Modeling and control of an Acrobot

Jens Troels Nielsen

1 Abstact

This article attempts to model and simulate the acrobot system. This is done by deriving the equations of motion from the potential and kinetic energies of the system, using Lagrange method. Furthermore, the study will examine the controllability of the acrobot, with intent on creating a controller. Finally, swing-up and LQR controllers will be implemented and tested.

2 Preface

All MATLAB scripts used for creating the controller and deriving equations are attached with the report. Additionally, the Python code used for simulation of the system is also attached and can be viewed in any IDE supporting Python.

3 Introduction

Underactuated robotics deal with systems that are not necessarily fully controllable. What that implies is that the system has more degrees of freedom than actuators. The acrobot is no different, it is a two-link robot arm with a passive joint at the shoulder region and an actuated joint at the elbow region. See figure 1 for an illustration of the acrobot. The acrobot got its name for the resemblance of an acrobat, only maneuvering with the waist joint.

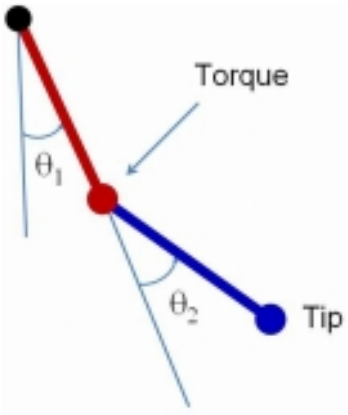

Fig. 1. Acrobot as a two-linked robotic arm with a passive shoulder joint and an actuated elbow joint. Angles for each of the links are denoted θ_1 and θ_2

The goal for this report is to investigate the acrobot, understanding the physics and creating a close approximation of it. This will include deriving equations of motion using Lagrange from the energy of the joints, examination of the equations with respect to controllability and implementing a model that can be simulated in python. Finally, an LQR controller and a swing-up controller will be implemented allowing the links to be stabilized on top of each other from given starting angles. Various benchmarks of the controller will be tested, displaying its full potential.

4 Modeling

One of the most important aspects of modeling is to create your coordinate system properly from the desired goals. The objective in this report was to create a controller to stabilize the acrobot with the two links standing on top of each other as shown in figure 2.

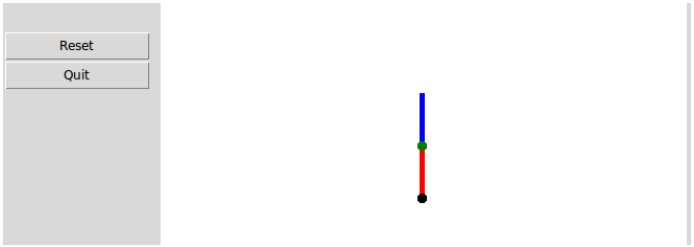

Fig. 2. Simulation screenshot depicting the goal for the balance controller

The zero state will be with both links pointing down. The moments of inertia are calculated from the links pivot points. The energies for the system are given as

$$T = T_1 + T_2 \tag{1}$$

$$T_1 = \frac{1}{2} I_1 \omega_1^2 \tag{2}$$

$$\begin{aligned} T_2 = \frac{1}{2}(m_2 l_1^2 + I_2 + 2 m_2 l_1 l_{c2} sin(\theta_2)\omega_1^2 \\ + \frac{1}{2} I_2 \omega_2^2 + (I_2 + m_2 l_1 l_{c2} cos(\theta_2)\omega_1 \omega_2 \end{aligned} \tag{3}$$

$$\begin{aligned} U = -m_1 g l_{c1} cos(\theta_1) - m_2 g(l_1 cos(\theta_1) \\ + l_2 cos(\theta_1 + \theta_2)) \end{aligned} \tag{4}$$

Where ω_i is the angular velocity for the respective link, l_{ci} is the center of mass for the respective link, g is the gravitational acceleration, l_i are the lengths of the links, I_i is the moment of inertia taking about the pivot of the link and

m_i are the masses. T_1 is the kinetic energy for the first link and T_2 is the kinetic energy for the second link. U denotes the potential energy of the system.

The equations of motion for the acrobot can be determined by using Lagrange's method on equations 1, 3 and 4.

To make Lagrange easier to use the coordinates of the system were generalized. This is done because using standard cartesian coordinates (x and y) would mean that at least one of them is dependant on the other. Since our coordinates are independent and complete we can define the generalized coordinates as $q = (\theta_1\theta_2)^T$, thus the Lagrange equation will become equation 6.

$$L = T - U \tag{5}$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = Q \tag{6}$$

As the potential energy is not time dependant, Lagrange can be rewritten more simplified.

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{q}_i} - \frac{\partial T}{\partial q_i} + \frac{\partial U}{\partial q_i} = Q \tag{7}$$

Running through equation 7 i'th times will yield the equations of motion for the acrobot system.

$$\begin{aligned}(I_1 + I_2 + m_2 l_1^2 + 2m_2 l1 l_{c2} cos(q_2))\ddot{q_1} \\ +(I_2 + m_2 l_1 l_{c2} cos(q_2))\ddot{q_2} - 2m_2 l1 l_{c2} sin(q_2)\dot{q_1}\dot{q_2} \\ -m_2 l1 l_{c2} sin(q_2){\dot{q_2}}^2 + (m_1 l_{c1} + m_2 l_1) g sin(q_1) \\ +m_2 g l2 sin(q_1 + q_2) = 0\end{aligned} \tag{8}$$

$$\begin{aligned}(I_2 + m_2 l_1 l_{c2} cos(q_2)\ddot{q_1} + I_2\ddot{q_2} \\ +m_2 l_1 l_{c2} sin(q_2){\dot{q_1}}^2 + m_2 g l_2 sin(q_1 + q_2) = \tau\end{aligned} \tag{9}$$

Where $\ddot{q_i}$ is the angular acceleration for the corresponding link *i*. Solving for the accelerations by substituting the equations 8 and 9 into each other, allows a basic model and system to be created and simulated. The accelerations are shown in attached MATLAB and Python code. The following link shows a simulation of the acrobot without any control.

https://youtu.be/O1GgJUD9Ip8 **(Acrobot with no control with starting angles 40 and -10 degrees)**

Arranging the equations of motion for the acrobot in manipulator form, allows a simpler way to create a linear system. A linear system can also be created by using a state-space model, thus solving for each of the states in the equations of motion and creating a state matrix **A** and a input matrix **B**. The manipulator form was chosen because it is more simple and a general model for all robotic systems. The linear system will be used to determine if the acrobot is controllable and by extension a possible controller. The equations of motion for a general robotic manipulator is seen in equation 10.

$$\boldsymbol{H(q)\ddot{q} + C(\dot{q}, q)\dot{q} + G(q) = B(q)u} \tag{10}$$

Where **q** is the state-vector, **H** is the inertial matrix, **C** is the matrix containing Coriolis forces, **G** is the matrix composed of potentials such as the gravitational force and **B** is the input matrix mapping the inputs of **u**. Equations 11, 12, 13, 14 are the robotic manipulator for the acrobot system.

$$\boldsymbol{H(q)} = \begin{bmatrix} I_1 + I_2 + m_2 l_1^2 + 2m_2 l1 l_{c2} & (I_2 + m_2 l_1 l_{c2} cos(q_2)) \\ (I_2 + m_2 l_1 l_{c2} cos(q_2)) & I_2 \end{bmatrix} \tag{11}$$

$$\boldsymbol{C(\dot{q}, q)} = \begin{bmatrix} -2m_2 l1 l_{c2} sin(q_2)\dot{q_2} & -m_2 l1 l_{c2} sin(q_2)\dot{q_2} \\ m_2 l1 l_{c2} sin(q_2)\dot{q_1} & 0 \end{bmatrix} \tag{12}$$

$$\boldsymbol{G(q)} = \begin{bmatrix} (m_1 l_{c1} + m_2 l_1) g sin(q_2) + m_2 g l_2 sin(q_1 + q_2) \\ m_2 g l_2 sin(q_1 + q_2) \end{bmatrix} \tag{13}$$

$$\boldsymbol{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{14}$$

5 Balance Controller

Consider the linear system in equation 15. In order to control this it is a requirement to obtain the state- and input matrices **A** and **B**. By linearizing (small angle approximateion) the manipulator equations of the acrobot both of these matrices can be obtained. As previously mentioned the goal of the controller was to stabilize the two joints on top of each other, thus the linearization is then done with respect to that goal.

$$\boldsymbol{\dot{x} = Ax + B}u \tag{15}$$

Using Taylor expansion around a fixed point yield equations 16 and 17.

$$\boldsymbol{A} = \begin{bmatrix} 0 & \boldsymbol{I} \\ -\boldsymbol{H}^{-1}\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{q}} & -\boldsymbol{H}^{-1}\boldsymbol{C} \end{bmatrix} \tag{16}$$

$$\boldsymbol{B} = \begin{bmatrix} 0 \\ -\boldsymbol{H}^{-1}\boldsymbol{B} \end{bmatrix} \tag{17}$$

In the case of the acrobot it is much simpler as the **C** matrix becomes 0 and the **G** matrix becomes equation 18.

$$\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{q}} = \begin{bmatrix} -g(m_1 l_{c1} + m_2 l_1 + m_2 l_2) & -m_2 g l_2 \\ -m_2 g l_2 & -m_2 g l_2 \end{bmatrix} \tag{18}$$

With **A** and **B** matrices obtained for the system, the controllability can be examined by using the ***ctrb(A,B)*** function in MATLAB, thus creating a controllability matrix. In order for the system to be controllable it has to be full rank, which is the case of the acrobot system (This can be seen done in attached MATLAB scripts). Confirming the system to be controllable, a controller can be created by using the MATLAB function ***lqr(A,B,Q,R)***, thus creating a linear feedback matrix **K**. This is the so called LQR controller and it can be customized by changing the **Q** and **R** matrices, which are traditionally positive and diagonal. Changing the values in the **Q** matrix will penalize the states of the system and changing the values in the **R** matrix will penalize the actions of the input.

The values chosen for this project was an initial guess and some fine-turning for optimizing the controller to bring the acrobot to an equilibrium state as fast as possible. The Q-values can be seen in equation 19 and the R-value was set to 1.

$$\boldsymbol{Q} = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 1000 \end{bmatrix} \tag{19}$$

The values were chosen, because penalizing the joint velocities will help the controller settle faster and because there is not torque limit, so the joints could have theoretically very high velocities, while the angles are bounded by wrap around made in the simulation.

With starting degrees of 190 for the first joint and -20 for the second the acrobot is easily able to stabilize. It takes about 8 seconds for it to be completely still as shown in figure 3. The figure also shows the joints swinging about their respective equilibrium points (π and 0).

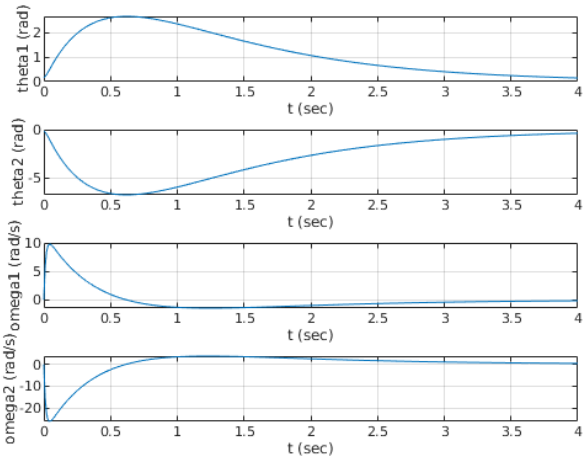

Fig. 3. Graph depicting the angles and velocities of the acrobot with LQR enabled. Starting angles were 190 degrees for the first joint and -20 degrees for the second joint

A video of the simulation can be found on the following youtube link:

https://youtu.be/B_O5fU19kao **(LQR for acrobot with starting angles 190 and -20)**

The LQR controller works as long as a few conditions are fulfilled. There is little-to-no excess energy in the system (the joints can't have high velocities) and the joint angles are in a certain vicinity of each other. If the joint angles are on the same side of the equilibrium point (180, 0 degrees) they can have a maximum of 8.5 degrees for the LQR to work properly. If the joint angles are on opposite sides of the equilibrium point they can have a maximum of 45 degrees. The following two youtube links show these two specific scenarios.

https://youtu.be/zEGmYGz2ovk **(LQR for acrobot with starting angles 188.5 and 8.5)**

https://youtu.be/zKSJNiUCf28 **(LQR for acrobot with starting angles 225 and -90)**

6 Swing-up Controller

Assuming the acrobot does not start in the desired angles for the LQR controller to work a swing-up controller is needed. It will funnel energy into the system to a point where the acrobot has enough energy to swing itself up desired angles of the joints and LQR will take over and stabilize it.

This was implemented by adding energy into the system until a threshold was met. Then switching to a PD controller in order to drive the second joint angle to 0, so the second link would be an extension of the first link, thus making it easier for the LQR to stabilize the system. Finally, it would switch to LQR once it gained the right angles for the LQR to work.

Equation 20 shows how energy is pumped into the system by evaluating the amount of energy needed to reach the equilibrium point.

$$u = (E - E_d)\dot{q_1} \tag{20}$$

Where E is the total energy for the system and E_d is the energy desired. After obtaining enough energy it would switch to the PD controller seen in equation 21.

$$u = q_2 K_p + (q_2 - q_2^*)K_d \tag{21}$$

Where q_2^* is the previous angle q_2 (the old error). The P and D gains were chosen by trial-and-error, to what seemed to accomplish the goal most efficiently. Figure 4 illustrates a graph of the swing-up controller. The sudden rises in the first joint angle is due to wrap around of the controller. As seen in the figure it pumps energy into the system and then slow settle until LQR can take over and stabilize it in the end.

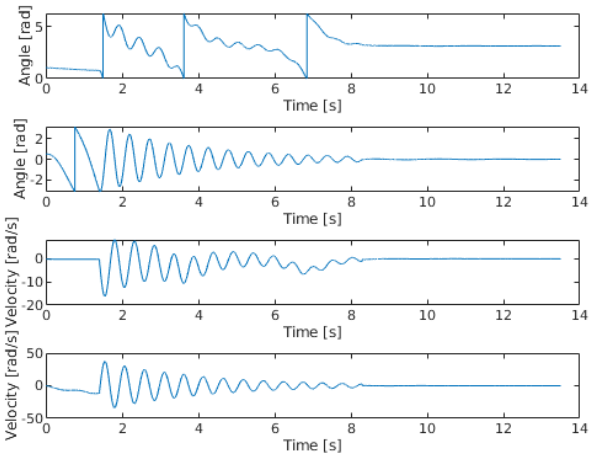

Fig. 4. Graph depicting the angles and velocities for the swing-up controller. Starting angles were 60 degrees for the first joint and 30 degrees for the second joint

The following youtube link will show the simulation of the controller

https://youtu.be/nwXxyafxZuU **(Swing-up acrobot with starting angles 60 and 30)**

This proved to be a rather volatile solution to the swing-up controller as it would only work sometimes. This was due to the fact that the PD controller would remove energy of the system depending on the starting angles of the joints, which meant that the system sometimes would lack energy, thus making the LQR unable to control the system.

7 Conclusion

By using the Lagrange method it was possible to derive equations of motion from the kinetic and potential energy of the acrobot system. This was without considering damping and friction as it would make the system very complex and have page long equations.

Obtaining the equations of motion made it possible to derive the acceleration, velocity and angle of the joints. This allowed a simulation of a simple acrobot system to be created.

Creating a controller for the system requires it to be controllable, thus rewriting the equations of motion to standard robotic manipulator equations was a great choice as the state- and input matrices could be easily obtained from these. Linearizing the manipulator equations about a desired equilibrium point yielded the needed matrices. The system was confirmed to be controllable and a LQR controller was developed for stabilization of the two links on top of each other.

As the acrobot system is non-linear but linearized, an LQR controller is not the optimal solution. This, coupled with the cost function of the LQR is mostly trial-and-error guesses, makes for high possibility that a better controller exist. The robustness of the LQR was not entirely tested as no external forces were applied to the system. The main reason for choosing a LQR controller is that it is easy to derive and it is a broad solution for controlling systems.

A swing-up controller was also attempted although this provided to be very volatile and unstable, as the desired energy for the system varied due to the fact that a PD controller was implemented to help drive the second joint angle to zero.

It should also be noted that a lot of the above conclusion are based on a disregard of the amount of torque allowed. Many real-life motors have a torque limit, but the above controllers had an infinite torque available and could add/remove as much energy as desired. Testing torque limits could be a good experiment for future work.