condition where the first angle was close to 0, and the second angle was 0. the effect of this was that the system started to move to the stable equilibrium point. With this test we were able to estimate the viscous friction coefficient for the first joint. The comparison between the real data and model estimation can be seen in Figure 5. The friction coefficient parameter approximated, based on the experimental data is $b_1 = 0.421$. Although the approximation is good, we can see that the form is not exactly the same. One way to explain this difference is that the robot arm contains also static friction which is not modelled. As it can be seen in Figure 5 the system does not reach the actual equilibrium point at 180^o due to the static friction. On the other hand based on the

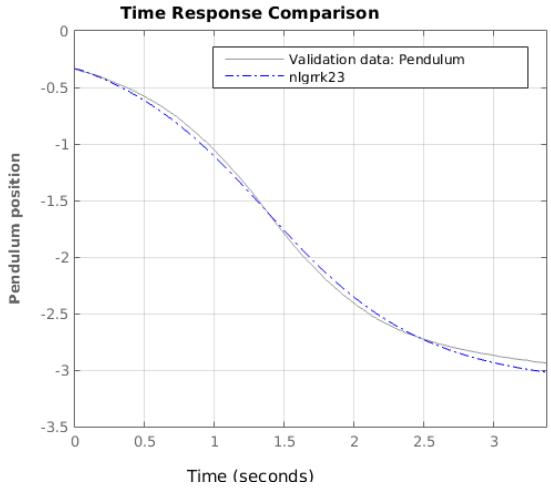

Fig. 5. Free fall approximation

equations and the measured parameters we observed that the inertia parameters have a relatively small effect on the overall system. That is why, they were considered to be very small values. For the aforementioned reasons another test case was developed for a better approximation. A constant input was applied on one of the joints, while the other joint was held in 0 angle. For this case we estimated both the input torque and the friction coefficients. This test was performed for both of the joints. The initial condition for this case is $x_0 = [0.49\ 0\ 0\ 0]^T$. Because of the effect of the gravity a nonlinear relation can be observed. We obtain $b_1 = 0.239$, and the constant input is $\tau_1 = -0.3924$. the minus sign here refers to the direction of the torque. The comparison between the experimental data and the estimated model data can be seen in Figure 6. The real system has an interesting behaviour at the

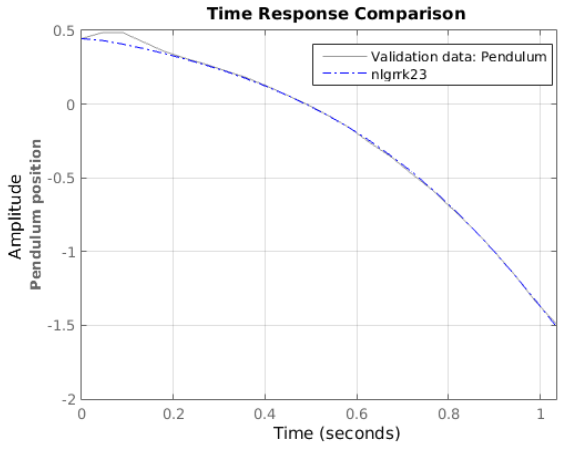

Fig. 6. Constant input approximation

beginning, which can be considered as a problem because of the unmodeled dynamics of the static friction, but now the approximation is much better compared to the free-fall case.

In the case of the second joint, the constant torque method was good also to estimate the friction coefficient, b_2 and the inertia parameters, I_{2x}, I_{2y} and I_{2z}.

In order to use our models, a conversion was necessary between torque and PWM. Another test method was developed, where we put the system in a position where the gravity has no effect on the second joint. Applying several PWMs to the system, we found out that there is a linear relation between the PWM and the joint speed in the no gravity case.

Based on this knowledge, we put the system back into the normal position, and we analysed the provided curve for different PWMs. Based on this analysis and the identified constant input the following relation was found between the PWM and the torque:

$$PWM = 834\tau + 15 + PWM_{offset} \tag{5}$$

It is also important to mention that the possible PWM values are in the range of $PWM \in [0, 2000]$, where a PWM command of 1000 corresponds to 0 torque, so $PWM_{offset} = 1000$. For clockwise rotation the range is [1050, 2000], and for counter-clockwise [0, 950]. There is a dead-zone between $950 - 1050$ where the torque is not enough to move the motors.

4. SIMPLE CONTROL

For the first iteration a simple control method was implemented on the real system. We linearized the system at 0 angles, then a discretization was applied with a sampling time $T_s = 0.045$. The general discrete state space representation has the following form:

$$\begin{aligned} x(k+1) &= A_d x(k) + B_d u(k) \\ y(k) &= C_d x(k) + D_d u(k) \end{aligned} \tag{6}$$

A_d, B_d, C_d, D_d are matrices representing the system. In our case the D_d matrix is a zero matrix with the appropriate dimension, and we have:

$$\begin{aligned} A_d &= \begin{bmatrix} 1 & 0 & 0.045 & 0 \\ 0 & 1 & 0 & 0.045 \\ 0.8355 & 0 & 0.6486 & 0 \\ 0 & 0.3464 & 0 & 0.6946 \end{bmatrix} \\ B_d &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1.4643 & 0 \\ 0 & 1.9085 \end{bmatrix} \\ C_d &= [I\ O] \end{aligned} \tag{7}$$

Where I is a 2-by-2 identity matrix and O is a 2-by-2 zero matrix. Based on these matrices an LQR regulator was designed (http://rocon.utcluj.ro/co/Files/Laborator%204%20-%20LQR.pdf). We chose the following diagonal elements for the Q and R diagonal matrices:

$$\begin{aligned} Q &= diag\,[15\ 15\ 1\ 1] \\ R &= diag\,[10\ 10] \end{aligned} \tag{8}$$

In the case of the Q matrix the first two elements refer to the angles of the two joints. We want these two to reach the desired position faster, so we use bigger values for them. The last two parameters refer to the angular velocities; these states are based on calculations and some errors may appear, so for this reason we use small values. The R matrix is for the inputs, we want to achieve the stable position, but with a relatively small energy effort, so that is why it contains those weights. The general state feedback control formula is represented in 9

$$u_k = -Kx_k \tag{9}$$

For our system the following control gain was found:

$$K = \begin{bmatrix} 1.65 & 0 & 0.22 & 0 \\ 0 & 1.10 & 0 & 0.20 \end{bmatrix} \tag{10}$$

We tested the robot arm with an initial condition different from 0 angle, and the task was to stabilize the system at 0. A comparison between experimental and simulation can be seen in Figures 7 and 8. As it can be seen the trajectories

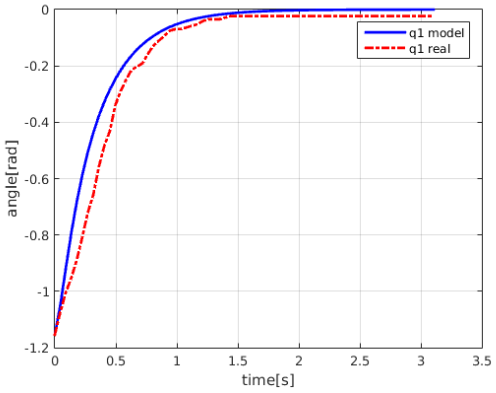

Fig. 7. Model and real system with LQR, first joint

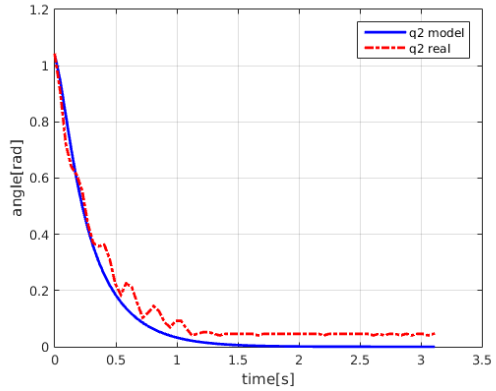

Fig. 8. Model and real system with LQR, second joint

are very close to each other. The differences can appear because of the unmodeled dynamics, or because of the calculation of the angular velocities.

In order to avoid the direct computation of the angular velocities a linear observer was designed based on the matrices in (7). The classical form of the discrete-time linear observer is the following:

$$\begin{aligned} \hat{x}(k+1) &= A\hat{x} + Bu(k) + L(y(k) - \hat{y}(k)) \\ \hat{y}(k) &= C\hat{x}(k) \end{aligned} \tag{11}$$

where $\hat{y}$ refers to the estimated value of the output state. Using the pole placement method, and the closed loop poles $p = [0.05\ 0.14\ 0.1\ 0.03]$, we found the following observer gain:

$$L = \begin{bmatrix} 0.29 & 0 \\ 0 & 0.34 \\ 0.72 & 0 \\ 0 & 0.34 \end{bmatrix} \tag{12}$$

Unfortunately we cannot apply this observer together with the LQR control because the reaction for the LQR control is fast, and the observer is not fast enough, so the ensemble becomes unstable. One option could be to make the observer more aggressive, but this is not applicable on the nonlinear system. For a case where the system has a slower reaction, i. e. the transient regime is longer, the observer can be applied. In the case of a nonlinear trajectory tracking control, the behaviour is much better, because the system has a slower reaction. This can be seen in Figure 9 and Figure 10 We also tried to use the angular

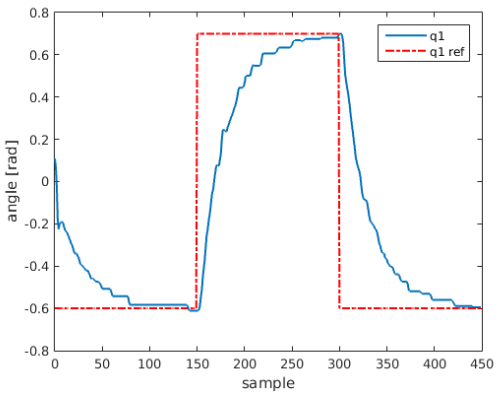

Fig. 9. Trajectory tracking with nonlinear control, first joint

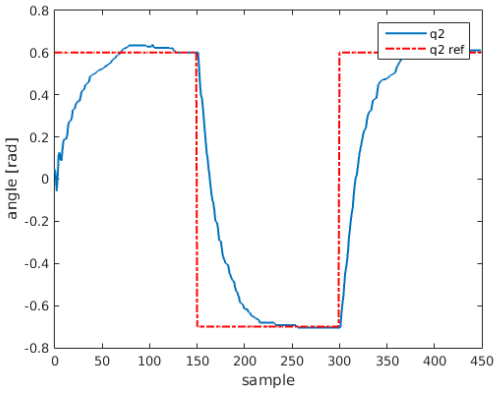

Fig. 10. Trajectory tracking with nonlinear control, second joint

velocity calculation method, but with this the system had an unstable behaviour.

5. CONCLUSION

Our system is a good start for a model based control. It can be used to test the correctness of a simple LQR controller or to estimate the angular velocities based on the model. Also due to the nonlinearities in the system it also can be used to test nonlinear controllers, as it can be seen in Figure 9 and 10. Because the sensors inside the motors measure only the angle, but not the angular velocities, we have two options to find these: by calculating them based on the joint angles; or by estimating them using an observer.

A next step on this part can be to improve the observer, to provide a more robust behaviour. This can be done by using nonlinear observer design.

6. USER'S MANUAL

6.1 Connection of the modules

The first step is to identify the components. We have the following:

- Robot arm with two joints
- Power supply converter
- Main power supply
- USB2Dynamixel controller

The robot arm is made from 2 AX-12A servo motors, this can be seen in Figure 11 The first motor is fixed

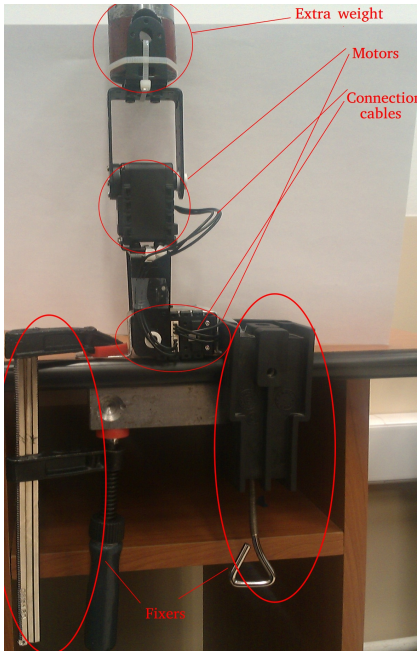

Fig. 11. Robot arm with fixers

to the table, using a 'U' shaped element and two fixers. The purpose of this is to hold still the whole system. Each motor has two input connectors. The power safety circuit converts the standard power supply connector into the motor's power connector. This element can be seen in Figure 12. This power safety circuit has three input (red circles in Figure 12). The first circle is the connector from the Main Power Supply, the second can be connected to the motor and the third to the USB2Dynamixel converter; they are interchangeable. In order to send information to the robot arm we have to connect it to a laptop. For this reason we have the USB2Dynamixel controller, which can be seen in Figure 13. Even though it is not straight forward, the connection to the motors is done through the connector shown in Figure 13. The other end goes directly to the laptop's USB input. The Main Power Supply can be seen in Figure 14. Finally the whole system together is shown on Figure 15. If everything is connected correctly then the red light on the power safety circuit should turn on, and also the LED on the motor should blink.

6.2 Matlab code

After everything is connected and checked, we can try to control the system from the laptop. If you are using Windows you can test the system by downloading

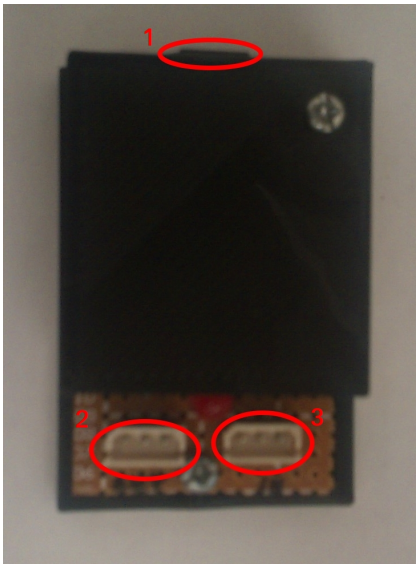

Fig. 12. Power supply converter

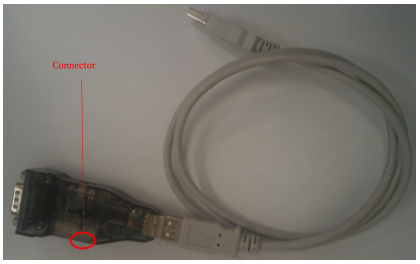

Fig. 13. USB2Dynamixel

Fig. 14. Power supply

the Morpheus GUI which is available from the webpage https://actuated.wordpress.com/ax12firmware/. This is optional.

Download the repository, and use the README.txt file.

Fig. 15. Connection of the modules