Error Estimation for Autonomous Suspended Load Flight[†]

Molly Salman, Aleksandra Faust, and Lydia Tapia
Adaptive Motion Planning Research Group, Dept. of Computer Science,
University of New Mexico, Albuquerque, NM 87131
{msalman10}@austincollege.edu {afaust, tapia}@cs.unm.edu

In an world of emerging technical advances and continual dangerous and undesirable situations for humans, it seems only logical that the next technological evolution will come in the way of autonomous robotic operation. For an agent or robot to be autonomous it must think and learn on its own while dealing with changing and unfamiliar environments. This thinking and learning can be accomplished through reinforcement learning, of which there are two major classes; discrete reinforcement learning and continuous reinforcement learning. While both have their strengths and weaknesses, both provide a method for finding an action selection policy that will allow an agent to make decisions that will optimize its progression through a task. However since the problems are not completely known before hand, the action selection policy can contain errors. In this research we attempt to get a better understanding of the error associated with autonomous flight while minimizing the swing of a suspended load by examining a known function and assessing its error. The research also focuses on comparing the accuracy of discrete reinforcement learning to a novel continuous real-time decision making algorithm, Action Selection through Axial Projection (ASAP).

[†]Salman supported by the Computing Research Association CRA-W Distributed Research Experiences for Undergraduates.

1 Introduction

There are many situations and tasks that are not suitable for humans. These situations/tasks might be too dangerous, too complicated, or simply undesirable and human involvement could have tragic outcomes. Search and rescue missions, explosive deactivation, and operations in space are a few examples. It would be ideal if robots could preform these tasks and enter these situations in our stead. Even more ideal would be if the robots could be autonomous, meaning that they would preform without being controlled by a human. This would require a robot to be able to think and learn on its own. They would have to make decisions and act in unfamiliar and often changing environments.

One of the main concerns with autonomous agents is how movement can be addressed. Autonomous motion-based robot task completion can be thought of in regards to the robot's control, obstacle avoidance, and decision making aimed at completing the task at hand. Current research aims to address this task completion through a methodology that overcomes the challenges of unknown dynamics and high-dimensional state spaces to enable the agent to perform decision making to accomplish a task safely [6], [17]. To accomplish the robots control problem, an agent needs to understand the systems dynamics and any constraints that are imposed on the system. This is important for the safety of the system and its surroundings. Since the control problem is often intractable, the preferred method is to learn it from examples and with aid of simulators. A reinforcement learning agent learns policy with respect to some observable reward and is a good candidate for learning the control of the system.

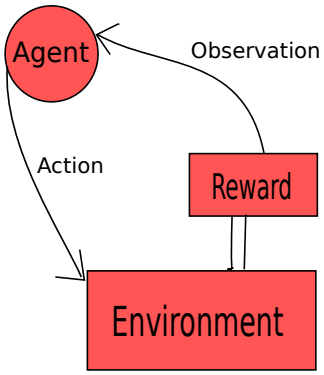


Figure 1: Agent reinforcement learning

When controlling the system with reinforcement learning, actions that the robot takes lead to a next state. It is through a series of actions and states that the robot eventually completes a task. The purpose of reinforcement learning is to find an action selection policy that maximizes accumulated reward over the lifetime of the task [11]. It does this by exploring the environment and constructing a value function, which is an estimated potential accumulated reward. There are two types of value functions. The first is a state value function, V, which associates states with an estimated potential accumulated reward. The second is an action value function, Q, that associates preforming an action with an estimated potential reward. Actions occur between states. V can be thought of as a measure of a state's quality, while Q can be considered as an action's quality [23].

When making an action selection policy we need to select an action that transitions the system to the highest possible valued state. This is called a greedy policy, $\pi(s) = argmax V(s')$, where $s' = D(s, a)$. D is the dynamics of the system, which is unknown. This formula also applies to Q where $\pi(s) = argmax Q(s, a)$. It is very hard to approximate Q using linear combinations of feature vectors. It is easier to find feature vectors for V. However, D is needed to evaluate the function, and as stated earlier, D is unknown.

Since D is not available there remain two options for creating an action selection policy, discretization and approximation. Discretization is a brute force tactic that breaks up states, and actions into small steps and calculates the action selection policy for each step. This can consume a lot of time and memory if the discretization is very fine. However, if the discretization is coarse, the resulting action policy has the potential to be rough which is not ideal for robots that need smooth movements to preform a task accurately [6]. This leaves us with approximation, or more specifically, action selection through axial projection (ASAP).

ASAP is a continuous reinforcement learning approach that continually samples the states in a system and determines a policy based on the average highest value.

1.1 Research Objective

The main objective for this research is to develop an understanding of the error associated with the autonomous flight of a quadrotor carrying a suspended load. Suspended loads are challenging because they swing and undulate based on the length of the tether and the motion of the quadrotor. The goal is for the quadrotor to fly to a specific location and learn how to minimize the swing of its suspended load. This is accomplished through a decision making policy for motion planning called action selection through axial projection (ASAP). When dealing with action selection policies the action selection step needs to find the highest valued action, $a^* = argmax_{a \in A} V(D(s,a))$, which does not scale well for high-dimensional action spaces. This is because action finding is an optimization problem on $Q(s,a) = V(D(s,a))$. Our goal is to find a^* , the true maximum. Functions V and D are not known, but their samples are. For a fixed state s, we could polynomially interpolate Q. At least $\frac{n+da}{n}$ samples are needed for $n - degree$ interpolation of a d_a -variate function.

To reduce the dimensionality of the optimization problem, we propose a divide and conquer approach. Instead of solving one multivariate optimization, we solve d_a univariate optimizations along the axes. We find a highest valued point on each axis, a_i . The composition of the axes action selections is the selection vector $a = [a_1..a_{d_a}]^T$. On each axis, our method finds the largest value isoline that the axis is tangent on. $a^* = [a_x^* a_y^*]^T$ is the true optimal action, while $a = [a_x a_y]^T$ is the action selected and the error is $\parallel a - a^* \parallel$. When the action components along the dimensions are independent, the error is zero, and the method finds true maximum.

The goal is to assess ASAP's method's quality and understand under what circumstances ASAP's method for action selection leads the system to a higher valued state, and to quantify the action selection error. To do that we will compare it to the optimal action selection and discrete action selection policies. We will measure the distance between selected actions, and the difference between resulting Q function values.

2 Related Work

When considering the task of learning for the quadrotor dealing with a suspended load, it is necessary to examine approximate value iteration[4], [7]. Approximate value iteration is used when examining continuous spaces which is the primary focus of this research.

Suspended load control has been studied extensively [8], [15], [16], and [20] for cargo equipped quadrotors. Palunko et al. successfully applied dynamic programming to solve swing-free trajectories for quadrotors[14], [16]. Dynamic programming requires that the dynamics of the system are known ahead of time, and is sensitive to the accuracy of the model, and to the start and goal states used. A reinforcement learning approach doesnt require a white box approach to the systems dynamics, and learning doesnt need to be repeated when the start state changes. Further, reinforcement learning is more suitable for compensating for the accumulated error resulting from model approximation. Lastly, while dynamic programming requires pre-calculating each trajectory, the approach presented here allows us to learn the problem once, and to generate any number of different trajectories with different starting positions using the same value function approximation. Palunko et al. [14] showed an example of manual obstacle avoidance of a quadrotor with suspended load by generating swing-free trajectories using DP. Bernard et al. [2] developed a controller for helicopters with suspended loads using two feedback loops. Hehn and DAndrea developed a controller for a quadrotor balancing an inverted pendulum [9]. Further examples of quadrotors interacting with the environment can be seen in aerial grasping [13, 18], ball throwing [19], and aerial robotic construction [24].

Swing-free trajectories have been studied outside of the UAV domain. They are important in industrial robotics with applications such as cranes in construction sites and for cargo loading in ports [1, 25]. Residual oscillation reduction is applicable to manufacturing environments where parts need to be transported in a limited space. Zameroski et al. [26] and Starr et al. [21] applied dynamic programming to reduce residual vibrations of a freely suspended payload.

Action selection plays a central role in reinforcement learning planning [22]. The latest trend is studying domains with continuous actions. Online optimistic sampling-based planners have been particularly popular

[3, 5, 12, 22]. Johnson and Hauser [10] developed a motion planning approach based on reachability analysis for trajectory tracking in acceleration-bound problems. Zhang et al. [27] proposed a sampling-based planner that finds intermediate goals for trajectory planning for systems with dynamical constraints.

3 Preliminaries

In this section we provide a description of ASAP continuous learning and discrete learning. We will also discuss the state spaces of the quadrotor.

When evaluating the error it was important to examine learning in discrete spaces against learning in continuous spaces. Discrete learning partitions the action space into a grid of a set size. At each step in the grid the agent uses reinforcement learning to determine the next step. The ASAP method for learning consists of selecting an action by finding the optimal action on each axis separately and then combining them together. Continuous learning is sometimes preferred to discrete learning because discrete learning can yield rough outputs if the discretization is coarse resulting in "bumpy" movements when smooth movements are what is desired. Also discrete learning can take up a lot of memory because of the shear number of steps [6].

State space refers to to information relevant for this problem, and is velocity and, position of vehicle's and load's centers of gravity. The action space is acceleration applied to the quadrotor's center of gravity. Acceleration has upper and lower limits of $\pm$ 3 m/s^2.

4 Methods

The following is a description of the steps and processes used to assess the error of the system. We start out small, looking simply at the distance between the optimal and ASAP actions, and later broaden our inquiry, examining the error when related to continuous action spaces versus discrete action spaces, to gain a more conclusive knowledge of the potential error.

4.1 Quadratic Equation

We begin by examining a Q function as quadratic function with changing parameters, and determine for what parameter values the ASAP method is successful. For the purpose of this study, we restrict our state and action space to two-dimensions. Thus we represent Q as

$$Q = a(x-1)^2 + b(y-2)^2 + cxy \tag{1}$$

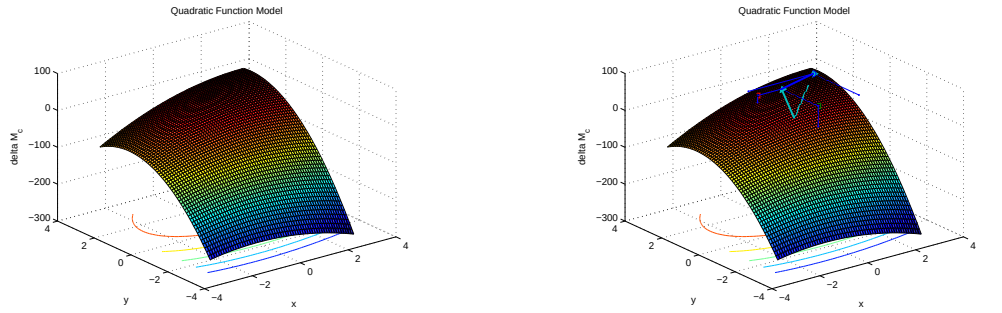
where $-3 < x < 3$, $-3 < y < 3$ fit the acceleration bounds. The area, given by x and y are actions that the quadrotor can take, known as the action space. The parameters that dictate the shape of the Q function are a, b, and c. Taking horizontal cross sections of the area and projecting them downward displays the isolines of the function. a, b, and c describe these isolines, where a is the semi-major, b is the semi-minor, and c the rotation of the isolines about the optimal action. The area and the isolines can be seen in figure Figure 2(a).

When looking at the parameters and the isoline topology they produced it became apparent that the isolines were modeling the three types of conic sections. Conic sections refer to the way in which a right circular conical surface is intersected by a plane. The three types are ellipses, parabolas, and hyperbolas. There is a fourth type of conic section, the circle, but it is a type of ellipse so it will not be discussed in this paper. Noting this relationship was valuable because it showed that we were on the right track of reducing our Q function to three parameters. The regular form of a quadratic function $Ax^2+Bxy+Cy^2+Dx+Ey+F = 0$ consists of six parameters A, B, C, D, E, and F. However, the shape of a conic section is dictated by only A, B, and C, or in our case a, b, and c. If $b^2-4ac < 0$ then the equation represents an ellipse. If $b^2-4ac = 0$ the equation is a parabola, and if $b^2-4ac > 0$ it is a hyperbola.

Knowing that our Q function is completely operable with 3 parameters also allows us to quantify the rotational value c, which is done in Section 4.5.

When looking at our Q function we concern ourselves with a few specific values. The optimal action, $(x^*, y^*) = argmax_{\{(x,y)|-3<x,y<3\}}Q(x,y)$ is a nonlinear optimization problem. The action value of the optimal action can be defined as $Z^* = Q(x^*, y^*)$. We are also concerned with the ASAP action (x_c, y_c) and

its action value. As well as the discretized action (x_d, y_d), and the associated action value. Action values are discussed in Section 4.4.



(a) Isoline Example (b) Distance between the optimal action and the ASAP action

Figure 2: (a) The isolines of the function are visible on the "floor" of the graph. (b) Shows the projections to the optimal and ASAP actions

4.2 Calculating Error Distance

The process of understanding the error began by comparing the distance between the optimal action and the ASAP action. This was accomplished using Euclidean distance. The distance between the two points is the value of the error, $\triangle M_c$, as seen in Figure 2(b).

$$\triangle M_c = \| (x_c, y_c) - (x^*, y^*) \| \tag{2}$$

This error changed in relation to changing the parameters of the function. A 3D matrix was created that contained all the distances for the different permutations of a, b, and c from -50 to 50. At this early stage in the research c was simply a value and had yet to be quantified with an actual degree of rotation. The calculation of rotation is discussed in Section 4.5.

4.3 Holding Parameters Fixed

The 3D matrix of distances proved to be too much data to analyze by graph (because we cannot graph 4 dimensions), let alone visualized. For this reason a was held fixed while c and b were changed. This was possible because b is a value proportional to a since they are modeling the dimensions of the isolines. Another way to write b would be $r(a)$ where r is some stretch value on a. By this logic we can redefine our Q function to be

$$Q = a(x-1)^2 + r(a)(y-2)^2 + cxy \tag{3}$$

The main reason a was chosen as the fixed value was because we hypothesized that rotation, c, would have the biggest impact on error for the system.

Using this method of holding a fixed, four cases where examined. The first case is when a is positive and $r(a)$ is positive, (+, +). The second and third cases are when a is positive and $r(a)$ is negative, (+, -), and visa versa, when a is negative and $r(a)$ is positive, (-, +). The fourth case is when both parameters are negative, (-, -). Figure 3(b) shows an example of this.

4.4 Action Value Function Difference

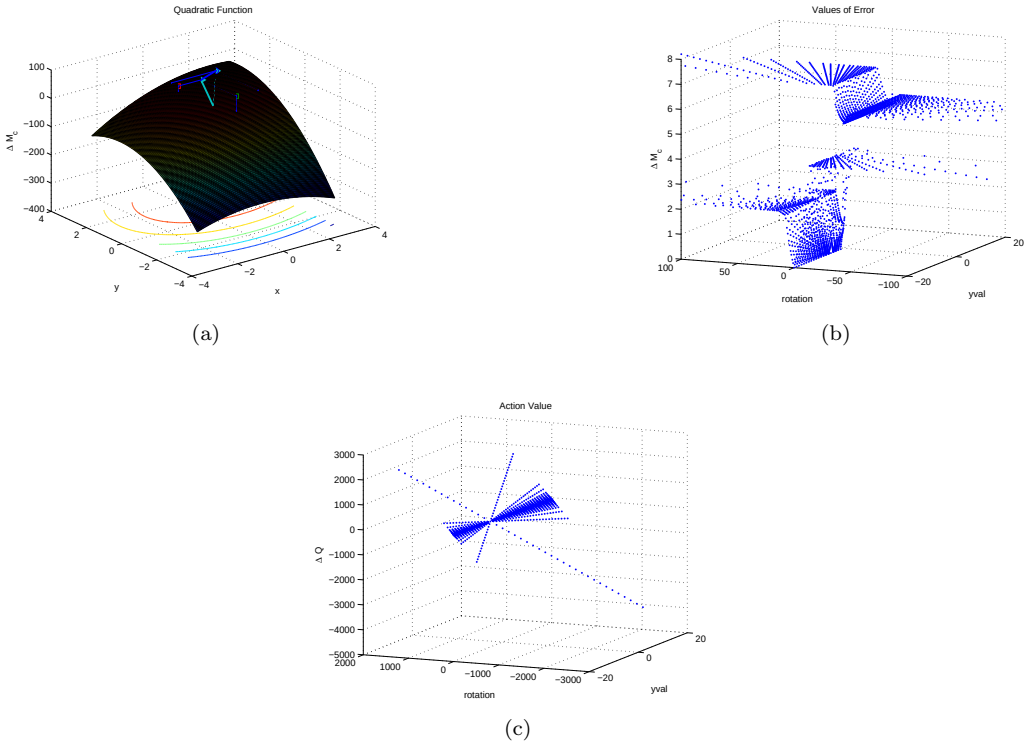
Another error to consider is if the quadrotor is actually making progress. This error is known as the action value difference, or $\triangle Q$. It can be assessed by the Q function as well. $\triangle Q$ is found by

$$\triangle Q_c = Q(x_c, y_c) - Q(0, 0) \tag{4}$$

for ASAP continuous action values and

$$\triangle Q_d = Q(x_d, y_d) - Q(0, 0) \tag{5}$$

for discrete action values. Through reinforcement learning the quadrotor is supposed to pick "more" optimal paths with each iteration. If the ASAP action or the discretized action has a greater Q value than the base action ($Q(0,0)$) then the quadrotor is making progress. If the Q value for the optimal is less than the base action then the quadrotor is not learning and making "bad" decisions. See Figure 3(c).



(a) (b) (c)

Figure 3: Using Equation 6 where $a = -5$ (a) The area of the function for which we are calculating the error; (b) Depicts the error for different values of $r(a)$ and $(a - r(a))tan2\theta$; (c) The action value function differences

4.5 Quantifying Rotation

Up until this point the value we had been using for c was simply that, a value. It did not correspond to an actual degree or radian of rotation. However, associating this value to an actual degree of rotation proved pretty simple to do. The quadratic Q function we have been using can be reduced to the form $Ax^2 + Bxy + Cy^2 = 0$ as stated in Section 4.1. Because of this we are able to use the formula $tan2\theta = \frac{B}{A-C}$ to find the actual c, where B is c, A is a, and C is $r(a)$. All that was needed was the θ value. A range for θ was selected from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$. For each value of $r(a)$ we iterated through the range of θ to calculate the rotation. Using radians of rotation the Q value was calculated and the error plotted as before and graph with respect to the actual rotational value.

Using this new understanding of the rotational value we can once again redefine our Q function.

$$Q = a(x-1)^2 + r(a)(y-2)^2 + (a - r(a))tan2\theta xy \tag{6}$$

4.6 Discrete vs. Continuous Action Spaces

Up until this point, the calculations had been made for continuous ASAP action spaces. It was important to look at the difference between the error in discrete spaces compared to that in continuous space. To do this

the step size was decreased significantly. The discrete maximum value was found by finding the maximum point in the discretized mesh. What is difficult is that sometimes the optimal action lays outside of the range of our discretized action space given different values of the parameters. If this occurred then that outside value was "snapped" back into the action space. The continuous maximum lay outside of the action space if a was greater than or equal to zero.

The ASAP action was compared to the optimal action along with comparing the discrete action to the optimal action.

The $\triangle Q$, the action value, was also calculated for the discrete space, $\triangle Q_d$, and the continuous ASAP space, $\triangle Q_c$, in order to compare the learning value for each type of reinforcement learning. See Section 4.4 and Figure 4.

5 Results

This section will discuss the results of this research and display the graphs associated with the results. The most telling data was collected when comparing discretized reinforcement learning to continuous ASAP reinforcement learning, as expected. Here, the most valuable data came when calculating the learning action value function, Q, and when comparing the continuous action to the optimal action, along with the discrete action to the optimal action.

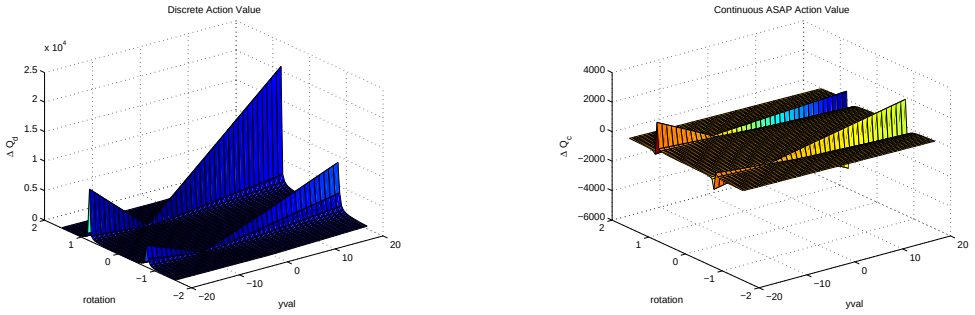
All calculations and graphs were completed using Matlab. To calculate optimal action

$$(x^*, y^*) = argmax_{\{(x,y)|-3<x,y<3\}} Q(x, y) \tag{7}$$

we used a built-in Matlab call 'fmincon' from the Optimization toolbox. The majority of the data and results found in this paper are shown using graphs. The graphs allow us to quickly see a relationship between hundreds of data points without having to look at charts of numbers.

5.1 Discrete Learning

As previously discussed, discrete learning is accomplished by discretizing the actions into small steps then evaluating the Q. Discretization is known to be effective in reinforcement learning and was confirmed by this research. It was shown that the $\triangle Q$ for a fine grain discretized space is always positive, meaning that the agent is always making progress when using a discrete reinforcement algorithm. This can be seen in Figure 4.



(a) Discrete Learning

(b) Continuous ASAP Learning

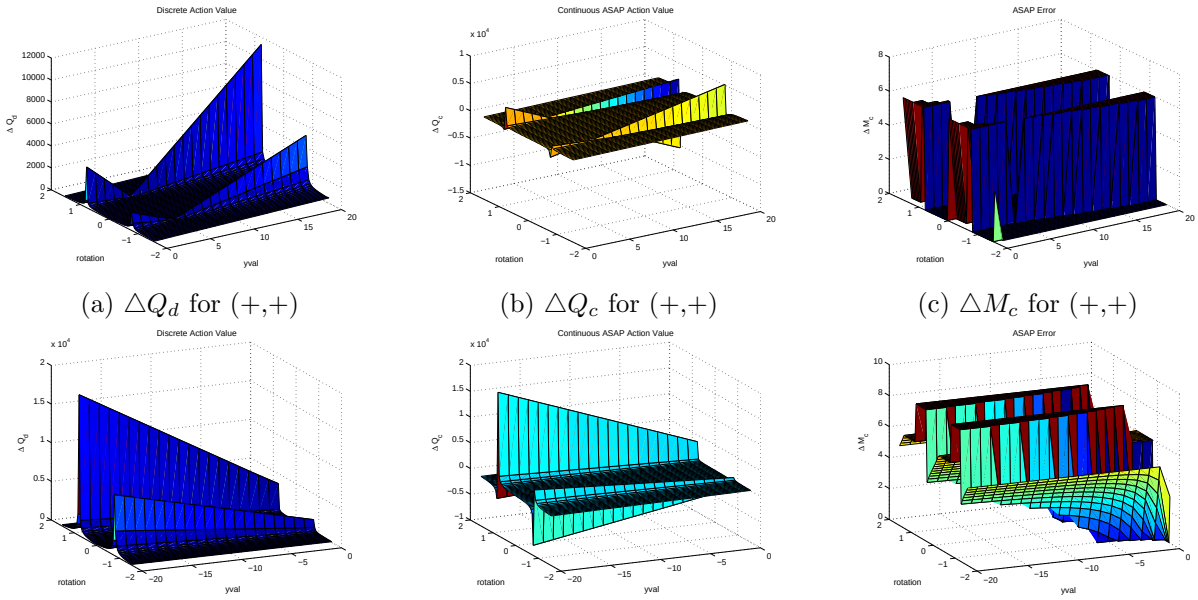
Figure 4: Using Equation 6 where $a = -8$ (a) positive action values for discrete learning. (b) ASAP action values

5.2 Continuous Learning

The primary focus of this research was to test and potentially justify ASAP. Here we are able to make some claims about it.

From the data it was apparent that there were four cases of parameters that effected the error in regards to continuous learning as mentioned in Section 4.3. The first case is when a and $r(a)$ are both positive, (+, +). The results for this case are shown in Figure 5 (a-c). They show that when $a = r(a)$ ASAP is selecting optimal actions. It was also noted that when rotation is zero ASAP is selecting optimal actions. This second observation was confirmed by the results of the second case (+, -) shown by Figure 5 (d-f). From these two cases it was observed that peaks and valleys for both discrete and ASAP action values occurred between 28° and 58° of rotation and $-28°$ and $-58°$ of rotation. This observation held true for all 4 cases of parameters. These results suggest that as the ratio between $\frac{r(a)}{a}$ increases and when the actions are coupled the most variation occurs in the vicinity of 28° and 58° of rotation and $-28°$ and $-58°$ of rotation, and that the action error increases proportionally to the $\frac{r(a)}{a}$ ratio.

The fourth case is when both parameters are negative, (-, -). The majority of the data comes from this fourth case. This is because two negative parameters result in the shape of the value function for a quadrotor with a load.



(a) $\triangle Q_d$ for (+,+) (b) $\triangle Q_c$ for (+,+) (c) $\triangle M_c$ for (+,+)

(d) $\triangle Q_d$ for (+,-) (e) $\triangle Q_c$ for (+,-) (f) $\triangle M_c$ for (+,-)

Figure 5: Using Equation 6 where $a = 5$ (a)(d) discretized action values. (b)(e) ASAP action values. (c)(e) ASAP error, $\triangle M_c$, values.

When examining continuous learning for the case of (-, -) it was possible to see that for some values of $r(a)$ and $(a - r(a))tan2\theta$ the action value, $\triangle Q_c$, was negative. This means that for some actions the ASAP function was resulting in the agent is taking a step backwards. See Figure 6 (a). We looked at these areas of negative learning and noted that they fell within an area that occurs when

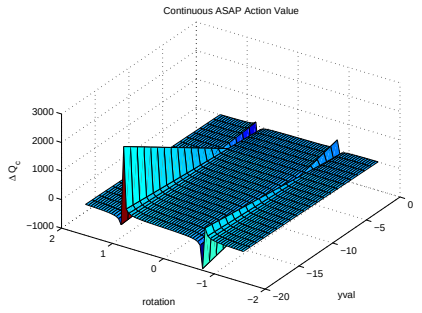
$$\frac{-r(a) - \sqrt{r(a)^2 - 4(a)((a - r(a))tan2\theta)}}{2(a)} < 0 \tag{8}$$

which is the equation for finding solutions to a quadratic formula $f(x) = ax^2 + r(a)x + (a - r(a))tan\theta$. By this we saw that when

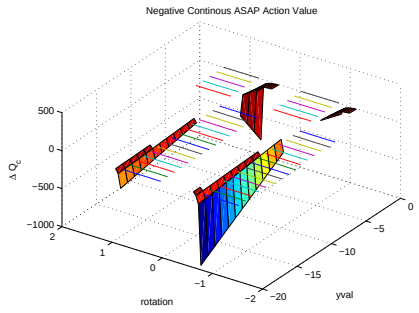
$$\frac{-r(a) - \sqrt{r(a)^2 - 4(a)((a - r(a))tan2\theta)}}{2(a)} > 0 \tag{9}$$

the continuous algorithm was positively learning. When the parameters resulted in this positive learning case, we calculated the continuous error using those same parameters and compared it onto the continuous negative learning action value graph (Figure 6 (b)). It was seen that the area of continuous error for positive

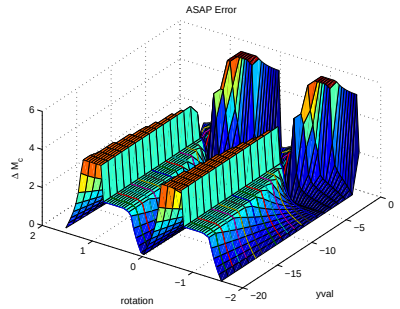
learning action value never intersected with the negative learning action value as long as the value of a was negative. From there we projected this area on to continuous error graph (Figure 6 (c)) and saw that it lay over the section of the graph where error was not at the maximum. This proves to us that the highest error when comparing the ASAP action to the optimal action occurred when the function has a negative ASAP continuous action value when parameters a and $r(a)$ were both negative.



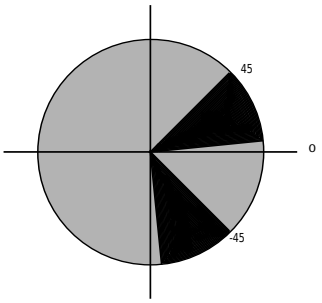
(a) Continuous Action Value (-,-)



(b) Negative Cont. Action Value (-,-)



(c) Continuous Error (-,-)



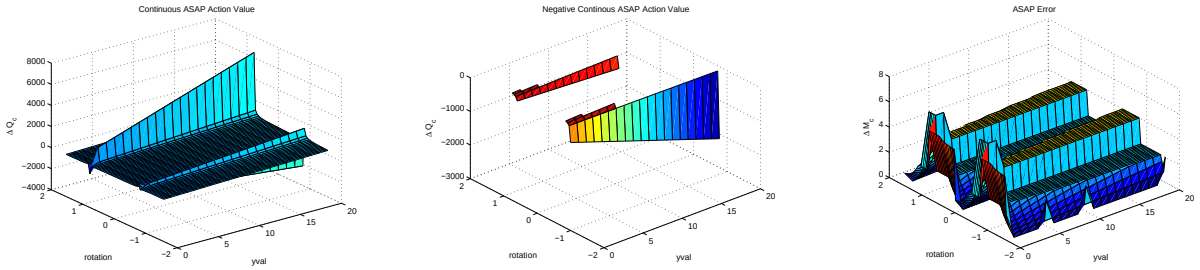
(d) Rotations of High Error (-,-)

Figure 6: Using Equation 6 where a = -5 (a) The action values for continuous ASAP learning. Here $\triangle Q_c$ is never equal to zero; (b) The areas where the action value is negative (less than zero) and the areas, marked by straight colored lines, for which Equation (9) holds true. (c) The areas, also marked by colored lines projected onto the action space, of negative action value correspond to the areas of high error. (d) Areas in black show for what values of $(a - r(a))tan2\theta$ that $\triangle M_c$ is highest.

Another observation worth mentioning is that the areas of highest error when comparing the ASAP action to the optimal action for the case of (-, -) occurred roughly between 5° and 45° of rotation and $-86°$ and $-45°$ of rotation. See Figure 6 (d). It should also be noted that error was zero when rotation was zero for the (-, -) case. This means that some coupling between actions is allowed.

In regards to the (-, +) case, where a is a negative value and $r(a)$ is a positive one the findings were not as conclusive. Equations 8 and 9 proved not to model the same relationship as seen in the (-, -) instance for this case. Other equations were tested such as $r(a) > \sqrt{4 * a * (a - r(a))tan2\theta}$, which tells us how many roots of the quadratic equation there are. On the first pass this equation seemed to work because it lay over the regions where continuous learning action was negative. However, upon further inspection it was seen that the area this equation was modeling was limited and did not encompass all of the areas on negative learning action when the value of a was changed. Further more, the areas of negative learning action did not correspond to the areas of high or low error for continuous error.

Even though we were only able to present conclusive data and observations when it came to the (-, -) case and a few minor observations about the (-, +) case, these results provide insight into the potential of the ASAP continuous learning function. This will contribute to future work with ASAP and continuous reinforcement learning for autonomous agents as a whole.



(a) $\triangle Q_d$ for (-,+) (b) Negative $\triangle Q_c$ for (-,+) (c) $\triangle M_c$ for (-,+)

Figure 7: Using Equation 6 where a = -5 (a) The action values for continuous ASAP learning. (b) The areas where the action value is negative (c) Depicts error for ASAP learning.

6 Conclusion

Addressing the problem of autonomous quadrotor swing-free flight in order to assist in dangerous and undesirable tasks is a rapidly developing area of research. A novel continuous learning approach to this problem, action selections through axial projection (ASAP) was asserted. We propose a manner in which to estimate the error of this algorithm. Error was assessed by comparing the distance between the optimal action and the continuous action. We apply this error estimation to both ASAP and discrete learning action spaces to determine if ASAP is the best option. It was shown that correlations between negative learning and high error could be made for ASAP learning for the case when the parameters of the Q function were negative. The results of this research contribute to the study of ASAP continuous learning and reinforcement learning for autonomous agents.

7 Acknowledgments

We would like the thank the Adaptive Motion Planning Research Group at University of New Mexico for all the support and help over the course of the summer.

References

[1] M. Agostini, G. Parker, H. Schaub, K. Groom, and I. Robinett, R.D. Generating swing-suppressed maneuvers for crane systems with rate saturation. *Control Systems Technology, IEEE Transactions on*, 11(4):471 – 481, July 2003.

[2] M. Bernard and K. Kondak. Generic slung load transportation system using small size helicopters. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3258 –3264, may 2009.

[3] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári. X-armed bandits. *J. Mach. Learn. Res.*, 12:1655–1695, July 2011.

[4] L. Buşoniu, R. Babuška, B. De Schutter, and D. Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, Boca Raton, Florida, 2010.

[5] L. Busoniu, A. Daniels, R. Munos, and R. Babuska. Optimistic planning for continuous-action deterministic systems. In *Accepted at The 2013 Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, April 2013.

[6] K. Doya. Reinforcement learning in continuous time and space. *Neural Computation*, pages 219–245, 2000.

[7] D. Ernst, M. Glavic, P. Geurts, and L. Wehenkel. Approximate value iteration in the reinforcement learning context. application to electrical power system control. *International Journal of Emerging Electric Power Systems*, 3(1):1066.1–1066.37, 2005.

[8] A. Faust, I. Palunko, P. Cruz, R. Feirro, and L. Tapia. Learning swing-free trajectories for uavs with a suspended load in obstacle-free envrionments. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2013.

[9] M. Hehn and R. D'Andrea. A flying inverted pendulum. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 763–770. IEEE, 2011.

[10] J. Johnson and K. Hauser. Optimal acceleration-bounded trajectory planning in dynamic environments along a specified path. In *ICRA*, pages 2035–2041, 2012.

[11] M. L. L. Kaelbling, Leslie Pack and A. W. Moore. Reinforcement learning: A survery. 1996.

[12] C. Mansley, A. Weinstein, and M. Littman. Sample-based planning for continuous action markov decision processes. In *Proceedings of International Conference on Automated Planning and Scheduling*, 2011.

[13] D. Mellinger, Q. Lindsey, M. Shomin, and V. Kumar. Design, modeling, estimation and control for aerial grasping and manipulation. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2668 –2673, sept. 2011.

[14] I. Palunko, P. Cruz, and R. Fierro. Agile load transportation : Safe and efficient load manipulation with aerial robots. *Robotics Automation Magazine, IEEE*, 19(3):69 –79, sept. 2012.

[15] I. Palunko, A. Faust, P. Cruz, L. Tapia, and R. Feirro. Suspended load manipulation using aerial robots and least square policy iteration. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4881–4886, 2013.

[16] I. Palunko, R. Fierro, and P. Cruz. Trajectory generation for swing-free maneuvers of a quadrotor with suspended payload: A dynamic programming approach. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2691 –2697, May 2012.

[17] N. Y. A. Pieter Abbeel. Exploration and apprenticeship learning in reinforcement learning. 2005.

[18] P. Pounds, D. Bersak, and A. Dollar. Grasping from the air: Hovering capture and load stability. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2491 –2498, may 2011.

[19] R. Ritz, M. Muller, M. Hehn, and R. D'Andrea. Cooperative quadrocopter ball throwing and catching. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pages 4972 –4978, oct. 2012.

[20] K. Sreenath, N. Michael, and V. Kumar. Trajectory generation and control of a quadrotor with a cable-suspended load – a differentially-flat hybrid system. In *IEEE International Conference on Robotics and Automation (ICRA)*, to appear, 2013.

[21] G. Starr, J. Wood, and R. Lumia. Rapid transport of suspended payloads. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1394 – 1399, april 2005.

[22] T. J. Walsh, S. Goschin, and M. L. Littman. Integrating sample-based planning and model-based reinforcement learning. In M. Fox and D. Poole, editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. AAAI Press, 2010.

[23] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

[24] J. Willmann, F. Augugliaro, T. Cadalbert, R. D'Andrea, F. Gramazio, and M. Kohler. Aerial robotic construction towards a new field of architectural research. *International Journal of Architectural Computing*, 10(3):439 – 460, 2012.

[25] J. Yang and S. Shen. Novel approach for adaptive tracking control of a 3-d overhead crane system. *Journal of Intelligent and Robotic Systems*, 62:59–80, 2011. 10.1007/s10846-010-9440-9.

[26] D. Zameroski, G. Starr, J. Wood, and R. Lumia. Rapid swing-free transport of nonlinear payloads using dynamic programming. *ASME Journal of Dynamic Systems, Measurement, and Control*, 130(4), July 2008.

[27] Y. Zhang, J. Luo, and K. Hauser. Sampling-based motion planning with dynamic intermediate state objectives: Application to throwing. In *ICRA*, pages 2551–2556, 2012.