# Optimization in Robotics (Acrobot) Control


Harvard University


AM205 Advanced Scientific Computing:
Numerical Methods

Final Project Report


*Authors:*
Hengte Lin
Taosha Wang
Yifan Wang

*Professor:*
Chris H. Rycroft


December 7, 2017

**Abstract**

Your abstract.

# 1 Introduction

An **acrobot**[1] (in Figure 1) is a planar two-link robotic arm in the vertical plane with an actuator at the elbow (the red point), but no actuator at the shoulder. An acrobot is a typical underactuated robots, which have less actuators than degrees of freedom. The control of underactuated systems has been an interesting topic in robotics industry for it "gives some reductions of numbers of necessary actuators, of the cost and of the weight of systems" [1].

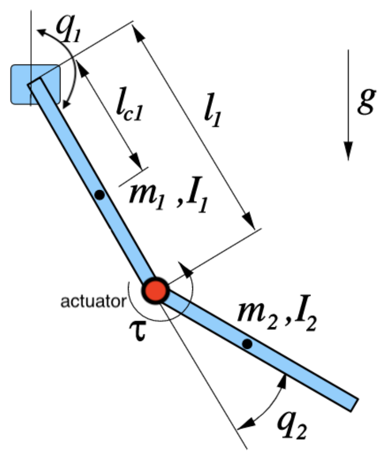


Figure 1: Acrobot (Modified on the basis of picture from [9])

An acrobot closely resembles the movement of a man, and is an important part of a robot. One of the most popular control task studied for the acrobot is the *swing-up* [2] task, in which the system must use the elbow torque to move the system into a vertical configuration and then balance. Previous studies have proposed many controllers for the Acrobot, such as energy based controllers [3][4], controllers based on partiallinearization [3][5],

[1]A link to an animation of Acrobot: `http://www.princeton.edu/~rvdb/WebGL/Acrobot.html`

tracking controller [6], back stepping controller [7], a controller based on the motion of the real gymnast [8] etc. Though optimal control is a powerful framework for specifying complex behaviors with simple objective functions, the computational tools cannot scale well to systems with state dimension more than four or five [9]. In this project, we attempt to find an optimal control solution that is valid from only a single initial condition, instead of solving for the optimal feedback controller for the entire state space. Thus, we represent the optimal control solution as a *trajectory*, $\mathbf{x}(\cdot)$, $\mathbf{u}(\cdot)$ rather than a feedback control function.

The rest of this report is organized as follows: In Section 2, we derive the motion equation of the acrobot using standard, manipulator equation form. In Section 3, we set up the Lagrangian equations and derive necessary conditions for the trajectory optimization problem. In Section 4, we present the results and runtime analysis of the trajectory optimization solutions. In Section 5, we apply a cutting-edge neural network model - the reinforcement learning algorithm to solve the trajectory optimization problem and compare the performance with traditional methods.

## 2 Dynamics of the Acrobot

In Figure 1, $q_1$ is the shoulder joint angle, $q_2$ is the relative joint angle at the elbow and $\mathbf{q} = [q_1, q_2]^T$. With the notations in Figure 1, we can derive the equations of motion in standard, manipulator equation form:

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \mathbf{B}u \tag{1}$$

where

$$\begin{aligned}
\mathbf{H}(\mathbf{q}) &= \begin{bmatrix} I_1 + I_2 + m_2 l_1^2 + 2m_2 l_1 l_{c2} c_2 & I_2 + m_2 l_1 l_{c2} c_2 \\ I_2 + m_2 l_1 l_{c2} c_2 & I_2 \end{bmatrix}, \\
\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) &= \begin{bmatrix} -2m_2 l_1 l_{c2} s_2 \dot{q}_2 & -m_2 l_1 l_{c2} s_2 \dot{q}_2 \\ m_2 l_1 l_{c2} s_2 \dot{q}_1 & 0 \end{bmatrix}, \\
\mathbf{G}(\mathbf{q}) &= \begin{bmatrix} m_1 g l_{c1} s_1 + m_2 g (l_1 s_1 + l_{c2} s_{1+2}) \\ m_2 g l_{c2} s_{1+2} \end{bmatrix}, \\
\mathbf{B} &= \begin{bmatrix} 0 \\ 1 \end{bmatrix},
\end{aligned}$$

and $u$ is the scalar control signal (torque acting on the second joint).

In the next steps we derive the deterministic dynamics from the above motion equations. First, we add frictions $f$ to the system:

$$\mathbf{B}u = \mathbf{B}\hat{u} - f\dot{\mathbf{q}} \tag{2}$$

Substituting equation (2) into the right hand side of equation (1), we get:

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \mathbf{B}\hat{u} - f\dot{\mathbf{q}} \tag{3}$$

Afterwards, $\ddot{\mathbf{q}}$ can be calculated as:

$$\begin{aligned} \ddot{\mathbf{q}} &= \mathbf{H}(\mathbf{q})^{-1}[-\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{G}(\mathbf{q}) + \mathbf{B}\hat{u} - f\dot{\mathbf{q}}] \\ &= \mathbf{H}(\mathbf{q})^{-1}[-\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{G}(\mathbf{q}) - f\dot{\mathbf{q}}] + \mathbf{H}(\mathbf{q})^{\mathbf{-1}}\mathbf{B}\hat{u} \end{aligned} \tag{4}$$

Let $\mathbf{x} = [q_1, q_2, \dot{q_1}, \dot{q_2}]^T$, we can calculate the first-order derivative $\dot{\mathbf{x}}$:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} = \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \mathbf{H}(\mathbf{q})^{-1}[-\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{G}(\mathbf{q}) - f\dot{\mathbf{q}}] \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \mathbf{H}(\mathbf{q})^{\mathbf{-1}}\mathbf{B} \end{bmatrix} \hat{u}, \tag{5}$$

in the format of $\dot{\mathbf{x}} = \mathbf{f_d}(\mathbf{x}, u) = \mathbf{a} + \mathbf{b}u$. The first order derivative $\dot{\mathbf{x}}$ is used in the deterministic dynamic function $\mathbf{x}[n+1] = \mathbf{x}[n] + \dot{\mathbf{x}}[n]$, which calculates the next state given previous state and the torque.

For simplicity, we set all constant scalars equal to 1, hence the above parameters $\mathbf{H}(\mathbf{q})$ and $-\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{G}(\mathbf{q})$ can be written as:

$$\begin{aligned} \mathbf{H}(\mathbf{q}) &= \begin{bmatrix} 3 + 2cos(q_2) & 1 + cos(q_2) \\ 1 + cos(q_2) & 1 \end{bmatrix} \\ -\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{G}(\mathbf{q}) &= \begin{bmatrix} \dot{q_2}(2\dot{q_1} + \dot{q_2})sin(q_2) + 2gsin(q_1) + gsin(q_1 + q_2) \\ -\dot{q_1}^2 - sin(q_2) + gsin(q_1 + q_2) \end{bmatrix} \end{aligned}$$

# 3 Trajectory Optimization

Define cost function as

$$g(x, u) = \frac{r}{2}u^2 + 1 - exp(kcos(x_1) + kcos(x_2) - 2k) \tag{6}$$

The constrained trajectory optimization problem in discrete time:

$$\begin{aligned} &\text{minimize}_{\mathbf{x}_1,\ldots,\mathbf{x}_N,\mathbf{u}_0,\ldots,\mathbf{u}_{N-1}} \sum_{n=0}^{N-1} g(\mathbf{x}[n], \mathbf{u}[n]), \\ &\text{subject to} \quad \mathbf{x}[n+1] = \mathbf{x}[n] + f_d(\mathbf{x}[n], \mathbf{u}[n]). \end{aligned}$$

We used Lagrange multipliers to derive the necessary conditions for our trajectory optimization problem:

$$L(\mathbf{x}[\cdot], \mathbf{u}[\cdot], \lambda[\cdot]) = \sum_{n=0}^{N-1} g(\mathbf{x}[n], \mathbf{u}[n]) + \sum_{n=0}^{N-1} \lambda^T[n] \left( f_d(\mathbf{x}[n], \mathbf{u}[n]) - \mathbf{x}[n+1] \right)$$

$$\begin{aligned} \forall n \in [0, N-1], \frac{\partial L}{\partial \lambda[n]} &= f_d(\mathbf{x}[n], \mathbf{u}[n]) - \mathbf{x}[n+1] = 0 \Rightarrow \mathbf{x}[n+1] = f(\mathbf{x}[n], \mathbf{u}[n]) \\ \forall n \in [0, N-1], \frac{\partial L}{\partial \mathbf{x}[n]} &= \frac{\partial g(\mathbf{x}[n], \mathbf{u}[n])}{\partial \mathbf{x}} + \lambda^T[n] \frac{\partial f_d(\mathbf{x}[n], \mathbf{u}[n])}{\partial \mathbf{x}} - \lambda^T[n-1] = 0 \\ \Rightarrow \lambda[n-1] &= \frac{\partial g(\mathbf{x}[n], \mathbf{u}[n])}{\partial \mathbf{x}}^T + \frac{\partial f_d(\mathbf{x}[n], \mathbf{u}[n])}{\partial \mathbf{x}}^T \lambda[n]. \\ \frac{\partial L}{\partial \mathbf{x}[N]} &= -\lambda[N-1] = 0 \Rightarrow \lambda[N-1] = 0 \\ \forall n \in [0, N-1], \frac{\partial L}{\partial \mathbf{u}[n]} &= \frac{\partial g(\mathbf{x}[n], \mathbf{u}[n])}{\partial \mathbf{u}} + \lambda^T[n] \frac{\partial f_d(\mathbf{x}[n], \mathbf{u}[n])}{\partial \mathbf{u}} = 0. \end{aligned}$$

# 4 Results

# 5 Reinforcement Learning

## References


[1] T. Henmi, Mingcong Deng, and A. Inoue. "Swing-up Control of the Acrobot Using a New Partial Linearization Controller Based on the Lyapunov Theorem". In: *2006 IEEE International Conference on Networking, Sensing and Control*. 2006, pp. 60–65. DOI: 10.1109/ICNSC.2006.1673118.

[2] M. W. Spong. "The swing up control problem for the Acrobot". In: *IEEE Control Systems* 15.1 (1995), pp. 49–55. ISSN: 1066-033X. DOI: 10.1109/37.341864.

[3] Y. Dong et al. "Energy-Based Control for a Class of Under-Actuated Mechanical Systems". In: *2008 Congress on Image and Signal Processing*. Vol. 3. 2008, pp. 139–143. DOI: 10.1109/CISP.2008.304.

[4] A. Inoue et al. "Swing-up and stabilizing control system design for an Acrobot". In: *2007 IEEE International Conference on Networking, Sensing and Control*. 2007, pp. 559–561. DOI: 10.1109/ICNSC.2007.372839.

[5] X. Z. Lai et al. "Comprehensive Unified Control Strategy for Underactuated Two-Link Manipulators". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39.2 (2009), pp. 389–398. ISSN: 1083-4419. DOI: 10.1109/TSMCB.2008.2005910.

[6] Z. Dengfeng et al. "An improved departed domain control strategy for acrobot". In: *2017 3rd IEEE International Conference on Control Science and Systems Engineering (ICCSSE)*. 2017, pp. 5–9. DOI: 10.1109/CCSSE.2017.8087884.

[7] T. Nam et al. "Swing-up Control and Singular Problem of an Acrobot System". In: 20(1) (2002), pp. 85–88.

[8] T. Henmi et al. "Swing-up control of an Acrobot having a limited range of joint angle of two links". In: *2004 5th Asian Control Conference (IEEE Cat. No.04EX904)*. Vol. 2. 2004, 1071–1076 Vol.2.

[9] Russ Tedrake. *Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation (Course Notes for MIT 6.832)*. http://underactuated.mit.edu/.