16.32 Final Project Report: Dynamic Programming for Minimum-Time Acrobot Control

Andrew Torgesen · May 4, 2020

1 Problem Definition

Feedback control of highly nonlinear and chaotic systems is rarely, if ever, a straightforward undertaking. For such systems, linear optimal feedback control laws fail to provide stability unless, perhaps, they are continually re-calculated by iteratively linearizing the nonlinear system about a pre-computed open-loop trajectory from a nonlinear solver. Alternative methods of nonlinear control are often tailored to specific systems by exploiting some idiosyncrasy in their dynamics, as with the methods described in [2].

Unlike nonlinear control methods based on open-loop trajectory optimization or ad hoc designs, dynamic programming offers an alternative closed-loop control option that can be applied to a wide class of highly nonlinear systems. In dynamic programming, heuristic search is used to derive a control policy for the entirety of the state space of the system. The general applicability of dynamic programming methods comes at the cost of high demands on memory usage and computational power in the derivation of the controller, though subsequent application of the derived control strategy requires significantly less resources.

This project aims to follow the lead of efforts in [4, 3, 1] to apply dynamic programming (in the form of value iteration of the Bellman update equation) to the minimum-time height task for the acrobot, which is a highly nonlinear and chaotic system. Defined in [1], the height task entails controlling the acrobot to reach a specified height with a near-zero velocity. Moreover, the minimum-time specification entails restricting the available inputs to some pre-defined saturation values: $u \in \{-\tau_{\max}, 0, \tau_{\max}\}$ to complete the height task as quickly as possible. This report details the efforts made to model the problem and the dynamic programming equations, implement them in C++, and apply the derived control strategy in simulation to achieve desired behavior with the acrobot as well as the (simpler) nonlinear inverted pendulum.

2 Methods

Because of the notoriety of high run times associated with dynamic programming methods when applied to systems with a state space dimensionality as high as that of the acrobot, it is advisable to perform test runs on a simpler, more tractable system for development and debugging. Thus, analysis and results are also supplied for the inverted pendulum system, whose smaller state space and more tame dynamics allow for additional insight into the derived control policy and underlying value iteration performance.

2.1 System Model: Inverted Pendulum

Figure 1 depicts the geometry of the state space of the inverted pendulum, where an input torque $\tau(t)$ is applied at the frictionless joint of a link of length l connected to a point mass m, resulting in the evolution of angle $\theta_1(t)$ against the force of gravity g.

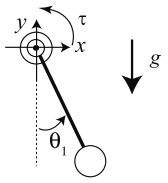

Figure 1: State space depiction for the inverted pendulum.

The two degree-of-freedom dynamic model for the inverted pendulum can be written concisely with state vector $x = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T = \begin{bmatrix} \theta_1 & \dot{\theta}_1 \end{bmatrix}^T$ as:

$$\dot{x} = \begin{bmatrix} x_2 \\ \frac{\tau - mgl\sin(x_1)}{ml^2} \end{bmatrix}. \tag{1}$$

For the analysis in this project, model parameter values of $m = 1.0$ kg, $l = 1.0$ m, and $g = 10.0$ m/s^2 are used.

The "goal state," or objective of the dynamic programming control strategy to be achieved in a minimum-time fashioned, is defined as the set of states for which the link is standing upright at or near its maximum height with a near-zero velocity:

$$\theta_1 \approx 180°, \dot{\theta}_1 \approx 0 \tag{2}$$

2.2 System Model: Acrobot

Figure 2 depicts the state space of the acrobot as a slight modification and extension of the inverted pendulum. An additional link of mass m_2, length l_2, and length to center-of-mass l_{c_2} is attached to the base link (m_1, l_1, l_{c_1}), and the torque τ is now applied at the center joint rather than at the base joint, causing the evolution of both $\theta_1(t)$ and $\theta_2(t)$ against the force of gravity g.

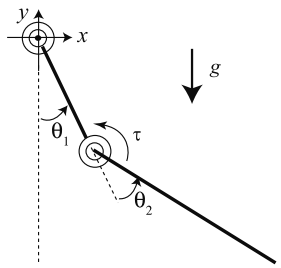

Figure 2: State space depiction for the acrobot.

The four degree-of-freedom dynamic model for the acrobot, with state vector $x = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T = \begin{bmatrix} \theta_1 & \theta_2 & \dot{\theta}_1 & \dot{\theta}_2 \end{bmatrix}^T$, can be expressed as:

$$\dot{x} = \begin{bmatrix} x_3 \\ x_4 \\ -(d_2\dot{x}_4 + \phi_1)/d_1 \\ \frac{\tau + \phi_1 d_2/d_1 - m_2 l_1 l_{c_2} x_1^2 \sin x_2 - \phi_2}{m_2 l_{c_2}^2 + I_2 - d_2^2/d_1} \end{bmatrix}, \tag{3}$$

where $d_1 = m_1 l_{c_1}^2 + m_2(l_1^2 + l_{c_2}^2 + 2l_1 l_{c_2}\cos x_2) + I_1 + I_2$, $d_2 = m_2(l_{c_2}^2 + l_1 l_{c_2}\cos x_2) + I_2$, $\phi_1 = -m_2 l_1 l_{c_2} x_4^2 \sin x_2 - 2m_2 l_1 l_{c_2} x_4 x_3 \sin x_2 + (m_1 l_{c_1} + m_2 l_1) g\cos(x_1 - \pi/2) + \phi_2$, $\phi_2 = m_2 l_{c_2} g\cos(x_1 + x_2 - \pi/2)$, $I_1 = m_1 l_1^2/12$, and $I_2 = m_2 l_2^2/12$. For the analysis in this project, model parameter values of $m_1 = 1.0$ kg, $l_1 = 1.0$ m, $l_{c_1} = 0.5$ m, $m_2 = 1.0$ kg, $l_2 = 2.0$ m, $l_{c_2} = 1.0$ m, and $g = 9.8$ m/s^2 are used.

A brief comparison of Equations 1 and 3 reveals the comparatively high nonlinearity of the acrobot compared to the inverted pendulum. In fact, as discussed in [3], the acrobot can be aptly referred to as a chaotic system, where a slightly perturbed initial condition $\theta_1(0)$, $\theta_2(0)$ leads to a vastly different state space evolution given the same input history $\tau(t)$.

Defined in [1], the set of goal states for the acrobot consist those in which the free end of the second link clears a pre-defined height h above the height of the base joint with a near-zero velocity:

$$-l_1\cos\theta_1 - l_2\cos(\theta_1 + \theta_2) \geq h, \dot{\theta}_1 \approx \dot{\theta}_2 \approx 0 \tag{4}$$

The conditions defined in Equation 4 are referred to collectively as the "height task" for the acrobot.

2.3 State Grid Discretization

To apply dynamic programming on a continuous system, the state space of the system must be discretized, bounded, and interpolated. The size of the resulting state space grid is ultimately the principal deciding factor of the run time of the control policy generation algorithm. Thus, for the state space discretization in this project, the symmetry of the inverted pendulum and acrobot systems is exploited in this project. As is shown in Tables 1 and 2, $x_1 = \theta_1$ is only discretized from 0 to 180°, and the interpolation algorithm negates the other state space variables to provide the mirrored scenario when necessary. This trick serves to at least halve the size of the state space grid for each system.

Obviously, because the inverted pendulum has half the number of degrees of freedom as the acrobot, it only requires a two-dimensional state space grid. However, in an effort to test the entirety of the state space generation, interpolation, and usage algorithms before tackling the acrobot, a full four-dimensional state space grid is used for the inverted pendulum, as depicted in Table 1. The variables x_2 and x_4 simply don't respond to control inputs and have no bearing on the determination of whether the state has reached a goal state, like the table suggests.

Table 2 gives the state space discretization and goal states for the acrobot. Due to the chaotic nature of the acrobot, the range of discretized angular velocities is expanded, also acknowledging that a wider velocities tends to be induced in θ_2 than in θ_1.

Table 1: State space discretization and goal state definition for the inverted pendulum.

x	$x_{\min}$	$x_{\max}$	**Goal**	Δx
x_1	$5°$	$175°$	$[175°]$	$10°$
x_2	$5°$	$355°$	$[x_2]$	$10°$
x_3	$-355°/s$	$355°/s$	$[-5, 5]°/s$	$10°$
x_4	$-355°/s$	$355°/s$	$[x_4]$	$10°$

Table 2: State space discretization and goal state definition for the acrobot. $[\Phi]$ is the set of members of x_1 and x_2 for which Equation 4 holds true.

x	$x_{\min}$	$x_{\max}$	**Goal**	Δx
x_1	$5°$	$175°$	$[\Phi]$	$10°$
x_2	$5°$	$355°$	$[\Phi]$	$10°$
x_3	$-715°/s$	$715°/s$	$[-10, 10]°/s$	$10°$
x_4	$-1615°/s$	$1615°/s$	$[-10, 10]°/s$	$10°$

With the given discretizations, the size of the inverted pendulum grid is $18 \times 36 \times 72 \times 72 = 3,359,232$ states, and the size of the acrobot grid is $18 \times 36 \times 144 \times 324 = 30,233,088$ states, or approximately 10 times the size of the inverted pendulum grid.

2.4 Dynamic Programming Algorithm

2.4.1 Value Iteration

With matched state space grid sizes, the same dynamic programming and state space interpolation routines are applied to both the inverted pendulum and the acrobot. Dynamic programming is performed through the use of the discounted value iteration algorithm for the Bellman update equation:

$$V^*_{k+1}[x] = \max_{u \in \mathcal{U}} R[x_{t+1}(x, u)] + \gamma V^*_k[x_{t+1}(x, u)] \tag{5}$$

To explain Equation 5, each node x in the state space is initialized with an associated value (or *opposite of cost-to-go*) $V^*_0[x] = 0$ and transition reward function value $R[x]$ (which is the reward obtained by transitioning

into x from a different state), which is equal to 1000 if x is a goal state and zero otherwise. Subsequently, each combination of state and possible input $\tau \in \mathcal{U} = \{-\tau_{\max}, 0, \tau_{\max}\}$ is iterated over to replace $V_k^*[x]$ with the maximum possible $V_{k+1}^*[x]$ given the available inputs. As Equation 5 suggests, each candidate $V_{k+1}^*[x]$ is obtained by simulating the evolution of the state over a time period given the input (or action) u and accounting for the possible reward and value gained for taking that action to transition to the new state x_{t+1}. For this project, simulation is carried out over a time step of $\Delta t = 0.2$ seconds using fourth-order Runge-Kutta integration. The discount factor $\gamma = 0.9 < 1$ is used to penalize state transitions that meander over time and do not arrive at a goal state sufficiently quickly. This value iteration process is repeated until the maximum value change across the state grid is smaller than a pre-defined value ϵ:

$$\max_x |V_{k+1}^*[x] - V_k^*[x]| < \epsilon \tag{6}$$

Once the value grid converges, the optimal control policy is then extracted by defining the optimal input at each state, $u^*[x]$, as the input from $\mathcal{U}$ that yields the greatest resultant value gain:

$$u^*[x] = \arg\max_{u \in \mathcal{U}} R[x_{t+1}(x,u)] + \gamma V^*[x_{t+1}(x,u)] \tag{7}$$

The derived values of $u^*[x]$ for each state can be written into memory and used as a closed-loop feedback control law, $u(x) = u^*[x]$, for the simulated system (again, using $\Delta t = 0.2$ seconds and fourth-order Runge-Kutta integration).

2.4.2 Interpolation

Because high-order integration is being performed on a discretized grid for both the policy generation and policy application phases, interpolation must be used to account for the majority of cases in which the state evolution does not fall cleanly onto another state grid point. For four-dimensional interpolation, the Natural Neighbor algorithm, derived from Voronoi tesselation techniques, is used. The algorithm can be understood intuitively for equally-spaced grid nodes as having each node surrounded by an n-dimensional cube, and an intermediate point is assigned a weighted average value based on how much "volume" from each neighboring grid point's cube the intermediate point's cube is "stealing."

Furthermore, during value iteration, if the evolved state ends up outside of the defined grid, then the corresponding value function for that state evolution $V_k^*[x_{t+1}]$ is assigned the extremely undesirable value of -1000, discouraging any inputs that lead to leaving the state grid.

3 Results

All of the following results correspond to a value iteration convergence criterion of $\epsilon = 0.001$ and input torque saturation value of $\tau_{\max} = 5$ N-m. $\tau_{\max}$ is carefully selected to be large enough to allow each system to overcome the force of gravity for most initial configurations, yet small enough to avoid losing control of the state velocities given the level of state grid discretization.

3.1 Algorithm Convergence

The value iteration algorithm exhibits vastly different convergence times between the inverted pendulum and the acrobot, which is directly attributable to the size of the underlying state grids. The inverted pendulum value iteration converges after 49 iterations at an average iteration time of 48 seconds, resulting in a total run time of **40 minutes**. By contrast, the acrobot value iteration converges after 68 iterations with an average iteration time of 705 seconds for an enormous run time of **13 hours**.

Figures 3 and 4 characterize the convergence properties of the value iteration algorithms for the inverted pendulum and acrobot, respectively. From the figures, it is apparent that each process exhibits two separate convergence modes with different slopes. Surface-level analysis of input maps like Figures 5 and 6 for different values of ϵ suggests that most of the changes in $u^*[x]$ occur during the first convergence mode, though further analysis is needed to determine why this is appears to be the case.

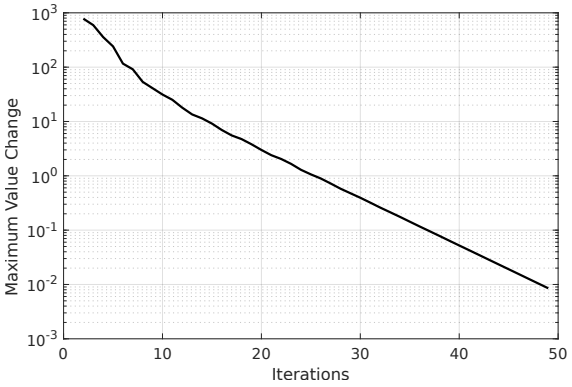

Figure 3: Value iteration convergence rate for the inverted pendulum. The maximum value change corresponds to the maximum difference in the value of Equation 5 over the entire state space grid for the current iteration.

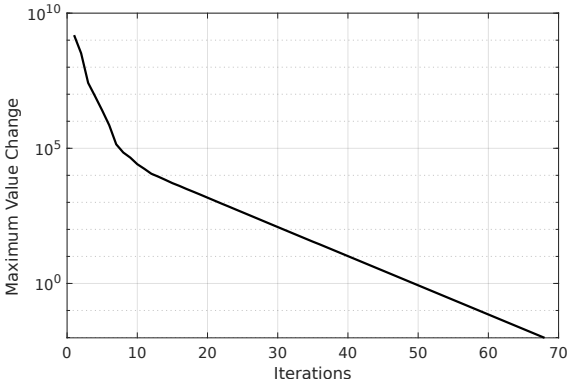

Figure 4: Value iteration convergence rate for the acrobot. The maximum value change corresponds to the maximum difference in the value of Equation 5 over the entire state space grid for the current iteration.

Once its value iteration has converged, the visualized geometry of the derived optimal control policy for the inverted pendulum provides some intuition for the shape of the cost-to-go in a minimum-time problem. Figure 5 clearly demarcates the switching curves for the minimum-time control in the phase plane of the inverted pendulum. Moreover, the curves are mostly consistent with the notion that torque must be applied such that a velocity away from the goal position can be counteracted *unless* that velocity is so great that it is actually more effective to aid the pendulum in swinging the long way around back to the goal state.

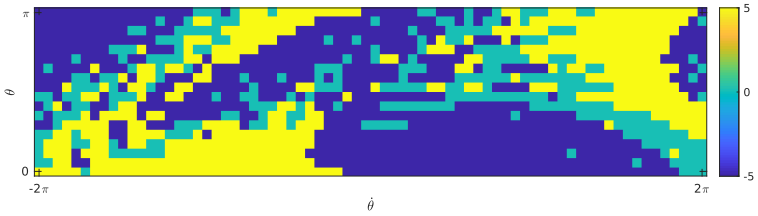

Figure 5: Optimal control policy for the minimum-time swing up problem for the inverted pendulum, derived with value iteration.

By contrast, the derived control policy for the acrobot pictured in Figure 6 appears to appeal to no sense of

geometric intuition. The Figure shows the control policy as a function of θ_1 and θ_2 at cross-sections of the four-dimensional policy for increasing values of $\dot{\theta}_1$ and $\dot{\theta}_2$. The apparent chaos of the minimum-time control policy points to the chaotic aspect of the acrobot.

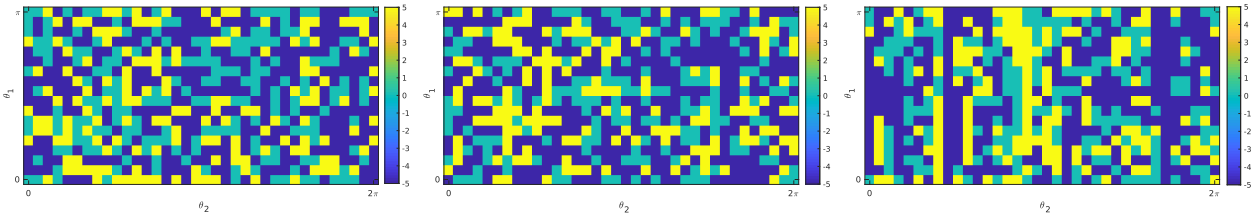

Figure 6: Optimal control policy for the minimum-time height task for the acrobot (visualized in cross-sections of increasing values of $\dot{\theta}_1$ and $\dot{\theta}_2$), derived with value iteration.

3.2 Policy Effectiveness

In order to provide a representative picture of the overall effectiveness of the derived control policies for the inverted pendulum and acrobot, 400 simulations from random initial configurations are run for each system.

First, the results of those 400 simulations are shown for the inverted pendulum, whose results once again provide some opportunity to corroborate with physical intuition. For the inverted pendulum, a maximum of 10 seconds is given for the control policy to achieve the swing up task in each simulated trial. Figures 7 and 8 depict the success rates of the control policy in each of the 400 randomized trials with $\tau_{\max}$ values of 3 and 5 N-m, respectively. It is apparent from comparing the two figures that 3 N-m is not sufficient to allow the inverted pendulum to overcome the force of gravity within 10 seconds when initialized well below the $y = 0$ line. With an allowed torque of 5 N-m, however, a success rate of about 60% is possible from a much wider range of initial conditions.

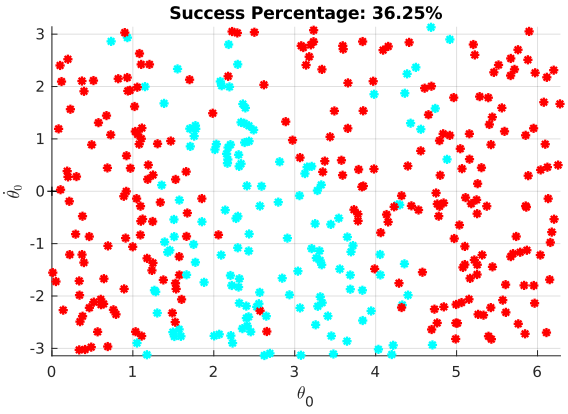

Figure 7: Simulated randomized trial results for the inverted pendulum control policy with $\tau_{max} = 3$ N-m. Successful trials are marked in cyan and unsuccessful trials are marked in red.

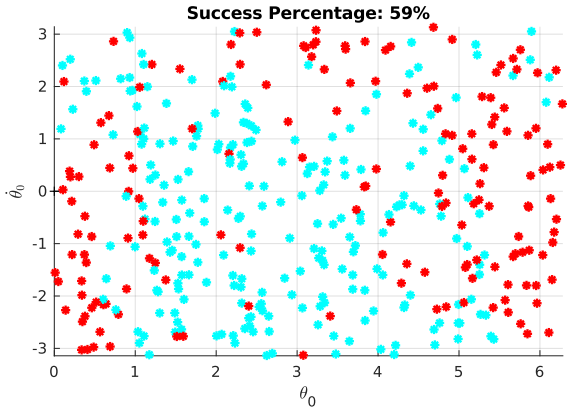

Figure 8: Simulated randomized trial results for the inverted pendulum control policy with $\tau_{max} = 5$ N-m. Successful trials are marked in cyan and unsuccessful trials are marked in red.

Figures 9-12 depict both successful and unsuccessful representative samples from the 400 simulation trial runs. It is shown that in the successful trials, the control policy is able to reach *and maintain* the goal state within approximately six seconds, albeit with some jitteriness. Conversely, in unsuccessful trials, the applied

torque is unable to build up the speed needed to swing the pendulum arm up within the allotted time (or, in some cases, within any amount of time).

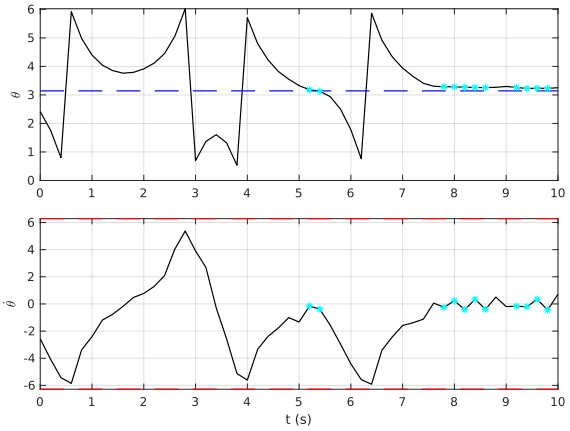

Figure 9: A successful trial run for the inverted pendulum starting from a randomized initial condition. Cyan markers indicate when the system satisfies the success criterion. Angle values given in radians.

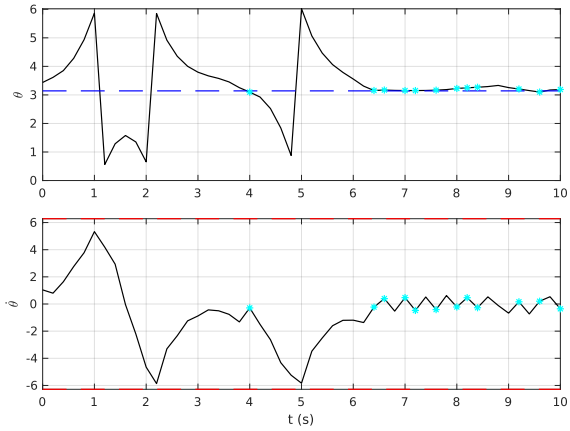

Figure 10: A successful trial run for the inverted pendulum starting from a randomized initial condition. Cyan markers indicate when the system satisfies the success criterion. Angle values given in radians.

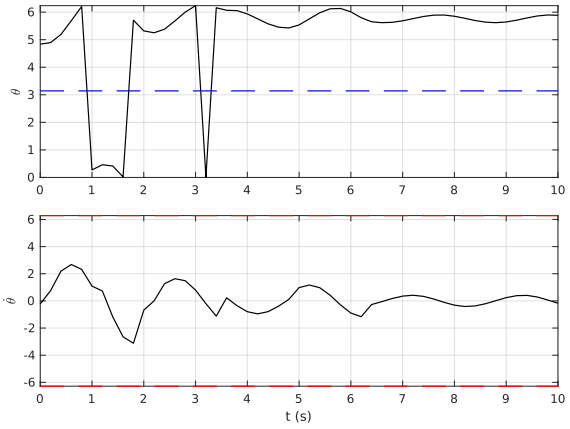

Figure 11: An unsuccessful trial run for the inverted pendulum starting from a randomized initial condition. Angle values given in radians.

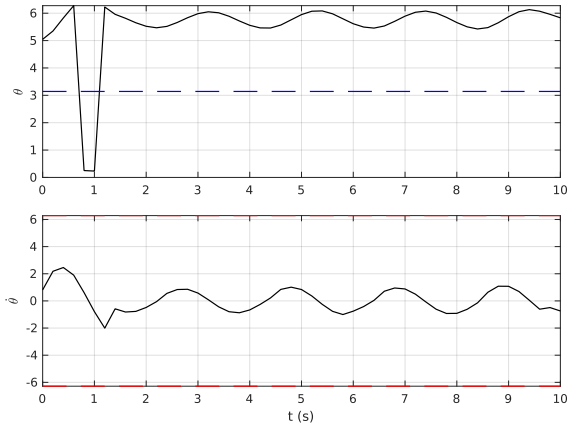

Figure 12: An unsuccessful trial run for the inverted pendulum starting from a randomized initial condition. Angle values given in radians.

For the 400 acrobot trials, random initial values for θ_1 and θ_2 are used, always initializing the angular velocities to zero. The control policy is given a maximum of 20 seconds to achieve the height task with $h = 2.0$ m. Figure 13 gives the success rate for all trials, which comes out to about 50%. It is apparent from the figure that both successful and unsuccessful trials are scattered relatively evenly throughout the configuration space.

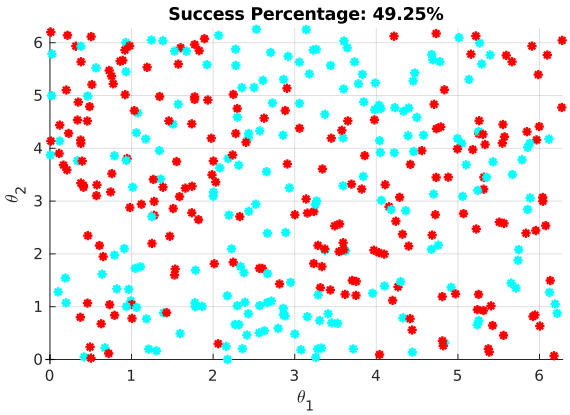

Figure 13: Simulated randomized trial results for the acrobot control policy with $\tau_{max} = 5$ N-m. Successful trials are marked in cyan and unsuccessful trials are marked in red.

Figures 14-17 depict representative successful and unsuccessful trial runs for the acrobot. In contrast with the inverted pendulum, the acrobot achieves the goal state at widely varying times depending on the initial condition and, more unfortunately, is always unable to maintain a goal configuration for long. Unsuccessful trial runs are difficult to parse, though one common theme appears to be that the acrobot is generally unable to lift its first link above the horizontal $y = 0$ line, as with the unsuccessful inverted pendulum runs.

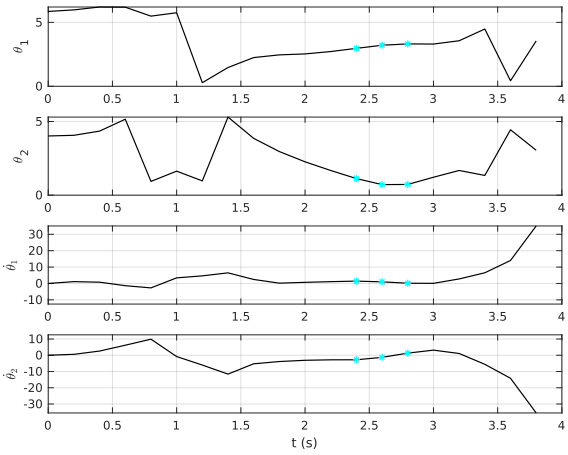

Figure 14: A successful trial run for the acrobot starting from a randomized initial condition. Cyan markers indicate when the system satisfies the success criterion. Angle values given in radians.

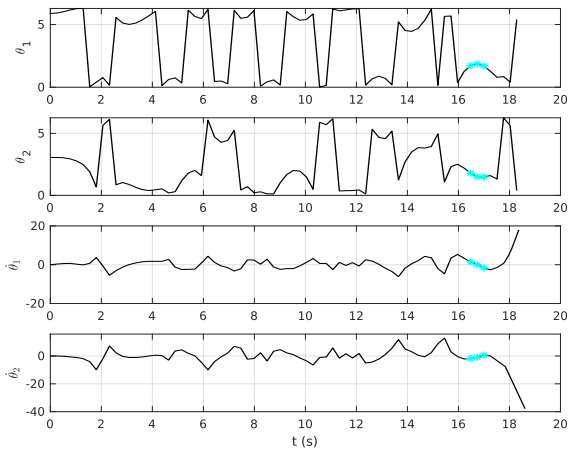

Figure 15: A successful trial run for the acrobot starting from a randomized initial condition. Cyan markers indicate when the system satisfies the success criterion. Angle values given in radians.

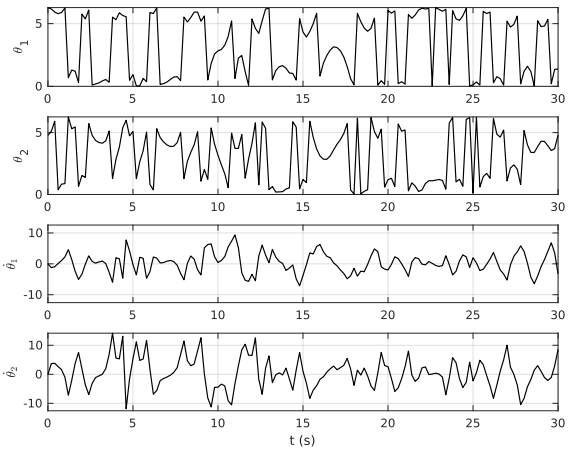

Figure 16: An unsuccessful trial run for the acrobot starting from a randomized initial condition. Angle values given in radians.

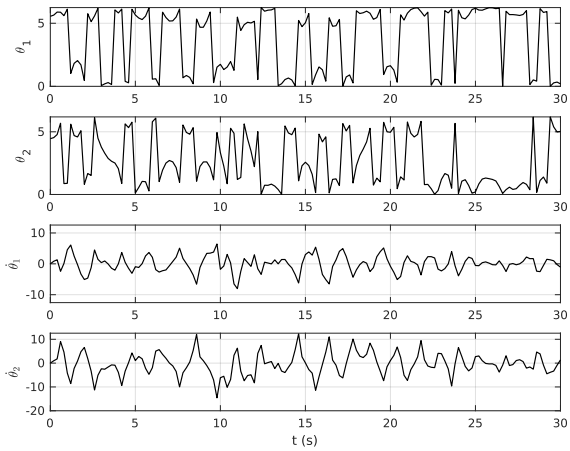

Figure 17: An unsuccessful trial run for the acrobot starting from a randomized initial condition. Angle values given in radians.

4 Discussion

The dynamic programming results for the inverted pendulum, given the relative simplicity of its dynamics, are easier to interpret in the context of physical intuition. However, despite this relative simplicity, the control policy is not able to perfectly control the inverted pendulum in a timely manner in all randomized cases. In some scenarios, there is evidence that this can perhaps be attributed to an underpowered torque saturation value. Despite this, an inspection of the visualized random trial results in Figure 8 suggests that an underpowered actuator cannot always be the reason, as with the unsuccessful trials found in the vicinity of the goal state.

A likely explanation is explored in [3], where it is noted that any discretization of the state space of a system governed by continuous dynamics must technically be considered as a stochastic process when simulated with value iteration. This entails approximately integrating Equation 5 over the set of possible alternative transition states given the input, all weighted by their relative probability from a probability distribution function. In [3], the authors show that introducing stochasticity into the value iteration algorithm allows for the derivation of a control policy that can achieve the goal state close to 100% of the time. Unfortunately, the increased accuracy comes at a huge computational cost, resulting in control policy generation run times on the order of multiple days, which simply is not feasible for the scope of this project.

The issue of stochasticity is only compounded for the acrobot, with its expanded state space and chaotic aspect. The results of this project show that having a relatively finely meshed state grid, while it can in principle achieve the desired goal state on a simulated system, cannot alone guarantee high performance under arbitrary initial conditions for highly nonlinear systems. That being said, the straight-forwardness of the value iteration algorithm is attractive for its general applicability, as has been demonstrated by this project's adaptation of the algorithm to fit two separate nonlinear systems with very little additional code (out of ≈ 500 lines of code written to implement this project from scratch, only ≈ 50 lines of additional code is used to accommodate two different dynamic systems, thanks to polymorphism). With the addition of probability integration terms, a tractable state space discretization, and a lot more expendable CPU time on a modern laptop, it is not unreasonable to assume that very high-percentage and versatile closed-loop control strategies can be derived for even highly nonlinear systems with dynamic programming.

All C++ and Matlab code used to produce these results, together with usage instructions, is available at

https://github.com/goromal/acrobot-dp.

References

[1] G. Boone. Minimum-time control of the acrobot. In *Proceedings of International Conference on Robotics and Automation*, volume 4, pages 3281–3287 vol.4, 1997.

[2] M. W. Spong. The swing up control problem for the acrobot. *IEEE Control Systems Magazine*, 15(1):49–55, 1995.

[3] R. Ueda and T. Arai. Dynamic programming for global control of the acrobot and its chaotic aspect. In *2008 IEEE International Conference on Robotics and Automation*, pages 2416–2422, 2008.

[4] R. Ueda, T. Arai, and K. Matsushita. Creation and compression of global control policy for swinging up control of the acrobot. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2557–2562, 2006.