IND E 599 PROJECT REPORT

Reinforcement Learning for Swing-Up Control of Acrobot

AKSHAY PAI RAIKAR **apr244@uw.edu**

Introduction:

Th Acrobot is an underactuated system which consists of two rods – "arms" fixed via pin joints wherein the joint between the two arms is actuated by a single motor. It can be thought of as a 2 degree-of-freedom planar robot with a single actuated joint. It is a highly simplified model of a human gymnast on a high bar, in which the passive first joint models the gymnast's hands on the bar, and the actuated second joint models the gymnast's hips. To mimic the motion of the gymnast going from a natural hanging position to a handstand on the bar, a control objective of swinging up the Acrobot from the downward position and balancing it at the upright position has been intensively studied in recent years. Prior to the work of M.W Spong [1] people had worked on Open Loop Control Strategies for and Sinusoidal Excitation strategies for swing up control. Several other strategies including Linear Regulator Control and Dynamic Programming have been attempted successfully in developing controllers for this system. The Swing Up Control assumes importance as it allows one to develop theory for further complicated robots with acrobatic behaviours. The techniques developed for controlling this system have been extended to dynamic locomotion. In this project the author aims to use active reinforcement learning methods to achieve Swing-Up control. This problem has become a benchmark problem to test Reinforcement Learning Algorithms ever since being discussed by Sutton and Barto [2] in their seminal book.

System Dynamics:

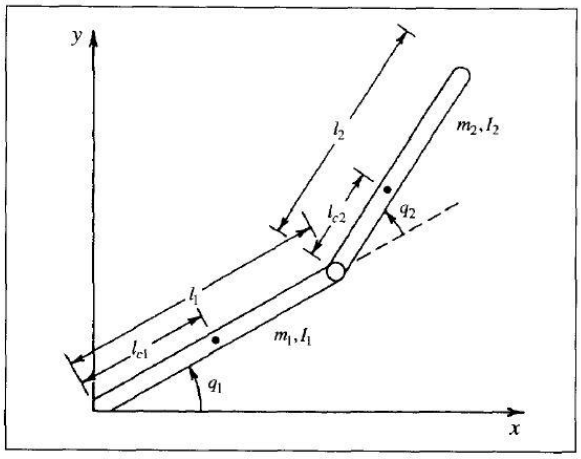


Fig.1 The Acrobot

The System Dynamic Equations were taken from [1], [2], [3] and are given by:

It must be noted that these were derived using the Method of Lagrange assuming no friction at the joints and only a single actuator at the joint linking Arm1 and Arm 2.

$$d_{11}\ddot{q}_1 + d_{12}\ddot{q}_2 + h_1 + \phi_1 = 0 \quad (1)$$

$$d_{12}\ddot{q}_1 + d_{12}\ddot{q}_2 + h_2 + \phi_2 = \tau \quad (2)$$

Where;

$$d_{11} = m_1 l_{c1}^2 + m_2(l_1^2 + l_{c2}^2 + 2l_1 l_{c2}\cos(q_2)) + I_1 + I_2 \quad (3)$$

$$d_{22} = m_2 l_{c2}^2 + I_2 \quad (4)$$

$$h_1 = -m_2 l_1 l_{c2} \sin(q_2)\dot{q}_2^2 - 2m_2 l_1 l_{c2} \sin(q_2)\dot{q}_2\dot{q}_1 \quad (5)$$

$$h_2 = m_2 l_1 l_{c2} \sin(q_2)\dot{q}_1^2 \quad (6)$$

$$\phi_1 = (m_1 l_{c1} + m_2 l_1) g \cos(q_1) + m_2 l_{c2} g \cos(q_1 + q_2) \quad (7)$$

$$\phi_2 = m_2 l_{c2} g \cos(q_1 + q_2) \quad (8)$$

The system parameters that we have assumed for our problem are taken from [4] [5]:

Parameter	*Description*	*Value*
l_1, l_2	Link lengths	1.0
l_{c1}, l_{c2}	Joint to mass center	0.5
m_1, m_2	Link masses	1.0
I_1, I_2	Link inertias	1.0
$\{\tau_{max}, \tau_{min}\}$	Torque range	{-1.0, 1.0}

From the linearization we know that the system has two fixed point given by (-π/2,0) which is a stable centre and (π/2,0) which is a saddle point.

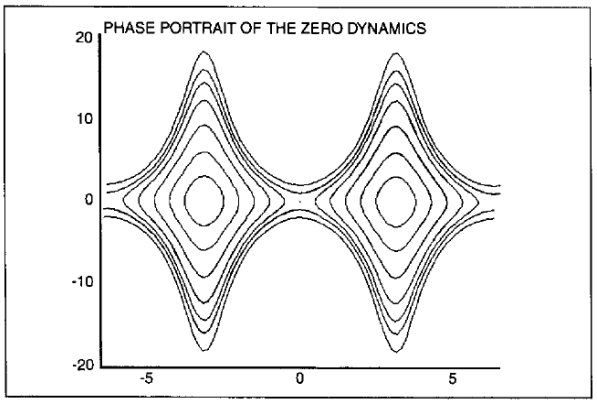


Fig.2 Phase Portrait of Acrobot System

Credits: M. W. Spong, The Swing Up Control Problem for the Acrobot

Note that the above in the above phase portrait taken from [1] shoes the stable centre and saddle point to be at (0,0) and (π,0) as the authors assumed a co-ordinate system wherein the angles are measured with respect to the vertical whereas in this report the angles are measured with respect to the horizontal. However, the phase portrait diagram is meant to convey the locations of stable centres and saddle points without loss of generality.

Our goal is to determine a control policy to stabilize the Acrobot system at the (π/2,0) position.

The dynamics are simulated by using Forward Euler Integration and the state vector is packaged as $x = [\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]^T$. Time step used for the problem is 0.05 seconds as quoted in [2].

Formulation of Problem as an MDP and Choice of Solution Method:

The Reinforcement Learning problem is defined by Sutton and Barto [2] to be the problem of learning from interaction to achieve a goal. The learner/decision maker is called the agent and the thing it interacts with is the environment. The agent and environment interact continually with the environment presenting new situations to the agent. The environment gives rise to rewards which the agent tries to maximize over time. Mathematical

Formulation of the RL Problem is done via **MDP**s or Markov Decision Processes. As per Sutton and Barto MDPs are all you need to know, to understand 90% of modern reinforcement learning.

A Markov Decision Process is a discrete time stochastic control process. At every time step, the process exists in some state $\mathbf{s}$, and the decision maker may choose any action $\mathbf{a}$ that is available in state $\mathbf{s}$. The process responds at proceeding time step by moving into a new state $\mathbf{s}'$. The new state $\mathbf{s}'$ is however chosen stochastically, and the process gives the decision maker a corresponding reward $\mathbf{R}(\mathbf{s}'|\mathbf{s},\mathbf{a})$. The probability that the process moves into its new state $\mathbf{s}'$ depends on the chosen action. It is given by the state transition function $\mathbf{P}(\mathbf{s}'|\mathbf{s},\mathbf{a})$. The state transitions however follow the Markov Property, where it is conditionally independent of all previous states and actions.

Therefore, to solve our control problem, it is imperative that we need to discretize our state space and action space.

Actions: In our case we have already assumed before that that our actuator is limited to a range between -1 N.m and 1 N.m. We will restrict the action space to three choices -1N.m where the acrobot swings leftward, 0 where the actuator doesn't exert any torque and +1 N.m where the acrobot swings rightward.

$$\mathbf{A} = [-\mathbf{1}, \mathbf{0}, +\mathbf{1}] \textbf{ N.m}$$

States: In case of our state space we have chosen a coarse discretization with angles being discretized into 5 bins and angular velocities being discretized into 3 bins with arbitrarily initialized values given by:

$$\boldsymbol{\theta_1} \in [-\boldsymbol{\pi}/\mathbf{2}, -\boldsymbol{\pi}/\mathbf{4}, \mathbf{0}, \boldsymbol{\pi}/\mathbf{4}, \boldsymbol{\pi}/\mathbf{2}] \textbf{ rad}$$

$$\boldsymbol{\theta_2} \in [-\boldsymbol{\pi}/\mathbf{2}, -\boldsymbol{\pi}/\mathbf{4}, \mathbf{0}, \boldsymbol{\pi}/\mathbf{4}, \boldsymbol{\pi}/\mathbf{2}] \textbf{ rad}$$

$$\dot{\boldsymbol{\theta}}_{\mathbf{1}} \in [-\boldsymbol{\pi}/\mathbf{4}, \mathbf{0}, \boldsymbol{\pi}/\mathbf{4}] \textbf{ rad/s}$$

$$\dot{\boldsymbol{\theta}}_{\mathbf{2}} \in [-\boldsymbol{\pi}/\mathbf{4}, \mathbf{0}, \boldsymbol{\pi}/\mathbf{4}] \textbf{ rad/s}$$

There were no constraints on the angular positions, but the angular velocities were restricted to $\dot{\boldsymbol{\theta}}_{\mathbf{1}} \in [-\mathbf{4\pi}, \mathbf{4\pi}]$ **rad/s and** $\dot{\boldsymbol{\theta}}_{\mathbf{2}} \in [-\mathbf{9\pi}, \mathbf{9\pi}]$ **rad/s**. This was in accordance with the problem statement provided by Sutton and Barto in [2].

Rewards: In general, MDP formulations have a clearly defined mapping function between to denote the rewards attained when transitioning from a state-action pair to another set of possible states. However, in the case of reinforcement learning no such aping is known and rewards and provided to the agent only upon achievement of certain pre-defined states.

The objective for the control of the system is to vertical balancing of the system which means that the tip of the 2nd arm of the acrobot must be in the goal A uniform rewards of **-1 is set for all states that are not the final goal state** and a **large reward of +100 for attaining final goal state**.

Transition Matrices: Since we are using Reinforcement learning based methods for control the system /agent has no knowledge of its environment and hence we cannot define any Transition Probability Function/Matrix that maps between current state-action pairs to new states.

Solution Approach and Explanation of the Method

The algorithm used for swing-up control of the Acrobot System is called SARSA. SARSA is also known as Temporal Differencing Control.

Initial reinforcement learning methods relied in the use of the Value function. Some examples of this method include Monte Carlo Prediction and Monte Carlo Control. Simply stated in these cases the system evaluates the returns from following a given action policy depending on the sequence of states that yield highest utility/return. This means that in the absence of a transition function for that state-action pairs the system needs to be simulated many times to take advantage of the Central Limit theorem to converge on a stable policy and state sequence.

Temporal Differencing is based off the concept that animal and human brains place importance to the temporality of a phenomenon to predict future instances of the being presented in the same state. Unlike the previous Monte Carlo Methods allows for Bootstrapping. In the MC methods one would have to wait for the end of an entire set of episodes to update the value function. In the case of TD learning one can update the utility function after progressing to the subsequent states.

The update rule is given by:

$$U(s_t) = U(s_t) + \alpha * [r_{t+1} + \gamma * U(s_{t+1}) - U(s_t)]$$

where the $Target = E_\pi[r_{t+1} + \gamma * U(s_t)]$ the term in the parentheses is the difference between the systems estimate of the utility function and the current utility function value of the present state. **α** is the **learning rate** or determines to what extent new information overrides old information of the environment and **ϒ** is the **discount factor** that accounts for the importance of future rewards.

However, a drawback of this method is that it requires a known policy to iterate over whereby it manages to select the right set of state sequences to achieve the goal state.

SARSA: This method is also known as Temporal Difference Control wherein we replace the Utility or value function that solely depends on the state of the system with the Q function which accounts for state -action pairs at every instance.

The update rule is this case is given by:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha * [r_{t+1} + \gamma * \mathrm{Q}(s_{t+1}, a_{t+1}) - \mathrm{Q}(s_t, a_t)]$$

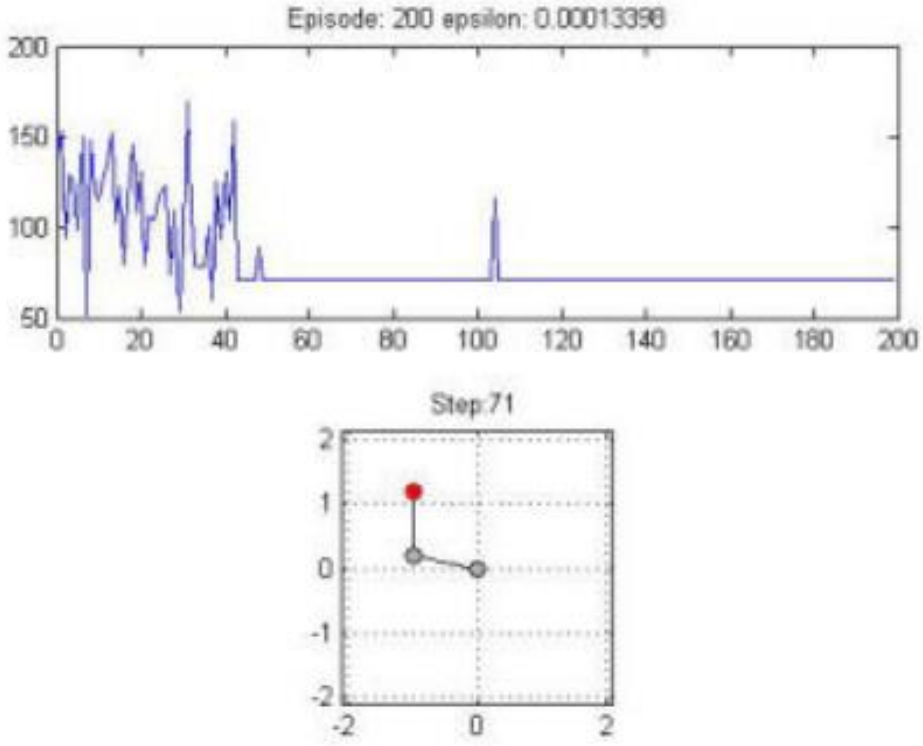
The estimation is based on the tuple State-Action-Reward-State-Action hence giving rise to the moniker SARSA.

The SARSA Algorithm as per [2] is as follows:

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Choose a from s using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g., ε-greedy)
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
    $s \leftarrow s'$; $a \leftarrow a'$;
  until s is terminal

Results and Conclusion:

In this project the author initializes the Q-table with initial values of zero and makes use of the ε-greedy strategy with a small value of **ε = 0.001** to ensure that the algorithm allows for exploration if the environment to not reach a sub-optimal solution. The small value of **ε** was needed because the task because long sequences of correct actions were required to do well on the task as per Sutton and Barto [2]. The learning rate **α = 0.5** and discount factor **ϒ =1**. It was important to have a discount value to encompass the learnings from previous episodes. The system was run for 200 episodes with a maximum of 1000 steps per episode.



References

[1] M. W. Spong, "The Swing Up Control Problem for the Acrobot," *ICRA 1994,* 1995.

[2] A. R. Sutto, Reinforcement Learning: An Introduction.

[3] R. Ueda, "Dynamic Programming for Global Control of The Acrobot and Its Chaotic Aspect," *ICRA,* 2008.

[4] G. Boone, "Minimum Time Control of the Acrobot".

[5] M. Patacchiola, "root@mpatacchiola:~$: Dissecting Reinforcement Learning-Part.1," 09 Dec 2016. [Online]. Available: https://mpatacchiola.github.io/blog/2016/12/09/dissecting-reinforcement-learning.html. [Accessed 05 May 2018].

[6] L. Johnson, Adaptive Swing-up and Balancing Control of Acrobot: Bachelor's Thesis, Cambridge, MA: Massachusetts Institute of Technology, 2009.