Abstract Dealing with high-dimensional feedback control problems is a difficult task when the classical dynamic programming principle (DPP) is applied. Existing techniques restrict the application to relatively low dimensions since the discretization techniques typically suffer from the curse of dimensionality. In this paper we introduce a novel approximation technique for the value function, which is a crucial tool for feedback control via the DPP. By running several open loop optimal control problems we are able to generate data that can be approximated by a kernel orthogonal greedy (KOGA) strategy which generates extreme sparse surrogates and enables rapid evaluations in high dimensions. Two numerical examples prove the performance of the approach and show that the method is able to deal with high-dimensional feedback control problems.

Keywords Kernel approximation, optimal feedback control, dynamic programming principle, greedy techniques
Data-driven surrogates of value functions and applications to feedback control for dynamical systems

A. Schmidt* B. Haasdonk*

* University of Stuttgart, Institute for Applied Analysis and Numerical Simulation, Pfaffenwaldring 57, 70569 Stuttgart - Germany
E-mail: {schmidt, haasdonk}@mathematik.uni-stuttgart.de

Abstract: Dealing with high-dimensional feedback control problems is a difficult task when the classical dynamic programming principle (DPP) is applied. Existing techniques restrict the application to relatively low dimensions since the discretization techniques typically suffer from the curse of dimensionality. In this paper we introduce a novel approximation technique for the value function, which is a crucial tool for feedback control via the DPP. By running several open loop optimal control problems we are able to generate data that can be approximated by a kernel orthogonal greedy (KOGA) strategy which generates extreme sparse surrogates and enables rapid evaluations in high dimensions. Two numerical examples prove the performance of the approach and show that the method is able to deal with high-dimensional feedback control problems.

Keywords: Kernel approximation, optimal feedback control, dynamic programming principle, greedy techniques

1. INTRODUCTION

Controlling technical systems is nowadays a problem of utmost importance. Examples range from small applications in our households, e.g. keeping the refrigerator at a desired temperature level, to very complex and safety critical applications like in the modern driving assistance systems or even autonomous driving features in our (future) cars.

In order to develop a controller, the real-world system is represented by a mathematical model that considers the control inputs and measurement outputs. The model is typically formulated as a system of partial differential equations (PDEs) or ordinary differential equations (ODEs). Based on those models, a control can then be obtained that satisfies certain goals like stability or minimizes prescribed performance criteria. Controllers can broadly be divided into two main groups: Open-loop and closed-loop controllers. In an open-loop control scenario, a control input is calculated once for the given scenario and state of the system, whereas a closed-loop system constantly takes the actual state into account and adjusts the control signal accordingly. It is an inherent consequence that closed-loop controllers provide greater stability properties against a-priori unknown disturbances, which is why we are primarily interested in this type of control. However, when it comes to computational and analytical complexity for obtaining the closed-loop and open-loop controllers, one clearly sees that the additional benefit of using closed-loop feedback control comes at a high computational cost compared to solving open loop systems. In this article we want to leverage the feedback control techniques by combining information gathered from few open-loop control solutions, by approximating the value function of the control problem by greedy kernel interpolation techniques.

It is due to the pioneering work of Bellman, who formulated the Dynamic Programming Principle (DPP), that we are able to formulate a closed expression to obtain feedback controls for almost all nonlinear control problems, see Bellman (1954); Bardi and Capuzzo-Dolcetta (1997). The key ingredient for feedback control via DPP lies in the value function, a function that gives at every point in the state space the minimum cost of the optimal control problem. Once this function is known or sufficiently well approximated, the feedback control problem is solved, since an analytical expression for the feedback law can be formulated based on the value function. A closed-form expression of an analytical solution to the value function can almost never be obtained, raising the need for numerical discretizations and approximations. Although many very efficient techniques exist for moderate ODE state space dimension, say \( n < 8 \), the computational complexity and memory requirements of classical grid-based schemes such as the semi-Lagrangian schemes and schemes based on sparse grids, typically increases exponentially with the dimension \( n \) of the problem. The phenomenon behind the increasing complexity is called curse of dimensionality (COD). See Alla et al. (2015); Garcke and Kröner (2017) for two popular techniques and Falcone and Ferretti (2013) for an introduction to the numerical approximation techniques. Recently, techniques that couple semi-Lagrangian schemes and kernel interpolations have been developed, see Junge and Schreiber (2015); Huang et al. (2006); Kang

* The authors would like to thank the German Research Foundation (DFG) for financial support of the project within the Cluster of Excellence in Simulation Technology (EXC 310/2) at the University of Stuttgart.
et al. (2017). However, the number of points required for these schemes still scales exponentially with the space dimension, rendering them infeasible in high dimensional, say \( n \geq 20 \).

This paper is organized as follows: We first set up the basic mathematical background about optimal feedback control and optimal open loop control in Section 2. Based on these techniques, we formulate the algorithm for approximating the value function in Section 3. Two tests in Section 4 show the performance of the approach, especially when it comes to high-dimensional problems, and highlight the open research questions, with which we close this work in Section 5.

2. SETTING AND BASIC MATHEMATICAL TECHNIQUES

2.1 Basics about Feedback Control

Throughout this article, we consider finite dimensional discrete-time optimal control problems on an infinite horizon of the form

\[
\begin{align*}
\min_{u \in \mathcal{U}} J(x, u) := & \sum_{k=0}^{\infty} g(y_k(u; x), u_k) \\
\text{s.t.} \quad & y_{k+1}(u; x) = f(y_k(u; x), u_k), \quad k \in \mathbb{N}_0 \\
& y_0(u; x) = x.
\end{align*}
\]

We assume that the state space is \( n \)-dimensional, i.e. \( y_k(u; x) \) and the initial value \( x \) are elements of \( \mathbb{R}^n \). The square-integrable sequence \( u = (u_1, u_2, \ldots) \in \mathcal{U} := \ell^2(\mathbb{U}) \) is called control and is later chosen in such a way that the cost functional \( J(x, u) \) is minimized.

The set of admissible controls \( \mathcal{U} \subset \mathbb{R}^m \) is assumed to be compact and to contain 0. The running cost function \( g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R} \) describes performance measures for the system and is typically chosen as a quadratic function, i.e. \( g(y, u) := \langle y, Qy \rangle + \langle u, Ru \rangle \) with a symmetric and positive semidefinite matrix \( Q \in \mathbb{R}^{n \times n} \) and symmetric and positive definite matrix \( R \in \mathbb{R}^{m \times m} \). The brackets \( \langle \cdot, \cdot \rangle \) denote the standard inner product on \( \mathbb{R}^n \). Throughout this paper, we always assume that all considered control problems have unique solutions, i.e. the solution to the difference equation (1) exists for all control sequences \( u \in \mathcal{U} \) and the cost functional achieves a minimum value for the (unique) optimal control \( u^* \in \mathcal{U} \) and corresponding trajectory \( y^* = (y_0^*(x; u^*), y_1^*(x; u^*), \ldots) \). We refer to Bardi and Capuzzo-Dolcetta (1997) for the technical details and verifications about these assumptions.

We first focus on the feedback control case, i.e. we try to find a feedback map \( F : \mathbb{R}^n \to \mathcal{U} \) from the state space to the set of admissible controls \( \mathcal{U} \) such that the closed loop system \( y_{k+1} = f(y_k, F(y_k)) \) is stabilized. Fortunately there exists a famous technique which allows us to directly characterize the solution to the feedback control problem. For that purpose we follow Bellman (1954) and his dynamic programming principle (DPP) and define the optimal value function \( v : \mathbb{R}^n \to \mathbb{R} \) of the control problem as

\[
v(x) := \inf_{u \in \mathcal{U}} J(x, u), \quad x \in \mathbb{R}^n.
\]

By the DPP we are able to formulate the following functional equation, which is also known as the Bellman equation, for the unknown value function \( v(x) \):

\[
v(x) = \inf_{a \in \mathcal{U}} \{ v(f(x, a)) + g(x, a) \}, \quad x \in \mathbb{R}^n. \tag{2}
\]

The value function is a great tool to synthesize feedback control, since, once it is known, the optimal control problem is solved very easily by picking a control that minimizes the right hand side of the Bellman equation, i.e. which satisfies

\[
u(x) \in \arg\min_{u \in \mathcal{U}} \{ v(f(x, a)) + g(x, a) \}. \tag{3}
\]

Several techniques exist to approximate the value function: As mentioned earlier, the numerical approximation of the value function via classical techniques suffers from several problems, most dominantly the curse of dimensionality. This results from Equation (2) being defined on an \( n \) dimensional space which has to be discretized, for example via grids, which is clearly infeasible for high-dimensional systems \( n \). This is indeed very restrictive, especially when it comes to systems that stem from semidiscretized PDEs, such as in our second example in Section 4. Different approaches that couple model order reduction techniques and control problems to end up with low-dimensional systems that are feasible for the numerical approximation have been tackled as a remedy, see Alla et al. (2017) for a recent comparison. However, the curse of dimensionality is always present and cannot be overcome by methods that rely on grid-based approximations.

2.2 Open-Loop Optimal Control

As we have seen in the previous section, although feedback control has great advantages in practice, it is a very expensive task when it comes to numerical computations. If we refrain from wanting to get feedback controls but only optimal open-loop control problems things become easier. We will thus briefly present one technique to solve open-loop control problems and show how this can help in the feedback control context in Section 3.

First of all, for computational procedures, we have to fix some finite horizon \( K \in \mathbb{N} \). We define the open-loop control problem on this finite horizon with the same functions as in the infinite horizon case and furthermore introduce a final weight term \( G : \mathbb{R}^n \to \mathbb{R}_+ \) which penalizes the state at the final step \( K \):

\[
\begin{align*}
\min_{u \in \mathcal{U}_K} J_K(x, u) := & \sum_{k=0}^{K-1} g(y_k(u; x), u_k) + G(y_K(u; x)) \\
\text{s.t.} \quad & y_{k+1}(u; x) = f(y_k(u; x), u), \quad k \in \mathbb{N}_0, k \leq K - 1, \\
& y_0(u; x) = x.
\end{align*}
\]

Here, the set \( \mathcal{U}_K = (u_0, u_1, \ldots, u_{K-1}) \) with \( u_i \in \mathcal{U} \) is the finite counterpart to \( \mathcal{U} \). Solving problems of the above type are often doable and, depending on the implementation and technique used for the numerical solution, it can be done rather efficiently. One technique which is very appealing, both analytically and numerically is the maximum principle of Pontriagin, that allows us to rephrase the optimal control problem into a boundary value problem for a difference equation, see for example Bryson (1975).
In the following, we assume that an optimal control policy \( u^* \) together with the optimal state trajectory \( y^* \) exists. We then define the Hamiltonian of the optimal control system as
\[
H(y, u, p) := g(y, u) + p^T f(y, u),
\]
where \( p \in \mathbb{R}^n \). Based on the Hamiltonian and its partial derivatives \( H_y, H_u \), we can write down the optimality system for the state \( y^* \), the optimal control sequence \( u^* \) and the costate (or adjoint state) sequence \( p = (p_0, p_1, \ldots, p_K) \):
\[
y_{k+1}(u^*; x) = H_y(y_k^*(u^*; x), u_k^*, p_{k+1}),
\]
\[
p_k = H_u(y_k^*(u^*; x), u_k^*, p_{k+1})),
\]
\[
u_k^* = \arg \min_{a \in U} H(y_k^*(u; x), a, p_{k+1}).
\]
Together with the boundary conditions \( y_0^*(u^*; x) = x \) and \( p_K = G_p(y_K(u^*; x)) \) these equations form a BVP for the system of difference equations that can be solved numerically.

### 2.3 Kernel Interpolation

We will use kernel-based approximation techniques, see for example Wendland (2005), since they are less prone to the curse of dimensionality. The basic kernel interpolation method works in the following way: Given a set of data points \( x_1, \ldots, x_\ell \in \mathbb{R}^n \) and corresponding target values \( v_1, \ldots, v_\ell \), we define an interpolant
\[
V(x) := \sum_{i=1}^\ell \alpha_i \Phi(||x - x_i||; p).
\]
Here, the function \( \Phi : \mathbb{R}^n_+ \rightarrow \mathbb{R} \) is a so called radial basis function (RBF), and \( p \) is an additional shape parameter, that can be chosen during the construction of the kernel expansion to match the data as good as possible. Typical examples for RBFs are the Gaussian \( \Phi(r; p) = e^{-(pr)^2} \) or the thin-plate spline \( \Phi(r; p) = (rp)^2 \ln rp \). To obtain the weights \( \alpha_i \) in the above expansion, we can set up and solve the linear system \( A\alpha = v \) for the values \( \alpha = (\alpha_1, \ldots, \alpha_\ell)^T \), with the matrix \( A_{ij} = \Phi(||x_i - x_j||^2; p) \) for \( 1 \leq i, j \leq \ell \) and the right hand side vector \( v = (v_1, \ldots, v_\ell)^T \in \mathbb{R}^\ell \). Note that this procedure can be carried out “offline”, i.e. it has to be performed only once. The evaluation of the interpolant is then very fast, as it only consists of evaluating equation (7). It is obvious that the quality of the interpolation greatly depends on the RBF \( \Phi \) and the parameter \( p \).

### 3. THE ALGORITHM

We will now present a novel approach for approximating the value function \( v(x) \) by using information from open loop optimal control solutions.

Consider an \( n \)-dimensional optimal control problem on an infinite horizon of the form (1) for which we want to determine a feedback control law via Equation (3). For that purpose, we need either the true value function or a suitable approximation to it.

The idea of the proposed algorithm is to approximate the infinite horizon optimal control problem by a finite horizon problem for which we can easily obtain optimal (open loop) controls. If we do this for a couple of different initial configurations, we are able to generate a large amount of data pairs \( (x_i, v(x_i)) \), i.e. of points and value function values. This data can then be used to set up a kernel approximant.

As a first step, we reformulate the infinite horizon control problem (1) as a finite horizon problem (4)–(5), where the horizon \( K \) should be chosen large enough to mimick the behaviour of the system on the infinite horizon. The penalization of the state at the final time can be chosen arbitrarily for example as \( G(y) := \|y\|^2 \) or simply as \( G(y) := 0 \). Given the reformulated system and an initial state \( x \in \mathbb{R}^n \), we solve the open-loop optimal control problem on the finite horizon and obtain the optimal policy and the corresponding state trajectory, denoted as \( u^* \) and \( y^*(u^*; x) \), respectively. Based on the calculated optimal solution, we can evaluate the cost functional (4) to get the approximate value of the value function at the point \( x \), denoted by \( v^* \). The DPP states that the remainder of the optimal control policy \( (u^*_{1, \ell}, u^*_{2, \ell}, \ldots) \) is optimal, when started at the point resulting from the optimal control up to time \( L \), i.e. from \( y^*_L(x) \). Therefore, we can define the following sequence, giving us the value function along the whole trajectory \( y^*(u^*; x) \):
\[
v_k^* := J_K(x, u) - \sum_{i=0}^k g(y_k^*(u^*; x), u^*), \quad k = 0, 1, \ldots, K.
\]

In other words, once the optimal control problem is solved, we get the value function not only at the initial state \( x \) but along the whole optimal trajectory. The application of the DPP thus provides us with an avalanche of data points and corresponding value function values which can be used in the subsequent kernel approximation step.

As a preprocessing step, we choose a set of initial values \( X = [x_1, \ldots, x_T] \) for a fixed number of training samples \( T \in \mathbb{N} \). For all elements \( x \in X \) we then solve the open-loop optimal control problems (4)-(5), calculate the value function along the solution trajectory and store the results as pairs \( (y^*, v^*) \). The algorithm for generating the data for the later kernel interpolation is summarized in Algorithm 1. Note that from an implementation point of view, the loop can be perfectly parallelized or run on a cluster or the cloud, such that the data generation can be sped up easily. Based on the data we are now able to formulate the algorithms that we use to create the interpolation.

### Algorithm 1 Data Generation for Kernel Approximation

1: function GenerateData(\( X \))
2: \( \text{Data} \leftarrow \{\} \)
3: for all \( x \in X \) do
4: \( y^*, u^* \leftarrow \) solution of (4)-(5) for initial state \( x \)
5: \( v^* \leftarrow \) calculate from Equation (8)
6: \( \text{Data} \leftarrow \text{Data} \cup \{(y^*, v^*)\} \)
7: end for
8: return Data
9: end function

A first natural implementation of the interpolation scheme is given by applying the kernel orthogonal f-greedy algorithm (KOFA), e.g. Wirtz and Haasdonk (2013). To this
end we define large matrices $Y \in \mathbb{R}^{n \times (T \cdot K)}$, $V \in \mathbb{R}^{1 \times (T \cdot K)}$ where we collect all precalculated data points, i.e. we combine all trajectories in $Y$ and the value function values along the trajectories in $V$. Based on these matrices, we can run the KOGA algorithm with prescribed tolerances, which we specify in Section 4 for each example. The result is a model that is very sparse, i.e. most of the $\alpha_i$ in (7) are 0, hence the surrogate can be evaluated very rapidly.

Note that, depending on the discretization, the number of points $K \cdot T$ available for the greedy loop for the KOGA algorithm can become very large when a large number of training points or a long control horizon is chosen. We thus propose an alternative to computing an approximation once for all training points, by incrementally building the approximant from single trajectories and corresponding value function values. The pseudocode for this approach is given in Algorithm 2. An outer loop walks over all training sequences $(y,v)$, generated from Algorithm 1. For each tuple we then run greedy algorithm: Let the absolute error for the trajectory $y$ at the time instance $k$ be defined by $\Delta(y,v,k) := |y_k - \tilde{v}(y_k)|$. We incrementally pick the point that maximizes this error indicator and include the maximized point and the target value to the training set for the kernel interpolation of $X,V$. By using this incremental technique, the computation of the approximant algorithm will run faster since it does not need to evaluate the interpolation over all $T \cdot K$ points after each extension of the kernel training data.

In all kernel interpolations we always explicitly include the point $0 \in \mathbb{R}^n$ together with the zero target value to the expansion, since we want to make sure that we always get $\tilde{v}(0) = v(0) = 0$, which is crucial for stability. Furthermore, the kernel interpolation does not consider positivity constraints. Indeed, we observed values $\tilde{v}(x) < 0$, which delivers improper interpolation values and feedback signals. Therefore, we limit the interpolation to $[0, \infty)$ by using $\max(\tilde{v}(x),0)$ at all relevant places. In particular, the feedback control from the kernel interpolation that we use in all experiments is defined via

$$\tilde{u}(x) \in \arg \min_{u \in U} \{ \max(\tilde{v}(f(x,a)),0) + g(x,a) \}.$$  

Algorithm 2 Multi-Stage Greedy Kernel Value Function Approximation

1: function MultiStageKOGA(Data)  
2: $V \leftarrow [0]$  
3: $X \leftarrow [0_{R^n}\in] \in \mathbb{R}^n$  
4: $\tilde{v}(x) := 0, x \in \mathbb{R}^n$  
5: for all $(y,v) \in Data$ do  
6: while $\max_{k=0,...,K} \Delta(y,v,k) > \varepsilon$ do  
7: $k \leftarrow \arg \max_{k=0,...,K} \Delta(y,v,k)$  
8: $V \leftarrow [V,v_k], X \leftarrow [X,y_k]$  
9: $\tilde{v} \leftarrow$ kernel interpolation of $X,V$  
10: end while  
11: end for  
12: return $\tilde{v}(x)$  
13: end function

### 4. NUMERICAL EXAMPLES

We now present two examples that show the performance of the presented approach. Both examples are systems of nonlinear ordinary differential equations that have an unstable equilibrium at the origin $y = 0$ which we want to reach by applying the approximated feedback control from Section 3. All calculations are carried out on a machine with eight Intel(R) Core(TM) i7-6700 CPUs with 3.4 GHz and 16 GB memory.

#### 4.1 Van der Pol Oscillator

The first model under consideration is the so-called Van der Pol oscillator. The system is governed by the nonlinear differential equation $\ddot{x} = (1 - x^2)\dot{x} - x + u$, which after rewriting as a first-order system and temporal discretization via an explicit Euler scheme with time step $\Delta t = 10^{-2}$ gives the right hand side of the difference equation in (1) for $y = (y^{(1)},y^{(2)})^T$.

$$f(y,u) = y + \Delta t \left((1 - y^{(1)^2})y^{(2)} - y^{(1)} + u\right).$$

We introduce the standard quadratic cost functional by defining $g(y,u) := \|y\|^2 + 1/2u^2$ and restrict the set of admissible controls to $U = [-1,1]$. The dynamics of this system have a characteristic oscillating structure: For all initial values $x \neq 0$, the dynamics evolve towards a limit cycle around the origin. Figure 1 shows two examples for the uncontrolled and controlled dynamics, as well as the control action that was required to drive the system to the (almost) zero state. Note that the norm of the solution vector of both open loop optimal controls at the final time $k = K$ is in the magnitude of $10^{-8}$, a continuation of the simulation for $k > K$ without applied control will again let the system evolve towards the limit cycle. This again shows the relevance of feedback control for this example.

For calculating the approximation, we have to restrict ourselves to a bounded domain, although the value function is defined on the whole $\mathbb{R}^n$. We therefore choose the box $\Omega := [-2,2]^2$ as our domain of interest. As a benchmark and for comparison purposes, we calculate the true value function by applying a value iteration (VI) algorithm.

![Fig. 1. Top: Uncontrolled and controlled states for two different initial conditions $x_1 = (0.5,1)^T$, $x_2 = (-1,2.5)^T$ for the VDP example. The solid lines are uncontrolled, dotted is the open-loop controlled state. Bottom: The corresponding control signals.](image-url)
VI basically calculates an approximation to the value function, by applying a fixed-point iteration to Equation (2), together with a suitable piecewise linear interpolation for the value function on a uniform grid, see Falcone and Ferretti (2013) for details. The calculation of the “true” value function was carried out on a discretization of $\Omega$, consisting of $301^2 = 90,601$ points. The resulting value function is plotted in Figure 2.

We solve the open-loop optimal control problems for $K = 2 \cdot 10^3$, i.e. up to the final time $t = 20$, by calculating the solution to the boundary value problem resulting from the first-order optimality conditions in Equation (6). We solve the continuous-time optimal control problem (i.e. for the original differential equation and the time-continuous adjoint equation) by using MATLAB’s bvp5c solver and then calculate the discrete-time values. Our numerical experiments confirm the feasibility of this ansatz, as we observed only negligible errors compared to solving the fully discrete system.

For applying the kernel interpolation technique described in this paper, we first choose a training set for the initial conditions. We pick a discretization parameter number $N \in \mathbb{N}$, define the step size $\delta := 5/N$ and define the training set as $A_N := \{y \in \Omega, y^T = (-2.5,-2.5) + (i \delta,j \delta), i,j = 0, \ldots, N\}$. For all following examples, we choose a tolerance of $\varepsilon = 0.05$ for the algorithms and pick the radial basis function $\Phi(r,p) := r \cdot p$. As a first test, we investigate the approximation error and the computation times with an increasing number of training points. The results are printed in Table 1. The error in both cases is measured in the relative $L^2(\Omega)$ norm: $E := \|v - \tilde{v}\|_{L^2(\Omega)}/\|v\|_{L^2(\Omega)}$. We see a similar decay in the approximation error for both algorithms with slightly better results for the KOGA algorithm, which is not surprising since it performs a global optimization over the whole training set. However, comparing the calculation times and the number of points that were included in the kernel expansion reveals the differences: The row entitled “Expansion” lists the number of centers for the kernel expansions that were picked by the algorithms. The expansion from the KOGA algorithm achieves the desired tolerance with fewer points compared to the other algorithm. Regarding the calculation times, we observe that the multi stage KOGA algorithm is about 2.3 times faster than the plain KOGA for the finest training set that was chosen during the test, see Table 1.

4.2 Advection Diffusion Equation with Nonlinear Reaction

We now present a second example, that demonstrates the strength of the proposed method when it comes to the approximation of value functions for high-dimensional control problems.

We consider the following nonlinear PDE on the domain $\Omega := (0,1)$ for the unknown state $w(\xi, t)$ with the scalar control input $u(t) \in U = \mathbb{R}$ and the constants $\alpha = 0.1$ and $\rho = 6$:

$$w_t - \alpha w_{xx} + w_x - \rho (w - w^3) + 1_{[0,2,0,7]} u = 0,$$

$$w(0,t) = w(1,t) = 0, t > 0,$$

$$w(\xi, 0) = w_0, \xi \in \Omega.$$

The operator $1_A$ is the characteristic function of the set $A$. We discretize the equation with a finite difference scheme into $n = 20$ points. A subsequent explicit Euler discretization with $\Delta t = 2.5 \cdot 10^{-2}$ yields the fully discrete system. The control objective is again a quadratic cost functional with $\Delta t = 2.5 \cdot 10^{-2}$ and the constants $\alpha = 0.1$ and $\rho = 6$.

$$w_t - \alpha w_{xx} + w_x - \rho (w - w^3) + 1_{[0,2,0,7]} u = 0,$$

$$w(0,t) = w(1,t) = 0, t > 0,$$

$$w(\xi, 0) = w_0, \xi \in \Omega.$$

For this example, we again choose the kernel $\Phi(r,p) := pr$ with a parameter that is selected by minimal training error and fix the tolerance $\varepsilon = 0.05$. We then again pick training sets of different size $N$, containing uniformly distributed random vectors in the box $[-1,1]^n$.

In this example, we compare the numerical evaluation of the cost functional for the true simulation, where we apply the control from the open-loop calculation, and the feedback controlled system, where we use our approximation of the value function in (3). We choose a test set consisting of 50 random vectors in the box $[-1,1]^n$, distinct from the training set. The results are presented in Table 4.2. We only show the results for the multi

---

**Fig. 2.** True value function from applying a value iteration scheme.

**Fig. 3.** Approximation of the value function. The black dots indicate the points picked by the KOGA greedy algorithm.
Table 1. Results for the approximation of the value function for the Van der Pol model.

| Expansion | Time $t$ | Error $E$ |
|-----------|----------|-----------|
| 10        | 1.2 s    | $2.55 \times 10^{-1}$ |
| 20        | 1.6 s    | $2.16 \times 10^{-1}$ |
| 30        | 1.9 s    | $1.29 \times 10^{-1}$ |
| 40        | 2.4 s    | $8.26 \times 10^{-2}$ |
| 50        | 2.9 s    | $5.43 \times 10^{-2}$ |

Table 2. Results for the nonlinear heat equation model. The last line $N = 42$ includes the two unstable equilibriums.

| # unstable | Runtime | Mean error |
|------------|---------|------------|
| 5          | 18      | 1.5 s      |
| 10         | 16      | 4.7 s      |
| 40         | 6       | 44.5 s     |
| 42         | 0       | 46.8 s     |

Fig. 4. Solution of the state for the nonlinear advection-diffusion-reaction model at initial and final time. The light lines in the background indicate the temporal evolution of the state.

5. CONCLUSION

In this paper we discussed a novel approach for solving high dimensional feedback control problems. The method relies on an interpolation of the value function of the problem by kernel methods, which are less prone to the curse of dimensionality than grid-based discretizations. The data for the interpolation was obtained from multiple open-loop optimal control solutions. The results obtained in the numerical examples look promising, especially when it comes to high-dimensional problems, stemming for example from discretized PDE problems. However, the method certainly can be improved at many points: The positivity constraint is now strictly enforced by cutting the interpolation. A better approach would be to directly enforce the interpolation to be positive. An error indicator based selection of the initial values for the optimal control solutions can furthermore add an additional benefit when it comes to runtime and approximation quality. Additional constraints like monotonicity along the optimal trajectory should be considered analytically and in the implementation, to ensure stability of the feedback controlled system.

REFERENCES

Alla, A., Falcone, M., and Kalise, D. (2015). An efficient policy iteration algorithm for dynamic programming equations. *SIAM Journal on Scientific Computing*, 37(1), A181–A200. doi:10.1137/130932284.

Bardi, M. and Capuzzo-Dolcetta, I. (1997). *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*. Birkhäuser Boston. doi:10.1007/978-0-8176-4755-1.

Bellman, R. (1954). The theory of dynamic programming. *Bull. Amer. Math. Soc.*, 60(6), 503–515.

Bryson, A.E. (1975). *Applied optimal control: optimization, estimation and control*. CRC Press.

Falcone, M. and Ferretti, R. (2013). *Semi-Lagrangian Approximation Schemes for Linear and Hamilton—Jacobi Equations*. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611973051.

Garcke, J. and Kröner, A. (2017). Suboptimal feedback control of PDEs by solving HJB equations on adaptive sparse grids. *Journal of Scientific Computing*, 70(1), 1–28. doi:10.1007/s10915-016-0240-7.

Huang, C.S., Wang, S., Chen, C., and Li, Z.C. (2006). A radial basis colocation method for Hamilton Jacobi Bellman equations. *Automatica*, 42(12), 2201 – 2207.

Junge, O. and Schreiber, A. (2015). Dynamic programming using radial basis functions. *Discrete Contin. Dyn. Syst.*, 35(9), 4439–4453. doi:10.3934/dcds.2015.35.4439.

Kang, W., Yakimenko, O., and Wilcox, L. (2017). Optimal control of uavs using the sparse grid characteristic method. In *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*, 771–776. doi:10.1109/ICCAR.2017.7942802.

Wendland, H. (2005). *Scattered Data Approximation*, volume 17 of Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge.

Wirtz, D. and Haasdonk, B. (2013). A vectorial kernel orthogonal greedy algorithm. *Dolomites Res. Notes Approx.*, 6, 83–100. Proceedings of DWCAA12.