Value Function Approximation and Model Predictive Control

Mingyuan Zhong*, Mikala Johnson*, Yuval Tassa†, Tom Erez† and Emanuel Todorov*
*Department of Applied Mathematics
University of Washington, Seattle, WA 98195
Email Mingyuan Zhong: Zhongmy@uw.edu
†Department of Computer Science
University of Washington, Seattle, WA 98195

***Abstract*—Both global methods and on-line trajectory optimization methods are powerful techniques for solving optimal control problems; however, each has limitations. In order to mitigate the undesirable properties of each, we explore the possibility of combining the two. We explore two methods of deriving a descriptive final cost function to assist model predictive control (MPC) in selecting a good policy without having to plan as far into the future or having to fine-tune delicate cost functions. First, we exploit the large amount of data which is generated in MPC simulations (based on the receding horizon iterative LQG method) to learn, off-line, the global optimal value function for use as a final cost. We demonstrate that, while the global function approximation matches the value function well on some problems, there is relatively little improvement to the original MPC. Alternatively, we solve the Bellman equation directly using aggregation methods for linearly-solvable Markov Decision Processes to obtain an approximation to the value function and the optimal policy. Using both pieces of information in the MPC framework, we find controller performance of similar quality to MPC alone with long horizon, but now we may drastically shorten the horizon. Implementation of these methods shows that Bellman equation-based methods and on-line trajectory methods can be combined in real applications to the benefit of both.**

I. Introduction & Motivation

Reinforcement Learning (RL) offers a compelling paradigm: instead of hand-tuning a controller using careful analysis of the specific system at hand, RL allows us to define high-level goals for the desired behavior and computes control rules through numerical optimization. The biggest challenge of RL is the curse of dimensionality — since the volume of the state space grows exponentially with added dimensions, any attempt to optimize a global policy a-priori becomes computationally infeasible in large, complex systems.

The alternative is to use trajectory optimization by focusing the limited computational resources on the most relevant states. Model Predictive Control (MPC) [1], [2] is an approach that applies trajectory optimization in real time to the current state of the system. Computing the policy in real time makes MPC very reactive, allowing it to generate a control law for any state of the system. However, reliance on on-line computation is also the primary limitation of MPC — computational resources are invariably limited, and on-line trajectory optimization for high-DOF, complex systems is a taxing task.

The main parameter used to regulate the computational demand of MPC is the length of the planning horizon. Akin to the search depth in classical AI applications (e.g., how many steps ahead a chess software considers before choosing a move), the choice of a particular horizon affects both the computational load (shorter horizons are easier to compute) and the level of performance (planning over shorter horizons results in poorer behavior). In some cases, we may find a "sweet-spot" for this trade-off. However, if real-time optimal control is required by the application and the longest horizon we can afford to compute in real time is too short to yield effective control, a different approach is needed.

In the course of simulation, MPC provides an approximation to the optimal value function and a near-optimal control policy near the states which the controller visits. We wish to exploit this data to avoid having to continually recalculate the value function at states that have previously been visited; further, we can use this data to shorten the horizon that MPC uses to calculate the optimal trajectory. In the extreme case, with exact replication of the value function or policy solely from data, we could use straight-forward greedy optimization to control the system (equivalent to MPC with a 1 time step horizon).

While RL global methods may be guaranteed to converge to the globally defined cost-to-go function or control policy in the limit of a large number of basis functions, neurons, or a high-degree of polynomial, in practice, these approximate methods must handle the finite error in representation which they generate. RL-based approximations often suffer from the curse of dimensionality, low accuracy, and low efficiency in reality. These shortcomings motivate us to find novel, practical methods of exploiting a globally defined cost-to-go function and control policy.

A. Related work

Finding a good way to combine the advantages of local and global dynamic programming is not a new endeavor. In particular Atkeson and colleagues have been actively exploring this area for the past decade [3], [4]. Their data-based approach involves saving the results of off-line trajectory optimization as a "trajectory library" and then selecting data from this library during real-time control. Similar results can also found in Real-Time Dynamical Programming [5], heuristics for

learning the game GO [6] and shortest path problem solving applications [7]. The main drawback of this approach is that it does not efficiently exploit computational resources in real time. If we already have the trajectory-optimization machinery at hand that runs in real time, why not use it in both the off-line and on-line contexts?

In some domains we may expect the optimal system to converge to a small volume of state space, e.g., when the optimal behavior leads to some target state or a limit cycle. In these cases, we may compute the accurate value function in that small region, and use it as a final cost for the MPC computation. This scheme is called *infinite-horizon MPC* [8], [9], [10], and it is guaranteed to produce the optimal behavior as long as the trajectory terminates in an area where the terminal cost accurately reflects the value function. Unfortunately, in high-DOF domains the system may find itself very far from the asymptotically-optimal region of state space. For example, consider a walking humanoid that has fallen on the ground; in this case, knowing the value function around the optimal gait does not provide enough information to figure out how to get up. On the other hand, it might well be that the system has already been in this perturbed state (or one similar to it), and it would be wise to find a way to reuse past experience to facilitate the current optimization.

B. Our contribution

The main contribution of this work is to demonstrate the results of interfacing model predictive control with value function approximation. First, leveraging machine learning techniques, we exploit the large quantities of data generated in a traditional MPC simulation to approximate the optimal value function globally. We use these global function approximations in the MPC framework as a final cost to help MPC make better decisions with shorter planning time. We demonstrate that, while these types of global function approximations may prove beneficial in some situations, there is great sensitivity to the quality of the approximation. As such, combining MPC with a fitted value function remains an open problem.

Second, we develop an alternative approach that reprsents a significant improvement over MPC alone. We obtain a global final cost by using the cost-to-go function generated by solving a linearly-solvable Markov Decision Process (LMDP), and we also warm-start MPC's policy search with the control policy also from solving the LMDP (both control policy and value function can be obtained in the same solve). The LMDP is an optimization problem explicitly related to the trajectory optimization problem which MPC solves, but it assumes certain structure in the mathematical formulation which differs from the MPC optimization problem. This structure reduces the Bellman equation to a linear functional equation, however we still need a function approximator in order to solve it numerically. We present results in which we use the solution of the LMDP, rather than trying to directly fit MPC data, as a final cost for MPC. With this value function approximation as a final cost and an extra "hint" from the computed LMDP policy, we are able to shorten the horizon of MPC while maintaining solution quality even on underactuated problems; the cart-pole and the acrobot. In the area of Bellman equation-based methods, this approach presents a novel way to recover complicated system behavior even from a coarse approximation of the cost-to-go function. In the area of MPC, this method serves as an automatic way to summarize the final cost.

II. Methods

In this section we explain details of our methods. First, we discuss MPC and how we propose to use global methods in conjunction with this on-line, trajectory-based method. Then, we discuss the two methods of obtaining a global value function approximation with which we experiment to improve MPC.

A. Model Predictive Control

1) Overview: Model Predictive Control, also known as Receding Horizon Control, solves the optimal control problem to a user-chosen finite time horizon. That is, from the given position in state space, MPC searches for an optimal policy in the neighborhood of the current position to some finite horizon. The first control of the computed optimal policy is applied to the system, and then MPC recomputes a new optimal trajectory from the new state (having not completed the old planned policy). The MPC algorithm is as follows.

1: **for** $i = 0$ to the end of simulation **do**
2: Initialize optimizer with a control sequence $\mathbf{u}^{init}_{i,\ldots,i+N-1}$
3: Solve for the optimal policy $\mathbf{u}_{i,\ldots,i+N-1}$
4: Apply only $\mathbf{u}_i$ to dynamical system, $\mathbf{x}_{i+1} = f(\mathbf{x}_i, \mathbf{u}_i)$
5: **end for**

The optimal policy in the line 3 of the algorithm is defined as the solution of the discrete-time problem

$$\min_{\mathbf{u}_{i,\ldots,i+N-1}} \left[\Sigma_{k=i}^{i+N-1} l(\mathbf{x}_k, \mathbf{u}_k) + v_F(\mathbf{x}_{i+N})\right] \qquad (1)$$

where $l(\mathbf{x}, \mathbf{u})$ represents the running cost, v_F is the final cost, and the states $\mathbf{x}_k$ and controls $\mathbf{u}_k$ are constrained by the discretized dynamics $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$.

There are multiple trajectory based optimization methods available. Having experimented with most of the techniques, we have found the iLQG method [1], [11], [12], which assumes a locally linear model of the dynamics and maintains a local value function of second order accuracy, to be the fastest and most robust overall. So, we use it exclusively in this work.

2) Discussion: The primary computational bottleneck in preventing MPC from running in real time, especially for high degree of freedom systems, is the need to plan far in advance at every time step. Therefore, it is desirable to add something to MPC that permits the same quality of policy at a shortened horizon, or, equivalently, a better policy with the same horizon.

From equation 1, one sees that MPC sums the running costs up through the end of the finite horizon and then anything that would define the system's behavior from time N to infinity is lumped into the final cost $v_F(\mathbf{x}_{i+N})$. That is, the final cost

should give information about the desirability of any state which the finite-horizon planner is able to reach. The more accurately the final cost describes the desirability of the state at the end of MPC's time horizon AND which directions in state-space increase this desirability, the less the method relies on the sum of the running costs. If the MPC horizon is fixed by limited computational resources, we obtain a better policy with a more descriptive final cost.

To further characterize a desirable final cost function, recall that MPC is a trajectory-based method of approximating and computing a solution to the Bellman equation

$$\min_{\mathbf{u}} l(\mathbf{x}, \mathbf{u}) + V(f(\mathbf{x}, \mathbf{u})) \tag{2}$$

where $V(f(\mathbf{x}, \mathbf{u})$ is the value function evaluated at the state $(\mathbf{x}, \mathbf{u})$ under the dynamics f and $l(\mathbf{x}, \mathbf{u})$ is the one-step cost function.

Drawing parallels between the Bellman equation and the MPC framework, we notice that the final cost in MPC $v_F(\mathbf{x}_{i+N})$ is the optimal value function for an infinite-horizon problem evaluated at the last state in the finite horizon trajectory. So, in the next sections we describe the results of employing two different methods of obtaining an approximation to the Bellman-defined value function. We will give these approximations to MPC as a final cost, and if the approximation is accurate (or even if it is inaccurate but useful information is still attainable), we will be able to achieve our aim of reducing the need for a long horizon for MPC.

A second aspect of trajectory-based MPC which makes it fast besides solving the local problem to only a finite horizon is recognizing that, if we only employ the first control of a computed optimal policy before recomputing a new trajectory from our new location, we expect that the optimal policy has not changed drastically from that of the previous time step. Thus, in line 2 of the MPC algorithm the entirety of the optimal policy computed at the last time step is used to start the search for the optimal policy at the new position. This "warm start" of the policy search permits real-time solution at every time step. However, it may constrain the solver to a local minimum; it does not permit the system to look for solutions that are perhaps better but which are too different from the solution already has in hand. Furthermore, in practice the online optimizer may not even converge to a local minimum because computational resources limit us to a single quasi-Newton optimization step, making good initialization even more imperative.

B. Data-driven value function approximation

1) Overview: First, we try to exploit the hard work of model predictive control simulations to make a global approximation to the optimal value function. As the iLQG-based MPC controller marches forward in time continually hypothesizing about optimal trajectories an avalanche of data is being generated. This data includes not only information about the state, but also a locally second-order approximation to the system's value function is generated at each visited state along with the gradient and Hessian of the value function. We use machine learning to fit a global value function to this data.

2) Models for function approximation: With all function approximation methods, there are many parameters that can be adjusted to achieve good fit. In broad terms, these typically include quantity, spacing, quality of the data, how and when the data is normalized, where the basis functions are placed, the spread of the basis functions, the cutoff, and the criterion defining what constitutes a good fit which ends the learning process. There are an infinite combination of these parameters, and while most methods of learning are adaptive and boast some provable convergence criteria, in practice they are highly sensitive. It is always important to check the fit on training data and on test data (on which the method was not trained), but even so, a good fit is commonly elusive. Rather than discuss further the general complications of fitting, which are well-known, we will provide just a couple examples of typical behavior of the function approximations we have chosen to use.

The function approximation models we employed for this work were

- Nearest neighbor
- Locally weighted projection regression
- Polynomial mixture of normalized Gaussians

The nearest neighbor is a global approximation that simply copies all the value function information (the value, the gradient, and the Hessian) from the closest neighbor. The closest neighbor is, here, defined as the data point with smallest Euclidean distance to the queried state after all data and the state of interest have been normalized. Nearest neighbor is, of course, very simple to implement, has no parameters besides the norm in which the nearest neighbor is calculated, and can be used effectively for quick base-line comparisons.

Locally weighted projection regression (LWPR) is a non-linear function approximation method which is applicable in high-dimensional spaces. The core is locally linear models spanned by a relatively small number of univariate regressions in directions of the input state-space selected with a weighted partial least-squares algorithm. LWPR learns rapidly and automatically updates the distance metrics of each local model using statistically significant stochastic cross-validation. LWPR provides benefit over many other techniques when the function to be learned is highly-nonlinear, when large amounts of data is available, and it is even very successful when the data is high-dimensional with redundant dimensions. Most control problems of interest are high-dimensional (though we explore basic behaviors in low-dimensional systems in this paper), so a method that scales well with dimension is valuable. Also, we know that the value function is very nonlinear, and we certainly have large amounts of data with which to train a model, so LWPR is a reasonable candidate for value function approximation. (It is worth nothing that the results presented here are not necessarily the best to showcase the strengths of LWPR. Namely, we use low-dimensional systems where other approximation methods work well, and we generate an approximation to the value function completely off-line.

LWPR is well-suited to on-line, adaptive approximations, as the update of linear weights can be done more efficiently than update of other models.) We use the MATLAB implementation provided by Klanke and Vijayakumar [13].

The third function approximation model is a mixture of Gaussians each weighted by a polynomial. Here the center of each normalized Gaussian basis function is determined through an expectation maximization procedure in an unsupervised learning step, and then coefficients of the polynomial mixture of the Gaussians are learned in a supervised fashion. The properties of a Gaussian mixture model are well-studied, and it is suitable for problems of moderate dimension.

3) Obtaining data: As previously mentioned, the data for the value function approximation was taken from direct MPC simulation. For each system data was generated by starting from a random initial pose also with randomly generated velocities. These initial states were chosen to sample from a variety of locations in the possible state space. All joint limits were respected, and the velocities were selected to be in ranges which had been seen in typical MPC-only simulation. The simulations were then run for a fixed simulation period. The time of simulation was sufficient for the system to achieve the goal state for most initial conditions.

After simulation, the trajectories were pruned to remove failed runs, transient behaviors, and other "suspicious" activity as indicated by outlying states or large regularization in the iLQG solver.

The main difficulties in obtaining data arose in defining useful initial states. The function approximation needs to be very accurate around the goal state, thus much data is required in that area. This does not present a problem as all simulations were run to the goal state so every simulated trajectory "saw" this region of interest. On the other hand, the approximation needs to provide valuable information to MPC especially when far away from the goal state, especially if, as was our experience, the value function approximation leads MPC out of its typical realms of operation (ostensibly in search of a better trajectory). As is common experience, to well-cover a large, high-dimensional state space in data is non-trivial.

C. Using a LMDP solution for value function approximation

1) Linearly-solvable MDPs: Linearly-solvable optimal control problems are a special class of optimal control problem that linearizes the Bellman equation. A Markov Decision Process is said to be linearized if the following conditions hold: (1) both the passive dynamics and the control policy are expressed as distributions, and they lie in the same subspace/subset, and (2) the cost function is defined by the KL divergence [14].

The linear MDP solves the optimality problem for the desirability function $z(\mathbf{x}) = \exp(-V(\mathbf{x}))$. In the MDP, the passive dynamics are given by $\mathbf{x}' \sim p(\cdot|\mathbf{x})$, and the running cost is defined in terms of distributions as follows.

$$l(\mathbf{x}, \mathbf{u}(\cdot|\mathbf{x})) = q(\mathbf{x}) + KL(\mathbf{u}(\cdot|\mathbf{x})||p(\cdot|\mathbf{x})) \tag{3}$$

where q is the state cost KL denotes the KL divergence representing control cost.

The minimized Bellman equation, linear in the desirability function, is given by

$$\exp\left(q(\mathbf{x})\right) z(\mathbf{x}) = \mathcal{G}[z](\mathbf{x}), \tag{4}$$

where

$$\mathcal{G}[z](\mathbf{x}) = \int_{\mathbf{x}' \in X} p(\mathbf{x}'|\mathbf{x}) z(\mathbf{x}') d\mathbf{x}', \tag{5}$$

is the next-state expectation under p. The optimal control policy is obtainable from $z(\mathbf{x})$ by

$$\mathbf{u}^*(\mathbf{x}'|\mathbf{x}) = \frac{p(\mathbf{x}'|\mathbf{x}) z(\mathbf{x}')}{\mathcal{G}[z](\mathbf{x})}. \tag{6}$$

In an average cost setting, the desirability function is the principal eigenfunction of $\exp\left(-\tilde{q}(\mathbf{x})\right) \mathcal{G}[z](\mathbf{x})$ and is guaranteed to be positive. For the first-exit formulation the value function is fixed at terminal states resulting in a linear equation.

2) LMDP approximation schemes for continuous state spaces: We have developed several numerical methods [15], [16], [17] to calculate the value function and the control policy in a continuous state space without losing the LMDP's linearity and efficiency. Among them is the aggregation method [17] which relates the continuous state space to discrete samples and solves the problem as a discrete state LMDP. The dynamics are assumed to be in the form:

$$d\mathbf{x} = \mathbf{a}(\mathbf{x})dt + B(\mathbf{x})(\mathbf{u}dt + \sigma d\omega), \tag{7}$$

where $\omega(t)$ represents Brownian motion, and σ is the noise magnitude. The cost function is in the form

$$l(\mathbf{x}, \mathbf{u}) = q(\mathbf{x}) + \frac{1}{2\sigma^2}\|\mathbf{u}\|^2. \tag{8}$$

The soft aggregation method defines the following two transition probability-like quantities to transform a LMDP on the original space ($\mathbf{x} \in \mathbb{R}^n$) to discrete clusters ($i = 1, 2...$).

1) Aggregation probability: $p(i|\mathbf{x}) = \phi_i(\mathbf{x})$, the "degree of membership of $\mathbf{x}$ in the aggregate state i"
2) Deaggregation probability: $p(\mathbf{x}|i) = d_i(\mathbf{x})$ the "degree to which i is represented by $\mathbf{x}$"

The desirability function is defined by deaggregation probabilities. Thus, the aggregation technique yields an approximation to the value function which is based on the logarithm of normalized Gaussians. Note that there are no polynomial factors in this approximation of the value functions which makes the function nearly a step function. A probability-like optimal control policy $\mathbf{u}^*(\mathbf{x}'|\mathbf{x})$ is obtained from (6) and its mean is used to recover the control signal $\mathbf{u}^*(\mathbf{x})$ through inverse dynamics. Note though that the LMDP may give instructions $\mathbf{u}^*(\mathbf{x}'|\mathbf{x})$ to go to somewhere unphysical.

We obtain $\mathbf{u}^*(\mathbf{x})$ in a discrete state space and, using the cluster positions, we are able to define desirable locations for the continuous system. In MPC, we set the final cost to $v_F(\mathbf{x}) = -\log(z(\mathbf{x}))$ and modify the MPC warm start of policy to the LMDP's control policy $\mathbf{u}^{init}_{i+N-1} = \mathbf{u}^*(\mathbf{x}^{init}_{i+N-1})$.

Also, the probabilistic framework introduces noise that is not present in MPC, so we are not approximating the value function of MPC. However, this discrepancy is irrelevant if the supplied final cost function helps nudge the optimizer in the right direction.

III. Numerical Results

In this section, we present the results of using the two different value function approximation methods with MPC as described above on simulated systems. We demonstrate our methods on the classic control examples; the inverted pendulum, the double pendulum and the acrobot.

The simplest system we investigate is the inverted pendulum. The goal is to swing a one-link pendulum up to the unstable vertical equilibrium and maintain stability. The cost of control is high, so that building momentum through swinging is required. This system has two degrees of freedom in the state (the angle of the pendulum and the angular velocity), and one control degree of freedom at the pivot point.

One level of difficulty up from the one-link pendulum is the double pendulum. This is a two link pendulum in which both joints ("shoulder" and "elbow") are actuated. The acrobot system is obtained by simply removing control at the shoulder such that the system is now underactuated. These systems both have four degrees of freedom in the state space (the two angles and the two angular velocities). The double pendulum has two controls and, as mentioned, the acrobot has only one. Here again, the goal is to swing both links up to the vertical position such that the pendulum is fully extended upward and stabilize.

In all the experiments, the state transitions are obtained by integrating the equations of motion. The starting states for simulation are randomly initialized within the limits of the positional degrees of freedom and with velocities on the order of those seen in unmodified MPC simulations.

A. Data-driven value function approximation

First, we present the results of using a value function learned from MPC simulation data. We start with the simplest system, the inverted pendulum, and then work toward more difficult problems.

1) Inverted pendulum: If we wish to use value function approximation to improve MPC, it is important to carefully consider and understand the properties of the chosen function approximation models and how these properties might translate into their usage in conjunction with MPC. We start with examining a two degree of freedom system as it permits full visualization of the state space and can give instruction about what problems to look out for with larger systems.

Figure 1a shows a fit by locally weighted projection regression. It was trained with 14600 data points. The parameters had to be set so that there was a large number of receptive fields (basis functions), specifically 451, to get this resolution in just a 2D problem. This model achieved less than 1% error on the training data though, and we can see it accurately captures both the linear feature in the middle of the state space and the periodic condition on the boundary when compared with the nearest neighbor approximation of figure 1b. One final thing to note in this model, is that while the value of the optimal-value function is nearly zero (-0.0025) at the goal state of $(0, 0)$, the gradient is not ($[-0.0299 - 0.0086]$).

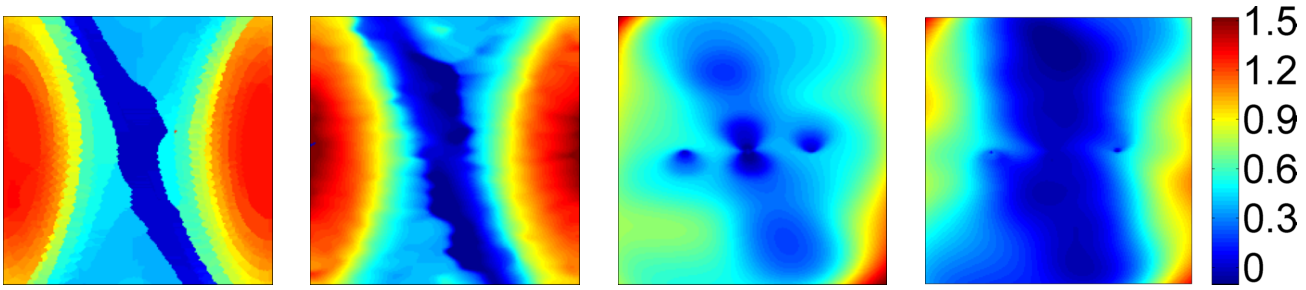
We contrast this behavior with a polynomial mixture of Gaussians model. A mixture of Gaussians with a first order polynomial is seen in figure 1c and a mixture with a second order polynomial is seen in figure 1d. Both figures typify a mixture of Gaussians, in that we see very circular structures, and it is not able to well-resolve the linear feature. In addition, training and storing a normalized mixture of Gaussian model with polynomials does not exploit any dimension reduction as LWPR does, so high order polynomials and training with as many data points as a LWPR model can be trained with is intractable.

With a learned model of the global value function in hand, we now investigate using it as a final cost in MPC. To do so, we choose random initial poses, and then allow MPC to run for a simulation time of 2 seconds. At the end of the simulation, the running cost at every state visited is summed over the complete trajectory. A lower cumulative running cost implies a more optimal policy. We simulate the four different methods; 1) MPC without any final cost, 2) MPC with nearest neighbor value function as final cost, 3) MPC with Locally Weighted Projection Regression (LWPR) as final cost, and 4) MPC with second order polynomial mixture of normalized Gaussians (polyMoG) as final cost. For each method, we vary the length of the MPC horizon between 4ms and 1.5s.

Figure 2a shows the percentage of trials in which each of the methods utilizing function approximation are able to achieve a lower cumulative cost than MPC alone (comparing all methods at the same MPC horizon). All the methods are able to achieve a lower running cost nearly all of the time at nearly all the horizons. Only the polynomial model performs poorly with short MPC horizon, but even so it is able to out-perform pure MPC at a horizon half the length of the original.

These results regarding how MPC performs with the value function approximation give a glimpse at an important piece of the puzzle in using these methods. That is the quality of the value function approximation. The quality of the LWPR and nearest neighbor models is very good, while the quality of the polyMoG model is relatively poor. We see that a highly accurate approximation seems to benefit us greatly, but it also appears that a poor approximation does not necessarily mean we cannot achieve some improvement. As we move to higher dimensional systems, achieving the level of accuracy of the value function that we obtained on this two dimensional problem is going to be practically impossible.

2) Double pendulum and acrobot: Assured that the general method works for a small problem, we next move to the fully-actuated double pendulum example. After obtaining a global value function approximation with our three techniques, we wish to give further insight into what information the approximation of the value function is able to capture. We note that while there are certainly point-wise errors in the approximation of the value function, a small relative error is



(a) LWPR (b) Nearest Neighbor (c) PolyMoG order 1 (d) PolyMoG order 2

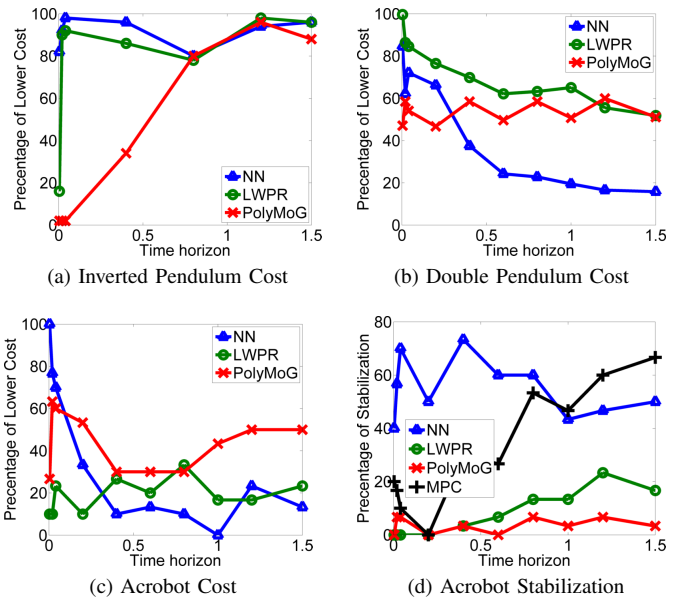
Fig. 1. Function approximation of inverted pendulum problem

achievable; indeed, not matching the data exactly is important. We wish for the approximate value function to help MPC avoid tracking toward local minima. As such, the approximation needs to have a smoother structure globally than the direct data yields. However, this global smoothness also hinders the method.

The norm of the gradient of the value function is not well-captured. The gradients are much smaller at certain critical states than the data indicates they should be; i.e., the approximation is smoother than the true value function. This smoothing of the value function prevents the method from being able to make aggressive control corrections. To make matters worse, there is a problem on the reverse side, too! That is, the gradient of the approximated function is frequently non-trivial even at states where they should be small or zero, namely near the goal state; LWPR is especially culpable in this matter.

We next examine the behavior of MPC when a global value function is used as a final cost. Figure 2b shows the percentage of trials in which each of the methods utilizing function approximation are able to achieve a lower cumulative cost than MPC alone. The LWPR model is able to achieve a lower final cost nearly always with short horizons; nearest neighbor also performs well in this regime. Their performance degrades somewhat with longer horizons, but is still able to achieve lower running cost than MPC more than half of the time. The polyMoG model does not show as much advantage at short horizons, but it is consistent throughout all horizons.

The results of the inverted pendulum and the fully-actuated double pendulum encourages us to further increase the difficulty of the control problem. Removing actuation at the shoulder to obtain the acrobot, we apply the same method and compare the results. Figure 2c shows the plots of the percentage of trials incurring lower running cost, and figure 2d shows the percentage of trials in which the various methods were able to stabilize at the goal state. On this problem, the nearest neighbor is the only method that performs better than MPC alone with high consistency and this is only at very short MPC horizons. The polynomial mixture of Gaussians approximation is not necessarily worse in terms of cost, but does not provide the benefit we hope to see.



(a) Inverted Pendulum Cost (b) Double Pendulum Cost

(c) Acrobot Cost (d) Acrobot Stabilization

Fig. 2. (a-c) Percentage of trials in which MPC+value function approximation incurred a lower running cost than MPC alone; (d) Percentage of acrobot trials able to stabilize

B. LMDP solution for value function approximation

The results of the previous section indicate that data-driven methods of approximating the value function, while in theory they work on any system, are limited by the learning process in high-dimensional systems with complicated value function topology. We now turn to using the LMDP solution for value function approximation to find a method that works well for more difficult systems.

1) Aggregation methods: Solving the LMDP by aggregation methods as described previously, we obtain a global value function approximation. However, using this value function as a final cost fails to provide successful results in achieving either swing up or stabilization in the underactuated acrobot. Recall that in solving the LMDP we also obtain the LMDP optimal policy. Warm starting MPC with this control policy and not providing a final cost also fails to improve MPC. Thus neither of the ingredients provided by the global method is able to help MPC. However, somewhat contrary to intuition, when

both the control policy and the value function are provided to MPC, the required horizon of planning for MPC can be reduced while still achieving the control tasks.

Figure 3b shows the percentage of 100 trials (initializing simulation from a random initial state) in which the controller is able to swing the pendulum to vertical and in which the controller is able to stabilize the pendulum after swing-up. In the acrobot problem, we need a 800 ms time horizon to swing up *and* stabilize the acrobot. This is in a sharp contrast contrast to the MPC method itself, which needs a 1500 ms horizon to obtain similar results on both problems. Thus using both the value function and the control policy derived from the LMDP is an effective method to shrink the horizon MPC needed. The LMDP solution helps MPC quickly discover the target, although a too short horizon limits the hybrid method's ability to stabilize the system. On the other hand, when the horizon is too long, the hybrid method often fails to achieve the task before the simulation is terminated due to the decaying effect of the policy warm-start. An example trial from each regime is depicted in Fig. 4.

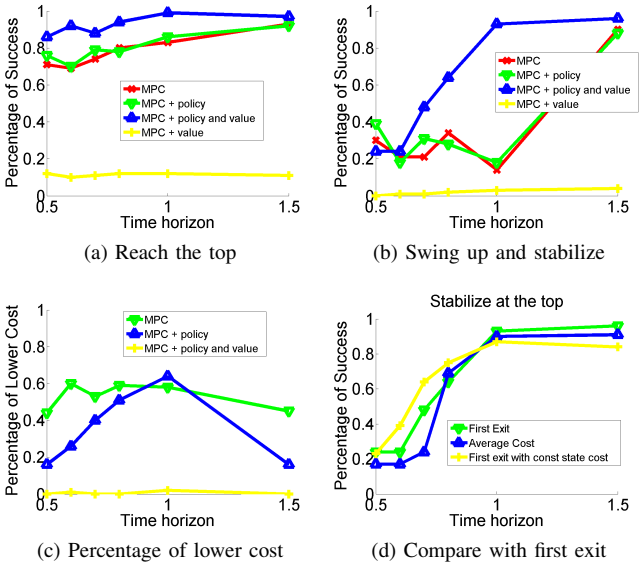
Compared to the data-driven approach, this combination does not generally lower the accumulated cost since the LMDP is actually solving a different optimal control problem. Yet the success rates show that the combination of LMDP and MPC is helpful in solving practical problems.

These results were obtained solving for the value function on a regular grid with approximately 10,000 bases in a 4D space; such grids are too coarse for aggregation methods alone to be effective. The computation time of the LMDP is around 10 seconds with 6 threads and utilizing the Mujoco solver [18]. The solution of the LMDP is done off-line. Extracting the control policy and the value function on-line after the LMDP solve is cheap given proper implementation, so MPC is made more efficient by the hybrid method.

The efficacy of the hybrid method is more noticeable still on single pendulum and double pendulum problems. For the double pendulum, a 100ms time horizon in MPC is all that is needed in the LMDP-MPC hybrid method in contrast to MPC alone which requires a 1500 ms horizon. Given success on the underactuated problem, this is not surprise, but we cannot provide more details due to page limitations.

2) First exit formulation: The flexibility of the LMDP formulation of the optimal control problem permits us to solve for a value function and control policy under a first exit formulation as well as in the average cost formulation we used previously. Interestingly, we find that in the first exit formulation, the aggregation method may be used to generate an acceptable control policy even when the state costs are not finely-tuned to describe the control problem. That is, the solution is foremost designed to obtain the goal state, and the state cost is of secondary concern. This approach puts less of a burden on the control engineer.

Figure 3d presents results in which we controlled the acrobot by solving the LMDP under the first exit formulation with the upward position assigned as the target. In the first case, the LMDP defined by a constant state cost and very



(a) Reach the top (b) Swing up and stabilize

(c) Percentage of lower cost (d) Compare with first exit

Fig. 3. Results of acrobot trials. (a) Success rate of swing up. (b) Success rate of swing and stabilization. (c) Percentage of lower running cost. (d) Success rate of swing up and stabilization of the acrobot under different LMDP formulations.

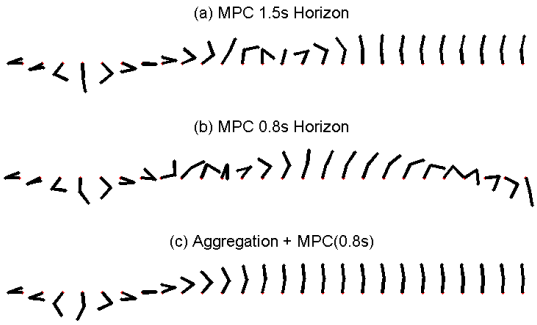


Fig. 4. Demonstration of the actual movement of acrobot under different approaches.

small control cost is solved; such a combination would yield a practically uncontrollable system under other formulations. In the other case, we again solve the LMDP in the first exit formulation, but we retain the well-tuned state cost. The percentage of trials in which swing up and stabilization are achieved is nearly equivalent to the case of the solution of the LMDP in an average cost formulation, but the first exit formulation is not reliant on a hand-tuned state cost.

3) Resolution of the value function: By using more bases in the LMDP solver the accuracy of the value function will be increased. However, for the acrobot, we can still reduce the MPC planning horizon with as few as 3000 bases, and the addition of more bases does not significantly decrease the needed horizon. It is particularly encouraging that the computational expense of a highly accurate LMDP solution is not required to provide benefit to MPC and that we can shorten the horizon needed for MPC quite dramatically using a global solution which is too inaccurate to be used by itself.

IV. Conclusions and Future Work

We finish with some broad observations from our current work and indicate the direction of our future efforts.

A. Shortening MPC's horizon automatically

The planning horizon required by MPC for effective control of a system is highly dependent upon the descriptiveness of the final cost function and on the method used for policy warm-start. Both our global methods for defining a final cost imply that an automatically synthesized final cost may permit shortening of the planning horizon. However, in the case of a value function learned from data, we have only achieved good results on very simple problems in spite of a great deal of experimentation. Better methods of function approximation in the highly-nonlinear and difficult high-dimensional spaces of complex systems need to be used to approximate the value function if any benefit to MPC is to be seen. Our LMDP results show that exploiting both a global value function and optimal policy is required to provide substantial benefit to MPC. All our results together show some promise of successful exploitation of global methods in on-line MPC, but more scaling to higher-dimensional systems remains to be demonstrated.

B. Expanding global methods

In the previous sections, we have described how we used global methods to improve MPC. We note that this is only half the story. Indeed, equally valuable is the improvement made to the global LMDP method by adding MPC to its solution. In our results, we obtain only a very rough estimate of cost-to-go function which is far from the accuracy required to achieve the desired control task. Yet, even a coarse global solution still contains information about the solution of the optimal control problem. On-line adaptation can exploit this helpful yet inaccurate information. Instead of increasing the accuracy of a global method solution by introducing more basis functions or neurons, it may be more practical to include on-line correction.

On-line planning can also expand a global method's applicability in real-time applications. First, the prediction generated by planning provides a last minute safety check of the control policy. Second, it decreases the demand on the limited computational power installed on robots. Rather than storing and computing solutions to the global Bellman equation locally on robots, we have the freedom to compute them on a remote server and send only the planned trajectories.

C. Future work

In the future, we plan to explore more ways to exploit other global methods in generating final costs and policies to assist MPC. In particular, we will seek to understand the hidden information in a coarse or inaccurate approximation of the optimal value function, especially in the linearly-solvable framework.

Acknowledgments

The authors would like to thank Dvijotham Krishnamurthy for helpful discussions. This work was supported by the US National Science Foundation.

References

[1] Y. Tassa, T. Erez, and W. Smart, "Receding horizon differential dynamic programming," *Advances in Neural Information Processing Systems*, vol. 20, pp. 1465–1472, 2008.

[2] M. Diehl, H. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear mpc and moving horizon estimation," *Nonlinear Model Predictive Control*, p. 391, 2009.

[3] C. Atkeson *et al.*, "Using local trajectory optimizers to speed up global optimization in dynamic programming," *Advances in neural information processing systems*, pp. 663–663, 1994.

[4] C. Atkeson and J. Morimoto, "Nonparametric representation of policies and value functions: A trajectory-based approach," *Advances in neural information processing systems*, p. 16431650, 2003.

[5] A. Barto, S. Bradtke, and S. Singh, "Learning to act using real-time dynamic programming," *Artificial Intelligence*, vol. 72, no. 1, pp. 81–138, 1995.

[6] S. Gelly and D. Silver, "Combining online and offline knowledge in uct," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 273–280.

[7] M. Fink, "Online learning of search heuristics," in *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS 2007)*, 2007, pp. 114–122.

[8] H. Chen and F. Allgöwer, "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability," *Automatica*, vol. 34, no. 10, pp. 1205–1217, 1998.

[9] B. Hu and A. Linnemann, "Toward infinite-horizon optimality in nonlinear model predictive control," *Automatic Control, IEEE Transactions on*, vol. 47, no. 4, pp. 679–682, 2002.

[10] T. Erez, Y. Tassa, and E. Todorov, "Infinite horizon model predictive control for nonlinear periodic tasks," *Manuscript under review*, 2011.

[11] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization." IROS, 2012.

[12] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proceedings of the 2005, American Control Conference, 2005.*, Portland, OR, USA, 2005, pp. 300–306.

[13] S. Vijayakumar, A. D'souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural Computation*, vol. 17, no. 12, pp. 2602–2634, 2005.

[14] E. Todorov, "Efficient computation of optimal actions," *Proceedings of the national academy of sciences*, vol. 106, no. 28, pp. 11 478–11 483, 2009.

[15] ——, "Eigenfunction approximation methods for linearly-solvable optimal control problems," in *Adaptive Dynamic Programming and Reinforcement Learning, 2009. ADPRL'09. IEEE Symposium on*. IEEE, 2009, pp. 161–168.

[16] M. Zhong and E. Todorov, "Moving least-squares approximations for linearly-solvable stochastic optimal control problems," *Journal of Control Theory and Applications*, vol. 9, no. 3, pp. 451–463, 2011.

[17] ——, "Aggregation methods for lineary-solvable markov decision process," in *World Congress of the International Federation of Automatic Control (IFAC)*, 2011.

[18] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," *Under Review*.