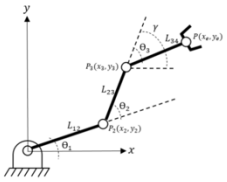
High Dimensionality, Value Iteration, and Barycentric Interpolation in Robotic Control
by Roman Aguilera

Motivation:

The purpose of this project was to explore methods of developing controllers for a high dimensional system. If we would like to perform robust robot control in the real world, with guarantees on performance, the states that the robot visits needs to be a computationally tractable set. In other words, we need to have a bound on the number of possible states that the system can visit. And once we have those bounds, we can determine meaningful statistics within that bounded state space, and thus create controllers with guarantees on performance based on those statistics. If we do not create bounds on the number of states that a controlled system can visit, the time to compute meaningful statistics would increase exponentially, and we would exceed spacetime computational affordances quickly.

1A) 

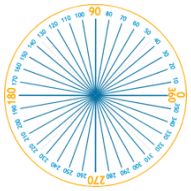
1B) 

Figure 1:
The image on the left (1A) is a 3-link robot arm. The image on the right (1B) shows the angular position quantized by 36 levels. The state space of the arm would be 6 dimensions (3 angular positions, 3 angular velocities). If each dimension has 36 quantization levels, the total number of states needed to create a mesh for a 3 link arm would be (36^6) = 2.1 billion states. If the value of each number all 6 dimensions is a number of type double, then a 3-link arm simulation based on this would need at least 104 GigaBytes to store the mesh.
([36 quantization levels per dimension]^[6 dimensional state space]) * [6 numbers per state] * [8 Bytes per number of type double] = 104 GigaBytes .
(It is worth noting that when attempting to design a controller for a system, the choice of spacing between states in a mesh is something to give careful consideration towards, and the choice of spacing would be different depending on the system to be controlled. But for the purposes of illustration, this example choice of discretization is mainly to show a low bound and for illustrative purposes. It is easy to visualize angular position to be divided in levels separated by 10 degrees. Moreover, velocity would need more than 36 quantization levels because it needs a finer discretization and velocity values do not wrap around like position, but again, we chose 36 quantization levels for each dimension for illustrative purposes and as a low bound.)

For example, if we chose a 3-link arm as a system to control, we would need 6 numbers to represent the state of the robot (3 angular positions, and 3 angular velocities). If we chose a discretization for each dimension to have 36 levels of quantization, we would need 36^6 total states to represent the resulting state space mesh (See Figure 1).
With this discretization, we would need (36^6) total states for a mesh, which is more than 2.1 billion states. And because each state would need 6 values (one value for each dimension in the state), we would need (36^6)*6 numbers to represent this mesh. If each number is of type double (8 Bytes), this would mean that we would need (36^6)*6*8=104.485 *10^9 Bytes. So as a low bound, we would need at least 104 GigaBytes in order to simply represent the mesh of states in this predetermined mesh.

```
Error using repmat
Requested 1626x1626x1626 (32.0GB) array exceeds maximum array size
preference (32.0GB). This might cause MATLAB to become unresponsive.
Error in ndgrid (line 72)
        varargout{i} = repmat(x,s);
Error in curse_of_dimensionality (line 24)
    [X,Y,Z] = ndgrid(0:res:1, 0:res:1, 0:res:1);
Related documentation
```

Figure 2: An example case where an array exceeds MATLAB's memory limitations of 32 GB. This does not include memory and time needed to do calculations in order to find a controller. To further illustrate an understanding of computational requirements, an array of 32 GB would take about 37 seconds to generate, a search for all elements in the array greater than 0 would take about 11 seconds, adding 1 to all elements in the array would take about 16 seconds in my computer (the same addition operation would crash the computer in the Dynamic Robotics Lab).

So we can see that this low-bound choice of discretization for a simple 3-link arm already exceeds the 32GB array allowance capacity for MATLAB in my computer(see Figure 2). The same memory limitation holds for the higher-performing computers we have at the Dynamic Robotics Lab. However, it is worth noting that array memory allowances in MATLAB can be reconfigured to push this limitation. However, if we chose to discretize a higher dimensional system, we would need to think more carefully about which states we actually care to mesh.

Once we determine the states we will care to include in our mesh (i.e. the countable set of states), and a transition function that encodes knowledge of how a state changes with respect to an external input, we can find a controller that gets us to a desired goal state. One such method of finding these controllers, and the method that was used in this project, is the Value Iteration Algorithm created by Sutton and Barto (see Figure 3).

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
  $\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$

Figure 3: Value Iteration Algorithm by Sutton and Barto.

I used this algorithm to find control policies for 2 systems, the first being Gridworld and the second being the Double Integrator. Gridworld was intended to be a preliminary introduction to Value Iteration and Barycentric Interpolation. The Double Integrator was intended to build upon Gridworld, and serve as an intermediary step towards more complex and higher dimensional systems. The code for these algorithms were written in MATLAB.

High Dimensionality, Value Iteration, and Barycentric Interpolation in Robotic Control
by Roman Aguilera

Gridworld:

For Gridworld, the problem is to find a control policy that determines what action would be taken, from any gridpoint in an M x N grid, to get to a goal state. In my implementation, the choice of possible actions were up, down, left, right, and diagonal movements. For the Value Iteration Algorithm, the initial value function was 1 at the goal state and -1000 for all other states. The value and policy functions converged to the image shown in Figure 4.

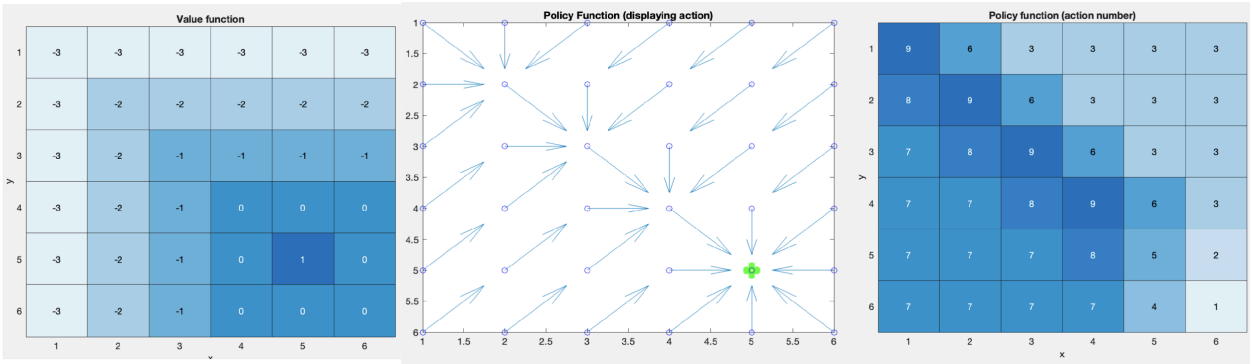


Figure 4: Final Value Function, Policy Function (with the action taken), and Policy Function (with action id number). The available integer actions for the control policy were {[1,1], [1,0], [1,-1], [0,1], [0,0], [0,-1], [-1,1], [-1,0], [-1,-1]}.

A second version of Gridworld was also posed, and in that version, I multiplied the value of the actions by 0.5 (e.g. an action of [-1, 1] would become [-0.5, 0.5]). The Value Function would converge to that of Figure 5.

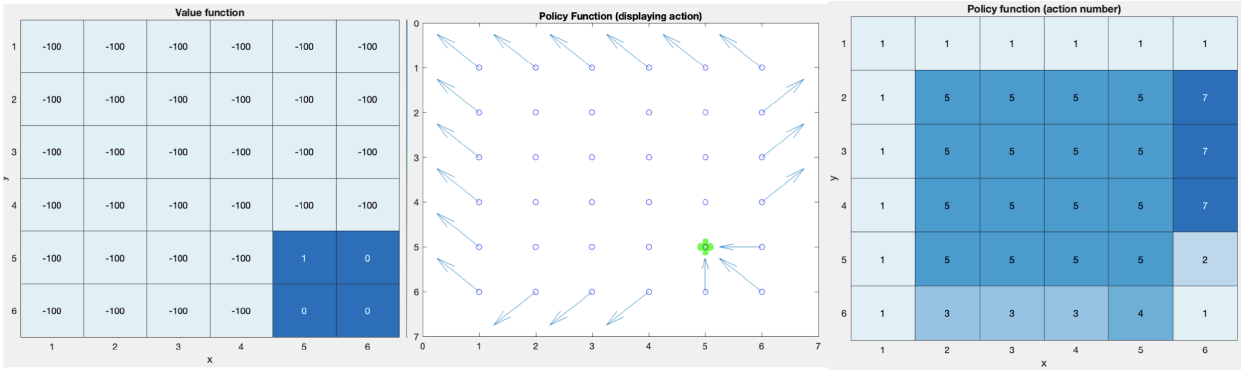


Figure 5: Final Value Function and Policy Function for Gridworld with Non-Integer Actions (Using Single Nearest Neighbor's Value to update the Value of the States). After an action is taken, we take the new state, find the nearest neighboring state, and use the nearest neighboring state's value to update the value of the state before the action was taken. The result is that the information from the goal state is only propagated to 3 neighboring states, the other states never update their values.

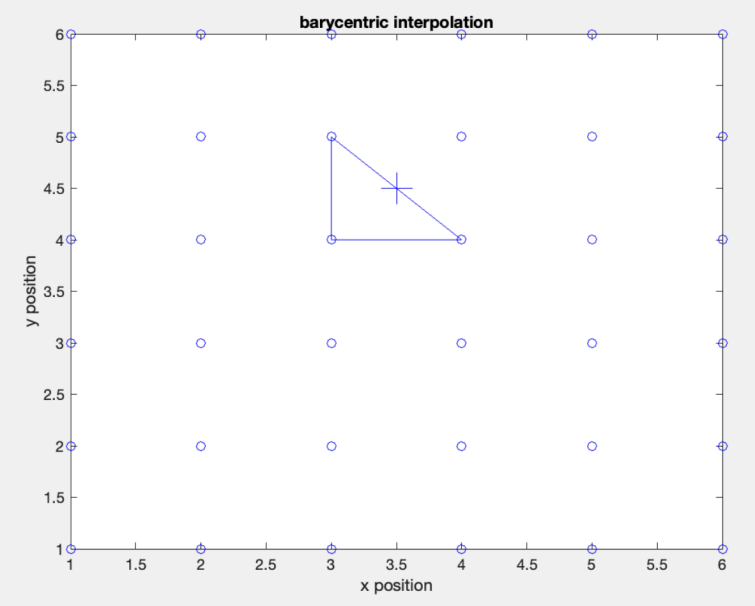
I noticed that if the multiplication factor was equal to or less than 0.5, then information would not propagate from the goal states to all other states. However, information would propagate if the multiplication factor was greater than 0.5. The reason this was the case is because when taking an action, the new state would not always end up in a spot that lands exactly on a state that is part of the predetermined mesh, so the value of the state before an action is taken is determined by taking an action, looking at the new state that you end up at, and using the value of the nearest neighboring state with respect to the new state. Only some states would have the possibility of going to a new state whose nearest neighbor has a value that is not -1000, and thus only those states

would have a value that changes from -1000 to a value that is closer to the value of the goal state.

Barycentric Interpolation:

We can resolve this issue of information non-propagation by implementing Barycentric Interpolation. The main idea is to find 3 nearby states in the mesh that can encapsulate the new state that is reached after a state transition (i.e. after an action is taken) (See Figure 6). The value of the new state is determined by weighing the values of encapsulating states, with all weights summing up to 1. First the nearby states are determined, then secondly, we can find the weights by formulating a linear inverse problem and using the MATLAB backslash operator to solve for the weights (See Figure 7).

6A)



6B)

$$\begin{pmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix} = \begin{pmatrix} 1 \\ x \\ y \end{pmatrix}$$

Figure 6: Barycentric interpolation. Top image (6A) shows new state [x,y] after a state transition (indicated by the cross + sign). The 3 nearest points on the mesh {[x1, y1], [x2,y2], [x3,y3]} are found such that a triangle created by the points encapsulates the new state. The image on the bottom (6B) shows the search for the weights λ=[λ1; λ2; λ3] formulated as a linear inverse problem.
The new state is a weighted sum of the 3 neighboring states
[x, y] = λ1*[x1, y1] + λ2*[x2,y2] + λ3*[x3,y3].

The value of the new state V([x,y]) is the weighted sum of the values of the 3 neighboring states, that is
V([x,y]) = λ1*[V([x1,y1]) + λ2*[V([x2,y2]) + λ3*[V([x3,y3]).

The linear inverse problem is Aλ=b, with
A = [[1,1,1]; [x1,x2,x3]; [y1,y2,y3]],
λ=[λ1; λ2; λ3],
b=[1;x;y]

High Dimensionality, Value Iteration, and Barycentric Interpolation in Robotic Control
by Roman Aguilera

We use the MATLAB backslash operation as follows:
λ = A\b

After implementing Barycentric interpolation, we see that the value function converges to one where information is propagated (See Figure 7).

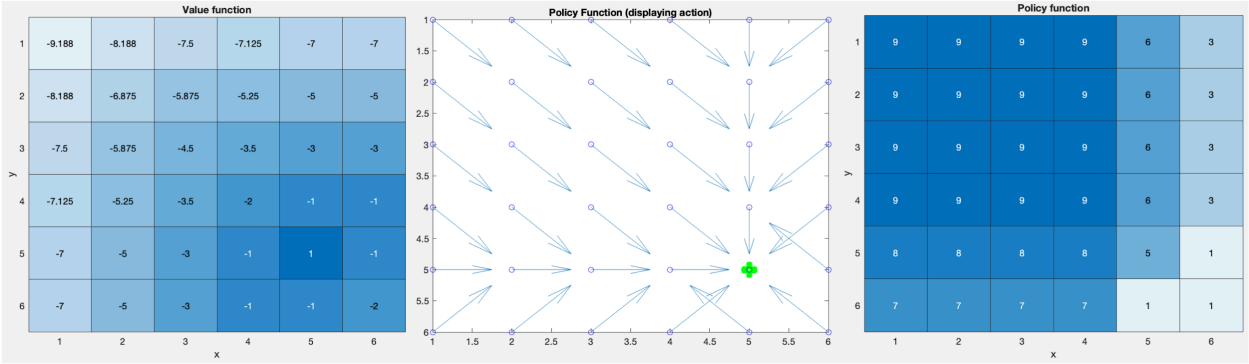
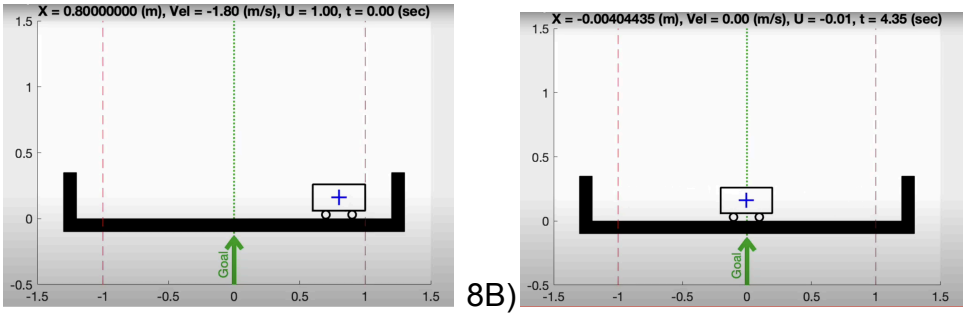


Figure 7: Final Value Function and Policy Function for Gridworld with Non-Integer Actions, and Using Barycentric Interpolation w/ Nearest Neighbors to interpolate value of the new state. The issue of information non-propagation is overcome, and all the states values are affected by the value of the goal state.

Double Integrator:

For the Double Integrator, the problem is to get a cart that can move along one axis from a given initial position and velocity, to reach a goal state of zero position and zero velocity. For this problem, we have one control input dimension, which is an input force that allows the cart to accelerate in either the positive or negative direction (See Figure 8).

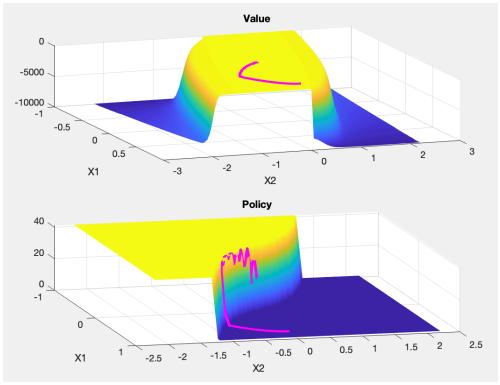


8A) 8B)

Figure 8: An image of the Double Integrator problem. The image on the left (8A) is the cart at an initial condition of 0.8 meters for position and -1.8 meters-per-second for velocity. The image on the right (8B) is the cart at the goal state of zero position and zero velocity.

One thing to note about the performance of the double integrator was that trajectories would look smooth until the state would approach the [0 position, 0 velocity] goal state (See Figure 9). Secondly, the optimal policy for the continuous time formulation for the double integrator would be to fully accelerate for half the time, and then fully decelerate for half of the time. The reason behind this discrepancy is because the double integrator problem essentially becomes a different problem depending on

how states, actions, and time are discretized. A different problem results in a different optimal solution.

9A) 

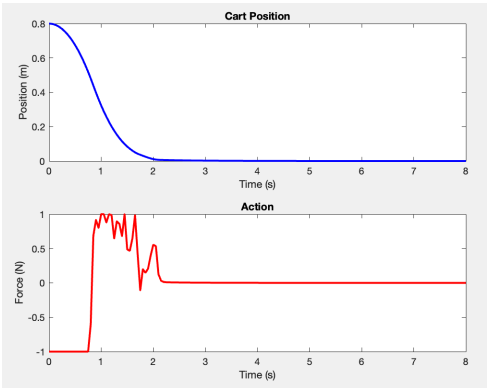
9B) 

Figure 9: The image on the left (9A) shows the value and policy functions, as well as a trajectory along those functions. The image on the right shows the same trajectory. As the cart a pproaches the zro position and zero velocity goal state, the trajectory if the input action is not smooth.

Code Optimizations:

For the Double Integrator, we also used barycentric interpolation and spent time figuring out how to make efficient use of MATLAB's matrix operations to create a faster alternative of the value iteration algorithm which replaced the use of a for loop that iterated over all states and actions. We achieved this by precomputing and storing the nearest neighbors for all states after a specific action was taken, as well as their associated Barycentric weights after an action was taken. Once those weights and nearest neighbors were calculated.

The value updates would be done in 3 steps. The first step is by doing an elementwise multiplication between (i) the current value function indexed at the state IDs of the nearest neighbors with (ii) the associated barycentric weights. The second step is, a using MATLAB's sum() function to do a summation along the dimension that corresponds to the barycentric weights, such that the Value at the current state and action would be

Vtemp(s, a) =

λ1(s,a)*Vtemp(snew1(s,a))+ λ2(s,a)*Vtemp(snew2(s,a))+λ3(s,a)*Vtemp(snew3(s,a)) .

The third step would be to set

Vtemp equal to Vtemp=r_onestep+alpha*Vtemp .

Alpha is the discount factor, typically a scalar value between 0 and 1. A value of alpha closer to 0 results in policies that favors short term decision making, and value of alpha closer to 1 favors policies with a sense of a longer horizon in decision making.

r_onestep is a scalar value, typically a scalar value less than 0, which is the cost to take an action. The fourth step would be to use MATLAB's max() function to take the maximum of Vtemp along the dimension that corresponds to the actions. The max() function would return (i) the maximum possible value for a corresponding state considering all possible actions taken from that state, and (ii) the index corresponding to the action that yields the maximum possible value for that state.

A second optimization was done by creating a matrix meshes for states, actions, and state-action id pairs. The meshes for state, action, and state-action id pairs were then passed as arguments into a rewritten transition function that evaluates state transitions for all state action pairs at once rather than a single state and action at a time.

Each instance of meshes were created by using MATLAB's ndgrid function, columnizing each of those matrixes that ndgrid() would return by using the index argument (:), and then concatenating those columnized matrices to form a mesh. The transition function used the state-action id's mesh to index the state mesh and action mesh multiple times for each state action pairs at once. This form of indexing indexing returns matrices that give the resulting state value and action value for each corresponding state action pair. Those state value and action value matrices were then used to replace the single state and single action variables from the old transition function to compute a matrix of all the corresponding "new" states at the same time. Multiplications from the original transition function were replaced with elementwise multiplications accordingly.

High Dimensionality, Value Iteration, and Barycentric Interpolation in Robotic Control
by Roman Aguilera

Acknowledgements:

First off, thank you to my advisor Katie Byl for the freedom she has given me throughout my time here at UCSB to pursue topics that interested me, for giving me very interesting research topics to work on, for her guidance, for allowing me a space in the Dynamic Robotics Lab, for your passion as a teacher, for your letter of recommendation for the Google-CAHSI Dissertation Fellowship, and for just being an overall nice person (sometimes too nice ☺).

Thank you to my committee Ambuj Singh, Daniel Lokshtanov, and Linda Petzold.

Ambuj thank you for allowing me into the the IGERT in Network Science Traineeship when I came into UCSB, thank you for supporting me in my early years at UCSB, thank you for serving as my faculty advisor before my transition into the Robotics Lab, and thank you for your recommendation to reach out to Katie when I mentioned my interest in Robotics.

Daniel thank you for being a part of my committee for always making yourself available, for always being friendly, and thank you for your academic rigor when teaching the Algorithms course when I took it.

Linda, thank you for always being a warm person to talk to and for making yourself available on short notice. Sorry I failed your numerical simulation course. I promise I still found the material very interesting.

I would like to thank my family members for their unconditional love and support.

Thank you to my father Ramon Aguilera Sr. for always being ready and willing to help, for visiting me when I was feeling low, for being willing to help through handywork (from building me my bedframe to finding a car for me when my Corolla broke down), for being a model of strength and discipline, and for always being proud of me (high five).

Thank you to my mother Maria Aguilera for always giving me a good laugh whenever I would come home to visit, for being a ray of sunshine, and for always making sure I am well fed when I visit home.

Thank you to my sister Maribel Aguilera-Hernandez for always wanting the best for me and always wanting to make sure our family is supported and doing well. Thank you for running for city council of Santa Maria and for always wanting to make a positive change for our community.

Thank you to my sister Marilu Aguilera-Leer, for supporting our family, for the motivational talks, for always listening, for allowing me to be open, for showing me how to be a gentle person, and for decorating the house for Christmas and family gatherings.

High Dimensionality, Value Iteration, and Barycentric Interpolation in Robotic Control
by Roman Aguilera

Thank you to my brother Ramon Aguilera Jr. for helping me make lifetime memories throughout the years, for really getting to know me when it came to gift giving, for helping me see things in a different perspective, and for always laughing at our inside jokes.

Thank you to my brothers in law Noe Hernandez and Jose Leer for being helpful in multiple ways throughout the years, for supporting my sisters, and for your calm presences.

Thank you to my niece Natalie, my nephews Aiden, Teddy, and Dracula for giving me countless memories throughout the years. Aiden and Natalie, I am especially very grateful to have been able to witness you both grow from birth to preschool while attending graduate school.

Thank you to my fellowship coordinators, Tim Robinson, Arica Lubin, Javier Read de Alaniz.

Tim, I appreciate you reaching out to me consistently especially in my early years here at UCSB and going beyond your role in the IGERT in Network Science to check in on how I was doing after the IGERT was completed.

Javier and Arica, thank you both for all the hard work you did for the Bridge to Doctorate Fellowship as well as for making it possible to focus wholeheartedly on my masters project during my last quarter. Thank you for organizing meetings throughout the quarter for the BD fellows to catch up. Arica thank you for consistently checking in with me and for maintaining a sense of positivity throughout my years here.

Thank you to my dear friends Sharad Shankar, Katherine Dickson for always giving their time and energy for emotional support, for giving me a place to sleep when I had none, and for always ensuring that I have a safe space with them. I cannot thank them enough.

Thank you to Sherry Chien for providing an extreme amount of emotional support in my last year at UCSB, for giving me some good memories, and for helping me apply to the Google-CAHSI dissertation fellowship. I almost lost hope and gave up on the application when I was feeling overwhelmed before it was due, but you helped me push through it at the end. Thanks to your support, I was able to submit the application, and thank fully I received the fellowship award, which was a source of lifechanging financial support for me.

Thank you to my good childhood friends Hammer Lazaro, Alex Solorio, Mark Anthony Rayas for being good friends to me throughout the years.

Thank you Hammer, for being my good friend since eighth grade, for cheering me up when I’m down, and being willing to have random calls to catch up.

Thank you Alex, for being my friend since eight grade, and for organizing our many camping trips to National Parks such as Zion, Kings Canyon, Horseshoe Bend, Grand Canyon. Those were very good memories.

High Dimensionality, Value Iteration, and Barycentric Interpolation in Robotic Control
by Roman Aguilera

Thank you Mark Anthony, for being my friend since fourth grade, for being willing to drive long distances for day trips, inviting me to local events, and for taking me out of the house when I needed a break.

I would like to thank my labmates for their support as well.

Thank you Sean Gillen, for helping me out with learning how to program in python, for helping me learn how to create gym reinforcement learning environments, for being a good project partner, for being a positive and helpful person in general, for reminding me to have confidence in myself and to recognize my worth, and for including me as one of your housemates.

Thank you Asutay Ozmen, for always being willing to help me in any way that you could, from discussing controls concepts to just talking about life in general), and for being a phenomenal and proactive head TA for Katie's Robotics and Control course.

Thank you to Guillaume, for answering the many questions I had about programming, neural nets, and reinforcement learning.

Thank you Nihar, for introducing me to trajectory optimization and for leaving an open invite to join you at Hollister Brewing Company.

Thank you Aditya for helping me and Aarti troubleshoot our course project, and for your guidance in getting started with learning about trajectory optimization.

Thank you It for talking with me about meditation, spirituality and mindfulness.

Thank you Christopher Cheney, for being a chill presence in the lab, for helping me brainstorm with my research projects, for being a nice person overall, for the many casual chats, and for organizing lab meetings, and for being a good listener.

Thank you Andrew Herdering, for helping me learn about dw service, for filling me in on the random bits of information such as the status of railroad networks in Europe, and for making me aware of people such as Kevin Knoedler.

Thank you to my community of graduate students throughout my years here at UCSB: Aimal Khankel, Tobias Mazal, Emmanuel Kayede, Marcela Areyano, Hyunjin Kim, Azzedin Jackson.

To those who should be in this acknowledgement, but I did not remember to include you, my sincerest apologies.

Thank you all for your support.
-Roman