2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Motion Planning for The Estimation of Functions*

Aneesh Raghavan, Giacomo Sartori and Karl Henrik Johansson

***Abstract*—We consider the problem of estimation of an unknown real valued function with real valued input by an agent. The agent exists in 3D Euclidean space. It is able to traverse in a 2D plane while the function is depicted in a 2D plane perpendicular to the plane of traversal. By viewing the function from a given position, the agent is able to collect a data point lying on the function. By traversing through the plane while paying a control cost, the agent collects a finite set of data points. The set of data points are used by the agent to estimate the function. The objective of the agent is to find a control law which minimizes the control cost while estimating the function optimally. We formulate a control problem for the agent incorporating an inference cost and the control cost. The control problem is relaxed by finding a lower bound for the cost function. We present a kernel based linear regression model to approximate the cost-to-go and use the same in a control algorithm to solve the relaxed optimization problem. We present simulation results comparing the proposed approach with greedy algorithm based exploration.**

I. Introduction

Exploration problems have received significant attention in system identification, dual control theory and reinforcement learning literature. The objective of exploration problems is to find paths in discrete state space (for eg. graphs) or continuous state space (of a dynamical system) which are optimal towards the learning of unknown parameters. The systems have additional tasks (eg. maximizing rewards) which eventually leads to trade-off. The trade-off between exploration vs. exploitation has been studied extensively in the context of multi-armed bandit problems, [1], [2]. Optimal exploration has been one of the driving forces for reinforcement learning, [3], [4], [5]. Dynamic programming and regression trees are tools that have been used extensively in exploration. The term exploration is also used in the context solving dynamic programming equations approximately [6], [7], [8].

In regression analysis, statistical learning theory and function approximation theory, it is well known that the functions learned or approximated are dependent on the data given. Optimal sampling for interpolation and regression problems have been studied, e.g. [9], [10]. We consider a motion planning problem where an agent explores the space it exists in to construct a "data set" which is then used to estimate a function. As the agent traverses through the space, data points are collected one after another, sequentially, to construct the data set. Two consecutive data points are constrained by the dynamics and the ability of the system to maneuver. Thus, the problem we consider is not a traditional sampling problem, where the data set is simply treated as a "set" without any "ordering" or constraints among the data points.

*Research supported by the Swedish Research Council (VR), Swedish Foundation for Strategic Research (SSF), and the Knut and Alice Wallenberg Foundation. The authors are with the Division of Decision and Control Systems, Royal Institute of Technology, KTH, Stockholm. Email: aneesh@kth.se, gsartori@kth.se, kallej@kth.se

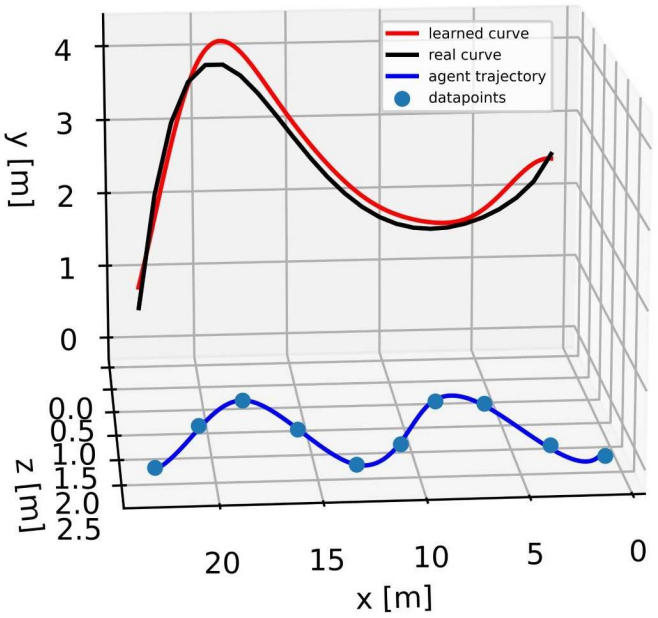

Fig. 1. Estimation of a function in $x-y$ plane by following a path in $x-z$ plane

A. Problem Description

We consider the problem of estimating a function by an agent as depicted in Figure 1. The agent lives in 3D Euclidean space, with three axes, $x-y-z$. The agent is able to move in the $x-z$ plane. The mapping from the independent variable x to the dependent variable y is plotted on the $x-y$ plane (depicted in black). By viewing the function from position (x,z), the agent is able to collect the data point (x,y). That is, the x co-ordinate of the agent (that determines the independent variable) along with the corresponding dependent variable are collected as data point (x,y). Over the planning horizon m, the agent moves through the $x-z$ plane (blue trajectory) and collects a finite number of data points, $\{(x_j,y_j)\}_{j=1}^{m}$ where $(x_j,\, y_j) \in \mathbb{R}^2$. The positions at which it collects data points are depicted as blue dots. For traversing through the $x-z$ plane, the agent has to pay a control cost. Using the data points collected, the agent estimates the function from x to y (depicted in red). The function estimated will depend on the data points collected and the estimation technique. The objective of the agent is to find a control policy executing which, it can collect data points which are optimal for estimation while minimizing its control effort. For the rest of the paper, we

2023 62nd IEEE Conference on Decision and Control (CDC) | 979-8-3503-0124-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/CDC49753.2023.10383869

use the term "function" and the term "curve" interchangeably. Though curves include circles, ellipses, etc., we use the term "curve" in a strictly function sense.

II. Problem Formulation

A. Abstraction

1) Estimation of the function: We consider the estimation technique given the data set $\{(x_l, y_l)\}_{l=1}^m$. Let $\mathcal{X} \subset \mathbb{R}$. Let $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a kernel and the RKHS generated by it be $(H, \langle\cdot,\cdot\rangle_H, ||\cdot||_H)$. For the definition of kernels and the spaces generated by them, we refer to [11]. We consider the hypothesis space to be H, i.e., we want to find a function $f \in H$ that best estimates the mapping from the independent variable x to the dependent variable y. The estimation problem, specifically the regression problem, is formulated as an optimization problem:

$$\min_{f\in H} \quad C(f);\ C(f) = \sum_{l=1}^{m}(y_l - f(x_l))^2 + \varrho||f||_H^2.$$

Let $\mathbf{K} = (K(x_j, x_k))_{jk} = (\langle K(x, x_k), K(x, x_j)\rangle_H)$ be the kernel matrix. It is well known that the solution to the problem is given by $f^*(x) = \sum_{l=1}^m \alpha_l^* K(x, x_l)$ (refer [11]), where $\boldsymbol{\alpha}^* = (\mathbf{K}^T\mathbf{K} + \varrho\mathbf{K})^{-1}\mathbf{K}^T\mathbf{y}$ and $\mathbf{y} = [y_1, \ldots, y_m]$. It can be shown that $C(f^*) = \mathbf{y}^T\Big[\mathbf{I} - \mathbf{K}(\mathbf{K}^T\mathbf{K} + \varrho\mathbf{K})^{-1^T}\mathbf{K}^T\Big]\mathbf{y}$. Thus, the error in fitting the data points depends on the data points collected. If we consider the "inference cost" as $C(f^*)$, then the estimation cost depends on both the components of the data points. However since y is a function of x, the minimum leaning cost is essentially a function of x alone. The function from x to y is unknown (which we are trying to estimate). Instead of minimizing the learning cost for a particular $\mathbf{y}$, i.e., $\mathbf{y} = [f(x_1), \ldots, f(x_m)]$ we consider minimizing the cost for any $\mathbf{y}$, which is equivalent to minimizing the 2 norm (or maximum eigenvalue) of $\bar{\mathbf{K}} = \mathbf{I} - \mathbf{K}(\mathbf{K}^T\mathbf{K} + \varrho\mathbf{K})^{-1^T}\mathbf{K}^T$.

2) The Control Problem: The state space of the agent is $\mathbb{R}^2$, while the set of actions (action space) that it can take is denoted by $\mathcal{U}$. The state of the system at time t is denoted by $\zeta(t) = [x(t), z(t)]$. The observation of the system at t is (x, y), where $x = x(t)$ and the mapping from x to y is to be estimated. No process noise or measurement noise is considered. The agent is modeled as a discrete time system with known dynamics,

$$\zeta(t+1) = \phi(\zeta(t), u(t)); C = [1, 0]$$
$$x(t+1) = C\zeta(t+1),\ t = 0, \ldots m-1.$$

Thus, given ζ_0, the initial state of the agent (where no data point is collected), the control problem for the agent is:

$$\min_{\{u(t)\}_{t=0}^{m-1}\in\mathcal{U}} \Big|\Big|\mathbf{I} - \mathbf{K}(\mathbf{K}^T\mathbf{K} + \varrho\mathbf{K})^{-1^T}\mathbf{K}^T\Big|\Big|_2 + \varsigma\sum_{l=0}^{m-1}||u(l)||^2$$
$$s.t\ \ \zeta(t+1) = \phi(\zeta(t), u(t)); x(t+1) = [1, 0]\zeta(t+1),$$
$$t = 0, \ldots m-1,\ \mathbf{K}_{kl} = \langle K(x, x(l)), K(x, x(k))\rangle_H. \qquad (1)$$

The cost function is a linear combination of the estimation cost and the control cost. The constraints ensure that (k, l) component of matrix $\mathbf{K}$ corresponds to $K(x(k), x(l))$, where $x(k)$(or $x(l)$) is the first component of $\zeta(k)$(or $\zeta(l)$), i.e., x co-ordinate of the system at time instant k(or l). This problem can be viewed as a control theoretic approach to the identification of its environment by an agent where the problem as been traditionally studied through vision and image processing techniques.

B. Literature Survey

Optimizing the maximum eigenvalue of a symmetric matrix has been studied in the literature from a numerical optimization perspective. We refer to [12], [13], [14] and the references there in. These papers consider the matrix function of the form $A(x) = \sum_{j=1}^m A_j x_j + A_0,\ \{x_j\}_{j=1}^m \subset \mathbb{R}$, where $\{A_j\}_{j=0}^m$ are symmetric matrices. They present numerical approaches based on convex optimization, specifically, interior point methods and semidefinite programming to solve the problem. This approach is not suitable for us as the matrix function we consider is not of the form above. Submodular optimization approach to study eigenvalues of submatricies has been considered in [15]. Though $\lambda_{max}(\cdot)$ might be convex, typically its not differentiable. There have been studies to approximate $\lambda_{max}(\cdot)$ by differentiable functions.

C. Contributions

Our contributions are as follows. 1. First, we prove that the minimization of the learning cost is equivalent to maximization of the smallest eigenvalue of the kernel matrix. Using a lower bound for the minimum eigenvalue of a square matrix, we find a lower bound for cost function described in equation (1) and obtain a relaxation of the control problem. 2. We analyze the relaxed optimization problem through dynamic programming and derive the equations satisfied by the value function. 3. We present a kernel based estimation of the cost-to-go and present a control algorithm using the same to approximately solve the relaxed control problem. 4. We present simulation results comparing the proposed control algorithm with a greedy algorithm based exploration.

The outline of the paper is as follows. In Section III, we present the relaxation of the control problem and the dynamic programming solution to the relaxed optimization problem. In Section IV, we discuss the estimation of the optimal cost-to-go using linear regression and present the control algorithm utilizing the approximate value function. In Section V, we present simulation results comparing kernel based control algorithm and a greedy algorithm based exploration. We discuss some concluding remarks and future work in Section VI. Notation: We denote vectors obtained by concatenating other vectors and matrices in boldface.

III. Analysis of the problem

A. Relaxation of the Control Problem

Though 2-norm of a matrix is a convex function, it is not necessary that the above optimization problem is convex. Since we consider a exploration problem it is bound to have a combinatorial aspect to it. We seek a relaxation of the

problem that it is tractable.We consider the eigendecomposition of $\mathbf{K}$ as $\mathbf{K} = O^T \Lambda O$. O is the orthonormal matrix obtained from the orthonormal eigenvectors of $\mathbf{K}$. Thus, $\bar{\mathbf{K}} = O^T[\mathbf{I} - \Lambda(\Lambda^2 + \varrho\Lambda)^{-1}\Lambda]O$. It follows that the eigenvalues of $\bar{K}$ are $1 - \dfrac{\lambda_i^2}{\lambda_i^2 + \rho\lambda_i}$ where λ_i is an eigenvalue of $\mathbf{K}$. Hence,

$$||\bar{\mathbf{K}}||_2 = \lambda_{\max}(\bar{\mathbf{K}}) = \frac{\varrho}{\lambda_{\min}(\mathbf{K}) + \varrho}.$$

Thus, minimizing the 2 norm of $\bar{\mathbf{K}}$ is equivalent to maximizing the minimum eigenvalue of $\mathbf{K}$. There are many lower bounds for $\lambda_{\min}(\mathbf{K})$ and include:

$$\lambda_{\min}(\mathbf{K}) \geq \left\{\frac{m-1}{Tr[\mathbf{K}]}\right\}^{m-1}(det(\mathbf{K})),$$

$$\lambda_{\min}(\mathbf{K}) \geq Tr[\mathbf{K}] - \frac{1}{det(\mathbf{K})}\frac{m^m}{(m-1)^{m-1}}\left\{\frac{Tr[\mathbf{K}]}{m+1}\right\}^{m+1},$$

$$\lambda_{\min}(\mathbf{K}) \geq \frac{Tr[\mathbf{K}]}{m} - \sqrt{\frac{(m-1)(mTr[\mathbf{K}^2] - Tr[\mathbf{K}]^2)}{m^2}}$$

For proof of these bounds, we refer to [16], [17], [18]. We consider the last inequality as the desired lower bound as it can be expressed completely using the components of $\mathbf{K}$. From the CBS inequality it follows that,

$$\begin{aligned} Tr[\mathbf{K}^2] &= \sum_{l=1,k=1}^{m} K^2(x_l, x_k) \\ &\leq \sum_{l=1,k=1}^{m} K(x_l, x_l)K(x_k, x_k) = Tr[\mathbf{K}]^2 \end{aligned}$$

Thus,

$$\begin{aligned} \lambda_{\min}^2(\mathbf{K}) &\geq \frac{Tr[\mathbf{K}]^2}{m^2} + \frac{(m-1)Tr[\mathbf{K}^2]}{m} - \frac{(m-1)Tr[\mathbf{K}]^2}{m^2} \\ &\quad -2\frac{Tr[\mathbf{K}]}{m}\sqrt{\frac{(m-1)(mTr[\mathbf{K}^2] - Tr[\mathbf{K}]^2)}{m^2}} \\ &\geq \frac{(m-1)Tr[\mathbf{K}^2]}{m} - \frac{(m-2)Tr[\mathbf{K}]^2}{m^2} - 2\frac{(m-1)Tr[\mathbf{K}]^2}{m^2} \\ &= \frac{(m-1)Tr[\mathbf{K}^2]}{m} - \frac{(3m-4)Tr[\mathbf{K}]^2}{m^2} \end{aligned}$$

We note that for m large enough the second term in the inequality becomes small and hence the R.H.S of the inequality stays positive. The objective is to maximize the lower bound. We redefine the control problem as:

$$\begin{aligned} &\min_{\{u(t)\}_{t=0}^{m-1} \in \mathcal{U}} \frac{(3m-4)Tr[\mathbf{K}]^2 - (m^2-m)Tr[\mathbf{K}^2]}{m^2} + \varsigma\sum_{l=0}^{m-1}||u(l)||^2 \\ &s.t\ \ \zeta(t+1) = \phi(\zeta(t), u(t)); x(t+1) = [1,0]\zeta(t+1), \\ &t = 0, \dots m-1,\ \mathbf{K}_{lk} = \langle K(x, x(l)), K(x, x(k))\rangle_H, \end{aligned} \tag{2}$$

where ζ_0 is given. The objective of this paper is to solve the above optimization problem. We note that the cost function in the optimization problem does not depend on the curve that is being estimated. It only depends on the kernel and the sampling points visited to learn the curve. Hence, irrespective of the "true" curve, given a kernel, the control algorithm should result in sampling points which maximize "information" for learning of the curve while minimizing control cost.

B. Dynamic Programming

We analyze the problem mentioned in equation (2) using a dynamic programming approach. We use the terms "value function" and "optimal cost-to-go" interchangeably. Let $a_m = \dfrac{3m-4}{m^2}$ and $b_m = \dfrac{m-1}{m}$. For convenience we denote the state at j by $\zeta_j = \zeta(j)$. The vector of all states until j is referred to as the augmented state and is denoted as $\boldsymbol{\zeta_j}$, i.e., $\boldsymbol{\zeta_j} = [\zeta_1, \dots, \zeta_j]$. Given, $\boldsymbol{\zeta_j}$, the stage cost at stage j is defined below (equation (3)) where $x_l = C\zeta_l,\ x = C\zeta$, and $\tilde{\phi}(\zeta, u) = C\phi(\zeta, u)$.

$$\begin{aligned} &S_j(\boldsymbol{\zeta_j}, u_j) = \sum_{l=1}^{j} 2\Big[a_m K(x_l, x_l)K(\tilde{\phi}(\zeta_j, u_j), \tilde{\phi}(\zeta_j, u_j)) \\ &- b_m K^2(x_l, \tilde{\phi}(\zeta_j, u_j))\Big] + (a_m - b_m)K^2(\tilde{\phi}(\zeta_j, u_j), \\ &\tilde{\phi}(\zeta_j, u_j)) + \varsigma||u_j||_2^2, j = 1, \dots, m-1,\ S_0(\zeta_{,0}\, u_0) = \\ &(a_m - b_m)K^2(\tilde{\phi}(\zeta_0, u_0), \tilde{\phi}(\zeta_0, u_0)) + \varsigma||u_0||_2^2 \end{aligned} \tag{3}$$

Thus $\sum_{j=0}^{m-1} S_j(\boldsymbol{\zeta_j}, u_j)$ is equal to the objective (cost function) of the optimization problem mentioned in (2) with incorporation of the constraints. We note that the stage cost at stage j not only depends on the current state ζ_j, but the entire history $\zeta_1, \dots, \zeta_{j-1}$. This is due to the learning cost, where the correlation between every pair of data points, $K(x_l, x_k)$, has to be taken into account. The dependency of $S_j(\cdot,\cdot)$ on $\boldsymbol{\zeta_j}$ deviates from the classical literature of dynamic programming, where $S_j(\cdot,\cdot)$ usually only depends on ζ_j. This dependency gets carried forward to the cost-to-go as well. The optimal cost-to-go for (2) at stage j, V_j, as function of $\boldsymbol{\zeta_j}$ is defined using the dynamic programming equations as follows,

$$\begin{aligned} &V_j(\boldsymbol{\zeta_j}) = \min_{u \in \mathcal{U}}\ S_j(\boldsymbol{\zeta_j}, u) + V_{j+1}((\boldsymbol{\zeta_j}; \phi(\zeta_j, u))), \\ &j = 0, \dots, m-1, \text{ and } V_m(\boldsymbol{\zeta_m}) = 0\ \forall\ \boldsymbol{\zeta_m} \in \mathbb{R}^{2\times m}, \end{aligned} \tag{4}$$

where $(\boldsymbol{\zeta_j}; \phi(\zeta_j, u)) = [\zeta_1, \dots, \zeta_j, \phi(\zeta_j, u)] \in \mathbb{R}^{2\times(j+1)}$. This set of equations can alternatively be expressed as,

$$V_j(\boldsymbol{\zeta_j}) = \sum_{l=j}^{m-1} \min_{u_l \in \mathcal{U}} S_l(\phi^{l-j}(\boldsymbol{\zeta_j}, u_j, u_{j+1}, \dots, u_{l-1}, u_l)), \text{where}$$

$$\phi^{l-j}(\boldsymbol{\zeta_j}, u_j, \dots, u_l) = \Big(\Big(\zeta_j^{l-j-1}; \phi^{l-j-1}(\boldsymbol{\zeta_j}, u_j, \dots, u_{l-1})\Big), u_l\Big),$$

$$\phi^{l-j-1}(\boldsymbol{\zeta_j}, u_j, \dots, u_k) = \phi\Big(\phi^{l-j-2}(\boldsymbol{\zeta_j}, u_j, \dots, u_{l-2}), u_{l-1}\Big),$$

$$\zeta_j^{l-j-1} = \Big(\zeta_j^{l-j-2}; \phi^{l-j-2}(\boldsymbol{\zeta_j}, u_j, \dots, u_{l-2})\Big), \phi^1(\boldsymbol{\zeta_j}, u_j, u_{j+1})$$

$$= \phi\big(\phi(\zeta_j, u_j), u_{j+1}\big), \zeta_j^1 = (\boldsymbol{\zeta_j}; \phi(\zeta_j, u_j)) \text{ for } l \geq j+2\ ,$$

$$\phi^1(\boldsymbol{\zeta_j}, u_j, u_{j+1}) = ((\boldsymbol{\zeta_j}; \phi(\zeta_j, u_j)), u_{j+1}), \phi^0(\boldsymbol{\zeta_j}, u_j) = (\boldsymbol{\zeta_j}, u_j).$$

The above equations only demonstrate the coupling between stage costs through the optimal control law and do not provide any remedy to solve the equations. If the equations were

indeed solvable, then the control algorithm is executed from state ζ_0 with the optimal control law at stage j as (a function of the augmented state)

$$u_j^*(\boldsymbol{\zeta_j}) = \min_{u \in \mathcal{U}} \; S_j(\boldsymbol{\zeta_j}, u) + V_{j+1}((\boldsymbol{\zeta_j}; \phi(\zeta_j, u))).$$

Thus, $\{u_j^*\}_{j=0}^{m-1}$ solves (2).

IV. Algorithms

In this section, we present a numerical approach to approximate $V(\cdot)$ defined in equation (4) and control algorithms to approximately solve the optimization problem in (2).

A. Kernel Based Approximation of Value function

Consider a radial kernel function, i.e., $K(x,y) = \psi(||x - y||)$, where $\psi(\cdot)$ is positive definite function. and $\psi(0) = 1$. The stage costs simplify to:

$$S_j(\boldsymbol{\zeta_j}, u) = \sum_{l=1}^{j} 2\left[a_m - b_m K^2(x_l, \tilde{\phi}(\zeta_j, u))\right] + (a_m - b_m) + \varsigma||u||_2^2, j = 1, \ldots, m-1, \; S_0(\zeta, u) = (a_m - b_m) + \varsigma||u||_2^2$$

A naive attempt to find $V_j(\cdot)$ satisfying equation (4) would be to set the gradient of $S_{m-1}(\cdot,\cdot)$ with respect to respect to u to zero.

$$\frac{\partial S}{\partial u} = \sum_{l=1}^{j} 2b_m \left[K(x_l, \tilde{\phi}(\zeta_j, u^*)) \frac{\partial K(x,y)}{\partial y}\bigg|_{\left(C\zeta_l, \tilde{\phi}(\zeta_j, u^*)\right)} \times \nabla_u \tilde{\phi}(\zeta, u)\Big|_{\left(\zeta_j, u^*\right)} \right] - \varsigma u^* = 0$$

Given this expression, finding a closed form expression for u^* does not seem to be reasonable. However, we note the $u^*(\zeta)$ depends on $K(C\zeta_{m-1}, \zeta_l)$, $\dfrac{\partial K(x,y)}{\partial y}$, etc. Ignoring the other dependencies, it is reasonable to consider the dependence of $u^*(\zeta)$ only on $\{K(C\zeta_{m-1}, C\zeta_l)\}_{l=1}^{m-2}$ (and thus $||u||_2^2$ on $\{K^2(C\zeta_{m-1}, C\zeta_l)\}_{l=1}^{m-2}$. Given that $K(\cdot,\cdot)$ is a kernel map, $K^2(\cdot,\cdot)$ is also a kernel map. We consider the following approximation procedure for V_{m-1}. For $j = 1, \ldots, m-1$, we uniformly sample p_j points, $E_j = \{\hat{\zeta}_{j,1}, \ldots, \hat{\zeta}_{j,p_j}\}$ from $\mathbb{R}^2$. We observe that learning costs and the control cost (as noted above) depend on $\{K^2(C\zeta_{m-1}, C\zeta_l)\}_{l=1}^{m-2}$. Thus, we choose our regressors as

$$K^2(C\zeta_{m-1}, C\hat{\zeta}_{j,l})\}_{j=1,l=1}^{j=m-2,l=p_j} \cup \{K^2(C\zeta_j, C\hat{\zeta}_{m-1,l})\}_{j=1,l=1}^{j=m-2,l=p_{m-1}}$$

and consider a linear regression model as

$$\hat{V}_{m-1}(\{\beta_{j,l}\}, \boldsymbol{\zeta_{m-1}}) = \sum_{j=1}^{m-2}\sum_{l=1}^{p_j} \beta_{j,l} K^2(C\zeta_{m-1}, C\hat{\zeta}_{j,l}) + \sum_{l=1}^{p_{m-1}} \beta_{m-1,l} \sum_{j=1}^{m-2} K^2(C\zeta_j, C\hat{\zeta}_{m-1,l}). \quad (5)$$

Due to the dependence of the learning cost and u^* only on the x component of ζ, we consider only the x component in the regression problem. For each vector in the Cartesian product $E = \prod_{j=1}^{m-1} E_j$, denoted by $\hat{\boldsymbol{\zeta}}_{\boldsymbol{l}}$, we solve the optimization problem,

$$V_{m-1}(\hat{\boldsymbol{\zeta}}_{\boldsymbol{m-1,l}}) = \min_{u \in \mathcal{U}} \; S_{m-1}(\hat{\boldsymbol{\zeta}}_{\boldsymbol{m-1,l}}, u).$$

Let $N_E = \prod_{j=1}^{m-1} p_j$. Thus, we obtain N_E number of evaluations of the function V_{m-1}, one for every vector $\hat{\boldsymbol{\zeta}}_{\boldsymbol{l}} \in E$. Our objective is to chose $\{\beta_{j,l}\}_{j=1,l=1}^{j=m-1,l=p_{m-1}}$, so that $\hat{V}_{m-1}(\{\beta_{j,l}\}, \hat{\boldsymbol{\zeta}}_{\boldsymbol{m-1,l}}) = V_{m-1}(\hat{\boldsymbol{\zeta}}_{\boldsymbol{m-1,l}}) \; \forall \hat{\boldsymbol{\zeta}}_{\boldsymbol{l}} \in E$. We have N_E number of linear equations while there are $N_V = \sum_{j=1}^{m-1} p_j$ number of variables. Thus, we have an overdetermined system of linear equations. Let,

$$\boldsymbol{\beta_{m-1}} = [\beta_{1,1} \ldots, \beta_{1,p_1}, \beta_{2,1}, \ldots, \beta_{2,p_2}, \ldots, \beta_{m-1,1} \ldots \beta_{m-1,p_{m-1}}]^T \in \mathbb{R}^{N_V}.$$

To find $\boldsymbol{\beta}$, we consider a least squares regression problem,

$$\min_{\boldsymbol{\beta_{m-1}} \in \mathbb{R}^{N_V}} ||\mathbf{K_{m-1}}\boldsymbol{\beta_{m-1}} - \mathbf{V_{m-1}}||, \quad (6)$$

where $\mathbf{V_{m-1}} = [V_{m-1}(\hat{\boldsymbol{\zeta}}_{\mathbf{1}}), \ldots, V_{m-1}(\hat{\boldsymbol{\zeta}}_{\boldsymbol{N_E}})]^T$ and K_{m-1} is defined as follows. For $1 \le l \le N_E$, define

$$l \bmod p_{m-1} = r_{m-1}, l_{m-2} = l/p_{m-1}, l_{m-2} \bmod p_{m-2}, = r_{m-2}. l_{m-3} = l_{m-2}/p_{m-2}, l_q \bmod p_q = r_q, \; q \ge 1, \; l_{q-1} = l_q/p_{q-1}, q \ge 2. \text{ If } r_{m-1} = 0, \text{then } r_{m-1} = p_{m-1}. \; r_q = r_q + 1, 1 \le q \le m-2. \Rightarrow l = \sum_{s=1}^{m-2}(r_s - 1)\left[\prod_{v=m-1}^{s+1} p_v\right] + r_{m-1}.$$

For $1 \le k \le N_V$, let $\bar{k}$ and $\bar{r}_k$ be such that

$$k = \sum_{s=0}^{\bar{k}} p_s + \bar{r}_k, 1 \le \bar{r}_k \le p_{k+1}, \text{ where } p_0 = 0, \bar{k} \ge 0.$$

The element in the lth row and kth column of matrix $\mathbf{K_{m-1}}$, $(\mathbf{K_{m-1}})_{l,k}$, is defined as,

$$K^2(\hat{\zeta}_{m-1,r_{m-1}}, \hat{\zeta}_{\bar{k}+1,\bar{r}_k}), \text{ if } 1 \le k \le \sum_{s=1}^{m-2} p_s$$
$$\sum_{j=1}^{m-2} K^2(\hat{\zeta}_{j,r_j}, \hat{\zeta}_{m-1,\bar{r}_k}), \text{ if } \sum_{s=1}^{m-2} p_s \le k \le N_V. \quad (7)$$

The solution to the optimization problem in (6) is given by the normal equations,

$$\boldsymbol{\beta^*} = \left(\mathbf{K_{m-1}}^T \mathbf{K_{m-1}}\right)^{-1} \mathbf{K_{m-1}}^T \mathbf{V_{m-1}}.$$

For $j = 1, \ldots, m-2$, we define $\hat{V}_j(\{\beta_{j,l}\}, \boldsymbol{\zeta_j})$ as

$$\hat{V}_j(\{\beta_{k,l}\}, \boldsymbol{\zeta_j}) = \sum_{k=1}^{j-1}\sum_{l=1}^{p_k} \beta_{k,l} K^2(C\zeta_j, C\hat{\zeta}_{k,l}) + \sum_{l=1}^{p_j} \beta_{j,l} \sum_{k=1}^{j-1} K^2(C\zeta_k, C\hat{\zeta}_{j,l}). \quad (8)$$

and $\mathbf{K_j}$ is defined similar to $\mathbf{K_{m-1}}$. A backward induction ($j = m-1, \ldots, 0$) based algorithm is described in algorithm 1 which results in approximate value functions being computed offline. During run time, given the initial state ζ_0, the control algorithm runs for m stages. The control input at stage j and augmented state $\boldsymbol{\zeta_j}$, is computed as a minimizer,

$$\underset{u \in \mathcal{U}}{\operatorname{argmin}}\ S_j(\boldsymbol{\zeta_j}, u) + \hat{V}_{j+1}((\boldsymbol{\zeta_j}; \phi(\zeta_j, u))),$$

and this control algorithm is referred to as "KVC" algorithm. Thus, $\{u_j^*\}_{j=0}^{m-1}$ approximately solves the optimization problem in equation (2) as the optimal cost-to-go is approximated by $\hat{V}_j$. The dynamic programming equations mentioned in subsection III-B and the approximation procedure mentioned above can be extended to $\zeta \in \mathbb{R}^N$ as along as the stage costs satisfy (3).

Algorithm 1 Kernel Based Approximation of Value Functions

procedure KBVF
  Initialize $p_1, \ldots p_{m-1} \in \mathbb{N}$
  $j \leftarrow 1$
  while $j \leq m-1$ **do**
    Uniformly sample $E_j = \{\hat{\zeta}_{j,1}, \ldots, \hat{\zeta}_{j,p_j}\} \in \mathbb{R}^2$.
    $j \leftarrow j+1$
  $j \leftarrow m-1$, $\hat{V}_m(\zeta) \leftarrow 0\ \forall \zeta \in \mathbb{R}^2$
  while $j \geq 0$ **do**
    Construct grid $\bar{E}_j = \prod_{l=1}^{j} E_l$,
    For all $\hat{\boldsymbol{\zeta}}_{\boldsymbol{j,l}} \in \bar{E}_j$ solve,

$$V_j(\hat{\boldsymbol{\zeta}}_{\boldsymbol{j,l}}) = \min_{u \in \mathcal{U}}\ S_j(\hat{\boldsymbol{\zeta}}_{\boldsymbol{j,l}}, u) + \hat{V}_{j+1}((\hat{\boldsymbol{\zeta}}_{\boldsymbol{j,l}}; \phi(\hat{\zeta}_{j,l}, u)))$$

    Define $\mathbf{K_j}$ as in equation (7)
    Define $\mathbf{V_j} = [V_j(\hat{\boldsymbol{\zeta}}_{\boldsymbol{j,1}}), \ldots, V_j(\hat{\boldsymbol{\zeta}}_{\boldsymbol{j,|\bar{E}_j|}})]^T$
    Solve for $\boldsymbol{\beta}_j^* = \left(\mathbf{K_j}^T \mathbf{K_j}\right)^{-1} \mathbf{K_j}^T \mathbf{V_j}$
    Define $\hat{V}_j(\boldsymbol{\zeta_j})$ as in equation (8).
    $j \leftarrow j-1$

B. Greedy Algorithm

In this algorithm, at any stage j we only consider the optimization of the one step stage cost, i.e., the cost-to-go from stage j to stage $j+1$. We ignore the cost-to-go from $j+1$ stage to the final stage, i.e., we consider $V_{j+1}(\cdot) = 0$. The approximation of $V_j(\cdot)$ is not needed. Given the initial state ζ_0, the control algorithm runs for m stages, where the control input at stage j is computed as

$$\min_{u \in \mathcal{U}}\ S_j(\boldsymbol{\zeta_j}, u).$$

V. Simulation Results

In this section, we present simulation results comparing two of the algorithms, namely, the KVC algorithm and the greedy algorithm. The simulation setup is described as follows. We considered the dynamics of the agent to be a stable discrete time LTI system, i.e., $\phi(\zeta, u) = A\zeta + Bu$, where,

$$A = \begin{bmatrix} 0.5 & 0.2 \\ 0 & 0.5 \end{bmatrix},\ B = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.K_1(x,y) = \exp\frac{-(x-y)^2}{2\sigma_1^2}.$$

$$\bar{A} = \begin{bmatrix} 1.25 & 0.2 \\ 0 & 0.5 \end{bmatrix}.\ K_2(x,y) = (xy + c)^d.$$

ς was set to 0.5. m was chosen to be 5. The initial state was chosen as $(x, z) = (30, 40)$. The set of values that the control can take, $\mathcal{U}$, was chosen as $[-10, 10]$. With respect to Algorithm 1, p_1, p_2, p_3 and p_4 were set to 10.

$$f(x) = -0.569\log(x) + \frac{2x^2 + x^3 + x^4 - 0.00008x^7}{17560} + 2.507,$$

is the function to be estimated. Now we describe the simulation runs. The description of the figures mentioned in the following is similar to the description of Figure 1 mentioned in subsection I-A, except that there are two data sets, trajectories and estimated curves. These have been obtained by plotting the results of the KVC and greedy algorithm in the same figure. Simulation run 1: The simulation results with the Gaussian kernel, K_1(with $\sigma_1 = 1$), are presented in Figure 2. We observe that the performance of the greedy algorithm and the KVC algorithm are similar. The data points collected by the two algorithms are close to each other at stages $j = 1, 2, 3, 4$ and are different at $j = 5$. This leads to different curves being estimated. Simulation run 2: The algorithms were simulated with polynomial kernel, K_2 (with $c = 1, d = 2$), while rest of the simulation set up was retained from simulation run 1. The results are plotted in Figure 3. By inspection, we note that the data points collected by the agent in simulation run 1 and 2 are different from each other. Since the kernels are different, the estimated functions are different as well. Simulation run 3: We considered the Gaussian kernel and an unstable dynamical system, $\phi(\zeta, u) = \bar{A}\zeta + Bu$, with $\bar{A}$ as above. We note that $(\bar{A}, B)$ is stabilizable. In Figure 4, we plot the the results of the greedy and KVC algorithm with $\zeta_0 = (0, 20)$. The data points and the trajectories followed are again close to each other. However, the data sets and hence, the curves estimated are different from simulation run 1 (Figure 2). We infer that curves estimated (and the trajectories followed) are function of the initial conditions. We note that ζ_0 has not been plotted in any of the figures as no data point is collected there. In two of the simulation runs, the data set collected by the agent using the greedy algorithm and the KVC algorithm are similar or "close" to each other. It is well known that dynamic programming suffers from the curse of dimensionality. In the KVC algorithm, the dimension of the augmented state, $\boldsymbol{\zeta_j}$ grows as the planning horizon m increases. Estimating $\hat{V}_j(\cdot)$ by computing it at 10^2 or 10^3 points in $\mathbb{R}^{2\times j}$ may not be sufficient to obtain a estimate that is close to the true value function. Further investigation is needed on the computational aspect of estimation procedure in the KVC algorithm.

VI. Conclusion and Future Work

To conclude, we considered a discrete time control problem with the objective of estimating a curve. The problem was analyzed using dynamic programming and an estimation method for the value function was presented. The estimated value function was utilized in the KVC control policy which approximately solves the control problem. As future work, following

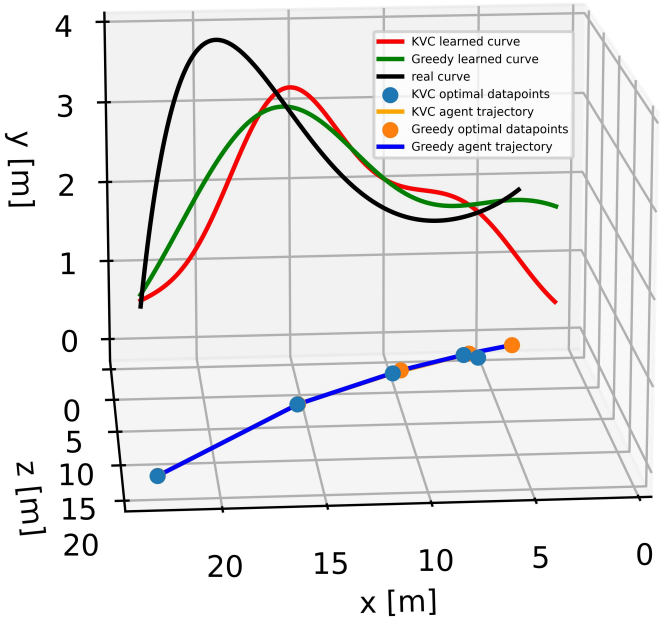

Fig. 2. Learning of a curve with trajectories generated by the KVC and greedy algorithm - Gaussian kernel (stable LTI system)

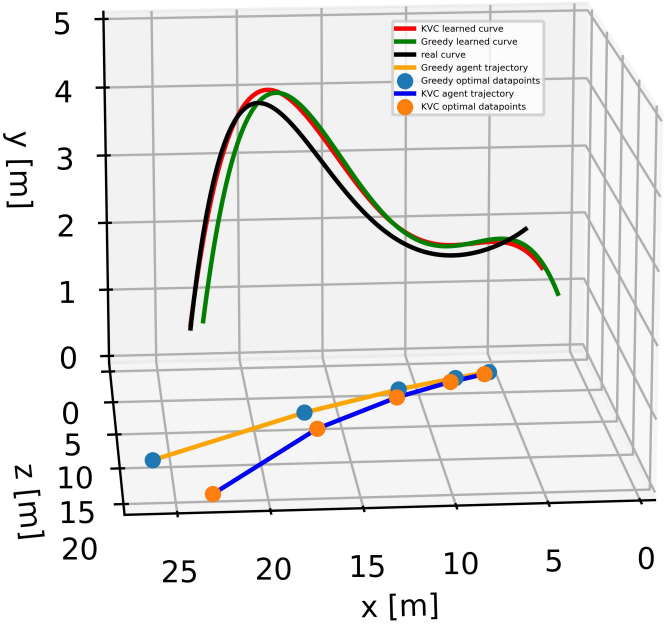

Fig. 3. Learning of a curve with trajectories generated by the KVC and greedy algorithm - polynomial kernel (stable LTI system)

are of interest: (a) " better " quantification of the inference cost; (b) quantifying the difference (in norm) between the estimated and true value function; (c) computationally efficient estimation of the value function.

REFERENCES

[1] A. N. Burnetas and M. N. Katehakis, "Optimal adaptive policies for markov decision processes," *Mathematics of Operations Research*, vol. 22, no. 1, pp. 222–255, 1997.

[2] O. Caelen and G. Bontempi, "A dynamic programming strategy to balance exploration and exploitation in the bandit problem," *Annals of Mathematics and Artificial Intelligence*, vol. 60, pp. 3–24, 2010.

[3] N. K. Jong and P. Stone, "Model-based exploration in continuous state spaces," in *SARA*. Springer, 2007, pp. 258–272.

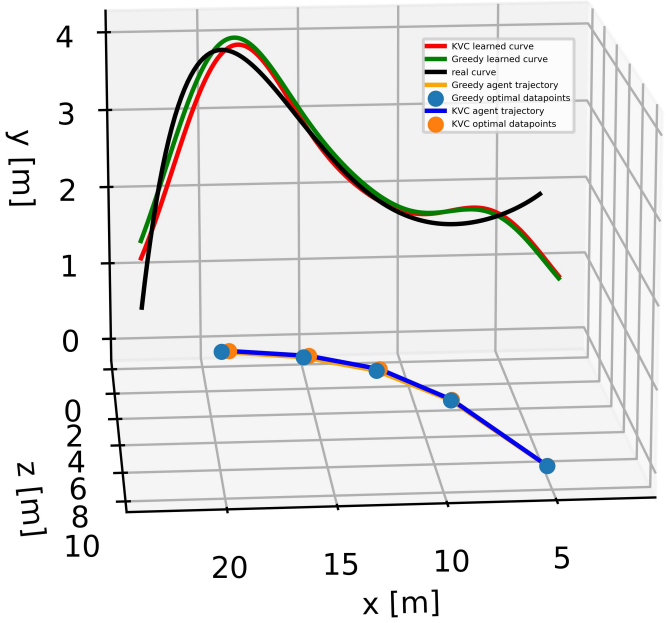

Fig. 4. Learning of a curve with trajectories generated by the KVC and greedy algorithm - Gaussian kernel (stabilizable LTI system)

[4] L. M. Zintgraf, L. Feng, C. Lu, M. Igl, K. Hartikainen, K. Hofmann, and S. Whiteson, "Exploration in approximate hyper-state space for meta reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12 991–13 001.

[5] A. Nouri and M. Littman, "Multi-resolution exploration in continuous spaces," *Advances in neural information processing systems*, vol. 21, 2008.

[6] S. M. LaValle, "From dynamic programming to rrts: Algorithmic design of feasible trajectories." *Control Problems in Robotics*, vol. 4, pp. 19–37, 2003.

[7] C. Atkeson and B. Stephens, "Random sampling of states in dynamic programming," *Advances in neural information processing systems*, vol. 20, 2007.

[8] S. Sanner, R. Goetschalckx, K. Driessens, and G. Shani, "Bayesian real-time dynamic programming," in *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-09)*. IJCAI-INT JOINT CONF ARTIF INTELL, 2009, pp. 1784–1789.

[9] Y. Shin and D. Xiu, "On a near optimal sampling strategy for least squares polynomial regression," *Journal of Computational Physics*, vol. 326, pp. 931–946, 2016.

[10] R. Wang and H. Zhang, "Optimal sampling points in reproducing kernel hilbert spaces," *Journal of Complexity*, vol. 34, pp. 129–151, 2016.

[11] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *The Annals of Statistics*, vol. 36, no. 3, pp. 1171–1220, 2008.

[12] M. L. Overton, "On minimizing the maximum eigenvalue of a symmetric matrix," *SIAM Journal on Matrix Analysis and Applications*, vol. 9, no. 2, pp. 256–268, 1988.

[13] F. Jarre, "An interior-point method for minimizing the maximum eigenvalue of a linear combination of matrices," *SIAM Journal on Control and Optimization*, vol. 31, no. 5, pp. 1360–1377, 1993.

[14] M. K. Fan and B. Nekooie, "On minimizing the largest eigenvalue of a symmetric matrix," *Linear Algebra and its Applications*, vol. 214, pp. 225–246, 1995.

[15] A. Clark, Q. Hou, L. Bushnell, and R. Poovendran, "Maximizing the smallest eigenvalue of a symmetric matrix: A submodular optimization approach," *Automatica*, vol. 95, pp. 446–454, 2018.

[16] H. Wolkowicz and G. P. Styan, "Bounds for eigenvalues using traces," *Linear algebra and its applications*, vol. 29, pp. 471–506, 1980.

[17] J. K. Merikoski and A. Virtanen, "Bounds for eigenvalues using the trace and determinant," *Linear algebra and its applications*, vol. 264, pp. 101–108, 1997.

[18] Q. Zhong and T.-Z. Huang, "Bounds for the extreme eigenvalues using the trace and determinant," *Journal of Information and Computing Science*, vol. 3, no. 2, pp. 118–124, 2008.