

Multi-level Optimal Control with Neural Surrogate Models[★]

Dante Kalise^{*} Estefania Loayza-Romero^{*}
Kirsten A. Morris^{**} Zhengang Zhong^{***}

^{*} *Department of Mathematics, Imperial College London (e-mail: {d.kalise-balza,k.loayza-romero}@imperial.ac.uk)*

^{**} *Department of Applied Mathematics, University of Waterloo (e-mail: kmorris@uwaterloo.ca)*

^{***} *Centre for Process Systems Engineering, Imperial College London (e-mail: z.zhong20@imperial.ac.uk)*

Abstract: Optimal actuator and control design is studied as a multi-level optimisation problem, where the actuator design is evaluated based on the performance of the associated optimal closed loop. The evaluation of the optimal closed loop for a given actuator realisation is a computationally demanding task, for which the use of a neural network surrogate is proposed. The use of neural network surrogates to replace the lower level of the optimisation hierarchy enables the use of fast gradient-based and gradient-free consensus-based optimisation methods to determine the optimal actuator design. The effectiveness of the proposed surrogate models and optimisation methods is assessed in a test related to optimal actuator location for heat control.

Copyright © 2024 The Authors. This is an open access article under the CC BY-NC-ND license (<https://creativecommons.org/licenses/by-nc-nd/4.0/>)

Keywords: Optimal actuator design, Optimal control of distributed parameter systems, Supervised learning and neural networks, Consensus-based optimisation.

1. INTRODUCTION

In engineering, an actuator is a device that materialises the control action within a physical system. Actuators can be mechanical, electrical, hydraulic, or magnetic (Kalise et al., 2018). The design of actuators (and sensors) is a fundamental engineering challenge arising, for instance, in vibration control through the design of piezoelectric actuators (Peng et al., 2005), and in active noise cancellation in automobiles and aircraft (Morris, 1998).

From a mathematical viewpoint, actuator design precedes the standard control design paradigm where, for a given control system, a control signal is synthesized to achieve a desired performance. Actuator design is concerned with the specification of a control-to-state map which determines the capabilities of the control signal. In this work, we follow an optimisation-based approach for actuator design and control synthesis. We develop a multi-level optimisation pipeline where the actuator design is at highest level of the hierarchy, and is optimised according to the performance of the associated optimal closed loop, which corresponds to the lower level problem.

We are interested in optimal actuator design problems where the underlying system dynamics are governed by partial differential equations (PDEs). This has been extensively studied in Morris (2010) for linear quadratic control problems, in Morris and Demetriou (2010); Kasiathan and Morris (2013) for H_2 and H_∞ controller design

objectives, in Privat et al. (2017) following a spectral approach, and in Kalise et al. (2018) in the framework of shape/topology optimisation. In our multi-level approach, this translates into a computationally demanding lower level problem, requiring the solution in the linear quadratic case of a large-scale Algebraic Riccati Equation (ARE), which renders evaluations of the higher level problem prohibitively expensive.

We propose the use of neural network surrogates in our multi-level optimal design/control pipeline to alleviate the computational burden associated to the lower level optimal control problem. The solution of this lower level problem is characterized by a parameter-dependent value function, where the parametric dependence is related to the higher level optimal actuator design. The evaluation of the value function and its gradient, which are intensively used for the parametric optimisation, is approximated by the use of a neural network trained by supervised learning as proposed in Nakamura-Zimmerer et al. (2021); Albi et al. (2022); Azmi et al. (2021). The use of a surrogate for the value function leads to a significant acceleration of the evaluation time for the lower level problem, enabling the use of both gradient-based and gradient-free methods for the solution of the higher level optimisation problems. This paper is the first step towards a more comprehensive workflow dealing with optimal actuator design for nonlinear PDEs.

The rest of the paper is structured as follows. In Sec. 2, we introduce the multi-level optimisation framework for optimal actuator and control design. In Sec. 3, we develop neural network surrogates to approximate the value function associated to the lower level of our hierarchy.

[★] This research was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/T024429/1 and a Discovery Grant from NSERC (Canada).

We propose the construction of both unstructured and structured surrogates, and evaluate their performance in a prototypical example. Having built a suitable surrogate, Sec. 4 is devoted to the presentation of optimisation algorithms to solve the min-max problem arising in the higher level of our hierarchy. We present a projected gradient descent ascent and a consensus-based method for saddle point problems and we illustrate their performance in a problem related to optimal actuator location for a thermal system.

2. A MULTI-LEVEL OPTIMISATION FRAMEWORK FOR OPTIMAL ACTUATOR/CONTROL DESIGN

We consider parameter-dependent linear dynamical systems described by

$$\frac{dz}{dt} = Az(t) + B(r)u(t), \quad z(0) = z_0 \quad (1)$$

where $A \in \mathbb{R}^{n \times n}$, and $B(r) \in \mathbb{R}^{n \times m}$ is matrix-valued function depending on the actuator location $r \in \mathbb{R}^m$. Such systems naturally arise after semi-discretization in space of systems governed by PDEs, where r parametrizes the location of m actuators. For the sake of simplicity, we assume that the control space is of the same dimension as the parameter space. However, the methodology developed in this work can be seamlessly adapted to work over spaces of different dimensions. We assume that each parameter coordinate can be varied over some compact set $\Omega \subset \mathbb{R}^m$.

The linear quadratic controller design aims at finding a minimising control $u(t) \in \mathbb{R}^m$ to the cost functional

$$J(u; z_0, r) = \int_0^\infty [z(t)^\top Q z(t) + u(t)^\top R u(t)] dt, \quad (2)$$

where $Q \in \mathbb{R}^{n \times n}$, $Q \succeq 0$, $R \in \mathbb{R}^{m \times m}$, $R \succ 0$, and $z(t) \in \mathbb{R}^n$ is determined by the dynamics (1).

For each parametric realisation r we assume that the pairs $(A, B(r))$ are stabilisable. The optimal cost-to-go or value function for a given initial condition z_0 and parameter r is

$$V(z_0, r) := \min_{u \in \mathcal{U}} J(u; z_0, r) = z_0^\top \Pi(r) z_0, \quad (3)$$

with $\mathcal{U} = L^2((0, +\infty); \mathbb{R}^m)$, denotes the linear space of square-integrable functions defined from $(0, +\infty)$ to \mathbb{R}^m . Note that expressing the value function as a quadratic function of z is only valid for linear-quadratic problems. Moreover, in this case, $\Pi(r)$ solves the parameter-dependent Algebraic Riccati Equation (ARE)

$$A^\top \Pi(r) + \Pi(r)A - \Pi(r)B(r)R^{-1}B(r)^\top \Pi(r) + Q = 0. \quad (4)$$

We cast the actuator design problem as a parametric optimisation problem,

$$\min_{r \in \Omega^m} V(z_0, r) = \min_{r \in \Omega^m} z_0^\top \Pi(r) z_0. \quad (5)$$

However, in general, the initial condition z_0 is not fixed. For example, we can optimise r according to the worst possible initial condition, that is,

$$\max_{\substack{z_0 \in \mathbb{R}^n \\ \|z_0\|=1}} \min_{u \in \mathcal{U}} J(u; z_0, r) = \max_{\substack{z_0 \in \mathbb{R}^n \\ \|z_0\|=1}} z_0^\top \Pi(r) z_0. \quad (6)$$

We refer the reader to Morris (2010); Edalatzadeh et al. (2021); Kalise et al. (2018) for the characterizations of solutions to the optimal actuator design problem.

Overall, the simultaneous optimisation of the control signal and the parameter, together with an optimality-based

characterisation of the space of initial conditions of interest, induces a hierarchy of costs which are cast as a multi-level optimisation problem

$$\min_{r \in \Omega^m} \max_{\substack{z_0 \in \mathbb{R}^n \\ \|z_0\|=1}} \min_{u \in \mathcal{U}} J(u; z_0, r) = \min_{r \in \Omega^m} \max_{\substack{z_0 \in \mathbb{R}^n \\ \|z_0\|=1}} V(z_0, r). \quad (7)$$

To eliminate the burden of computing the solution of the ARE (4) at every evaluation of $V(z_0, r)$ within an algorithm for the outer min-max problem, we propose to build a closed-form surrogate for $V(z_0, r)$ using neural networks.

3. SURROGATES FOR THE VALUE FUNCTION WITH NEURAL NETWORKS

In a nutshell, deep learning is about realising complex tasks such as speech, or image recognition, and language translation, among others, by means of parameterised functions called neural networks (NNs). A neural network consists of neurons that are ordered into layers. We have three different kinds of layers: input, hidden and output layers (Svozil et al., 1997). Here, we are interested in building neural network surrogates $V_\theta(z_0, r) : \mathbb{R}^{n+m} \rightarrow \mathbb{R}$ for the value function $V(z_0, r)$, described in (3). In what follows, all the neural networks considered are single hidden layer feedforward NN with ReLu and/or `softplus` as activation functions.

3.1 Unstructured Surrogates

To train the neural network, we collect $N_z \times N_r$ joint pairs for the initial state z_0 and the design parameter r from a uniform sampling of $\|z_0\| \leq 1$ and Ω^m , respectively. To generate output data, we calculate $\hat{y} := z_0^\top \Pi(r) z_0$ for each pair of initial state and design parameters. To train the neural network parameters θ , we consider the mean squared error loss function

$$\mathcal{L}(\theta) = \frac{1}{N_z \times N_r} \sum_{i=1}^{N_z \times N_r} \left(V_\theta(z_0^{(i)}, r^{(i)}) - \hat{y}^{(i)} \right)^2. \quad (8)$$

It is worth highlighting that from optimal control theory one expects the value function to be non-negative since the solution of the ARE is positive semi-definite. However, with this unstructured approach, this cannot be guaranteed. In the following, we propose a structured surrogate model by approximating the matrix $\Pi(r)$ instead.

3.2 Structured Surrogates

As the value function (3) is a quadratic function of the initial state and requires the solution of the ARE (4), we consider alternatively to construct a NN directly approximating $\Pi(r)$, which is a positive semi-definite matrix, thus guaranteeing the non-negativeness of the value function.

We define a NN surrogate model to learn the unique Cholesky decomposition of symmetric positive definite matrices L_θ and the solution of ARE (4) is approximated by $\Pi(r) \approx \Pi_\theta := L_\theta(r)L_\theta^\top(r)$. The diagonal elements of $L_\theta(r)$ are expected to be positive; thus, we define the matrix valued surrogate

$$L_{\theta}(r) = \begin{bmatrix} f_{11}(r) & & & 0 \\ \vdots & \ddots & & \\ f_{n1}(r) & \dots & f_{nn}(r) & \end{bmatrix}$$

where

$$f_{ij} = \begin{cases} \varphi(NN_{ij}) + \varepsilon, & \text{if } i = j, \\ NN_{ij}, & \text{otherwise.} \end{cases} \quad (9)$$

with NN_{ij} a two-layer neural network mapping r to a scalar, φ an activation function and ε a small enough positive constant used to guarantee the diagonal entries of the matrix are positive. A similar concept of learning a symmetric positive definite NN was previously explored by (Xu et al., 2021).

For the purpose of training, we generate N design parameter samples from Ω^m and thereby the corresponding Riccati solutions $\Pi(r_i)$. The loss function is hence defined as

$$\mathcal{L}(\theta) = \sum_{i=1}^N \|\Pi_{\theta}(r_i) - \Pi(r_i)\|_{\text{fro}}^2,$$

where $\|\cdot\|_{\text{fro}}$ is the Frobenius norm.

3.3 Numerical verification

We consider a prototypical example related to optimal actuator location problem for a heating system, similarly as in Kalise et al. (2018). After semi-discretisation in space, the finite-dimensional linear dynamics are

$$\begin{aligned} \frac{dz}{dt} &= Az(t) + B(r)u(t), & t \in [0, +\infty[, \\ z(0) &= z_0, \end{aligned} \quad (10)$$

where $A := KM^{-1}$, with

$$\begin{aligned} K_{ij} &= \int_0^{\pi} \phi'_i(x)\phi'_j(x) dx & M_{ij} &= \int_0^{\pi} \phi_i(x)\phi_j(x) dx \\ B(r)_{i\ell} &= \int_{r_{\ell}-\delta\pi}^{r_{\ell}+\delta\pi} \phi_i(x) dx, \end{aligned}$$

with the functions $\phi_i(x) = \sin(ix)$ on $[0, \pi]$, for all $i, j = 1 \dots, n$ and $\ell = 1 \dots, m$. The matrices appearing in (2) are given by $Q = M$ and $R = \text{id}_{m \times m}$. Here, the parameter r represents the location of m patches of length 2δ inside the domain, each one associated to a scalar control signal. Moreover, each entry of $z(t)$ is commonly referred as a “mode”.

In the following, we demonstrate the performance of unstructured and structured surrogates $V_{\theta}(z_0, r)$ for a problem with $n = 3, \delta = 0.005$, and $m = 1$, i.e., a single actuator. In this work, all NN surrogates are built using PyTorch.

For the unstructured surrogate, we consider $\hat{z}_0 := \prod_{j \in 1 \dots n} \mathcal{Z}_j$, where $\mathcal{Z}_j := \{-1 + (i-1)/3.5 \mid i \in (1, \dots, 8)\}$ and $\hat{r} := \{i\pi/100 \mid i \in (1, \dots, 99)\}$. We calculate the value $z_0^{\top} \Pi_n(r) z_0$ from $\{(z_0, r) \mid z_0 \in \hat{z}_0, r \in \hat{r}\}$, thereby 51200 input and output data for training the neural network. We construct the surrogate neural network with one hidden layer with 128 neurons and ReLu as the activation function. The training optimisation is solved with the Adam solver with the learning rate $1e-3$ and 223000 iterations.

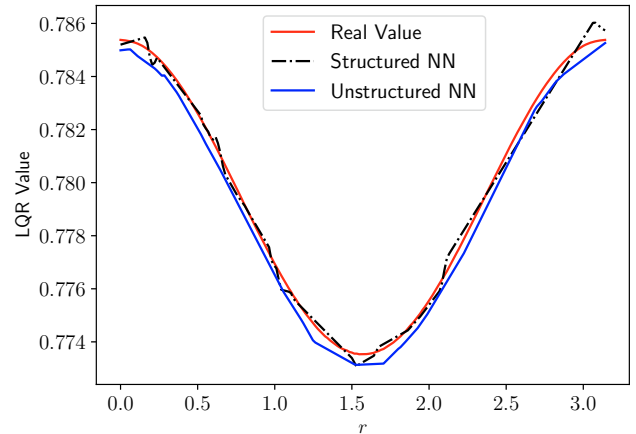


Fig. 1. Worst-case LQR values corresponding to true $\max_{\|z_0\|=1} V(z_0, r)$, structured and unstructured surrogates $\max_{\|z_0\|=1} V_{\theta}(z_0, r)$ for $m = 1$ and $n = 3$. Both surrogates effectively approximate the value function without violating the non-negativeness constraint, the structured surrogate generates a more accurate solution.

For the structured surrogate we compute $\Pi(r)$ at $r \in \hat{r} := \{i\pi/120 \mid i \in (1, \dots, 119)\}$, thereby 120 solutions of the Riccati equation are applied for training the neural network. Each element of the surrogate matrix takes NN with one hidden layer with 128 neurons. It uses ReLu as the activation function and φ as softplus (function guaranteeing the positive definiteness). The parameter ε is set to $1.192e-7$, i.e. the default smallest representable number for float32 in PyTorch. The training optimisation is solved with the Adam solver with the learning rate $1e-3$ and 6000 iterations.

In Fig. 1, we show the values of the real value function and the approximated unstructured and structured NN surrogates. It can be seen that while both surrogates effectively approximate the value function without violating the non-negativeness constraint, the structured surrogate generates a more accurate solution.

We further consider the performance of the structured surrogate $V_{\theta}(z_0, r)$ on an approximation of (10) with $n = 10, m = 2$. We calculate $\Pi(r)$ at $r := (r_1, r_2) \in \{(r_1, r_2) \mid r_1, r_2 \in \hat{r}\}$, where $\hat{r} := \{(i-1)\pi/19 \mid i \in (1, \dots, 20)\}$, thereby 400 solutions of the Riccati equation are applied for training the neural network. Fig. 2 shows the heat map corresponding to the absolute error of $|\max_{\|z_0\|=1} V(z_0, r) - \max_{\|z_0\|=1} V_{\theta}(z_0, r)|$, from where it can be observed that the surrogate yields an accurate approximation of the true value function.

The training process of the unstructured surrogate is limited by computational challenges, as it requires significantly more training data – $(N_x)^n$ times more – compared to the structured surrogate. (N_x) stands for the amount of samples consider for each of the n modes of the initial condition.

4. SURROGATE OPTIMISATION

The true value function $V(z_0, r)$ and its gradient are expensive to evaluate as they require the solution of an

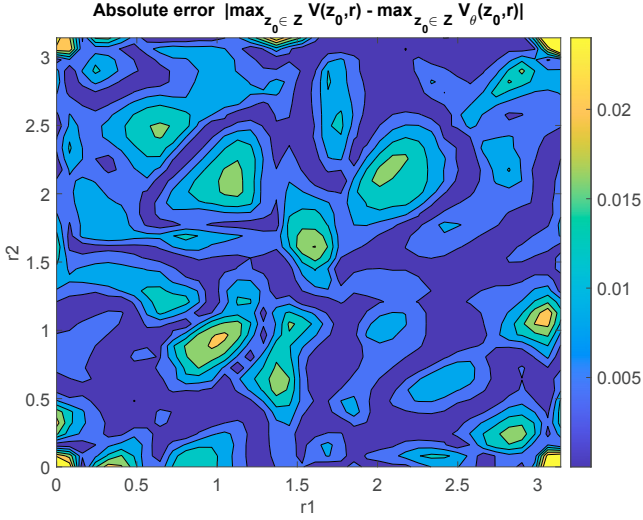


Fig. 2. The heat map of absolute training error using the structured NN for $n = 10$ and $m = 2$, with the surrogate constructed as indicated in Subsec. 3.2.

ARE and additional sensitivity relations in the case of the gradient. Instead, the NN surrogate $V_\theta(z_0, r)$ and its gradient have closed-form expressions that can be readily evaluated using automatic differentiation. In the following, we use the trained surrogate to develop gradient-based and gradient-free algorithms for the solution of the min-max optimisation problem

$$\min_{r \in \Omega^m} \max_{z_0 \in Z} V_\theta(z_0, r) \quad (11)$$

with $Z = \{z \in \mathbb{R}^n : \|z\| = 1\}$ and $\Omega^m = [0, \pi]^m$.

4.1 Projected gradient descent ascent (PGDA)

One natural candidate for solving problem (11) is the gradient descent-ascent (GDA) algorithm, which, at each iteration, performs gradient descent over the variable r with the step-size η_r and gradient ascent over the variable z_0 with the step-size η_{z_0} . We replace the admissible set by the convex set $\bar{Z} = \{z \in \mathbb{R}^n : \|z\| \leq 1\}$ without altering the solutions of the problem thanks to the convexity of V_θ w.r.t. z_0 . The method is described in Alg. 1.

Algorithm 1 Projected gradient descent-ascent

Require: Objective V_θ , initial guess $z_0^{\text{init}}, r^{\text{init}}$, step-size η_{z_0}, η_r , and feasible set \bar{Z}, Ω^m

- 1: $k \leftarrow 0$
- 2: $z_0^k \leftarrow z_0^{\text{init}}$ and $r^k \leftarrow r^{\text{init}}$
- 3: **while** $k \leq K$ **do**
- 4: $z_0^{k+1} = \text{Proj}_{\bar{Z}} [z_0^k + \eta_{z_0} \nabla_{z_0} V_\theta(z_0^k, r^k)]$
- 5: $r^{k+1} = \text{Proj}_{\Omega^m} [r^k - \eta_r \nabla_r V_\theta(z_0^k, r^k)]$
- 6: $k \leftarrow k + 1$
- 7: **end while**
- 8: **return** z_0^k, r^k

Even when considering linear dynamics, using NN surrogates to approximate the value function may result on a lack of convexity-concavity. Moreover, it is well known that PGDA fails to converge to a global Nash equilibrium, see e.g., (Huang et al., 2024). Therefore, in this study, we

aim to overcome this limitation by employing a consensus-based algorithm, as proposed by (Huang et al., 2024), to address the min-max surrogate optimisation problem (11).

4.2 Consensus-based method for saddle points

A consensus-based algorithm (CBO) is an agent-based global optimisation algorithm, which mimics interacting agents communicating over a weighted mean (Totzeck, 2021). CBO methods use a finite number of agents, which are formally stochastic processes, to explore the domain and to form a global consensus about the minimiser as time passes. The dynamics of the agents are governed by two competing terms. A drift term that drags the agents towards a momentaneous consensus point and a diffusion term that randomly moves agents according to a scaled Brownian motion, featuring the exploration of the energy landscape of the cost.

In the min-max setting, two sets of agents corresponding to the minimiser $\{(r)^i\}_{i=1}^{N_1}$ and maximiser $\{(z_0)^i\}_{i=1}^{N_2}$ are considered, respectively. The agent simulations are realised using the standard Euler-Maruyama time discretisation with the step size Δt . The algorithm can be summarised in the procedure described in Alg. 2. For more details, we refer the reader to Huang et al. (2024).

Algorithm 2 Consensus-based Optimisation

Require: Objective V_θ , discrete time step Δt , number of iterations K , parameters $\lambda_1, \lambda_2, \sigma_1, \sigma_2, \alpha, \beta$, number of particles N_1, N_2 , initial laws $\rho_{z_0,0}$ and $\rho_{r,0}$

- 1: Generate the particles' initial positions
- 2: $\{(r)_0^i\}_{i=1}^{N_1} \sim \rho_{r,0}$ and $\{(z_0)_0^i\}_{i=1}^{N_2} \sim \rho_{z_0,0}$
- 3: $k \leftarrow 0$
- 4: **while** $k \leq K$ **do**
- 5: Compute $(Z_0)_\alpha^r$ as given in (12a)
- 6: $(z_0)_{k+1}^i = (z_0)_k^i - \lambda_2 \Delta t \nu(z_0) + \sigma_2 \text{diag}(\nu(z_0)) W_k^{z_0, i}$,
- 7: Compute $(R)_\beta^{z_0}$ as given in (12b)
- 8: $(r)_{k+1}^i = (r)_k^i - \lambda_1 \Delta t \nu(r) + \sigma_1 \text{diag}(\nu(r)) W_k^{r, i}$,
- 9: $k \leftarrow k + 1$
- 10: **end while**
- 11: **return** z_0^k, r^k

Steps 6, and 8 in Alg. 2 are computed for all $i = 1, \dots, N_1$ for $(r)_{k+1}$ and $i = 1, \dots, N_2$ for $(z_0)_{k+1}$. Moreover, we use the notation $\nu(r) = (r)_k^i - (R)_\alpha^{z_0}$ and $\nu(z_0) = (z_0)_k^i - (Z_0)_\beta^r$. The consensus point $\left((R)_\alpha^{z_0} \left(\hat{\rho}_{r,k}^{N_1} \right), (Z_0)_\beta^r \left(\hat{\rho}_{z_0,k}^{N_2} \right) \right)$ is defined by

$$(R)_\alpha^{z_0} = \int r \frac{\omega_\alpha \left(\int z_0 d\hat{\rho}_{z_0,k}^{N_2}(z_0), r \right)}{\left\| \omega_\alpha \left(\int z_0 d\hat{\rho}_{z_0,k}^{N_2}(z_0), \cdot \right) \right\|_{L_1(\hat{\rho}_{r,k}^{N_1})}} d\hat{\rho}_{r,k}^{N_1}(r), \quad (12a)$$

$$(Z_0)_\beta^r = \int z_0 \frac{\omega_{-\beta} \left(z_0, \int r d\hat{\rho}_{r,k}^{N_1}(r) \right)}{\left\| \omega_{-\beta} \left(\cdot, \int r d\hat{\rho}_{r,k}^{N_1}(r) \right) \right\|_{L_1(\hat{\rho}_{z_0,k}^{N_2})}} d\hat{\rho}_{z_0,k}^{N_2}(z_0), \quad (12b)$$

where $\omega_\alpha(z_0, r) := \exp(-\alpha V_\theta(z_0, r))$ and $\omega_{-\beta}(z_0, r) := \exp(\beta V_\theta(z_0, r))$. Here $\hat{\rho}_{z_0,k}^{N_2}(z_0)$ and $\hat{\rho}_{r,k}^{N_1}(r)$ denote the

empirical measures of minimiser and maximiser agents at iteration k , respectively.

To penalise iterates leaving the set $Z \times \Omega^m$, we use a modified objective function, i.e. penalty with a positive value for the minimiser and with a negative value for the maximiser in the objective $V := V_\theta + \mu(-\|z_0\| - 1) + \sum_l ([r - r_{ub}]_+^l + [r_{lb} - r]_+^l)$, where μ is a scalar penalizing the constraint violation, $[\cdot]_+^l$ is the l -th component of the positive part of a vector and r_{lb} , r_{ub} are low and upper bound of the box constraints of r , respectively.

4.3 Numerical tests

We return to the example presented in Section 3.3 related to optimal actuator location. We set $n = 10, m = 2$ and we solve the surrogate min-max problem (11). The heat maps corresponding to the real and surrogate worst-case value functions are shown in Figs. 3 and 4, respectively.

For PGDA we select $K = 2000, \eta_r = 3 \times 10^{-4}, \eta_{z_0} = 10^{-3}$ and initial guess $z_0^{\text{init}} = [0.5, \dots, 0.5], r_\ell^{\text{init}} = [2.5, 2.5]$. For CBO-SP, we set $K = 2000, \alpha = \beta = 10^{15}, \lambda_1 = 2, \lambda_2 = 0.1, \sigma_1 = \sigma_2 = 2, \mu = 10000$ and initial law corresponding to normal distributions $\rho_{r,0} = \mathcal{N}(r_0^{\text{init}}, \text{diag}(1.5, 1.5)), \rho_{z_0,0} = \mathcal{N}(z_0^{\text{init}}, \text{diag}(1.5, \dots, 1.5)), N_1 = N_2 = 300$.

The solutions obtained by PGDA and CBO-SP after K iterations are $r_{\text{CBO-SP}} = [2.0472, 1.1830]$ and $r_{\text{PDGA}} = [1.9465, 1.1927]$, respectively, are displayed in Fig. 4. Both algorithms converge to similarly accurate solutions, distributing the actuators symmetrically in $[0, \pi]$. From the surrogate value function it can also be observed that actuators can be interchanged, as expected.

We validate the closed-loop performance of the optimal parameter and associated optimal control signals $r_{\text{CBO-SP}}$. For reference, we consider a sub-optimal solution where both actuators are placed at the origin $r = [0, 0]^T$.

Fig. 5, depicts the state of the three first modes when the actuators are placed in a sub-optimal position ($r = [0, 0]^T$) and the initial conditions are chosen randomly, whereas in Fig. 6 we use the worst-case initial condition and optimal actuator positions obtained with CPO-SP. We can observe that in the case of a non-optimal position even for a random initial condition the modes converge significantly slower than in the optimal case. In fact, even in the presence of worst initial condition, placing the actuators properly allows the system states to be controlled to zero within 0.7 seconds.

5. CONCLUSION AND FUTURE WORK

We have shown that multi-level control problems related to optimal actuator/sensor design, where the optimal parameter is linked to closed-loop performance, can be computationally alleviated by the use of neural network surrogates. This surrogate can be endowed with additional structures which are relevant to the optimisation, such as non-negativeness or convexity. Moreover, the use of a neural network surrogate with a closed-form expression after training, allows fast evaluation of the low-level value function and its gradient using automatic differentiation, which is considerably cheaper than sensitivity expressions

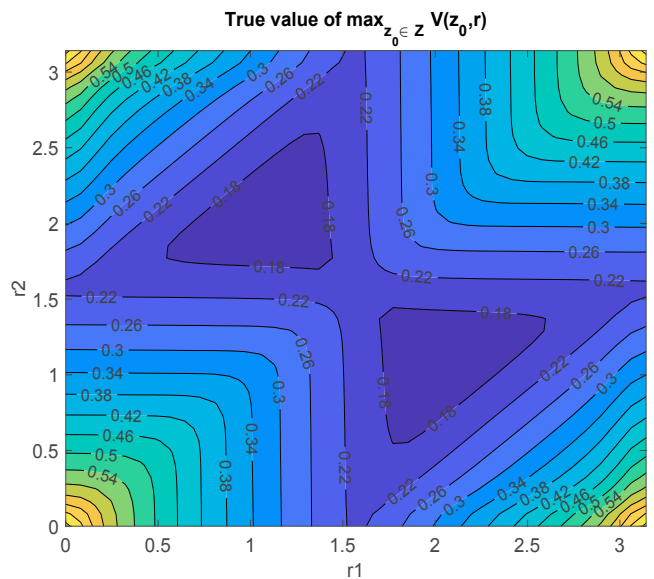


Fig. 3. The heat map corresponding to the real worst-case value function.

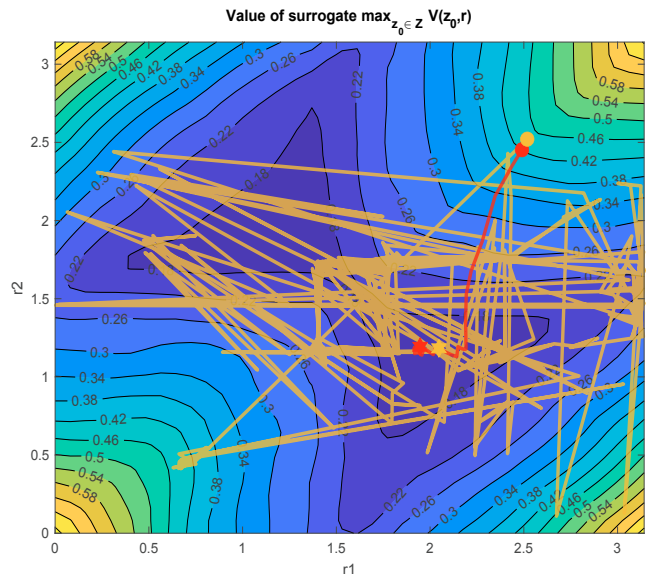


Fig. 4. The heat map corresponding to the NN surrogate worst-case value function. Red: solution trajectories of PGDA, Magenta: consensus point trajectories of CBO-SP. Initial points are depicted with a circle and end points with a star.

for the value function requiring adjoint calculations. This enables the use of both gradient-based and gradient-free solvers for the higher-level min-max problem.

Our main interest in this subject is the development of an effective pipeline for optimal actuator design and location in nonlinear distributed parameter systems. We note that while some parts of the proposed methodology may seem redundant for the linear-quadratic problem presented in this work, such as the use of a gradient-free solver for the min-max problem, they will become essential in the nonlinear case, where we can no longer rely on a Riccati-based approach to characterize the value function of the low-level optimal control problem.

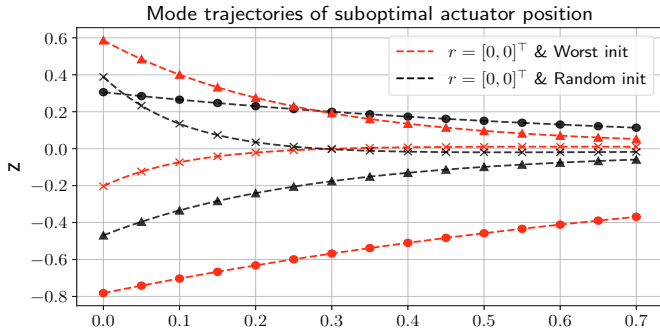


Fig. 5. Mode trajectories of suboptimal actuator position. The first three modes are demonstrated with circle, triangle and cross, respectively.

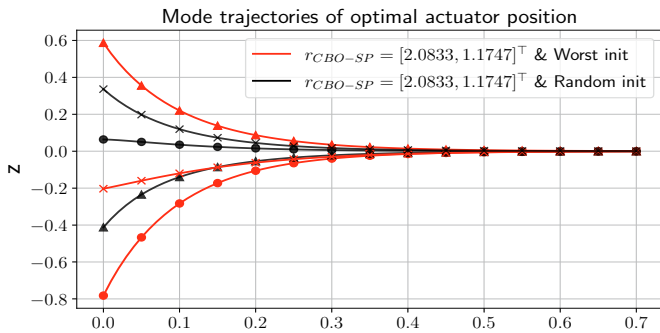


Fig. 6. Mode trajectories of optimal actuator position. The first three modes are demonstrated with circle, triangle and cross, respectively.

REFERENCES

- Albi, G., Bicego, S., and Kalise, D. (2022). Gradient-augmented supervised learning of optimal feedback laws using state-dependent Riccati equations. *IEEE Control Syst. Lett.*, 6, 836–841.
- Azmi, B., Kalise, D., and Kunisch, K. (2021). Optimal feedback law recovery by gradient-augmented sparse polynomial regression. *J. Mach. Learn. Res.*, 22, Paper No. 48, 32.
- Edalatzadeh, M.S., Kalise, D., Morris, K.A., and Sturm, K. (2021). Optimal actuator design for the euler-bernoulli vibration model based on lqr performance and shape calculus. *IEEE Control Systems Letters*, 6, 1334–1339.
- Huang, H., Qiu, J., and Riedl, K. (2024). Consensus-based optimization for saddle point problems. *SIAM Journal on Control and Optimization*, 62(2), 1093–1121.
- Kalise, D., Kunisch, K., and Sturm, K. (2018). Optimal actuator design based on shape calculus. *Mathematical Models and Methods in Applied Sciences*, 28(13), 2667–2717.
- Kasinathan, D. and Morris, K. (2013). \mathbb{H}_∞ -optimal actuator location. *IEEE Transactions on Automatic Control*, 58(10), 2522–2535.
- Morris, K. (1998). Noise reduction in ducts achievable by point control.
- Morris, K. (2010). Linear-quadratic optimal actuator location. *IEEE Transactions on Automatic Control*, 56(1), 113–124.

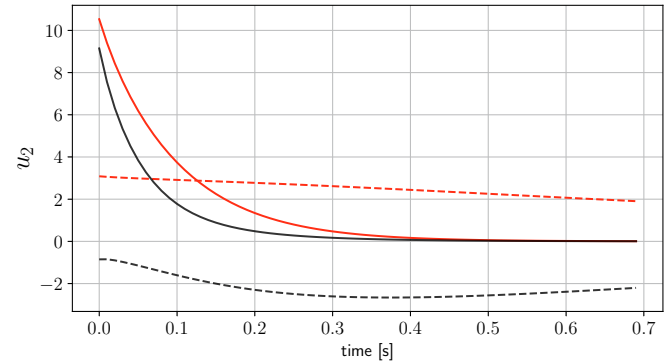
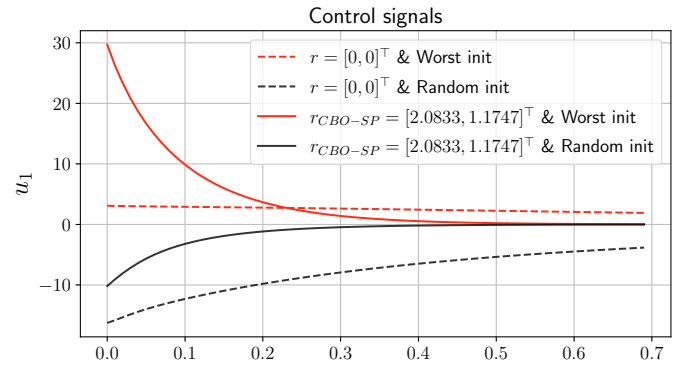


Fig. 7. Feedback control signals of the optimised actuators u_1 and u_2 at $r_{CBO-SP} = [2.0472, 1.1830]^T$ and non-optimised ones placed at $r = [0, 0]^T$.

- Morris, K. and Demetriou, M.A. (2010). Using h_2 -control metrics for the optimal actuator location of infinite-dimensional systems. In *Proceedings of the 2010 American Control Conference*, 4899–4904. IEEE.
- Nakamura-Zimmerer, T., Gong, Q., and Kang, W. (2021). Adaptive deep learning for high-dimensional Hamilton-Jacobi-Bellman equations. *SIAM J. Sci. Comput.*, 43(2), A1221–A1247.
- Peng, F., Ng, A., and Hu, Y.R. (2005). Actuator placement optimization and adaptive vibration control of plate smart structures. *Journal of Intelligent Material Systems and Structures*, 16(3), 263–271.
- Privat, Y., Trélat, E., and Zuazua, E. (2017). Actuator design for parabolic distributed parameter systems with the moment method. *SIAM J. Control Optim.*, 55(2), 1128–1152.
- Svozil, D., Kvasnicka, V., and Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39(1), 43–62.
- Totzeck, C. (2021). Trends in consensus-based optimization. In *Active Particles, Volume 3: Advances in Theory, Models, and Applications*, 201–226. Springer.
- Xu, K., Huang, D.Z., and Darve, E. (2021). Learning constitutive relations using symmetric positive definite neural networks. *Journal of Computational Physics*, 428, 110072.