Construction of Embedded Markov Decision Processes for Optimal Control of Non-Linear Systems with Continuous State Spaces

Nikovski, D.; Esenther, A.

TR2011-081 December 2011

Abstract

We consider the problem of constructing a suitable discrete-state approximation of an arbitrary non-linear dynamical system with continuous state space and discrete control actions that would allow close to optimal sequential control of that system by means of value or policy iteration on the approximated model. We propose a method for approximating the continuous dynamics by means of an embedded Markov decision process (MDP) model defined over an arbitrary set of discrete states sampled from the original continuous state space. The mathematical similarity between sets of barycentric coordinates (convex combination) and probability mass functions is exploited to compute the transition matrices and initial state distribution of the MDP. Barycentric coordinates are computed efficiently on a Delaunay triangulation of the set of discrete states, ensuring maximal accuracy of the approximation and the resulting control policy.

IEEE Conference o Decision and Control and Eurpoean Control Conference (CDC-ECC)

Construction of Embedded Markov Decision Processes for Optimal Control of Non-Linear Systems with Continuous State Spaces

Daniel Nikovski and Alan Esenther
Mitsubishi Electric Research Laboratories
201 Broadway, Cambridge, MA 02139, USA
E-mail: {nikovski,esenther}@merl.com

Abstract—**We consider the problem of constructing a suitable discrete-state approximation of an arbitrary non-linear dynamical system with continuous state space and discrete control actions that would allow close to optimal sequential control of that system by means of value or policy iteration on the approximated model. We propose a method for approximating the continuous dynamics by means of an embedded Markov decision process (MDP) model defined over an arbitrary set of discrete states sampled from the original continuous state space. The mathematical similarity between sets of barycentric coordinates (convex combinations) and probability mass functions is exploited to compute the transition matrices and initial state distribution of the MDP. Barycentric coordinates are computed efficiently on a Delaunay triangulation of the set of discrete states, ensuring maximal accuracy of the approximation and the resulting control policy.**

Index Terms—**optimal control, embedded Markov chains, dynamic programming, Markov decision process models**

I. Introduction

Many dynamical systems have state spaces that are naturally continuous, but can be controlled successfully by applying controls from a relatively small discrete set, applied at discrete moments in time. The state of such systems is a real-valued vector $x \in R^d$, where d is the dimensionality of the (continuous) state space of the dynamical system, and the controls $a \in A$ are discrete and belong to the set A. The dynamics of the control system are described by the general form $x_{k+1} = f(x_k, a_k)$, where x_k is the state of the system at time t_k, a_k is the control applied at time t_k, f is an arbitrary non-linear function, and the system evolves in discrete time such that $t_k = k\Delta t$ for a suitably chosen constant interval Δt.

The goal of sequential optimal control is to select a sequence of actions a_0, a_1, a_2, $\ldots$, such that some performance measure dependent on the states traversed by the systems and the controls applied to it is optimized. One example of such a performance measure is the cumulative cost over a finite horizon of K steps $J = \sum_{k=0}^{K} g(x_k, a_k) + h(x_K)$, where $g(x_k, a_k)$ is a suitably chosen running cost, and $h(x_K)$ is a terminal cost associated with the final state x_K. Another possible performance measure is the discounted cumulative cost over an infinite horizon $J = \sum_{k=0}^{\infty} \gamma^k g(x_k, a_k)$, where $0 < \gamma < 1$ is a discounting factor. General analytical methods for solving this optimization problem exactly for arbitrary functions f, g, and h are not known, although solutions for special cases have been known and used for a long time, such as the linear quadratic regulator (LQR) that can be applied when the control action a is real-valued, the function f is linear, and the functions g and h are quadratic in the state x and control a [1]. However, in the general case, the function f is not linear, and the cost functions g and h are not quadratic in the state and control. In such cases, the optimal control has to be found by means of numerical methods [2], or reinforcement learning [3], [4].

An alternative strategy for solving the optimal control problem is to convert the continuous-state-space dynamical system into a Markov decision process (MDP) with discrete state space and solve it by means of existing algorithms such as policy iteration and value iteration [5]. The algorithm proposed in this paper is one such method.

A discrete-space MDP is described by a discrete set of states S such that the MDP occupies one of these states $s_k \in S$ at any time t_k, and a transition probability function $p(s_{k+1}|s_k, a_k) = Pr(s_{k+1}|s_k, a_k)$ that expresses the probability of being in state s_{k+1} at time t_{k+1} if the MDP was in state s_k at time t_k and control (action) a_k was applied at that time. Similarly to the dynamical system described above, the MDP evolves in discrete time ($t_k = k\Delta t$), and the controls (actions) $a \in A$ are discrete and belong to a relatively small set A. The goal is to optimize a performance measure $R = \sum_{k=0}^{K} r(s_k, a_k)$, much like in the continuous case.

The similarities between the class of continuous-state-space systems that we are considering, and MDPs, are that both evolve in discrete time under the effect of a small number of discrete actions, and both seek to optimize a performance criterion defined over states and actions. The two major differences are in the type of state used (continuous $x \in R^d$ vs. discrete $s \in S$) and in the way state evolution is described (function $f(x,a)$ vs. probability transition function $p(s_{k+1}|s_k,a_k)$). The objective of the conversion method, then, is to construct a state set S embedded in R^d and a transition function $p(s_{k+1}|s_k,a_k)$ for every triple (s_{k+1}, s_k, a_k) such that $s_{k+1} \in S$, $s_k \in S$, and $a_k \in A$. After the MDP is constructed, an optimal policy $a_k = \pi^*(s_k)$ that maps states to optimal controls can be found for every $s_k \in S$, by using well-known algorithms such as policy iteration and value iteration [6], [4]. From this MDP policy, a control law $a_k = \mu^*(x_k)$ for the continuous-state-space system can be obtained, as described below.

II. A Method for Constructing Embedded MDPs for Dynamical Systems with Continuous State Spaces

The proposed method is based on similarities in the mathematical properties of probability functions and convex combinations. A probability mass function specifies the probability that a random variable is equal to some specified value. For the case of MDPs, the transition function $p(s_{k+1}|s_k,a_k)$ is such a (conditional) probability mass function, conditioned on the starting state s_k and the applied control a_k. The random variable for which the probability function is specified is the successor state s_{k+1}. If the size $|S|$ of the state set S is N, let $s^{(1)}$, $s^{(2)}$, $\ldots$, $s^{(N)}$ be an enumeration of all states. The elements of the transition function can then be defined as $p_i \doteq Pr(s_{k+1} = s^{(i)}|s_k,a_k) = p(s^{(i)}|s_k,a_k)$. From the axiomatic properties of probability mass functions, then, it is always true that $\sum_{i=1}^{N} p_i = 1$, and $0 \le p_i \le 1$, $i = 1,\ldots,N$.

On the other hand, a convex combination of N vectors y_i, $i = 1,\ldots,N$ is defined as $\sum_{i=1}^{N} c_i y_i$, such that $\sum_{i=1}^{N} c_i = 1$, and $c_i \ge 0$, $i = 1,\ldots,N$. By comparing the two definitions, it can be observed that probability mass functions and the set of coefficients defining a convex combination obey exactly the same mathematical constraints, and a valid probability function can be used as coefficients of a valid convex combination, and vice versa. We will use this fact to construct all transition functions of the MDP as sets of coefficients for suitably defined convex combinations.

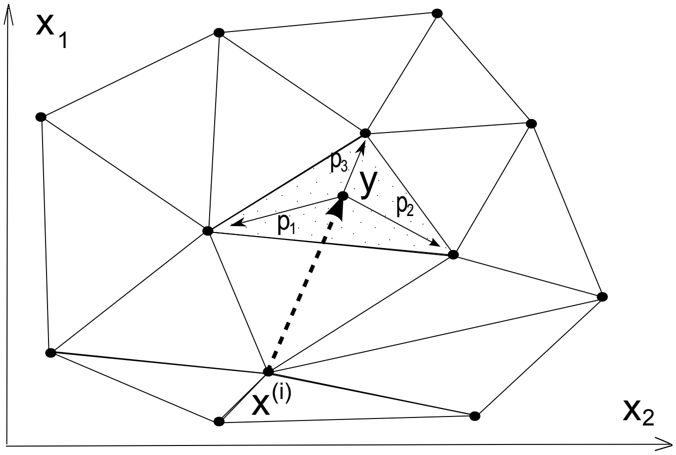

Figure II.1. A Delaunay triangulation on a set of vertices sampled from the embedding two dimensional space. The dashed line shows the transition from some starting state $x^{(i)}$ under action a resulting in end state $y = f(x^{(i)},a)$. The simplex (here, triangle) containing the end state y is shown with a dotted background, and the barycentric coordinates p_1, p_2, and p_3 of y are computed with respect to the vertices of that simplex. These coordinates are also the transition probabilities from $x^{(i)}$ under action a to the states corresponding to these vertices in the resulting MDP.

A. Conversion Algorithm

Step 1: The algorithm starts with selecting N states $s^{(1)}$, $s^{(2)}$, $\ldots$, $s^{(N)}$ such that each of them corresponds to a continuous state $x^{(i)} \in R^d$. Every $x^{(i)}$ is a point (vector) in d-dimensional Euclidean space. We will call these points *anchor points* and will denote the set of all anchor points by $X = \{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}$. Any selection method could be used, for example sampling the continuous state space uniformly, imposing a kind of a regular grid and using its vertices as $x^{(i)}$, or following a set of trajectories using one or more preset policies, and recording the resulting states. The last method is especially applicable to systems where the accessible state space is only a relatively small subset of the entire state space, for example specific linear subspaces, or general manifolds. After selection, the selected anchor points are stored in a suitable data structure, for example the d-by-N matrix B, where each column is one anchor point.

Step 2: Find the Delaunay triangulation $DT(X)$ of the set of points X [7], [8]. The Delaunay triangulation consists of simplices, each of which has $d+1$ vertices, such that each of these vertices is a member of X, i.e., an anchor point. In two-dimensional space ($d = 2$, the plane), the simplices are triangles; in three-dimensional space ($d = 3$), the simplices are tetrahedra, etc. Let there be M such simplices. Store the

Delaunay triangulation in a suitable data structure, for example the $d+1$-by-M matrix D, where each column corresponds to a simplex, and the $d+1$ entries in the column contain indices to the anchor points in the matrix B.

Step 3: For every starting state $s^{(i)}$ and control $a^{(l)}$, repeat the following sub-steps:

Step 3.1: Retrieve the anchor point $x^{(i)}$ that corresponds to state $s^{(i)}$.

Step 3.2: Use the system function f of the continuous dynamical system to find the successor point y of $x^{(i)}$ under control $a^{(l)}$: $y = f(x^{(i)}, a^{(l)})$. In general, the successor point y does not coincide with any of the pre-selected anchor points $x^{(i)}$, $i = 1, \ldots, N$.

Step 3.3: Find the simplex in $DT(X)$ that contains point y. To this end, traverse all M simplices in $DT(X)$ and repeat the following steps for every simplex m, $m = 1, \ldots, M$:

Step 3.3.1: Retrieve the last, $d+1$st vertex of simplex m, and store it in vector q.

Step 3.3.2: Create a d-by-d matrix E, whose column j contains the difference $v_{m,j} - q$ between the j-th vertex of simplex m (denoted here by $v_{m,j}$) and the vector q, for $j = 1, \ldots, d$.

Step 3.3.3: Find the d-dimensional vector c such that $Ec = (y-q)$ by solving the set of d simultaneous linear equations.

Step 3.3.4: Compute the $d+1$st element of the vector c as $c_{d+1} = 1 - \sum_{j=1}^{d} c_j$.

Step 3.3.5: For every element c_j, $j = 1, \ldots, d+1$, if $c_j < 0$, then simplex m does not contain point y; increase m by one and go back to Step 3.3.1 to test the next simplex. If all $c_j \geq 0$, $j = 1, \ldots, d+1$, then this simplex contains point y; exit the loop of Step 3.3, and go to the next step.

Step 3.4: At this point, the $d+1$-dimensional vector c contains coefficients that define a valid convex combination such that $y = \sum_{j=1}^{d+1} c_j v_{m,j}$. Moreover, it defines a valid probability transition function, since all of its entries are positive and sum up to unity.

Step 3.5: In order to construct a complete transition probability distribution over all possible N successor states, we perform the following step for each state $s^{(i)}$, $i = 1, \ldots, N$. If $s^{(i)}$corresponds to one of the vertices of the simplex m, that is, $x^{(i)} = v_{m,j}$ for some j, then the corresponding transition probability of the MDP is $p_i = Pr(s_{k+1} = s^{(i)} | s_k, a_k) = p(s^{(i)} | s_k, a_k) \doteq c_j$; otherwise, $p_i \doteq 0$.

Conceptually, we can think of this algorithm as a way of converting the system dynamics represented by the function f to an equivalent probabilistic representation involving only a small set of points $x^{(i)}$ embedded into the original continuous state space of the system. If the system starts in one of these few points, the successor state y, in general, will not coincide with another one of these points. However, we can identify the $d+1$ points that define a simplex that completely encloses the successor state y, and can think that the system has transitioned not to point y itself, but to the vertices of this simplex with various probabilities, instead. The probabilities are equal to the convex decomposition of point y with respect to the vertices of the simplex, also known as the barycentric coordinates of that point within the simplex. The similarities between convex combinations (barycentric coordinates) and probability mass functions required by the MDP formalism make this conversion possible.

In order to speed up computations, the inverse $E^{-1}$of matrix E can be pre-computed and stored for every simplex in the Delaunay triangulation, and then used in step 3.3.3 to find c using $c = E^{-1}(y-q)$, rather than solving a set of linear equations every time.

Another possibility for computational speed up is to use such an order of traversal of the simplices of the Delaunay triangulation that would result in faster discovery of the simplex containing the end point y. It is reasonable to expect that, in most cases, the simplex that encloses the successor state y will be generally closer to it than other simplices. If the centroid of each simplex (i.e., the average of all its vertex points) is pre-computed, and the Euclidean distance between each centroid and y is computed, the simplices of the Delaunay triangulation can be traversed in increasing order of that distance in Step 3.3.

Finally, the set of N anchor points can be triangulated in any other manner, if Delaunay triangulation is too expensive in higher-dimensional state spaces. For $d = 2$, an efficient $O(NlogN)$ algorithm for computing Delaunay triangulations exists [8]; for $d > 2$, (non-Delaunay) triangulations can be computed by starting with an initial simplex of $d+1$ points, and adding the remaining $N-d-1$ points one by one, each time sub-dividing the simplex that contains the point into $d+1$ new simplices, or adding a new simplex, if no existing simplex contains the point. Since finding the containing simplex takes at most $O(N)$ computation, the overall complexity of the triangulation step is $O(N^2)$. This is also the complexity of Step 3 of the algorithm, and hence the final complexity of constructing the MDP.

B. Optimal Control Based on the Constructed MDP

Given a constructed MDP, an optimal policy $a = \pi^*(s)$ for that MDP can be found by means of policy iteration or value iteration [6]. For example, for the discounted case, we can evaluate the value function $V(s^{(i)})$ of the MDP by means of repeated applications of Bellman backups of the form

$$\begin{aligned} V(s^{(i)}) &= \max_a\{g(s^{(i)}) + \gamma \sum_{j=1}^{N} Pr(s^{(j)}|s^{(i)}, a)V(s^{(j)})\} \\ &= \max_a\{g(s^{(i)}) + \gamma \sum_{j=1}^{N} p_j V(s^{(j)})\} \end{aligned}$$

for each state $s^{(i)}$, $i = 1, N$, until convergence of the value function $V(s^{(i)})$. The optimal policy for the MDP is then $\pi^*(s^{(i)}) = argmax_a Q(s^{(i)}, a)$, where we make use of the auxiliary function $Q(s^{(i)}, a) \doteq g(s^{(i)}) + \gamma \sum_{j=1}^{N} p_j V(s^{(j)})$. The computational complexity of this solution is only $O(Nd)$, since in each iteration of the value iteration algorithm, a Bellman backup is performed for each of the N anchor states, and, although the sum is over all N successor states $s^{(j)}$, there are non-zero transition probabilities to only $d+1$ of them (identified in Step 3.3. of the conversion algorithm above), and they can be either stored explicitly during conversion, or the transition probabilities can be placed in sparse matrices.

The ultimate goal, however, is to find a control law $a = \mu^*(x)$ that is a mapping from the *continuous* state x, as opposed to the discrete state of the MDP s. By recognizing that our method introduces uncertainty about the state the system is in, we can use several control strategies from the field of partially observable Markov decision processes (POMDP) [9]:

Nearest anchor point: find the closest anchor point $x^{(i)}$ to x in the embedding continuous space in terms of Euclidean distance, and use the optimal action for the corresponding MDP state $s^{(i)}$: $a = \pi^*(s^{(i)})$.

Largest vote: find the simplex m that contains x, and compute the barycentric coordinates c of x with respect to the $d+1$ vertices $v_{m,j}$, $j = 1, \ldots, d+1$ of that simplex, as described in Step 3.3 of the algorithm above. Then, if $a_j = \pi^*(s^{(j)})$, where $s^{(j)}$ is the state corresponding to vertex $v_{m,j}$, we can use c_j as an individual vote for action a_j, and execute the action that has the highest cumulative vote over all $d+1$ vertices.

Highest expected merit: use the barycentric coordinates to estimate the merit $\hat{Q}(x, a)$ of the individual action a taken in state x as $\hat{Q}(x, a) = \sum_{j=1}^{d+1} c_j Q(s^{(j)}, a)$, and use the control law $\mu^*(x) = argmax_a \hat{Q}(x, a)$. Given that the barycentric coordinates c can be interpreted as individual probabilities that the MDP is in one of its discrete states, the function $\hat{Q}(x, a)$ is indeed the exact expected merit of taking action a at the continuous state x.

III. Experimental Validation

The proposed method was validated on a mountain car test problem popular in the field of reinforcement learning (Fig. 3.1) [3]. The task is to drive an under-powered car located in a valley up a steep mountain road to the right. The difficulty is that gravity is stronger than the car's engine, and even at full throttle the car cannot accelerate up the steep slope. The only solution is to first move away from the goal and up the opposite slope. Then, by applying full throttle the car can build up enough momentum to carry it up the steep slope even though it is slowing down the whole way. The reward is -1 for all state/action combinations that do not result in the goal (a position to the right of the valley, at the top of the right slope), and 1 for states where the car is at the goal position (regardless of the velocity and action). The three possible actions are $a^{(1)} = +1$, full throttle forward; $a^{(2)} = -1$, full throttle reverse; and $a^{(3)} = 0$, zero throttle. The state of the continuous system $x = [l, v]^T$ is described by the car's position l and velocity v. Assuming a control period of length one time unit and simplified physics, the dynamical system is described by the equations ([3]):

$$l_{k+1} = l_k + v_k$$

$$v_{k+1} = v_k + 0.001a_k - 0.0025cos(3l_k)$$

The state variables were bounded by $-1.2 \leq l \leq 0.6$, $-0.7 \leq v \leq 0.7$; the starting state was $l_0 = -0.5$, $v_0 = 0$, and the goal was defined as $l \geq 0.5$.

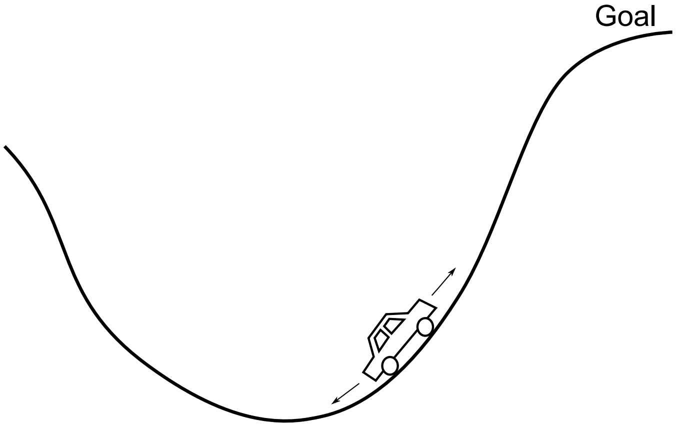

Figure III.1. An under-powered car on a mountain slope that can reach the goal only by means of building up momentum going up and down the opposite slope.

The state space of the continuous system was sampled by imposing a regular mesh grid on the two-dimensional state space. In experiments, the number N of states of the MDP was varied by changing the number n of rows, respectively columns, of that grid, such that $N = n^2$. The algorithm for constructing MDPs was implemented in Matlab, and the resulting

n	N	Iterations	Steps to goal
3	9	273	150
4	16	247	108
5	25	232	140
6	36	431	102
7	49	253	171
8	64	313	127
9	81	351	133
10	100	301	103
20	400	284	137
50	2500	24	152

Table I
NUMBER OF ITERATIONS NECESSARY FOR CONVERGENCE OF VALUE ITERATION, AND NUMBER OF STEPS UNTIL THE GOAL IS REACHED, FOR VARIOUS SIZES OF THE STATE SET OF THE MDP.

MDP was solved by means of value iteration with discount factor $\gamma = 0.99$, as implemented in a publicly available MDP toolbox [10]. This toolbox uses sparse matrices to represent the transition probabilities of the MDP, and benefits directly from the sparsity of the transition probability functions computed by the proposed method. After value iteration converged, the resulting policy was executed, using the highest-expected-merit control law, and the number of steps until the goal was reached was recorded. Table 1 shows the experimental results for several values of N, including the number of iterations needed until the value function iteration converged.

The value function obtained for $N = 400$ after value iteration has converged is shown in Fig. 3.2, and the trajectory to the goal resulting from executing the resulting policy is shown in Fig. 3.3.

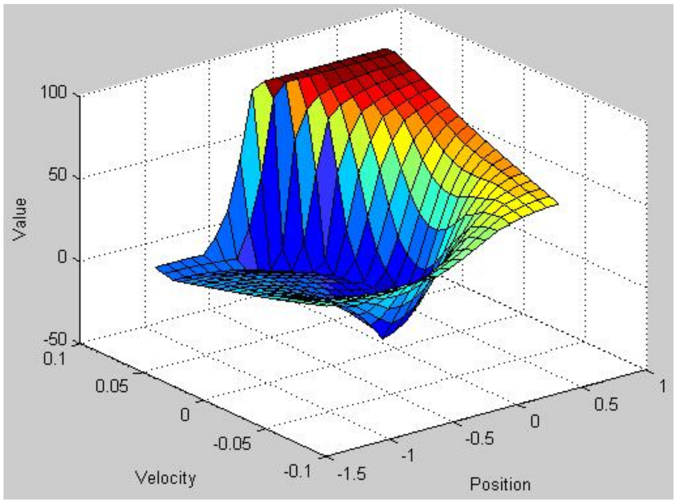

Figure III.2. Value function for $N = 400$ states of the MDP.

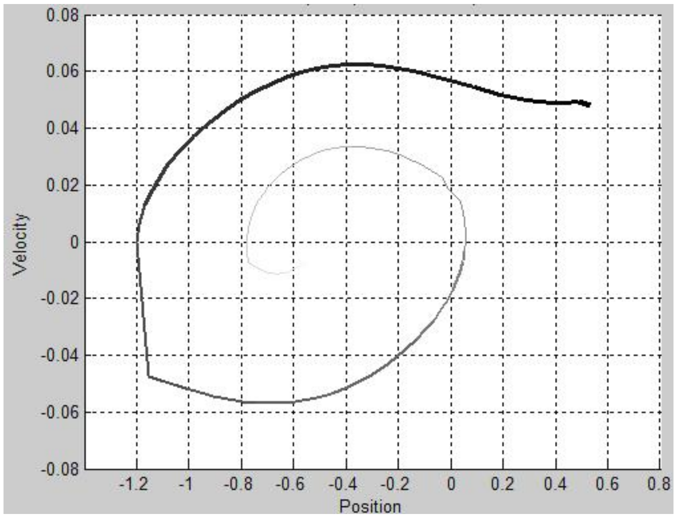

Figure III.3. Trajectory resulting from executing the optimal policy determined by solving the MDP with $N = 400$ states. Darker/thicker line indicates more recent points in the traversal along the trajectory.

IV. RELATED WORK

As noted above, many numerical methods for solving optimal control problems have been proposed in the control systems, operations research, and reinforcement learning communities. The specific instance of the problem that we are considering in this paper, namely continuous state space and discrete time and actions, is a special case of the more general optimal control problem, where all variables (state, actions, and time) are continuous. As such, it can be solved by means of general optimal control methods, including both indirect and direct methods. In indirect methods, the problem is formulated as a two-point boundary-value problem, and its solutions are the desired control and state trajectories. A disadvantage of such indirect methods is that solving boundary-value problems numerically can be very difficult, error prone, and computationally demanding. In direct methods, the state and control trajectories are approximated by means of appropriate parametric approximators, and the corresponding parameters are estimated by means of general-purpose nonlinear optimization methods. An example is the direct collocation method, currently implemented in powerful modeling and optimization tools and languages such as JModelica/Optimica [11]. Although such methods can be very fast and effective, the direct result of their computation is not a control law, but an optimal state and control trajectory, whereas the method proposed in this paper computes an entire control law over the entire state space. Furthermore, the MDP constructed by the proposed algorithm can be extended to handle uncertainty in system dynamics by modifying its transition probabilities, whereas direct optimal control methods would have to solve a stochastic optimal control problem, which is much harder to solve than the deterministic case.

The reinforcement learning and operations research communities have also proposed multiple algorithms for solving sequential optimal decision and control problems, commonly incorporating the MDP framework. One of the dominant ideas has been to use universal value function approximators, such as feed-forward neural networks, radial-basis functions, k-nearest neighbor, etc., to represent the value function of a sequential decision problem over the entire state space of the problem, and then minimize the residual in the Bellman equation at select points, fitting the parameters of the function approximators in the process. To a certain degree, the method proposed in this paper is based on the same idea for using a universal function approximator to represent the value function, in this case piecewise linear approximation over a collection of simplices. However, two important differences exist. First, the proposed method approximates the dynamical system by an MDP, and then calculates the value function of the MDP *exactly*, as opposed to approximating the value function of the original system in the process of estimating it. This has important consequences as regards the convergence guarantees of the method. Since the method constructs a standard MDP, the convergence of the solution procedure used, such as value iteration, policy iteration, linear programming, etc., is guaranteed, and the rates of convergence can be estimated based on existing research [6]. In contrast, the convergence of value iteration when used with an arbitrary universal function approximator is not at all guaranteed, and research has shown that many popular function approximator schemes may in fact lead to divergence [12].

The second difference between the proposed method and most solution methods from the field of reinforcement learning such as Q-learning and TD(λ) is that such methods use the system dynamics only as a source for sampling system transitions, whereas the proposed method uses the system dynamics directly for the exact calculation of the transition probabilities of the MDP. This has the practical consequence that once the MDP model is constructed, finding the optimal control law over the entire state space is very fast ($O(Nd)$), whereas estimating value functions and optimal control policies from sampled system transitions can be excruciatingly slow.

V. Conclusion

A method for solving sequential optimal control problems with arbitrary non-linear dynamics in continuous state spaces was proposed, and its operation was verified on a popular test problem. The method reduces the continuous-space dynamics to a Markov decision process defined over a number of anchor points sampled from the original state space. The embedding d-dimensional continuous state space is triangulated into simplices of $d+1$ vertices each, and the non-linear dynamics are approximated over this triangulation by constructing suitable transition probability functions of the MDP. Unlike previous approaches, the states of the MDP are not the subdivisions of the embedding spaces (simplices, in this case) themselves, but the vertices of these simplices. Since the original dynamical system will almost never be at one of these vertices, its actual state x is represented as a probability function over the vertices of the simplex that contains x. By exploiting the mathematical similarities between barycentric coordinates (convex combinations) and probability mass functions, it is possible to treat the state x as a probability distribution over discrete states of the MDP, and compute an optimal control law for that continuous state based on the optimal policy for the embedded MDP.

References

[1] R. F. Stengel, *Optimal Control and Estimation*. Mineola, NY: Dover, 1986.

[2] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, Massachusetts: Athena Scientific, 2000.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning*. Cambridge, Massachusetts: The MIT Press, 1998.

[4] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, Apr. 30 1996. [Online]. Available: http://arxiv.org/abs/cs/9605103

[5] J. Kushner and P. Dupuis, *Numerical Methods for Stochastic Control Problems in Continuous Time*. Heidelberg: Springer Verlag, 2001.

[6] M. L. Puterman, *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. New York, NY: John Wiley & Sons, Inc., 1994.

[7] F. P. Preparata and M. I. Shamos, *Computational Geometry*. Heidelberg: Springer Verlag, 1990.

[8] S. Skiena, *The Algorithm Design Manual (Second Edition)*. Springer, 2008.

[9] A. R. Cassandra, L. P. Kaelbling, and J. A. Kurien, "Acting under uncertainty: Discrete bayesian models for mobile robot navigation," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

[10] I. Chadès, M.-J. Cros, F. Garcia, and R. Sabbadin, "Markov Decision Processes (MDP) Toolbox," http://www.inra.fr/mia/T/MDPtoolbox/.

[11] J. Åkesson, K.-E. Årzén, M. Gäfvert, T. Bergdahl, and H. Tummescheit, "Modeling and optimization with optimica and JModelica.org - languages and tools for solving large-scale dynamic optimization problems," *Computers & Chemical Engineering*, vol. 34, no. 11, pp. 1737–1749, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.compchemeng.2009.11.011

[12] G. J. Gordon, "Stable function approximation in dynamic programming," in *Proceedings of the International Conference on Machine Learning*, 1995, pp. 261–268.