Control by Gradient Collocation: Applications to Optimal Obstacle Avoidance and Minimum Torque Control

Paul Ruvolo, Tingfan Wu, and Javier R. Movellan
Machine Perception Laboratory
University of California, San Diego
La Jolla CA, USA

***Abstract*— We present a new machine learning algorithm for learning optimal feedback control policies to guide a robot to a goal in the presence of obstacles. Our method works by first reducing the problem of obstacle avoidance to a continuous state, action, and time control problem, and then uses efficient collocation methods to solve for an optimal feedback control policy. This formulation of the obstacle avoidance problem improves over standard approaches, such as potential field methods, by being resistant to local minima, allowing for moving obstacles, handling stochastic systems, and computing feedback control strategies that take into account the robot's (possibly non-linear) dynamics. In addition to contributing a new method for obstacle avoidance, our work contributes to the state-of-the-art in collocation methods for non-linear stochastic optimal control problems in two important ways: (1) we show that taking into account local gradient and second-order derivative information of the optimal value function at the collocation points allows us to exploit knowledge of the derivative information about the system dynamics, and (2) we show that computational savings can be achieved by directly fitting the gradient of the optimal value function rather than the optimal value function itself. We validate our approach on three problems: non-convex obstacle avoidance of a point-mass robot, obstacle avoidance for a 2 degree of freedom robotic manipulator, and optimal control of a non-linear dynamical system.**

I. INTRODUCTION

The problem of computing optimal controllers to guide a robot from an initial configuration to a goal in the presence of obstacles has been widely studied in the robotics literature. One popular approach for solving this problem is based on constructing artificial potential functions [5], [11]. These approaches work by placing a repulsive potential around obstacles in the robot's environment and a basin of attraction around the goal. The control signal for the robot at each point in time is proportional to the negative gradient of the potential function at the current point. While this approach is computationally lightweight, it suffers from several important drawbacks: including both a lack of a clearly defined notion of optimality and the possibility for the robot to get stuck in local minima. While previous approaches [2], [11] have sought to address the local minima problem in various ways, these approaches are not applicable to the general obstacle avoidance problem; focusing instead on special cases (see Section VI for a more thorough discussion of the existing literature). Here we show that with an appropriate function approximation scheme, collocation approaches designed to compute optimal controllers for nonlinear systems [12], [3], [1], [15], [14] can effectively solve the obstacle avoidance problem.

In Section III we describe our method for converting the obstacle avoidance problem into the problem of optimally controlling a nonlinear diffusion in continuous state, action, and time. The main idea is to first specify an objective (or reward) function that penalizes the robot for intersecting obstacles and rewards the robot for reaching the goal. Secondly, we use collocation methods to convert this myopic reward function into a control policy that maximizes reward over the long term (and thereby reaches the goal while avoiding obstacles).

Recently, there has been a surge of interest in collocation methods for solving continuous state and action control problems [12], [3], [1], [15], [14]. These algorithms work by computing an approximate solution to the Hamilton Jacobi Bellman (HJB) Equation (a sufficient condition for deriving an optimal control policy) at a finite set of states. One of the key contributions of this work is to propose a function approximation architecture that is well-suited to the obstacle avoidance problem with its characteristic multimodal reward functions (where the modes are due to the presence of obstacles). In addition, our method modifies the typical optimization criterion for collocation methods to focus on satisfying the gradient and second-derivative of the HJB equation rather than the HJB equation itself. Due to the choice of this particular optimization criterion, we call our approach *Gradient Collocation* (GC). This change allows the user more control over the parameterization of the space of control laws considered by the algorithm, focuses the optimization on producing good policies, and also comes with a computational savings under certain conditions. In Section VII we provide experimental validation for our proposed method on three problems: obstacle avoidance for a non-convex obstacle, obstacle avoidance for a robotic manipulator, and control of a non-linear dynamical system.

II. STATEMENT OF THE OPTIMAL CONTROL PROBLEM

Most processes of interest in robotics can be modeled as a controlled stochastic differential equation of the following

form

$$dX_t = a(X_t)dt + b(X_t)U_t dt + c(X_t)dB_t \quad (1)$$
$$U_t = \pi(X_t, t) \quad (2)$$

where $X_t \in \mathbb{R}^d$ is a random vector specifying the state of the system (typically joint angles and angular velocities), $a(X_t)$ represents the uncontrolled deterministic dynamics (a combination of inertial matrix, centrifugal-Coriolis matrix, and gravitational/viscous torques), $b(X_t)$ is the transpose of the Kinematic Jacobian matrix, $U_t \in \mathbb{R}^m$ is the control signal, $c(X_t)$ is a matrix controlling the amount of noise in the system dynamics, dB_t is a Brownian motion differential, and π is a deterministic feedback controller that maps states and times into control signals.

First, we propose a method to solve the finite horizon stochastic optimal control problem, and later (see Section III) we show how this formulation can be used to solve the obstacle avoidance problem. The value of a feedback controller π takes the following form

$$v_t(x \mid \pi) = \mathbb{E}\Big[\int_t^T e^{-\frac{1}{\tau}(s-t)} r_t(X_s, U_s) ds + e^{-\frac{1}{\tau}(T-t)} g_T(X_T) \;\Big|\; \pi \Big] \quad (3)$$

where $v_t(x|\pi) \in \mathbb{R}$ is the value of starting in state x at time t when controlling a robot with controller π, $\tau > 0$ is a known time discount rate (which can be set to ∞ if no discounting is desired), $r_t(X_t, U_t) \in \mathbb{R}$ is a known instantaneous reward rate, and $g_T(X_t) \in \mathbb{R}$ is a known terminal reward function.

Our goal is to find a feedback controller π^* that provides maximum value. It is well known that the value $v_t(x)$ achieved by the optimal controller satisfies the Hamilton-Jacobi-Bellman equation (HJB):

$$-\nabla_t v_t(x) = \max_u \Big\{ -\frac{1}{\tau} v_t(x) + r_t(x,u) + \big(a(x) + b(x)u\big)' \nabla_x v_t(x) + \frac{1}{2} \mathrm{Trace}\big[c(x)c(x)^\top \nabla_x^2 v_t(x)\big] \Big\} \quad (4)$$
$$v_T(x) = g_T(x) \quad (5)$$

where $\nabla_t v_t$, $\nabla_x v_t$ are the partial derivatives with respect to time and state, and ∇_x^2 is the Hessian matrix with respect to state. We make the simplifying assumption that the reward rate takes the following form:

$$r_t(x,u) = g_t(x) - \frac{1}{2} u' q_t u \quad (6)$$

where g_t is an arbitrary known function of the state and q_t is a known symmetric positive definite matrix. Since u appears only linearly and quadratically in Equation 4 the maximization over u can be computed analytically:

$$u_t = q_t^{-1} b(x)' \nabla_x v_t(x) \quad (7)$$

The form of the previous equation shows that if we could compute the gradient of the optimal value function for all states and times we could easily derive an optimal controller. This transforms the control problem to the more convenient problem of approximating the optimal value function. Substituting the optimal u_t back into the HJB equation we get:

$$-\nabla_t v_t(x) = -\frac{1}{\tau} v_t(x) + g_t(x) + \frac{1}{2} u_t(x)' q_t u_t(x) + a(x)' \nabla_x v_t(x) + \frac{1}{2} \mathrm{Trace}\big[c(x)c(x)^\top \nabla_x^2 v_t(x)\big] \quad (8)$$
$$u_t(x) = q_t^{-1} b(x)' \nabla_x v_t(x) \quad (9)$$
$$v_T(x) = g_T(x) \quad (10)$$

We have now removed the minimization operator from the HJB equation. Next, we show how the removal of the minimization operation allows us to more efficiently solve the HJB equation and in turn compute an optimal policy.

III. Reduction of Obstacle Avoidance to a Continuous Control Problem

Before discussing our method for solving the control problems presented in the previous section, we show how to formulate the problem of obstacle avoidance as a continuous state, action, and time control problem.

We assume that we are given a model of how the control signals for our robot probabilistically affect the state differential. That is we assume that we are given a, b, and c in Equation 1. Additionally, we assume that the quadratic penalty on the control signal, q_t, is given. In order to fully specify the control problem we also have to define the terminal reward $g_T(x)$ as well as the state reward rate $g_t(x)$. Given a desired end configuration for the robot $x^\star$ we define the terminal reward $g_T(x) = -(\psi(x) - x^\star)^\top \Lambda (\psi(x) - x^\star)$ where Λ is a given symmetric positive definite matrix that specifies the magnitude of the penalty for the robot not reaching the goal at the terminal time and ψ is a function that maps from the state space of the robot to a potentially different coordinate system (e.g. world coordinates). An example of the role that the ψ function could play would be to map from the joint angles of a robotic manipulator to the position of its end effector. In this case, the reward would be quadratic based on the distance in world coordinates between the end-effector and the goal. The usage of a quadratic penalty based on the distance to the goal is not required by our algorithm, however, we have found that a quadratic penalty worked well for our experiments.

We define the state reward rate to penalize the robot from contacting obstacles. We use a collection of weighted radial-basis functions to represent the location of obstacles in the environment. Similarly to the terminal reward function, the obstacles can be defined in a different coordinate system than the state-space of the robot. Specifically,

$$g_t(x) = \sum_{i=1}^{d} \frac{1}{z} \sum_{j=1}^{z} \alpha_i e^{-(\psi_j(x) - \mu_i)' \Sigma_i^{-1} (\psi_j(x) - \mu_i)} \quad (11)$$

where each α_i is a scalar specifying the penalty for contacting the obstacle, the functions $\psi_1 \ldots \psi_z$ specify the center of

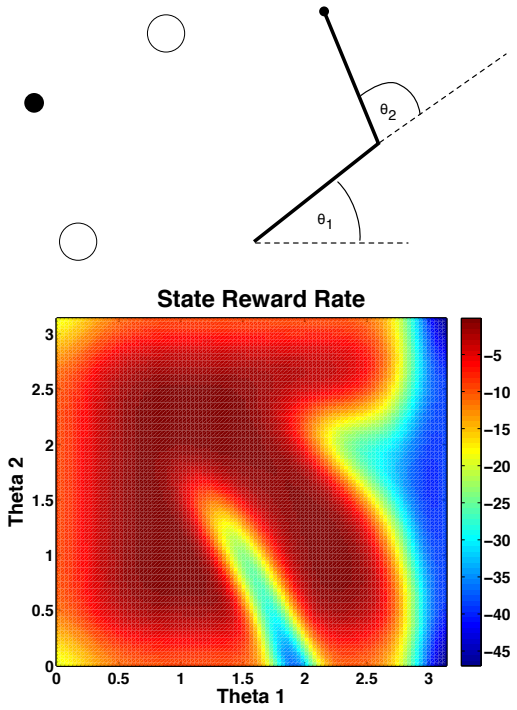

Fig. 1. **Top:** a schematic of a two degree of freedom planar robot arm with two obstacles (white circles) and one goal position (black filled circle). **Bottom:** the reward rate constructed to formulate the given obstacle avoidance as a control problem. The reward function is created by penalizing the robot based on the proportion of arm segments that are in contact with the obstacle at a particular configuration of joint angles.

each of the z segments of the robot in the coordinate system of the obstacles, and the sum over j from 1 to z approximates the proportion of overlap of the robot with the obstacles by evaluating the overlap at the center of each of the z segments. For example, in the two degree of freedom robot manipulator shown in Figure 1 the state of the robot is given by the two joint angles θ_1 and θ_2, whereas, the positions of the obstacles are given in world coordinates. In order to compute the reward of a particular joint angle configuration of the robot, we compute the penalty for overlap between the center of each segment (where the segments are small divisions of the two rigid bodies that compromise the manipulator) of the robot arm in a particular configuration (θ_1, θ_2) with the obstacles. If the robot has joint limits, then these are also enforced by virtual obstacles at the edges of the robot's range of motion.

The state reward rate for the manipulator in Figure 1 is shown in the bottom panel. Note that even though the obstacles are represented as gaussian bumps in the world coordinate system, the induced state reward rate is much more complex due to the interaction between the geometry of the robot arm and the positions of the obstacles. The choice of the number of segments m to use to detect potential collisions with the obstacles is up to the system designer (more segments will result in more accurate collision detection).

IV. Collocation for Computing an Approximately Optimal Controller

Given a description of an obstacle avoidance problem as a controlled stochastic diffusion we seek to compute an optimal controller by finding a solution to the HJB equation, i.e., to find a value function v that satisfies it for all states and times. However, solving the HJB exactly is not computationally tractable except for a limited number of special cases.

One method for obtaining an approximate solution to the HJB is to use collocation methods in which we select a value function from some parameterized family of functions that satisfies the HJB as closely as possible (in the least squares sense) for a finite set of states and times. In the current work we specify the collocation points using a uniform grid over a given region of the state space. Alternatively, iterative approaches to selecting the collocation points and solving for an optimal control law are also possible and are compatible with our approach (for example see [15]).

Once we have the collocation points, we then define an objective function for selecting a candidate value function from some parameterized family. In addition we discretize the HJB in time using a Backwards Euler approach on a finite set of time points $\mathcal{T} = \{t_1 \cdots t_n\}$ such that $T = t_n \leq t_{n-1} \leq \cdots \leq t_1 = 0$. The parameters w_t of the optimal value function at each point in time t are computed using a recursive backwards pass that involves solving an optimization problem for each time step in $\mathcal{T}$. The objective function minimized in [12], [3], [1], [15], [14] is the squared difference between the the lefthand and righthand side of the HJB equation. For the terminal time T we find the parameters w_T by minimizing the sum of squared differences between the terminal reward and the value function estimate at the set of collocation points, x_T, at time T:

$$w_T = \underset{w}{\operatorname{argmin}} \sum_{x \in x_T} \Big(g_T(x) - v_T(x|w) \Big)^2 \tag{12}$$

For other time steps $t \in \mathcal{T}$ we assume we have already computed the parameters of the value function at the previous time step $s \in \mathcal{T}$, $s > t$. We then find the value of w_t by minimizing the sum of squared differences between the left and right side of the HJB equation at the set of collocation points x_t.

$$\begin{aligned} w_t = \underset{w}{\operatorname{argmin}} \sum_{x \in x_t} \Big(& \frac{v_s(x \mid w_s) - v_t(x \mid w)}{s - t} \\ & - \frac{1}{\tau} v_s(x \mid w_s) + g_s(x) \\ & + \frac{1}{2} u_s(x \mid w_s)' q_s u_s(x \mid w_s) + a(x)' \nabla_x v_s(x \mid w_s) \\ & + \frac{1}{2} \operatorname{Trace}\big[c(x) c(x)' \nabla_x^2 v_s(x \mid w_s) \big] \Big)^2 \end{aligned} \tag{13}$$

Another objective function for selecting a value function that we propose for the first time in this work, is to satisfy the gradient and second-order derivatives of the HJB as closely as possible. The motivation for this alternate optimization

criterion is twofold: (1) it allows us to use information about the first and second derivatives of the system passive dynamics, controlled dynamics, and Brownian motion gains at the collocation points, (2) it focuses the optimization procedure on accurately approximating derivative information of the value function which is what is needed for the computation of optimal actions (see Equation 7).

An additional benefit of enforcing gradient and second-order derivative information is that it is possible to parameterize the derivative of each dimension of the value function independently as opposed to having one parametric function for the entire value function. This creates d objective functions at each time step that can be optimized independently at reduced computational cost compared with fitting one approximator for the entire value function (see Section V for a discussion of this computational savings). This change creates the following objective for fitting the ith terminal gradient function:

$$w_{T,i} = \underset{w}{\operatorname{argmin}} \left\{ \sum_{x \in x_T} \left(\left(\frac{\partial}{\partial x_i} \left(g_T(x) - v_T(x|w) \right) \right)^2 + \sum_{j=1}^{d} \left(\frac{\partial^2}{\partial x_i \partial x_j} \left(g_T(x) - v_T(x|w) \right) \right)^2 \right) \right\} \tag{14}$$

The objective function for the ith gradient function at time t in the backward pass assuming the optimal weights w_s have already been computed at time s is given by:

$$w_{t,i} = \underset{w}{\operatorname{argmin}} \left\{ \sum_{x \in x_t} \left(\left(\frac{\partial}{\partial x_i} e_t(x, w, s, w_s) \right)^2 + \sum_{j=1}^{d} \left(\frac{\partial^2}{\partial x_i \partial x_j} e_t(x, w, s, w_s) \right)^2 \right) \right\} \tag{15}$$

$$\begin{aligned} e_t(x, w, s, w_s) = & \frac{v_s(x \mid w_s) - v_t(x \mid w)}{s - t} \\ & - \frac{1}{\tau} v_s(x \mid w_s) \\ & + \frac{1}{2} u_s(x \mid w_s)' q_s u_s(x \mid w_s) \\ & + a(x)' \nabla_x v_s(x \mid w_s) \\ & + \frac{1}{2} \text{Trace}\left[c(x) c(x)' \nabla_x^2 v_s(x \mid w_s) \right] \end{aligned} \tag{16}$$

Fitting independent function approximators for each dimension of the gradient does not always allow for efficient reconstruction of a corresponding value function (in fact there is no guarantee that the approximation of the gradient computed in this fashion has a well-defined antiderivative), however, the value function itself is never needed for computing the optimal actions for the robot at runtime. The reason for this is that the optimal control law given by Equation 7 only relies on the gradient of the value function. In this work we choose to directly parameterize the gradient of the value function and use the gradient and second-order derivatives of the HJB at the collocation points as our optimization criterion (see Equations 14-16). In our experience, the choice of this formulation not only leads to a more computationally efficient algorithm, but also to greater numerical stability when compared to standard collocation approaches. However, a detailed comparison of parameterizing the value function vs. the gradient of the value function is beyond the scope of this document.

We parameterize each dimension of the gradient of the value function at time $t \in \mathcal{T}$ as a linear combination of known basis functions,

$$\frac{\partial}{\partial x_i} v_t(x \mid w_t) = \alpha_t(x)' w_{t,i} \phi(x) \;\; \forall i \in \{1 \dots d\} \tag{17}$$

where $\phi(x) \in \mathbb{R}^p$ is a vector of known features designed to approximate $\frac{\partial v}{\partial x_i}$, $w_{t,i} \in \mathbb{R}^{o \times p}$ is a time-dependent matrix of weight parameters, and $\alpha_t(x) \in \mathbb{R}^o$ is a vector that encodes the relative influence of each row of $w_{t,i}$ on the partial derivative of the state, x. At a high-level the motivation for this choice is that $\phi(x)$ encodes the basis for an approximator of the value function gradient designed to be accurate in a local region of the state space, the rows of $w_{t,i}$ define the parameter weights for each of the o value function derivative approximators, and the locality vector α_t specifies the local region of influence of each of the o copies of the value function derivative approximator.

Specifically, we define α_t using the locations of a set of vectors $\mu_{t,1} \dots \mu_{t,o}$ and a kernel function, k. The i^{th} element of $\alpha_t(x)$ takes the following form:

$$(\alpha_t(x))_i = \frac{k(x, \mu_{t,i}, \sigma)}{\sum_{j=1}^{o} k(x, \mu_{t,j}, \sigma)} \tag{18}$$

$$k(x, \mu, \sigma) = \exp\{-(x - \mu)' \sigma (x - \mu)\} \tag{19}$$

Where σ is a fixed, symmetric positive definite matrix. Since σ is fixed, the value function gradient is determined by the vectors $\mu_{t,i}$ and by the weight matrices $w_{t,1} \dots w_{t,d}$. By substituting this parameterization of the value function gradient into our objective function (see Equations 14-16) it is easy to see that each optimal $w_{t,i}$ can be found by solving an ordinary least squares problem.

While our framework allows for arbitrary feature functions $\phi(x)$, it is of interest to let $\phi(x)$ contain a constant and the d elements of x. This models each component of the gradient of the value function as a convex combination of linear functions and results in a control policy that is a mixture of linear feedback controllers.

V. Computational Complexity Considerations

Here we compare the relative computational complexity of parameterizing the value function vs. the gradient of the value function. In order to achieve the equivalent expressivity of a mixture of linear approximators for each dimension of the value function gradient one must parameterize the value function using a mixture of local approximators containing a constant, the d state dimensions, and the $\frac{d(d+1)}{2}$ products and coproducts of the state dimensions.

The computational bottleneck of the algorithm is performing the linear regression step needed to compute the optimal parameter weights $w_{t,i}$. The complexity of linear regression is $O(mn^2)$ where m is the sample size and n is the number of features to be estimated. For the direct parameterization of the value function we have $d + \frac{d(d+1)}{2}$ linear regression instances generated for each of the m collocation points (based on satisfying first and second-order derivatives of the HJB) and a total of $k\left(d + \frac{d(d+1)}{2}\right)$ parameters (for k local function approximators). In total the computational complexity of this approach is $O(mk^2d^6)$.

If we instead directly parameterize each dimension of the gradient we have to perform d fits with each having $k(d+1)$ parameters and $m(d+1)$ total examples (based on the gradient and second-order derivatives of the HJB). This gives a total computational cost for the proposed method of $O(mk^2d^4)$.

A speed up for both methods can be achieved If the collocation points and function approximator locations (μ) are constant over the discrete time-points in $\mathcal{T}$. In this case by precomputing the pseudoinverse for the design matrix, a matrix multiplication can be substituted for the linear regression step at each time step in $\mathcal{T}$.

VI. Relation to Prior Work

Least squares collocation is a popular approach for solving PDEs. A collocation method using radial basis functions was proposed in [4]. A similar approach was used in [3], [1] for solving finite horizon HJB control equations. The work of Simpkins and Todorov [12] introduced collocation methods for solving infinite horizon robot control problems and was inspirational to us, however, the approach was not directly applicable to the finite horizon problems we consider here. Todorov and Tassa have also explored the use of collocation methods for finite-horizon control problems [15]. While they introduce a basic blueprint for the application of collocation methods to the class of nonlinear finite-horizon control problems considered here, their work does not explore many of the particular choices needed to achieve good performance on obstacle avoidance problems. In particular we found that using the appropriate function approximator (in this case a mixture of locally linear functions to parameterize the gradient) was crucial to achieving good performance. Additionally, the approach of Todorov and Tassa was geared toward finding local solutions to the control problem (using iterative collection of collocation points with a candidate controller and controller improvement), we focus here on computing globally optimal control policies.

There are other approaches to obstacle avoidance that build on the artificial potential function approach but try to overcome some of its key limitations [9], [8], [10], [16], [2], [11], however, each of these approaches either is not applicable to solving the full range of robot control problems we address here or else does not allow for a well-defined and flexible notion of optimality. Also, we are not the first to conceptualize the problem of obstacle avoidance as an optimal control problem (see [13], [6] for example). However, our method provides a key contribution in introducing new continuous collocation methods from optimal control to solving the obstacle avoidance problem. Additionally, compared to these two other works Gradient Collocation is: (1) applicable to a very general class of control-affine diffusions (capable of describing most biological and mechanical motor control problems) (contrary to [13]) and (2) able to solve the original continuous state, action, and time problem without the need for discretization of the state space (as is done in [6]).

VII. Experiments

Here we show the results of running our algorithm, Gradient Collocation, on three problems from robotic obstacle avoidance and non-linear control.

A. Non-convex Obstacle Avoidance

In this experiment we test Gradient Collocation on a 2-d obstacle avoidance problem. The state reward rate generated to represent the obstacles using the procedure from Section III is shown in Figure 2. The obstacle avoidance problem involves moving a point mass robot from an initial point in the state space to the goal state (shown in black in Figure 2). For this experiment we used a kinematic model of the robot where the control signals of the robot specify the desired velocity at each point in time. Also we assume a deterministic system. Thus, according to Equation 1 we have $a(x) = 0$, $b(x) = I_2$, $c(x) = 0$ (where I_2 is the 2x2 identity matrix). Additionally, we specify a time horizon of 5 seconds and use a .01s time discretization interval.

We use a 13×13 grid of collocation points uniformly distributed over the region $[-3, 9] \times [-3, 9]$. Additionally, we use the same grid as the locations for the local value function gradient approximators. The σ matrix for each local approximator was set to $\frac{1}{2.5} I_2$.

The results of running Gradient Collocation in terms of average reward over trajectories starting at each of the points in the 13×13 grid was -13.93. In order to compare this result to the potential function approach we computed the control signal by performing gradient descent directly on the sum of the reward rate and terminal reward functions (where the reward rate function specifies the obstacles and the terminal reward function specifies the goal). In order to be fair to the potential function method we performed a grid search over many possible weightings of the two reward functions and report on the best performing weighting. Even with this search over possible weightings the average performance for the potential function method was -131.1 (which is about 10 times worse than Gradient Collocation). The reason that the potential function approach works so poorly is that no matter how we weight the goal vs. the obstacle the points that start in the middle of the u-shaped obstacle are effectively trapped in a local minima of the potential function and are unable to escape.

B. Obstacle Avoidance for a 2-DoF Robotic Manipulator

Next, we test our algorithm's ability to perform obstacle avoidance when the state space is defined in terms of the

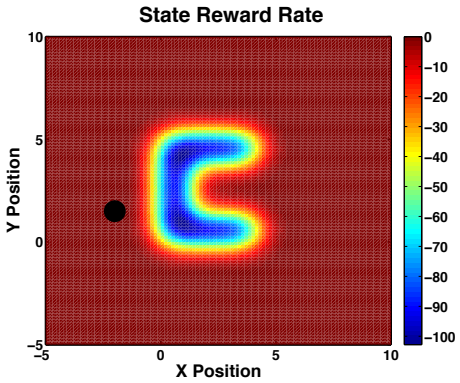

Fig. 2. The state reward rate used in the experiment of moving a point mass robot to the goal (black dot) while avoiding the regions of low reward (the sideways u-shaped region).

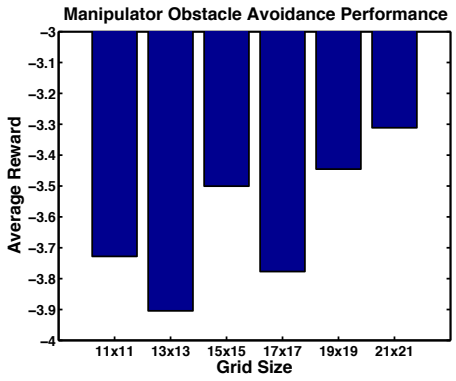

Fig. 3. The average reward of our algorithm when using varying sized grids of function approximators. For instance 11×11 means that the parameterization of the value function gradient had $11 \times 11 = 121$ local approximators.

joint angles of a robotic manipulator and the obstacle and goal positions are given in world coordinates (see Figure 1). Specifically, our goal is to move a 2-link robotic arm from an initial joint angle configuration such that the end effector reaches the goal (shown in black), while maintaining the constraint that none of the parts of the arm can touch the obstacles. Figure 1 shows both a schematic of the robotic arm, the goal, and the obstacles, and also the state reward rate used to enforce both the joint limits of the robot (each constrained to $[0, \pi]$) and the fact that the robot arm should not contact any obstacles. The reward rate was computed by dividing the robot arm into 500 evenly spaced segments and then applying Equation 11. For this experiment we used a kinematic model of the robotic arm where the control signals of the robot specify the desired angular velocities at each point in time. Also we assume a deterministic system. Thus, according to Equation 1 we have $a(x) = 0$, $b(x) = I_2$, $c(x) = 0$ (where I_2 is the 2x2 identity matrix). Additionally, we used a time horizon of 2 seconds.

We ran our algorithm multiple times with differing resolutions for both the number of collocation points as well as the number of local function approximators. For each run, an identical grid of collocation points and local approximator centers was distributed uniformly over the area $[0, \pi] \times [0, \pi]$. The performance for each run was evaluated on a 15×15 grid of starting joint angle configurations uniformly distributed over this same area. Figure 3 shows the results in terms of average reward for our algorithm with different resolutions. The peak performance for Gradient Collocation was obtained with a 21×21 grid of local function approximators. A representative trajectory that our algorithm learns is shown in Figure 4. The robot learns to first contract its elbow so that when it rotates about its shoulder in order to reach the target the forearm link does not come into contact with the obstacles. Once the arm is successfully through the two obstacles the end effector moves toward the target.

To compare Gradient Collocation with the artificial potential function approach we performed a similar evaluation as we did for the non-convex optimization avoidance problem. That is we created an artificial potential function as a weighted sum of the goal and obstacle functions and searched over a large set of possible weights. The maximum average reward over the 15×15 grid of starting locations for the artificial potential function approach was -10.48 compared to the best performance achieved with our approach which was -3.31.

C. Nonlinear Control: Minimum Torque Robot Control

We performed an experiment on the popular inverted pendulum problem. The goal was to swing up a pendulum from a given starting configuration (see Figure 5) to a straight up, $\theta = \pi$, configuration in a fixed amount of time while achieving minimal speed at the terminal time. We put a quadratic cost on the torque generated by the motor, thus effectively limiting the available torque. Without this limitation the optimal solution would be to perform the swingup in one motion. However, the addition of the minimum torque requirement makes the problem considerably harder. Minimum torque solutions for the inverted pendulum swingup problem display a characteristic "pumping" motion where energy is added to the system over a series of oscillations around $\theta = 0$ before a final attempt to swing the pendulum up is made. The number of such oscillations depends on the specific parameters of the problem. The deterministic system dynamics have the following form:

$$M\ddot{\theta} + G(\theta) = \tau + \tau_\nu(\dot{\theta}) \tag{20}$$

where M is the pendulum's moment of inertia, $G(\theta)$ is the Gravitational torque, τ is the motor's torque, and $\tau_\nu(\dot{\theta}) = -\nu\dot{\theta}$ is a viscous friction torque with a coefficient of viscous friction ν. Additionally, $\gamma \in \mathbb{R}$ defines the magnitude of a Brownian motion process noise that acts on $\dot{\theta}$. Thus the goal is to compute the value τ for each state and time in order to swing the pendulum up in a fixed amount of time. Specifically, we used a time horizon of 2 seconds discretized into 10 millisecond intervals, no discounting, no state-reward rate, a control cost-rate equal to $\frac{1}{2}\tau^2$, and a terminal state reward of $-5(\pi - \theta)^2 + \dot{\theta}^2$ (thus the highest reward is

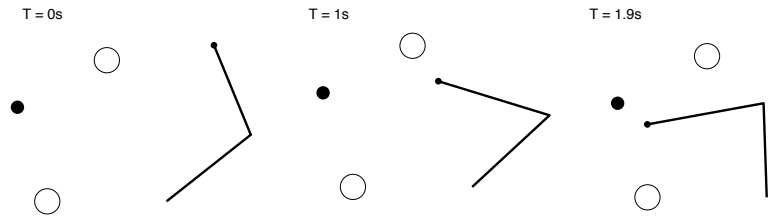

Fig. 4. A typical trajectory for the task of guiding the end effector of a 2-link robotic arm toward a goal without contacting obstacles. The configuration of the robotic arm at three points in the time interval of $[0, 2]s$ are shown. The robot learns to first bend its elbow so that it can proceed toward the goal without contact the obstacle. The terminal time is at $T = 2s$.

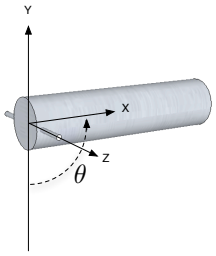

Fig. 5. Schematic of the inverted pendulum system. The system is controlled by exerting a torque around the base of the pendulum. The goal is to guide the pendulum to $\theta = \pi$ (pendulum arm facing up) using minimal torque in a fixed amount of time. Additionally a gravitational force is exerted on the pendulum as well as a viscous friction force around the base of the pendulum.

given for reaching the goal state), and no noise ($c(x) = 0$). Additionally the mass of the pendulum was $M = 1kg$ and the coefficient of viscous friction was $\nu = .4$.

We compared the performance of Gradient Collocation with the popular iLQR approach to nonlinear control (for a discussion of this algorithm see [7]). We compared the two approaches on a 21×21 grid of starting states spread uniformly over the region $\theta \in [-\pi, 3\pi]$ and $\dot{\theta} \in [-5, 5]$. The proposed method used the grid of starting states for both the collocation points and for the location vectors for the local approximators (μ). For the iLQR method we performed independent runs of iLQR for each candidate starting location. Our method achieved an average reward of -15.35 vs. -34.57 for iLQR.

VIII. Conclusion

We proposed an approach, named Gradient Collocation, for finding global approximately optimal solutions to robot control problems in continuous state, action, and time. Our method is well-suited to computing optimal controllers for a robot in the presence of obstacles as well as for solving control problems with non-linear dynamics. Our approach worked well on two obstacle avoidance problems: avoiding non-convex obstacles and avoiding obstacles with a robotic manipulator. Additionally, our approach significantly outperformed a widely used local approach, iLQR, for solving nonlinear control problems on the inverted pendulum task.

Future work will test Gradient Collocation's performance on both higher dimensional control problems as well as stochastic obstacle avoidance problems.

References

[1] H. Alwardi, S.Wang, L. S. Jennings, and S. Richardson. An adaptive least-squares collocation radial basis function method for the hjb equation. *Journal of Global Optimization*, 2011.

[2] J. Barraquand and J. Latombe. Robot motion planning: A distributed representation approach. *The International Journal of Robotics Research*, 10(6):628, 1991.

[3] C. S. Huang, S. Wang, C. S. Chen, and Z. C. Li. Aradial basis collocation method for Hamilton–Jacobi–Bellman equations. *Automatica*, pages 2201–2207, 2006.

[4] J..Kansa. Multiquadrics—a scattered data approximation scheme with applications to computational fluid-dynamics—ii: Solutions to parabolic, hyperbolic and elliptic partial differential equations. *Computers and Mathematics with Applications*, 17:169–178, 2000.

[5] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90, 1986.

[6] S. Lavalle and P. Konkimalla. Algorithms for computing numerical optimal feedback motion strategies. *The International Journal of Robotics Research*, 20(9):729–752, 2001.

[7] W. Li and E. Todorov. Iterative linear-quadratic regulator design for nonlinear biological movement systems. In *Proceedings of the First International Conference on Informatics in Control, Automation, and Robotics*, pages 222–229. Citeseer, 2004.

[8] S. Lindemann, I. Hussein, and S. LaValle. Real time feedback control for nonholonomic mobile robots with obstacles. In *Decision and Control, 2006 45th IEEE Conference on*, pages 2406–2411. IEEE, 2006.

[9] S. Lindemann and S. LaValle. Smoothly blending vector fields for global robot navigation. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 3553–3559. IEEE, 2005.

[10] S. Lindemann and S. LaValle. Simple and efficient algorithms for computing smooth, collision-free feedback laws over given cell decompositions. *The International Journal of Robotics Research*, 28(5):600, 2009.

[11] E. Rimon and D. Koditschek. Exact robot navigation using artificial potential functions. *Robotics and Automation, IEEE Transactions on*, 8(5):501–518, 1992.

[12] A. Simpkins and E. Todorov. Practical numerical methods for stochastic optimal control of biological systems in continuous time and space. In *Adaptive Dynamic Programming and Reinforcement Learning, 2009. ADPRL'09. IEEE Symposium on*, pages 212–218. IEEE, 2009.

[13] S. Sundar and Z. Shiller. Optimal obstacle avoidance based on the hamilton-jacobi-bellman equation. *Robotics and Automation, IEEE Transactions on*, 13(2):305–310, 1997.

[14] Y. Tassa and E. Todorov. High-order local dynamic programming. In *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2011 IEEE Symposium on*, pages 70–75. IEEE, 2011.

[15] E. Todorov and Y. Tassa. Iterative local dynamic programming. In *Adaptive Dynamic Programming and Reinforcement Learning, 2009. ADPRL'09. IEEE Symposium on*, pages 90–95. IEEE, 2009.

[16] L. Zhang, S. LaValle, and D. Manocha. Global vector field computation for feedback motion planning. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 477–482. IEEE, 2009.