Multi-fidelity robust controller design with gradient sampling

Steffen W. R. Werner*,1 Michael L. Overton*,2
Benjamin Peherstorfer*,3

*Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA.
[1]Email: steffen.werner@nyu.edu, ORCID: 0000-0003-1667-4862
[2]Email: overton@cims.nyu.edu, ORCID: 0000-0002-6563-6371
[3]Email: pehersto@cims.nyu.edu, ORCID: 0000-0002-1558-6775

Abstract: Robust controllers that stabilize dynamical systems even under disturbances and noise are often formulated as solutions of nonsmooth, nonconvex optimization problems. While methods such as gradient sampling can handle the nonconvexity and nonsmoothness, the costs of evaluating the objective function may be substantial, making robust control challenging for dynamical systems with high-dimensional state spaces. In this work, we introduce multi-fidelity variants of gradient sampling that leverage low-cost, low-fidelity models with low-dimensional state spaces for speeding up the optimization process while nonetheless providing convergence guarantees for a high-fidelity model of the system of interest, which is primarily accessed only in the last phase of the optimization process. Our first multi-fidelity method initiates gradient sampling on higher fidelity models with starting points obtained from cheaper, lower fidelity models. Our second multi-fidelity method relies on ensembles of gradients that are computed from low- and high-fidelity models. Numerical experiments with controlling the cooling of a steel rail profile and laminar flow in a cylinder wake demonstrate that our new multi-fidelity gradient sampling methods achieve up to two orders of magnitude speedup compared to the single-fidelity gradient sampling method that relies on the high-fidelity model alone.

Keywords: nonsmooth optimization, multi-fidelity methods, robust control, linear dynamical systems, H-infinity norm

Mathematics subject classification: 37N35, 37N40, 65K10, 90C30, 90C59

1. Introduction

Robust controllers are a ubiquitous tool to overcome uncertainties in the control of real-world applications resulting from the gap between mathematical modeling and reality. Constructing such controllers via minimizing the $\mathcal{H}_\infty$-norm of closed-loop systems is numerically challenging for at least two reasons. First, the optimization objective induced by $\mathcal{H}_\infty$-control leads to a challenging optimization problem due to its nonsmooth and

nonconvex nature. Second, each evaluation of the objective entails computing the $\mathcal{H}_\infty$-norm, which incurs costs that grow rapidly with the dimension of the state space of the system model. Gradient sampling methods [19,24,34] can handle the nonsmooth, nonconvex objectives underlying $\mathcal{H}_\infty$-control; however, each evaluation of the objective remains computationally expensive. We introduce multi-fidelity approaches that build on gradient sampling and leverage hierarchies of low-fidelity models of the system of interest for speeding up the optimization while still providing convergence guarantees for the high-fidelity model of the system. Our new multi-fidelity variants of gradient sampling make finding $\mathcal{H}_\infty$-controllers tractable for models of systems with high-dimensional state spaces, where relying on the expensive high-fidelity model alone quickly becomes computationally prohibitive.

Multi-fidelity methods for optimization have a long tradition, especially in the engineering community; see, e.g., the survey [53]. Early work on multi-fidelity optimization was based on trust-region methods [1,5,29,30,57]. Other works use a combination of reduced and full models in optimization [50,51,54,63] and especially target optimization under uncertainty, where the objective depends on stochastic auxiliary variables [25,35,36,44,45,52]. In the context of uncertainty quantification, warm-starting iterative processes is a common multi-fidelity approach; see, e.g., [3]. There are also derivative-free multi-fidelity methods [38,39,62]; however, these still require a smooth objective function and thus are not well-suited to nonsmooth optimization problems arising in $\mathcal{H}_\infty$-control.

There is a large body of work on reduced modeling for control and control for large-scale systems; see, e.g., [8,13,46,56]. The problem of efficiently designing $\mathcal{H}_\infty$-controllers for large-scale systems has been addressed before from different view points. While in [42] a new large-scale $\mathcal{H}_\infty$-norm computation routine was used to improve performance of optimization algorithms, reduced-order surrogates were instead exploited in [15]. In [12,43], analytical formulas for (suboptimal) $\mathcal{H}_\infty$-controllers are used rather than an optimization algorithm, relating the low-order controller design problem under additional assumptions to the solution of large-scale sparse nonlinear matrix equations.

The multi-fidelity variants of gradient sampling that we introduce in this work can cope with nonconvex, nonsmooth objectives and at the same time leverage low-fidelity models for reducing the optimization costs. Our first multi-fidelity variant initiates gradient sampling on higher fidelity models with starting points obtained from cheaper, lower fidelity models. The second variant uses an ensemble of gradients of low- and high-fidelity models. Numerical experiments demonstrate that speedups of up to two orders of magnitude can be obtained compared to single-fidelity gradient sampling that uses the high-fidelity model alone.

The paper is organized as follows. We first discuss $\mathcal{H}_\infty$-control and gradient sampling methods in Section 2. We then introduce two new multi-fidelity variants of gradient sampling in Section 3. We present numerical results for both variants on two real-world applications, controlling the cooling of a steel rail profile and control of a laminar flow in a cylinder wake in Section 4. Conclusions are drawn in Section 5.

2. Mathematical preliminaries

This section reviews the concepts of linear state-space systems, robust $\mathcal{H}_\infty$-controller design and the gradient sampling method.

2.1. Dynamical systems and feedback controllers

Consider a finite-dimensional open-loop state-space model of the form

$$G\colon \begin{cases} E\dot{x}(t) = Ax(t) + B_1 w(t) + B_2 u(t), \\ z(t) = C_1 x(t) + D_{11} w(t) + D_{12} u(t), \\ y(t) = C_2 x(t) + D_{21} w(t) + D_{22} u(t), \end{cases} \tag{1}$$

where $x(t) \in \mathbb{R}^n$ are the internal states, $u(t) \in \mathbb{R}^{m_2}$ the control inputs, $w(t) \in \mathbb{R}^{m_1}$ the disturbances, $z(t) \in \mathbb{R}^{p_1}$ the performance of the system and $y(t) \in \mathbb{R}^{p_2}$ the measurements. The matrices describing the model have corresponding dimensions: $E, A \in \mathbb{R}^{n\times n}$, $B_1 \in \mathbb{R}^{n\times m_1}$, $B_2 \in \mathbb{R}^{n\times m_2}$, $C_1 \in \mathbb{R}^{p_1\times n}$, $C_2 \in \mathbb{R}^{p_2\times n}$, $D_{11} \in \mathbb{R}^{p_1\times m_1}$, $D_{12} \in \mathbb{R}^{p_1\times m_2}$, $D_{21} \in \mathbb{R}^{p_2\times m_1}$ and $D_{22} \in \mathbb{R}^{p_2\times m_2}$; see, e.g., [31, 64]. We consider the system (1) without any direct feed-through term, i.e., $D_{22} = 0$, to simplify the exposition. In the general case with $D_{22} \neq 0$, it is described in [64, Sec. 14.7] how one may first construct a controller K with transfer function $K(s)$ for the system with $D_{22} = 0$ and then obtain the controller for the system with $D_{22} \neq 0$ from $K(s)\big(I_{p_2} + D_{22}K(s)\big)^{-1}$. Also, we assume the matrix pencil $\lambda E - A$ in (1) to be regular, i.e., there exists a $\lambda \in \mathbb{C}$ such that $\lambda E - A$ is invertible, so that (1) has a classical frequency domain representation in terms of a transfer function. The goal is to construct a continuous-time (finite-dimensional) feedback controller, which maps the measurements taken from (1) onto an appropriate control signal, $K\colon y \mapsto u$. The controller takes the form of a linear state-space model with

$$K\colon \begin{cases} \dot{x}_{\mathrm{K}}(t) = A_{\mathrm{K}} x_{\mathrm{K}}(t) + B_{\mathrm{K}} y(t), \\ u(t) = C_{\mathrm{K}} x_{\mathrm{K}}(t) + D_{\mathrm{K}} y(t), \end{cases} \tag{2}$$

where $A_{\mathrm{K}} \in \mathbb{R}^{n_{\mathrm{K}}\times n_{\mathrm{K}}}$, $B_{\mathrm{K}} \in \mathbb{R}^{n_{\mathrm{K}}\times p_2}$, $C_{\mathrm{K}} \in \mathbb{R}^{m_2\times n_{\mathrm{K}}}$ and $D_{\mathrm{K}} \in \mathbb{R}^{m_2\times p_2}$. Here, $n_{\mathrm{K}} \in \mathbb{N}$ is the order of the controller, assumed to be a fixed number that is much smaller than the state-space dimension n of the system to be controlled, so $n_{\mathrm{K}} \ll n$. Note that, in contrast to the open-loop system (1), the controller (2) does not have a descriptor (mass) matrix E_{K}; this is motivated by engineering practice that avoids the use of active algebraic constraints in the controller. The control loop of (1) is closed by connecting the controller (2) with the system (1), which yields the closed-loop system $G_{\mathrm{c}}\colon w \mapsto z$ with

$$G_{\mathrm{c}}\colon \begin{cases} E_{\mathrm{c}}\dot{x}_{\mathsf{c}} = A_{\mathrm{c}} x_{\mathsf{c}} + B_{\mathrm{c}} w(t), \\ z(t) = C_{\mathrm{c}} x_{\mathsf{c}} + D_{\mathrm{c}} w(t), \end{cases} \tag{3}$$

where the system matrices are given by

$$\begin{aligned} E_{\mathrm{c}} &= \begin{bmatrix} E & 0 \\ 0 & I_k \end{bmatrix}, & A_{\mathrm{c}} &= \begin{bmatrix} A + B_2 D_{\mathrm{K}} C_2 & B_2 C_{\mathrm{K}} \\ B_{\mathrm{K}} C_2 & A_{\mathrm{K}} \end{bmatrix}, \\ B_{\mathrm{c}} &= \begin{bmatrix} B_1 + B_2 D_{\mathrm{K}} D_{21} \\ B_{\mathrm{K}} D_{21} \end{bmatrix}, & C_{\mathrm{c}} &= \begin{bmatrix} C_1 + D_{12} D_{\mathrm{K}} C_2 & D_{12} C_{\mathrm{K}} \end{bmatrix}, \\ D_{\mathrm{c}} &= D_{11} + D_{12} D_{\mathrm{K}} D_{21}. \end{aligned} \tag{4}$$

2.2. $\mathcal{H}_\infty$ controller design

The requirement for the feedback controller (2) that we consider here is the stabilization of the closed-loop system (3), i.e., the design of (2) ensures that the closed-loop matrix

pencil $sE_c - A_c$ is regular and that all of its finite eigenvalues lie in the open left half-plane. Thus, we define the set of stabilizing controllers as

$$\mathcal{K} = \{(A_K, B_K, C_K, D_K) \mid \lambda \in \mathbb{C} \text{ with } \det(\lambda E_c - A_c) = 0 \;\Rightarrow\; \mathrm{Re}(\lambda) < 0\}.$$

Let $\|\cdot\|_{\mathcal{H}_\infty}$ denote the $\mathcal{H}_\infty$-norm, defined for the closed-loop system (3) by

$$\|G_c\|_{\mathcal{H}_\infty} := \sup_{\lambda \in \mathbb{C}, \mathrm{Re}(\lambda) \geq 0} \|G_c(\lambda)\|_2,$$

with the transfer function $G_c(s) = C_c(sE_c - A_c)^{-1}B_c + D_c$, where $s \in \mathbb{C}$; see, e.g., [4]. In optimal $\mathcal{H}_\infty$-control, a controller $K \in \mathcal{K}$ is sought as a solution to the constrained minimization problem

$$\min_{K \in \mathcal{K}} \|G_c\|_{\mathcal{H}_\infty}. \tag{5}$$

The task of $\mathcal{H}_\infty$-optimal control can be interpreted as finding a stabilizing controller that minimizes the worst-case amplification of all admissible disturbances.

In this paper, we focus on the case where the open-loop system (1) and, consequently, the closed-loop system (3), are described by large-scale sparse systems of differential-algebraic equations. The spectral abscissa of the pencil $sE_c - A_c$ is the real part of its rightmost finite eigenvalue; we denote this by

$$\alpha(A_c, E_c) := \max\{\mathrm{Re}(\lambda) \mid \lambda \in \mathbb{C} \text{ with } \det(\lambda E_c - A_c) = 0\}. \tag{6}$$

The maximum peak of the spectral norm of the transfer function on the imaginary axis is known as the $\mathcal{L}_\infty$-norm, which is for the closed-loop system (3) given by

$$\|G_c\|_{\mathcal{L}_\infty} := \sup_{\omega \geq 0} \|G_c(\mathrm{i}\,\omega)\|_2, \tag{7}$$

where i denotes the imaginary unit, and the supremum is over the nonnegative imaginary axis because the data are real.

Using (6) and (7), the $\mathcal{H}_\infty$-norm is

$$\|G_c\|_{\mathcal{H}_\infty} = \begin{cases} \|G_c\|_{\mathcal{L}_\infty} & \text{if } \alpha(A_c, E_c) < 0, \\ \infty & \text{otherwise.} \end{cases} \tag{8}$$

Now we define our objective function to be minimized as

$$f(x) := \|G_c\|_{\mathcal{H}_\infty} \tag{9}$$

with the design variable

$$x = \begin{bmatrix} \mathrm{vec}(A_K) \\ \mathrm{vec}(B_K) \\ \mathrm{vec}(C_K) \\ \mathrm{vec}(D_K) \end{bmatrix} \in \mathbb{R}^N \text{ where } N = n_K^2 + n_K m_2 + p_2 n_K + p_2 m_2, \tag{10}$$

defining a controller (2) via the matrices $K = (A_K, B_K, C_K, D_K)$, with the closed-loop system matrices defining G_c in (8) depending on K via (4). It is also convenient to define the constraint function

$$h(x) := \alpha(A_c, E_c), \tag{11}$$

where again the closed-loop system matrices A_c and E_c depend on the controller matrices $K = (A_K, B_K, C_K, D_K)$ via (4). Using this notation, the optimization problem (5) may be equivalently given as either

$$\min_x f(x) \quad \text{or} \quad \min_{x:h(x)<0} f(x). \tag{12}$$

This optimization problem is challenging because the $\mathcal{H}_\infty$-norm (8) is nonconvex and, at points x where the supremum in (7) is attained at more than one value of ω, nonsmooth. However, f is locally Lipschitz on the set of stabilizing controllers $\{x \in \mathbb{R}^N : h(x) < 0\}$.

2.3. Gradient sampling method

The gradient sampling method was presented by Burke, Lewis and Overton in 2005 [24], along with an extensive convergence theory that was subsequently refined by Kiwiel in 2007 [34]. The algorithm is nondeterministic in the sense that it generates (samples) gradients at randomly generated points within a given ball around a given iterate. In this paper, we rely on the detailed summary of the method and its convergence theory in the survey by Burke, Curtis, Lewis, Overton and Simões [19]. The main convergence result for Alg. GS of [19] (with specific parameter choices) is stated as Theorem 3.1 there: *Suppose that f is locally Lipschitz on $\mathbb{R}^N$ and continuously differentiable on an open set with full measure. Then, with probability one, Alg. GS is well defined and does not terminate, and generates a sequence of iterates for which either the function values diverge to $-\infty$, or every cluster point of the sequence is Clarke stationary for f.* Clarke stationarity is a standard measure of stationarity for locally Lipschitz, nonsmooth functions [18].

The gradient sampling method relies on the computation of the function f and its gradient ∇f at the sequence of iterates generated by the method, using a "gradient paradigm" [6], as opposed to a more standard "subgradient paradigm" often used for nonsmooth functions. The former asserts that, since locally Lipschitz functions are differentiable almost everywhere by Rademacher's theorem, and since in practice, it is essentially impossible to verify whether a nontrivial function f is differentiable or not at a given iterate x, a method can reasonably compute an approximate gradient at any given point, for example, by ignoring "ties" in a max function. The idea is that it is only in the limit of the sequence of iterates that the function is actually not differentiable. Of course, sampled gradients computed at nearby points in this way may vary greatly, and the gradient sampling algorithm exploits this property. These key points are discussed at greater length in the references given above.

The gradient sampling method has been applied to solve $\mathcal{H}_\infty$-norm optimization and related stabilization problems since it was first introduced [20,21,23]. We follow the same basic strategy used in [20]: first, in order to find a stabilizing controller for the $\mathcal{H}_\infty$-norm optimization problem described in Section 2.2, we apply gradient sampling to the constraint function $h(x)$ defined in (11); then, once a point x^0 with $h(x^0) < 0$ has been found, we apply gradient sampling to the $\mathcal{H}_\infty$-norm objective f defined in (9), initialized at x^0. If this results in f being evaluated at a non-stabilizing controller, the function value ∞ that is returned will result in the controller being rejected by the line search; according to the gradient sampling convergence theory, as long as f is differentiable at x^0, the line search must eventually return a new point x^1 with $f(x^1) < f(x^0)$. The functions f and h are differentiable almost everywhere (in the former case, almost everywhere on $\mathcal{K}$), and the formulas for their gradients may be derived from the formulas for the gradients of the $\mathcal{H}_\infty$-norm and the spectral abscissa given in Appendices A and B, respectively.

3. Multi-fidelity gradient sampling

In this section, we introduce two multi-fidelity versions of the gradient sampling method to design controllers for high-fidelity models for which a hierarchy of cheap low-fidelity models are available. We first introduce the notation of hierarchies of models in Section 3.1. Then we define our two new methods: Gradient sampling with multi-fidelity restarts in Section 3.2 and gradient sampling with multi-fidelity approximate gradients in Section 3.3.

3.1. Hierarchies of models

We consider the situation where there is a hierarchy of L models of the form (1) available. The accuracy of the models increases with a corresponding index from level 1 to level L, the most accurate model. We find such a situation, for example, when (1) is given as spatial discretization of partial differential equations, where the model hierarchy with levels $\ell = 1, \ldots, L$ is due to different refinements of the discretization. The hierarchy of models gives rise to a hierarchy of objective functions for $\mathcal{H}_\infty$-controller design:

$$f^\ell(x) = \|G_{\mathrm{c}}^\ell\|_{\mathcal{H}_\infty}, \tag{13}$$

with $\ell = 1, \ldots, L$. A key point to note is that the dimension N of the vector x in (13) representing the controller $K = (A_{\mathrm{K}}, B_{\mathrm{K}}, C_{\mathrm{K}}, D_{\mathrm{K}})$ is independent of the model level ℓ. Instead of (4), we now have closed-loop system matrices defined by

$$E_{\mathrm{c}}^\ell = \begin{bmatrix} E^\ell & 0 \\ 0 & I_k \end{bmatrix}, \quad A_{\mathrm{c}}^\ell = \begin{bmatrix} A^\ell + B_2^\ell D_{\mathrm{K}} C_2^\ell & B_2^\ell C_{\mathrm{K}} \\ B_{\mathrm{K}} C_2^\ell & A_{\mathrm{K}} \end{bmatrix}, \quad B_{\mathrm{c}}^\ell = \begin{bmatrix} B_1^\ell + B_2^\ell D_{\mathrm{K}} D_{21}^\ell \\ B_{\mathrm{K}} D_{21}^\ell \end{bmatrix},$$
$$C_{\mathrm{c}}^\ell = \begin{bmatrix} C_1^\ell + D_{12}^\ell D_{\mathrm{K}} C_2^\ell & D_{12}^\ell C_{\mathrm{K}} \end{bmatrix}, \quad D_{\mathrm{c}}^\ell = D_{11}^\ell + D_{12}^\ell D_{\mathrm{K}} D_{21}^\ell,$$

where the matrices superscripted by ℓ are the open-loop system matrices. The corresponding transfer functions of the closed-loop systems are $G_{\mathrm{c}}^\ell(s) = C_{\mathrm{c}}^\ell (s E_{\mathrm{c}}^\ell - A_{\mathrm{c}}^\ell)^{-1} B_{\mathrm{c}}^\ell + D_{\mathrm{c}}^\ell$.

Our aim is to find a controller that is optimal with respect to the high-fidelity objective function f^L, while leveraging the less accurate but cheaper objective functions f^ℓ on levels $\ell = 1, \ldots, L-1$. The objective functions have gradients $\nabla f^1, \ldots, \nabla f^L$, which are increasingly more expensive to compute as ℓ increases; see Appendix A for the formulas.

Besides hierarchies of discretizations, the model hierarchy may alternatively be obtained via model reduction techniques. These allow the computation of reasonably accurate, cheap-to-evaluate surrogates that can serve as low-fidelity models in our setting. See, for example, [9,10,16,17,55] for overviews on potential methods, or [12,43] for model reduction methods in the context of $\mathcal{H}_\infty$-controller design.

In the following, our starting point is a hierarchy of objective functions $f^1, \ldots, f^L$ that are ordered from cheap to expensive and less accurate to more accurate but we make no assumptions on where the objective functions originate. It is sufficient to have an oracle that allows the evaluation of the functions f^ℓ and their gradients ∇f^ℓ at the design variable x corresponding to the given controller K. Besides hierarchies of objective functions, we must also at least implicitly consider hierarchies of constraint functions $h^\ell(x)$. We return to this topic below.

3.2. Restarted multi-fidelity gradient sampling (RMFGS)

Our restarted multi-fidelity gradient sampling (RMFGS) method uses controllers obtained with lower fidelity models to warm-start the optimization for controllers of higher fidelity models.

Algorithm 1 Restarted multi-fidelity gradient sampling (RMFGS).

Input: Initial point $x^0 \in \mathbb{R}^N$,
sample size $q_\ell \geq N+1$, initial sampling radii $\epsilon_{\ell,0} > 0$,
initial stationarity targets $\nu_{\ell,0} > 0$,
termination tolerances $\epsilon_{\ell,\text{opt}} \in (0, \epsilon_{\ell,0}), \nu_{\ell,\text{opt}} \in (0, \nu_{\ell,0})$,
reduction factors $\theta_{\ell,\epsilon} \in (0,1), \theta_{\ell,\nu} \in (0,1)$, and
line search parameters $\beta_\ell \in (0,1)$, $\gamma_\ell \in (0,1)$, for $\ell = 1, \ldots, L$.
Output: Approximation $x^k \in \mathbb{R}^N$ to a minimizer of f^L.

Initialize $k = 0$.
for $\ell = 1$ **to** L **do**
  if $f^\ell(x^k)$ is not finite **then**
    Apply stabilization step for f^ℓ to x^k.
  end if
  Set $\nu_{k+1} = \nu_{\ell,0}$ and $\epsilon_{k+1} = \epsilon_{\ell,0}$.
  repeat
    Independently sample $\{x^{k,1}, \ldots, x^{k,q}\}$ uniformly from $\mathcal{B}(x^k, \epsilon_k)$.
    Compute g^k as the solution of $\min_{g \in \mathcal{G}^{\ell,k}} \frac{1}{2}\|g\|_2^2$, where

$$\mathcal{G}^{\ell,k} = \text{conv}\left\{\nabla f^\ell(x^k), \nabla f^\ell(x^{k,1}), \ldots, \nabla f^\ell(x^{k,q})\right\}.$$

    Compute $x^{k+1}, \epsilon_{k+1}, \nu_{k+1}$ using Algorithm 2 with inputs
    $x^k, g^k, f^\ell, \epsilon_k, \nu_k, \theta_{\ell,\epsilon}, \theta_{\ell,\nu}, \epsilon_{\ell,\text{opt}}, \nu_{\ell,\text{opt}}, \beta, \gamma$.
    Increment $k \leftarrow k+1$.
  until $(x^k == x^{k-1})$ **and** $(\epsilon_k == \epsilon_{k-1})$ **and** $(\nu_k == \nu_{k-1})$.
end for

3.2.1. Multi-fidelity restarts

The proposed RMFGS approach iterates over the levels $\ell = 1, \ldots, L$ and, at each level ℓ, solves an optimization problem of the form (12) with the objective function f^ℓ, where the initial guess is the solution of the previous level. So, letting $x^{k_{\ell-1}}$ denote the final iterate at level $\ell - 1$, the initial guess at level $\ell \geq 2$ is $x^{k_{\ell-1}}$. The motivation for RMFGS is that the objective functions become progressively more accurate with increasing level ℓ, and thus, the solution $x^{k_{\ell-1}}$ at the previous level $\ell - 1$ should be a good starting point at the current level ℓ, implying that fewer gradient sampling steps are necessary than with a generic initial guess. Hence, the aim is to take many iterations on lower levels where the initial starting points are poor but where objective and gradient evaluations are cheap, while taking fewer of the expensive evaluations on higher levels as the starting points get closer to a minimizer of the high-fidelity objective function f^L.

For any level ℓ, the function f^ℓ is monotonically decreasing on $\{x^k\}$ as k increases from $k_{\ell-1}$ to k_ℓ. Note, however, that for $\ell < L$, there is no guarantee that the high-fidelity objective f^L is lower at x^{k_ℓ} than it was at $x^{k_{\ell-1}}$. Indeed, it might not even be finite, since the objective function is finite only if the closed-loop system is stable, and even if this is the case for the model at one level, it might not be at another level.

3.2.2. Algorithmic description of RMFGS

The new method is summarized in Algorithm 1. The main difference from the original (single fidelity) gradient sampling method [19, Alg. GS] is the new outer loop starting in

Line 2 of Algorithm 1, which iterates over the available levels $\ell = 1, \ldots, L$. Lines 7 to 12 consist of an inner iteration describing the single fidelity gradient sampling method using the objective function f^ℓ and its gradients ∇f^ℓ at the current level. This has three parts:

(a) In Line 8, sampling gradients uniformly from $\mathcal{B}(x^k, \epsilon_k)$, the 2-norm ball around the current iterate x^k with radius ϵ_k.

(b) In Line 9, computing the vector g^k, which is easily done by standard software for convex quadratic programming. As explained in [19, Sec. 6.1], the vector $-g^k$ is not only a descent direction for f^ℓ, but more importantly it is a *stabilized* or *robust* descent direction, which allows for longer steps to be taken in the line search in the next part.

(c) In Line 10, the computation of the gradient sampling step as described in Algorithm 2, which includes checking the convergence criteria, updating the algorithm parameters accordingly, and, if the termination criteria are not yet met, updating the current iterate using a line search along $-g^k$.

The inner iteration for a given f^ℓ terminates when the gradient sampling step has no effect, i.e., if the new iterate is the same as the previous one and the sampling radius and stationarity target did not change. Looking at Algorithm 2, we see that this can only occur if the algorithm satisfies the convergence criteria specified by the parameters. According to the gradient sampling theory, this must happen eventually; see [19, Cor. 6.1], taking into account the initialization of the parameters in Algorithm 1. In practice, it is necessary to set a limit on the number of steps in each inner iteration, both because of the possible effects of rounding errors and to limit the overall computation time. Likewise, in theory, the line search in Line 8 of Algorithm 2 must terminate in a finite number of steps, although in practice, because of rounding errors, a limit must be placed on this and the line search terminated if this limit is reached. Whichever way the iteration for level $\ell < L$ terminates, the method continues with the next model level in the outer loop. In this case, the current iterate x^k is the final iterate x^{k_ℓ} of level $\ell < L$ and the initial iterate of level $\ell + 1$.

The algorithm allows for its parameters to depend on the level ℓ so that adjustments for each level are possible. The last step of the outer loop in Algorithm 1 is gradient sampling with the objective function of interest f^L, i.e., each step of the inner loop in Algorithm 1 is as expensive as each step of classical single-fidelity gradient sampling. In terms of global computational costs in comparison to the single fidelity method [19, Alg. GS], we can potentially save function as well as gradient evaluations using Algorithm 1, under the assumption that the computed approximations of minimizers on each level are indeed good initial guesses for optimization on subsequent levels.

Algorithm 2 implements the update step of gradient sampling and is the same as in Alg. GS in [19], except for the differentiability check of the objective function f at the next iterate $\hat{x}$. This check is needed in theory in order to be able to rigorously state the convergence results in [19], but in practice, with the inevitable rounding errors incurred in floating point arithmetic, it makes little or no sense to attempt it. As already noted, our objective functions are differentiable almost everywhere, and while encountering a point where the function is actually not differentiable is not technically a probability zero event, it may be considered extremely unlikely in practice. This issue is discussed further in [19, Sec. 6.4.2].

Algorithm 2 Gradient sampling step.

Input: Iterate $x \in \mathbb{R}^N$, vector $g \in \mathbb{R}^N$, objective function f,
current sampling radius ϵ and stationarity target ν,
reduction factors θ_ϵ and θ_ν, termination tolerances ϵ_{opt} and ν_{opt}, and
line search parameters β and γ.

Output: Updated iterate $\hat{x}$, sampling radius $\hat{\epsilon}$ and stationarity target $\hat{\nu}$.

1: **if** $(\|g\|_2 \leq \nu_{\text{opt}})$ **and** $(\epsilon \leq \epsilon_{\text{opt}})$ **then**
2: Set $\hat{\nu} = \nu$, $\hat{\epsilon} = \epsilon$ and $\hat{t} = 0$.
3: **else**
4: **if** $\|g\|_2 \leq \nu$ **then**
5: Set $\hat{\nu} = \theta_\nu \nu$, $\hat{\epsilon} = \theta_\epsilon \epsilon$ and $\hat{t} = 0$.
6: **else**
7: Set $\hat{\nu} = \nu$ and $\hat{\epsilon} = \epsilon$.
8: Set $\hat{t} = \max\left\{t \in \{1, \gamma, \gamma^2, \ldots\} : f(x - tg) < f(x) - \beta t \|g\|_2^2\right\}$.
9: **end if**
10: **end if**
11: Update $\hat{x} = x - \hat{t}g$.

3.3. Approximate multi-fidelity gradient sampling (AMFGS)

A valid criticism of Algorithm 1 is that although our primary interest is in minimizing the highest fidelity model f^L, this does not enter the computation until the gradient sampling algorithm has been run on all lower fidelity objectives $f^1, f^2, \ldots, f^{L-1}$. Although we justified this by arguing that the final iterate for one level should be a good starting point for the next level, an alternative viewpoint is that we might want to involve the highest fidelity model f^L at earlier stages of the computation. This can be done efficiently by using f^L as the objection function from the beginning, but replacing the expensive gradient sampling of f^L by gradient sampling of the cheaper models $f^1, f^2, \ldots, f^{L-1}$.

3.3.1. Multi-fidelity ensembles of gradients

In the AMFGS method, we retain the idea of an outer loop over all L levels, but, unlike in the RMFGS method, we involve the high-fidelity function f^L at every stage of the outer loop. For this reason, we enforce the property that the high-fidelity function f^L is monotonically decreasing on $\{x^k\}$ as k increases. However, although we evaluate f^L at every iterate x^k, and in the line search that produces these iterates, only at the final level L we actually sample $q \geq N + 1$ gradients of the high-fidelity function f^L. At all earlier levels, we sample gradients of lower fidelity functions instead. Thus, we replace the definition

$$\mathcal{G}^{\ell,k} = \text{conv}\left\{\nabla f^\ell(x^k), \nabla f^\ell(x^{k,1}), \ldots, \nabla f^\ell(x^{k,q})\right\}$$

in Line 9 of Algorithm 1 by

$$\mathcal{G}^{\ell,k} = \text{conv}\left\{\nabla f^L(x^k), \nabla f^\ell(x^{k,1}), \ldots, \nabla f^\ell(x^{k,q})\right\}.$$

3.3.2. Algorithmic description of AMFGS

The AMFGS method is summarized in Algorithm 3. The basic structure of the algorithm is the same as that of Algorithm 1. However, a major difference between them is that in AMFGS, we are minimizing the high-fidelity objective function f^L at *all* levels $\ell =$

Algorithm 3 Approximate multi-fidelity gradient sampling (AMFGS).

Input: Initial point $x^0 \in \mathbb{R}^N$,
sample size $q_\ell \geq N+1$, initial sampling radii $\epsilon_{\ell,0} > 0$,
initial stationarity targets $\nu_{\ell,0} > 0$,
termination tolerances $\epsilon_{\ell,\text{opt}} \in (0, \epsilon_{\ell,0}), \nu_{\ell,\text{opt}} \in (0, \nu_{\ell,0})$,
reduction factors $\theta_{\ell,\epsilon} \in (0,1), \theta_{\ell,\nu} \in (0,1)$, and
line search parameters $\beta_\ell \in (0,1)$, $\gamma_\ell \in (0,1)$, for $\ell = 1, \ldots, L$.
Output: Approximation $x^k \in \mathbb{R}^N$ to a minimizer of f^L.

1: Initialize $k = 0$.
2: **if** $f^L(x^0)$ is not finite **then**
3: Apply stabilization step for f^L to x^0.
4: **end if**
5: **for** $\ell = 1$ **to** L **do**
6: Set $\nu_{k+1} = \nu_{\ell,0}$ and $\epsilon_{k+1} = \epsilon_{\ell,0}$.
7: **repeat**
8: Independently sample $\{x^{k,1}, \ldots, x^{k,q}\}$ uniformly from $\mathcal{B}(x^k, \epsilon_k)$.
9: Compute g^k as the solution of $\min_{g \in \mathcal{G}^{\ell,k}} \frac{1}{2}\|g\|_2^2$, where

$$\mathcal{G}^{\ell,k} = \operatorname{conv}\left\{\nabla f^L(x^k), \nabla f^\ell(x^{k,1}), \ldots, \nabla f^\ell(x^{k,q})\right\}.$$

10: Compute $x^{k+1}, \epsilon_{k+1}, \nu_{k+1}$ using Algorithm 2 with inputs
$x^k, g^k, f^L, \epsilon_k, \nu_k, \theta_{\ell,\epsilon}, \theta_{\ell,\nu}, \epsilon_{\ell,\text{opt}}, \nu_{\ell,\text{opt}}, \beta, \gamma$.
11: Increment $k \leftarrow k+1$.
12: **until** $(x^k == x^{k-1})$ **and** $(\epsilon_k == \epsilon_{k-1})$ **and** $(\nu_k == \nu_{k-1})$.
13: **end for**

$1, \ldots, L$, while in RMFGS, at level ℓ, we minimize the objective f^ℓ. Consequently, each step of level ℓ of method AMFGS (Algorithm 3) is computationally more expensive than the corresponding step in RMFGS (Algorithm 1). However, for $\ell < L$, it is less expensive than a step at level L of either method due to the use of cheaper-to-evaluate approximations in the gradient computations of the sampled evaluation points in Line 9 of Algorithm 3. A key point, however, is that at the current iterate x^k, we use the gradient of the high-fidelity objective function f^L in the definition of $\mathcal{G}^{\ell,k}$, regardless of the level ℓ in the outer loop. This guarantees that $-g^k$ is a descent direction for f^L, although how "robust" of a descent direction it is depends on how well the sampled gradients of f^ℓ approximate gradients of f^L. If the approximation is not very good, the result may be that the line search needs to take a very short step to obtain a reduction in f^L along $-g^k$. The main differences between Algorithms 1 and 3 are the definition of $\mathcal{G}^{\ell,k}$ and that the function we pass to Algorithm 2 is f^ℓ in the first case and f^L in the second case. Note that both methods, Algorithms 1 and 3, boil down to the classical (single-fidelity) gradient sampling method from [19, Alg. GS] in the last step of each outer loop, so the rationale for both methods is ultimately to provide a good starting point for this final optimization at level L.

3.4. Stabilization

As explained in Section 2.3, in order to obtain initial points for minimization of the $\mathcal{H}_\infty$-norm objective, it may be necessary to first apply gradient sampling to the stabilization constraint function. Thus, in Algorithm 1, in order to initiate gradient sampling optimization of f^ℓ at step ℓ of the outer loop, it may be necessary to first apply gradient sampling

to the corresponding constraint function h^ℓ. This applies not only at level 1, but at higher levels as well, because there is no guarantee that at level $\ell > 1$, the function f^ℓ is finite at the starting point x^k, even though $f^{\ell-1}$ is necessarily finite there. However, we note that this stabilization step at level $\ell > 1$ was never needed in our computational results presented in Section 4.

In contrast, for Algorithm 3, at most one initial stabilization is necessary, to obtain a point x^0 where f^L is finite.

3.5. Theoretical guarantees

Provided step ℓ in the outer loop of Algorithm 1 is initiated at a point where f^ℓ is finite and differentiable, the convergence theory given in [19] states that, in exact arithmetic, and the absence of maximum iteration limits, eventually the convergence criteria imposed by the parameters $\epsilon_{\ell,\text{opt}}$ and $\nu_{\ell,\text{opt}}$ must be satisfied. It is important to note that these stopping criteria, namely

$$\|g^{\ell,k_\ell}\|_2 \leq \nu_{\ell,\text{opt}} \quad \text{and} \quad \epsilon_{\ell,k_\ell} \leq \epsilon_{\ell,\text{opt}}$$

essentially provide an approximate Clarke stationarity certificate. More precisely, if the parameters $\epsilon_{\ell,\text{opt}}$ and $\nu_{\ell,\text{opt}}$ were set to zero, then all cluster points of the resulting sequence of iterates must be Clarke stationary for f^ℓ (see [19, Thm. 6.1]), which amounts to a first-order optimality condition given the Clarke regularity of f^ℓ [24, p. 753]. However, for $\ell < L$, no such statement can be made about step ℓ in the outer loop of Algorithm 3, because the gradients sampled are not gradients of f^L. In contrast, the statement *can* be made about the final step $\ell = L$ in the outer loop of Algorithm 3.

4. Numerical examples

In this section, we present results of applying the new multi-fidelity gradient sampling algorithms to two applications. We start by introducing two special cases of the general system (1) that we will use. We then describe the experimental setup, and subsequently present the computational results.

4.1. Two open-loop systems

We test the new methods for the design of $\mathcal{H}_\infty$-controllers on two special instances of open-loop systems (1) that are motivated by applications discussed subsequently. First, we consider systems of the form

$$\begin{aligned} E\dot{x}(t) &= Ax(t) + Bw_1(t) + Bu(t), \\ z_1(t) &= Cx(t), \\ z_2(t) &= u(t), \\ y(t) &= Cx(t) + w_2(t). \end{aligned} \tag{14}$$

In (14), the disturbances are separated into two independent parts $w_1(t)$ and $w_2(t)$, where $w_1(t)$ has the same influence on the system dynamics as the controls and $w_2(t)$ disturbs the measurements taken for the controller. Also, the performance of the system consists of the non-disturbed measurements taken for the controller and the control signal itself. Note that an open-loop system of the form (14) is known in the literature as *normalized*

Table 1: Properties of models used in numerical experiments.

		rail example	cylinder example
Discretization levels and state dimensions	$\ell = 1$	$n = 109$	$n = 6\,618$
	$\ell = 2$	$n = 371$	$n = 10\,645$
	$\ell = 3$	$n = 1\,357$	$n = 22\,060$
	$\ell = 4$	$n = 5\,177$	—
	$\ell = 5$	$n = 20\,209$	—
Inputs	system (14)	$m_1 = 13$, $m_2 = 7$	$m_1 = 14$, $m_2 = 6$
	system (15)	$m_1 = 3$, $m_2 = 4$	$m_1 = 3$, $m_2 = 3$
Outputs	system (14)	$p_1 = 13$, $p_2 = 6$	$p_1 = 14$, $p_2 = 8$
	system (15)	$p_1 = 6$, $p_2 = 6$	$p_1 = 8$, $p_2 = 8$

linear-quadratic Gaussian (LQG) formulation; see, e.g., [12,43]. We may write (14) in the form (1) by defining

$$B_1 = \begin{bmatrix} B & 0 \end{bmatrix}, \quad B_2 = B, \qquad C_1 = \begin{bmatrix} C \\ 0 \end{bmatrix}, \qquad C_2 = C,$$
$$D_{11} = 0, \qquad D_{12} = \begin{bmatrix} 0 \\ I_{m_2} \end{bmatrix}, \quad D_{21} = \begin{bmatrix} 0 & I_{p_2} \end{bmatrix}, \quad D_{22} = 0.$$

As a second instance of (1), we consider

$$\begin{aligned} E\dot{x}(t) &= Ax(t) + B_1 w(t) + B_2 u(t), \\ z(t) &= C_2 x(t) + D_{12} u(t), \\ y(t) &= C_2 x(t) + D_{21} w(t). \end{aligned} \tag{15}$$

Due to the nature of the benchmark problems that we use, the performance and control measurements are based on the same state observations, i.e., we have $C_1 = C_2$ in (1). The feed-through term D_{12} is taken as the first columns ($m_2 \leq p_2$) or rows ($m_2 > p_2$) of the $\max(m_2, p_2)$-dimensional identity matrix, and the feed-through term D_{21} as the first columns ($m_1 \leq p_2$) or rows ($m_1 > p_2$) of the $\max(m_1, p_2)$-dimensional identity matrix.

For the controller design in both cases, we consider only the problem formulation of the controller (2) without a feed-through term, i.e., $D_K = 0$, which is in line with known analytically derived formulas for the construction of (suboptimal) $\mathcal{H}_\infty$-controllers for (14) and (15); see, e.g., [12,28].

4.2. Experimental setup

In our experiments, we set the parameters of the multi-fidelity gradient sampling algorithms as shown in Table 2. Note that we set iteration limits on each level of the multi-fidelity algorithms. These values are varied with the problem and are listed in the column headed "Max. Iters." in the tables that appear below. In these tables, the point $x^{k\ell}$ denotes the final iterate at level ℓ. In the case of the rail example, we steadily decrease the

Table 2: Algorithm parameters used in numerical experiments.

	HFGS	RMFGS	AMFGS
Init. sampling radii, stationarity targets	$\epsilon_0 = 0.1$, $\nu_0 = 0.1$	$\epsilon_{1,0} = \nu_{1,0} = 0.1$ $\epsilon_{2,0} = \nu_{2,0} = 0.01$ $\epsilon_{3,0} = \nu_{3,0} = 0.001$ $\epsilon_{4,0} = \nu_{4,0} = 10^{-4}$ $\epsilon_{5,0} = \nu_{5,0} = 10^{-4}$	$\epsilon_{1,0} = \nu_{1,0} = 0.1$ $\epsilon_{2,0} = \nu_{2,0} = 0.01$ $\epsilon_{3,0} = \nu_{3,0} = 0.001$ $\epsilon_{4,0} = \nu_{4,0} = 10^{-4}$ $\epsilon_{5,0} = \nu_{5,0} = 10^{-4}$
Termination tol.	$\epsilon_{\text{opt}} = 10^{-4}$, $\nu_{\text{opt}} = 10^{-4}$	$\epsilon_{1,\text{opt}} = \nu_{1,\text{opt}} = 10^{-4}$ $\epsilon_{2,\text{opt}} = \nu_{2,\text{opt}} = 10^{-4}$ $\epsilon_{3,\text{opt}} = \nu_{3,\text{opt}} = 10^{-4}$ $\epsilon_{4,\text{opt}} = \nu_{4,\text{opt}} = 10^{-4}$ $\epsilon_{5,\text{opt}} = \nu_{5,\text{opt}} = 10^{-4}$	$\epsilon_{1,\text{opt}} = \nu_{1,\text{opt}} = 0.01$ $\epsilon_{2,\text{opt}} = \nu_{2,\text{opt}} = 0.001$ $\epsilon_{3,\text{opt}} = \nu_{3,\text{opt}} = 10^{-4}$ $\epsilon_{4,\text{opt}} = \nu_{4,\text{opt}} = 10^{-4}$ $\epsilon_{5,\text{opt}} = \nu_{5,\text{opt}} = 10^{-4}$
Reduction factors	$\theta_\epsilon = 0.1$, $\theta_\nu = 0.1$	$\theta_{\ell,\epsilon} = \theta_{\ell,\nu} = 0.1$ for $\ell = 1, \ldots, L$	$\theta_{\ell,\epsilon} = \theta_{\ell,\nu} = 0.1$ for $\ell = 1, \ldots, L$
Line search	$\beta = 10^{-4}$, $\gamma = 0.5$	$\beta_\ell = 10^{-4}$ $\gamma_\ell = 0.5$ for $\ell = 1, \ldots, L$	$\beta_\ell = 10^{-4}$ $\gamma_\ell = 0.5$ for $\ell = 1, \ldots, L$

Table 3: Number of sampled gradients per problem instance.

	rail example		cylinder example	
	system (14)	system (15)	system (14)	system (15)
# sampled gradients q	32	26	34	24

maximum number of allowed iterations per level as the computed iterates approach a minimizer of the highest fidelity objective. In the case of the cylinder example, we observed some stagnation in the lowest fidelity objective for high maximum iteration numbers, perhaps resulting from a mismatch in the approximation to the highest fidelity objective. Therefore, we chose here a smaller maximum iteration number than for the second level. The number of sampled gradients for all methods and in all problem instances is set to $q_\ell = q = N + 2$, for $\ell = 1, \ldots, L$. The resulting numbers are listed in Table 3. All methods are initialized with a randomly generated controller based on the same random seed, which is then stabilized by a gradient sampling method applied to the constraint function (11).

We compare RMFGS and AMFGS to the single-fidelity gradient sampling method from [19, Alg. GS] applied directly to the high-fidelity objective function f^L, denoted subsequently as HFGS. We compare the results for the different methods by comparing the evolution of the high-fidelity objective f^L on the iterate sequence $\{x^k\}$. In the case of RMFGS, which does not access f^L until its final outer loop, we computed $f^L(x^k)$ *a*

posteriori.

For each problem instance that we solve, since we do not know the minimal value of f^L, it is convenient to define

$$f_{\min} := \min\left(f^L(x_{\text{HFGS}}), f^L(x_{\text{RMFGS}}), f^L(x_{\text{AMFGS}})\right),$$

where the three quantities on the right-hand side are respectively the minimal values of f^L found by the three different methods. Then, in the figures below, for each problem instance we show two different plots of the evolution of $f^L(x^k)$. In the plots on the left, the vertical axis shows the values of f^L computed by each of the three methods, with different symbols indicating the discretization level, i.e., the index of the outer loop in the case of RMFGS and AMFGS. For HFGS, only the highest fidelity discretization symbol is used. In the plots on the right, the vertical axis shows the relative error

$$\frac{f^L(x^k) - f_{\min}}{f_{\min}},$$

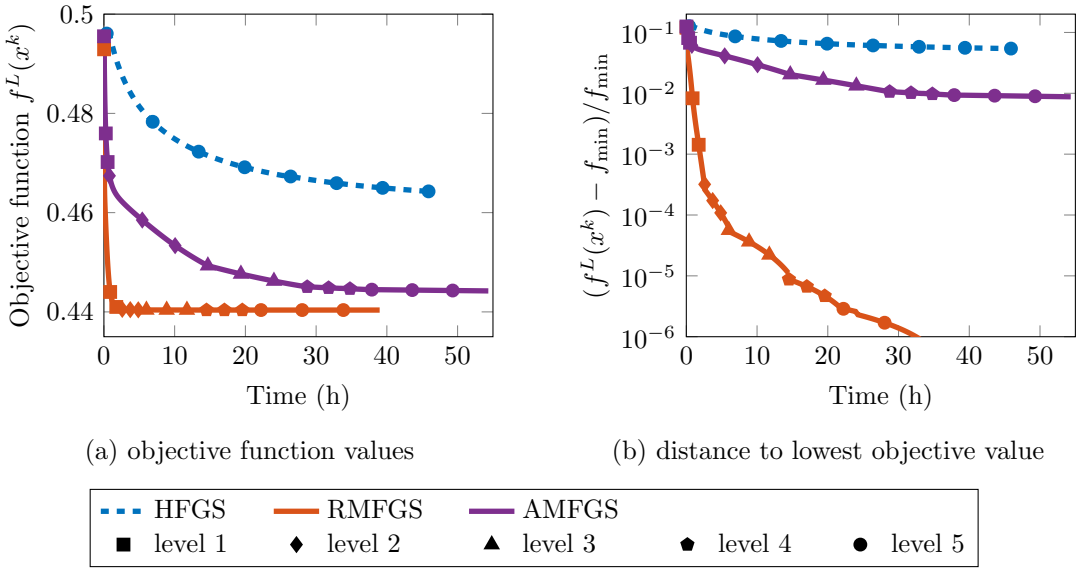
using $f_{\min}$ as our best estimate of the true minimal value. In both cases, the horizontal axis shows the running time in hours.

The experiments were run on compute nodes of the `Greene` high-performance computing cluster of New York University using 16 processing cores of the Intel Xeon Platinum 8268 24C 205W CPU at 2.90 GHz and 16 GB main memory. We used MATLAB 9.9.0.1467703 (R2020b) running on Red Hat Enterprise Linux release 8.4 (Ootpa). For the single-fidelity gradient sampling method, we used the implementation in HANSO, Hybrid Algorithm for Non-Smooth Optimization, version 3.0 [47]. The new multi-fidelity codes are also based on this. All the examples discussed below, except the first two levels of the rail example, use MATLAB's sparse data structure. For the computation of the $\mathcal{H}_\infty$-norm we employ the `normTfMaxPeak` and `normTfPeak` routines from ROSTAPACK (RObust STAbility PACKage), version 3.0 [41]; see also [14] for the implemented algorithms. As `normTfPeak` does not do a stability check, we implemented this using MATLAB's `eigs` function. The source code, data and results of the numerical experiments are open source/open access and available at [61].

4.3. Optimal cooling of a steel rail profile

We consider the heat flow on a two-dimensional cross section of a steel bar for optimal cooling; see [58] for further details and [59] for the data set. The underlying heat equation is discretized on multiple grid levels using finite elements. The resulting dimensions of the two open-loop systems (14) and (15) can be found in the rail example column of Table 1.

We first consider the example formulation (14). The results are shown in Figure 1 and Table 4. Even a quick glance reveals that both new methods are faster and more accurate than the single fidelity method HFGS, with RMFGS faster and more accurate than AMFGS. Indeed, even level 1 of the RMFGS method obtains about the same value for f^L, without even accessing the high fidelity model, as the final value found by HFGS, in less than 0.1 h, compared to 45 h required by HFGS to reach the same value of f^L. Furthermore, although the plot on the left side of Figure 1 suggests that RMFGS stagnates, the plot on the right side shows that this is not the case, with additional digits of accuracy steadily attained as the hierarchy level of RMFGS is increased. Overall, to reach the same objective function value, RMFGS achieves a speedup of 452 compared to HFGS and AMFGS achieves a speedup of 30. For all methods, the stabilization of the initial guess took only a single step of gradient sampling for the spectral abscissa constraint function. Even for Algorithm 1, no subsequent stabilization steps were required.



(a) objective function values

(b) distance to lowest objective value

Figure 1: Rail example with formulation (14): To reach the final objective function value found by HFGS, RMFGS achieves a speedup of 452 and AMFGS achieves a speedup of 30 in comparison. Additionally, RMFGS and AMFGS ultimately obtain lower objective function values than those found by using only the high-fidelity model in HFGS.

The second experiment that we consider for this application is for formulation (15). The disturbances are set to be the lower boundary temperatures and the controls are restricted to the boundary temperatures of the upper segments; see also [11, Sec. 3.2] where the same setup is used. The results are shown in Figure 2 and Table 5. In this case, although the results in absolute terms are not as much in favor of the new methods as they were for the previous example, in relative terms, RMFGS is much better than either of the other methods, and AMFGS gives much better results than the single fidelity method until after 10 h of computation. RMFGS and AMFGS reach the same level of the final objective function value of HFGS in about 1.5 h and both provide at the end of the iterations a smaller objective function value than HFGS. All methods needed only a single gradient sampling step to stabilize the closed-loop system at initialization.

4.4. Robust stabilization of laminar flows in a cylinder wake

We now consider the stabilization of laminar flow in a two-dimensional wake resulting from a circular obstacle. The flow is modeled as the linearization of the Navier-Stokes equations at Reynolds number 90 around the unstable non-zero steady state; see [7] for details. The spatial discretization is obtained with Taylor-Hood finite elements resulting in open-loop systems of the forms (14) and (15) described by differential-algebraic equations, i.e., the E matrices are singular. The model matrices have been obtained in differently sized discretizations using the codes from [7]. The resulting dimensions of the systems are given in the cylinder example column of Table 1.

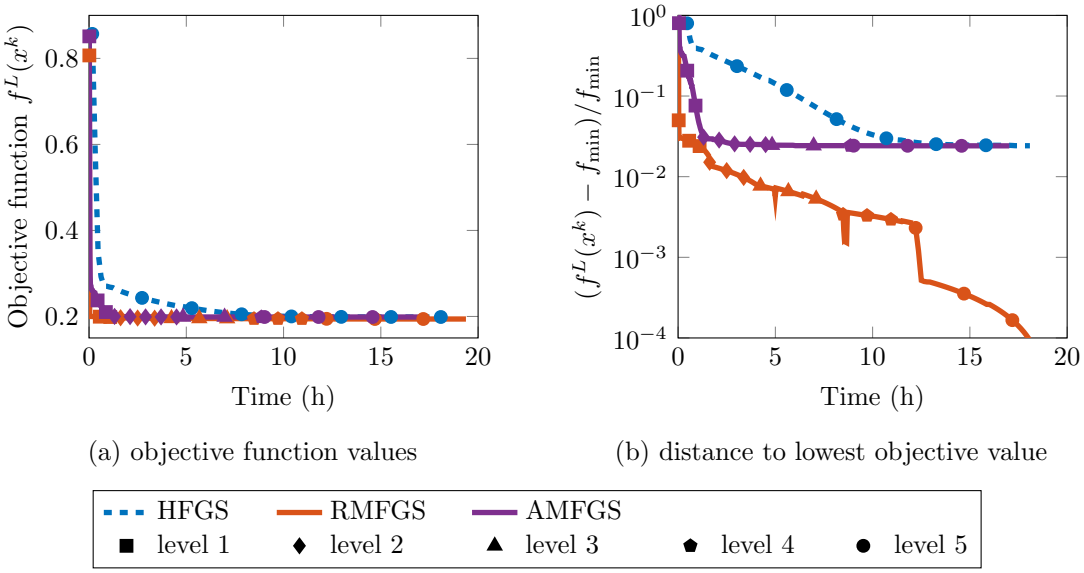
We first consider the formulation (14). The results of the computations can be found in Figure 3 and Table 6. The RMFGS method provides the lowest final objective function value of all methods within about the same runtime as HFGS. AMFGS converges in less than half of the runtime than that of RMFGS and HFGS but to a different objective function value than the one found by the other 2 methods, higher by a factor of

Table 4: Rail example with formulation (14): The table reports the wall-clock time of the computations, the number of iterations taken versus the maximum allowed number and the objective function values corresponding to the low-fidelity models (in case of RMFGS) and high-fidelity models.

		Time (h)	Iters./Max. Iters.	$f^\ell(x^{k_\ell})$	$f^L(x^{k_\ell})$
HFGS		45.895	120 / 120	—	0.464284
RMFGS	level 1	2.5594	5 000 / 5 000	0.440143	0.440511
	level 2	3.3656	1 000 / 1 000	0.440312	0.440399
	level 3	8.5062	500 / 500	0.440365	0.440375
	level 4	7.4379	100 / 100	0.440372	0.440372
	level 5	17.113	50 / 50	—	0.440370
		38.982	6 650 / 6 650	—	0.440370
AMFGS	level 1	0.7683	66 / 5 000	—	0.467422
	level 2	13.901	1 000 / 1 000	—	0.449315
	level 3	13.983	500 / 500	—	0.445053
	level 4	8.8870	100 / 100	—	0.444489
	level 5	16.805	50 / 50	—	0.444229
		54.363	1 716 / 6 650	—	0.444229

about 1.0058. An interesting point to observe here that we did not see earlier is that for RMFGS, the high-fidelity objective function value $f^L(x^k)$ is not monotonically decreasing as k increases. Particularly between 5 and 15 h, the high fidelity function value f^L increases. This indicates a mismatch in the approximation of the high-fidelity model by the low-fidelity model. Such convergence behavior cannot occur for AMFGS, which directly optimizes the high-fidelity objective function f^L. Indeed, in the region between 10 and 15 h, the objective function values obtained by AMFGS are smaller than for RMFGS and HFGS. However, when the discretization is refined, RMFGS overtakes AMFGS and eventually obtains a significantly better result. As previously, all three methods needed only a single gradient sampling step to stabilize the initial controller.

Finally, we consider the formulation (15) for the cylinder example. The original controls of the benchmark example are modeled to steer the flow velocities in horizontal and vertical directions behind the circular obstacle. We consider only the first half of these controls to introduce disturbances into the system, which is, for example, the case when control units are defective. The second half of the controls remain as given for the design of feedback controllers. The results for this example are shown in Figure 4 and Table 7. As earlier, RMFGS performs much better than AMFGS, which in turn performs much better than HFGS, obtaining lower values of f^L in less runtime. It requires AMFGS 10 h more computation time than RMFGS to reach a value of f^L that agrees with RMFGS to two digits. Compared to the final objective function value of HFGS, AMFGS performs around 2 times faster than HFGS and RMFGS is around 4 times faster than HFGS. For all three methods, only a single gradient sampling step is necessary to stabilize the initial guess for



(a) objective function values

(b) distance to lowest objective value

Figure 2: Rail example with formulation (15): To reach the final objective function value found by HFGS, RMFGS achieves a speedup of 17 and AMFGS a speedup of 2 in comparison.

the controller.

As an alternative to the relatively expensive gradient sampling method, we also experimented with using the BFGS method, which has proved very effective in other nonsmooth optimization applications [27, 37, 48]. However, we found that, particularly for the cylinder example, the behavior of gradient sampling was more consistent and reliable, perhaps reflecting its very satisfactory convergence theory, which is not shared by BFGS.

5. Conclusions

We have introduced two multi-fidelity gradient-sampling approaches for the robust control of expensive, high-fidelity models that leverage low-cost, low-fidelity models for speedup. The numerical examples demonstrate that speedups of several orders of magnitude can be achieved compared to a single-fidelity approach that uses the high-fidelity model alone. Furthermore, our RMFGS (Restarted Multi-Fidelity Gradient Sampling) method, which does not access the highest fidelity model until the final phase of the computation, consistently outperforms our AMFGS (Approximate Multi-Fidelity Gradient Sampling) method, which uses the high fidelity model throughout the computation, using lower fidelity gradients in the sampling step. One might have expected the opposite, since AMFGS monotonically reduces the high-fidelity objective function on the sequence $\{x^k\}$. However, as the cylinder example demonstrated (see Figure 3), even when RMFGS fails to reduce the high-fidelity function on a lower level of optimization, it can still recover when it continues to the next level of optimization. In fact, its robustness seems to reflect its stronger convergence properties. As explained in Section 3.5, the convergence guarantees of the gradient sampling algorithm apply at every level of the RMFGS method, while, because of the approximate gradients used by AMFGS, they apply only at the final level of AMFGS, which, in a sense, means that its convergence guarantees are no stronger than those of HFGS. One could argue that the consequence of this is that the result of optimization on one level of RMFGS really does provide a good starting point for optimization at the next

Table 5: Rail example with formulation (15): The table reports the wall-clock time of the computations, the number of iterations taken versus the maximum allowed number and the objective function values corresponding to the low-fidelity models (in case of RMFGS) and high-fidelity models.

		Time (h)	Iters./Max. Iters.	$f^{\ell}(x^{k_\ell})$	$f^{L}(x^{k_\ell})$
HFGS		18.085	120 / 120	—	0.198720
RMFGS	level 1	1.5963	5 000 / 5 000	0.197222	0.197473
	level 2	2.6002	1 000 / 1 000	0.195428	0.195475
	level 3	4.2480	500 / 500	0.194404	0.194740
	level 4	3.6196	100 / 100	0.194148	0.194543
	level 5	7.3114	50 / 50	—	0.194028
		19.375	6 650 / 6 650	—	0.194028
AMFGS	level 1	1.2626	86 / 5 000	—	0.200531
	level 2	3.4359	69 / 1 000	—	0.198870
	level 3	3.8725	29 / 500	—	0.198732
	level 4	0.2885	1 / 100	—	0.198732
	level 5	8.1653	50 / 50	—	0.198707
		17.043	235 / 6 650	—	0.198707

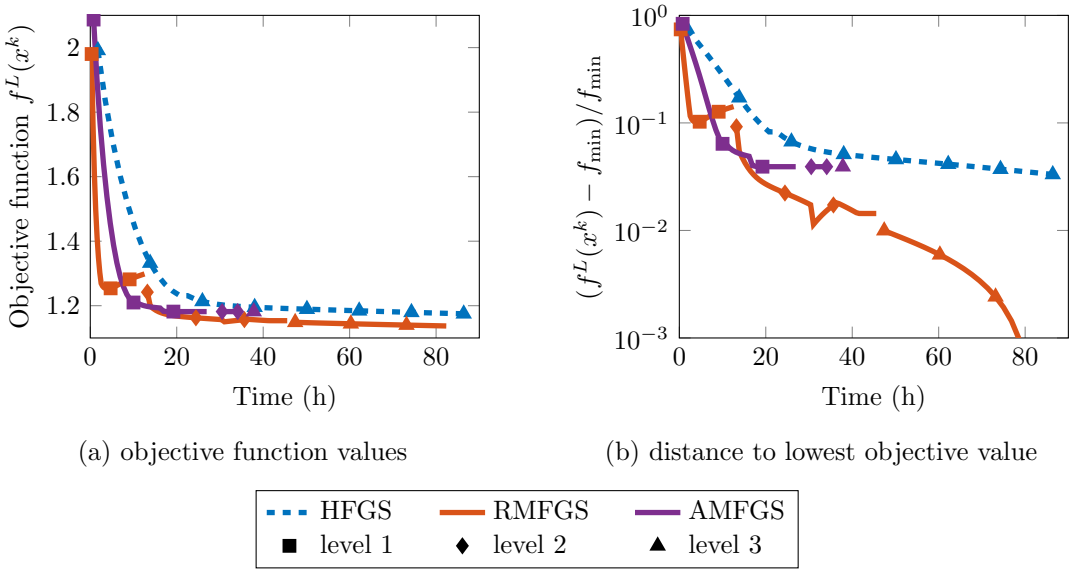
level; the same argument cannot be made for AMFGS.

An interesting question that we leave for future work is what convergence guarantees one might be able to derive for a variant of RMFGS where the discretization level increases without bound so that it asymptotically approximates a limit objective function that is computationally intractable. Such a situation can be found when the dynamical system stems from a discretization of an underlying partial differential equation and the limit $\ell \to \infty$ means driving the mesh width to zero to asymptotically approximate the continuous solution of the partial differential equation and its corresponding objective function. Such a setting is considered in the context of uncertainty quantification in, e.g., [26, 32, 49].

Acknowledgments

The authors were partially supported by the National Science Foundation under Grant No. 2012250. The third author was additionally supported by the National Science Foundation under Grant No. 1901091. This material is based upon work supported by the National Science Foundation under Grant No. DMS-1439786 and by the Simons Foundation Grant No. 50736 while the first and third author were in residence at the Institute for Computational and Experimental Research in Mathematics in Providence, RI, during the "Model and dimension reduction in uncertain and dynamic systems" program.

We would like to thank Tim Mitchell of the Max Planck Institute in Magdeburg, Germany, who provided a prerelease of ROSTAPACK version 3.0 (now publicly available) for the initial numerical experiments of this work, for the useful discussions with him about



(a) objective function values

(b) distance to lowest objective value

Figure 3: Cylinder example with formulation (14): AMFGS requires less than half of the runtime time of HFGS to converge but it converges to a different objective function value, higher by a factor of about 1.0058. RMFGS finds the lowest final objective function value of all three methods.

$\mathcal{H}_\infty$-norm computations and his valuable comments on a draft of this manuscript.

A. Gradients of the $\mathcal{H}_\infty$-norm of the closed-loop system

For the use of gradient sampling in $\mathcal{H}_\infty$-controller design, the gradients of the $\mathcal{H}_\infty$-norm (8) of the closed-loop system (3) with respect to the controller matrices from (2) are needed. These are well-known in the $\mathcal{H}_\infty$-control community, and, for the case of an identity descriptor matrix in (1), i.e., $E = I_n$, they can be found, for example, in [40]. We summarize these gradients here for completeness and include also the case of descriptor matrices as in (1). We are concerned with computing the gradients at a given design variable given by (10). We need to assume that, given these controller variables, the supremum in (7) is attained only at one finite point $\omega_{\mathcal{H}_\infty}$, with $\|G_c(\mathrm{i}\,\omega_{\mathcal{H}_\infty})\|_2 = \|G_c\|_{\mathcal{H}_\infty}$, and that the largest singular value of $G_c(\mathrm{i}\,\omega_{\mathcal{H}_\infty})$ is simple. Then the $\mathcal{H}_\infty$-norm of the closed-loop system (3) is indeed differentiable and its gradients with respect to the closed-loop system matrices are given by

$$\begin{aligned} \nabla_{A_c}\|G_c\|_{\mathcal{H}_\infty} &= Z^{-\mathsf{H}} C_c^{\mathsf{T}} u v^{\mathsf{H}} B_c^{\mathsf{T}} Z^{-\mathsf{H}}, & \nabla_{B_c}\|G_c\|_{\mathcal{H}_\infty} &= Z^{-\mathsf{H}} C_c^{\mathsf{T}} u v^{\mathsf{H}}, \\ \nabla_{C_c}\|G_c\|_{\mathcal{H}_\infty} &= u v^{\mathsf{H}} B_c^{\mathsf{T}} Z^{-\mathsf{H}}, & \nabla_{D_c}\|G_c\|_{\mathcal{H}_\infty} &= u v^{\mathsf{H}}, \end{aligned} \tag{16}$$

where $Z = \mathrm{i}\,\omega_{\mathcal{H}_\infty} E_c - A_c$, and u and v are the right and left singular vectors corresponding to the largest singular value of $G_c(\mathrm{i}\,\omega_{\mathcal{H}_\infty})$. Note that the gradient with respect to E_c is not needed since it does not involve any of the controller matrices, i.e., it contains no optimization variables for which the gradients need to be evaluated. However, the matrix E_c plays a role in (16) in terms of the frequency-dependent matrix pencil Z. Using the chain rule of differentiation we can directly obtain the requested gradients with respect to the controller matrices from (16). Additionally applying realification to the single terms, since we are only interested in the design of controllers realized by real-valued matrices,

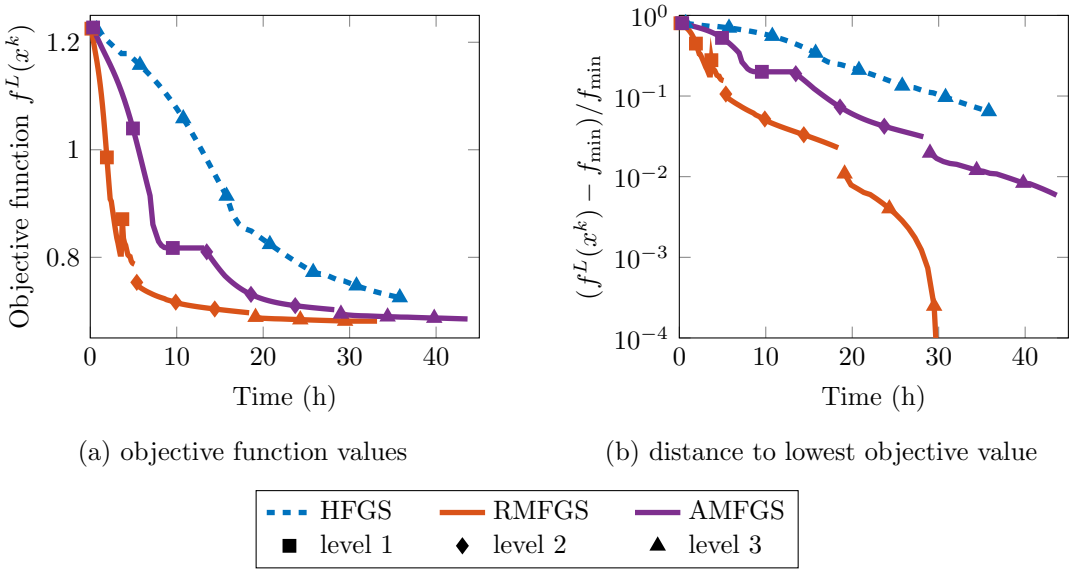
Table 6: Cylinder example in formulation (14): The table reports the wall-clock time of the computations, the number of iterations taken versus the maximum allowed number and the objective function values corresponding to the low-fidelity models (in case of RMFGS) and high-fidelity models.

		Time (h)	Iters./Max. Iters.	$f^{\ell}(x^{k_\ell})$	$f^{L}(x^{k_\ell})$
HFGS		86.390	50 / 50	—	1.174923
RMFGS	level 1	12.552	40 / 40	1.399568	1.298534
	level 2	33.007	50 / 50	1.152272	1.153551
	level 3	36.802	20 / 20	—	1.137206
		82.360	110 / 110	—	1.137206
AMFGS	level 1	26.924	35 / 40	—	1.181741
	level 2	7.1769	2 / 50	—	1.181741
	level 3	3.7238	1 / 20	—	1.181741
		37.852	38 / 110	—	1.181741

yields the following results:

$$
\begin{aligned}
\nabla_{A_\mathrm{K}} \|G_\mathrm{c}\|_{\mathcal{H}_\infty} &= \mathrm{Re}\left(\begin{bmatrix} 0 & I_{n_\mathrm{K}} \end{bmatrix} \nabla_{A_\mathrm{c}} \|G_\mathrm{c}\|_{\mathcal{H}_\infty} \begin{bmatrix} 0 \\ I_{n_\mathrm{K}} \end{bmatrix} \right), \\
\nabla_{B_\mathrm{K}} \|G_\mathrm{c}\|_{\mathcal{H}_\infty} &= \mathrm{Re}\left(\begin{bmatrix} 0 & I_{n_\mathrm{K}} \end{bmatrix} \nabla_{A_\mathrm{c}} \|G_\mathrm{c}\|_{\mathcal{H}_\infty} \begin{bmatrix} I_n \\ 0 \end{bmatrix} C_2^\mathsf{T} \right) \\
&\quad + \mathrm{Re}\left(\begin{bmatrix} 0 & I_{n_\mathrm{K}} \end{bmatrix} \nabla_{B_\mathrm{c}} \|G_\mathrm{c}\|_{\mathcal{H}_\infty} D_{21}^\mathsf{T} \right), \\
\nabla_{C_\mathrm{K}} \|G_\mathrm{c}\|_{\mathcal{H}_\infty} &= \mathrm{Re}\left(B_2^\mathsf{T} \begin{bmatrix} I_n & 0 \end{bmatrix} \nabla_{A_\mathrm{c}} \|G_\mathrm{c}\|_{\mathcal{H}_\infty} \begin{bmatrix} 0 \\ I_{n_\mathrm{K}} \end{bmatrix} \right) \\
&\quad + \mathrm{Re}\left(D_{12}^\mathsf{T} \nabla_{C_\mathrm{c}} \|G_\mathrm{c}\|_{\mathcal{H}_\infty} \begin{bmatrix} 0 \\ I_{n_\mathrm{K}} \end{bmatrix} \right), \\
\nabla_{D_\mathrm{K}} \|G_\mathrm{c}\|_{\mathcal{H}_\infty} &= \mathrm{Re}\left(B_2^\mathsf{T} \begin{bmatrix} I_n & 0 \end{bmatrix} \nabla_{A_\mathrm{c}} \|G_\mathrm{c}\|_{\mathcal{H}_\infty} \begin{bmatrix} I_n \\ 0 \end{bmatrix} C_2^\mathsf{T} \right) \\
&\quad + \mathrm{Re}\left(B_2^\mathsf{T} \begin{bmatrix} I_n & 0 \end{bmatrix} \nabla_{B_\mathrm{c}} \|G_\mathrm{c}\|_{\mathcal{H}_\infty} D_{21}^\mathsf{T} \right) \\
&\quad + \mathrm{Re}\left(D_{12}^\mathsf{T} \nabla_{C_\mathrm{c}} \|G_\mathrm{c}\|_{\mathcal{H}_\infty} \begin{bmatrix} I_n \\ 0 \end{bmatrix} C_2^\mathsf{T} \right) \\
&\quad + \mathrm{Re}\left(D_{12}^\mathsf{T} \nabla_{D_\mathrm{c}} \|G_\mathrm{c}\|_{\mathcal{H}_\infty} D_{21}^\mathsf{T} \right).
\end{aligned}
\tag{17}
$$

Given the $\mathcal{H}_\infty$-frequency point $\omega_{\mathcal{H}_\infty}$, the gradients in (17) can be cheaply obtained. This is especially the case when A_c and E_c are large-scale and sparse by using appropriate factorizations of the matrix products above. There have been recent advances in the computation of the $\mathcal{L}_\infty$-norm of large-scale sparse systems [2, 14], which also yield an efficient approximation of $\omega_{\mathcal{H}_\infty}$.



(a) objective function values

(b) distance to lowest objective value

Figure 4: Cylinder example with formulation (15): Both RMFGS and AMFGS obtain smaller objective function values than HFGS does and in a shorter runtime, corresponding to speedups of 4 and 2, respectively.

B. Gradients of the spectral abscissa for initial stabilization

The gradients of (6) with respect to the controller matrices (2) are well known in the literature for the standard system case, i.e., $E_c = I_{n+n_K}$; see, for example, [22] and the implementation in [40]. Let the design variable be given by (10). We need to assume that the spectral abscissa of the corresponding matrix pencil (A_c, E_c) is attained at only one eigenvalue in the closed upper half of the complex plane, say λ_α with $\mathrm{Re}(\lambda_\alpha) = \alpha(A_c, E_c)$, and that this eigenvalue is simple. Then the spectral abscissa is indeed differentiable, with the gradient, with respect to A_c, given by

$$\nabla_{A_c} \alpha(A_c, E_c) = w v^{\mathsf{H}},$$

where v is the right generalized eigenvector of λ_α and w is the corresponding left eigenvector, normalized with respect to the inner product with E_c, i.e., such that

$$w^{\mathsf{H}} E_c v = 1.$$

Note that we do not need the gradient with respect to E_c since this matrix does not contain any matrix of the controller (2). Applying the chain rule and realification of the resulting terms, since we are only interested in controllers with real-valued matrices, yields the gradients of interest given by

$$\begin{aligned}
\nabla_{A_K} \alpha(A_c, E_c) &= \mathrm{Re}\left(\begin{bmatrix} 0 & I_{n_K} \end{bmatrix} \nabla_{A_c} \alpha(A_c, E_c) \begin{bmatrix} 0 \\ I_{n_K} \end{bmatrix} \right), \\
\nabla_{B_K} \alpha(A_c, E_c) &= \mathrm{Re}\left(\begin{bmatrix} 0 & I_{n_K} \end{bmatrix} \nabla_{A_c} \alpha(A_c, E_c) \begin{bmatrix} I_n \\ 0 \end{bmatrix} C_2^{\mathsf{T}} \right), \\
\nabla_{C_K} \alpha(A_c, E_c) &= \mathrm{Re}\left(B_2^{\mathsf{T}} \begin{bmatrix} I_n & 0 \end{bmatrix} \nabla_{A_c} \alpha(A_c, E_c) \begin{bmatrix} 0 \\ I_{n_K} \end{bmatrix} \right), \\
\nabla_{D_K} \alpha(A_c, E_c) &= \mathrm{Re}\left(B_2^{\mathsf{T}} \begin{bmatrix} I_n & 0 \end{bmatrix} \nabla_{A_c} \alpha(A_c, E_c) \begin{bmatrix} I_n \\ 0 \end{bmatrix} C_2^{\mathsf{T}} \right).
\end{aligned}$$

Table 7: Cylinder example with formulation (15): The table reports the wall-clock time of the computations, the number of iterations taken versus the maximum allowed number and the objective function values corresponding to the low-fidelity models (in case of RMFGS) and high-fidelity models.

		Time (h)	Iters./Max. Iters.	$f^\ell(x^{k_\ell})$	$f^L(x^{k_\ell})$
HFGS		35.803	50 / 50	—	0.725212
RMFGS	level 1	5.1153	40 / 40	0.786143	0.787230
	level 2	13.262	50 / 50	0.696799	0.696859
	level 3	14.795	20 / 20	—	0.681304
		33.172	110 / 110	—	0.681304
AMFGS	level 1	13.169	40 / 40	—	0.817351
	level 2	15.055	50 / 50	—	0.702572
	level 3	15.415	20 / 20	—	0.685316
		43.663	110 / 110	—	0.685316

The right-most eigenvalues and eigenvectors of large-scale sparse matrix pencils can be efficiently computed using an Arnoldi or Krylov-Schur method with the shift-and-invert operator and a suitable shift σ with a real part larger than or close to $\alpha(A_c, E_c)$; see, e.g., [33, 60]. The shift σ can be efficiently updated during an optimization approach using the previous computations of $\alpha(A_c, E_c)$. In our numerical experiments, we use the `eigs` function from MATLAB, which in its latest version implements the Krylov-Schur algorithm [60].

References

[1] N. M. Alexandrov, J. E. Dennis Jr., R. M. Lewis, and V. Torczon. A trust-region framework for managing the use of approximation models in optimization. *Structural optimization*, 15:16–23, 1998. doi:10.1007/BF01197433.

[2] N. Aliyev, P. Benner, E. Mengi, and M. Voigt. A subspace framework for $\mathcal{H}_\infty$-norm minimization. *SIAM J. Matrix Anal. Appl.*, 41(2):928–956, 2020. doi:10.1137/19M125892X.

[3] T. Alsup, L. Venturi, and B. Peherstorfer. Multilevel Stein variational gradient descent with applications to Bayesian inverse problems. In J. Bruna, J. Hesthaven, and L. Zdeborova, editors, *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference*, volume 145, pages 93–117, 2022. URL: https://proceedings.mlr.press/v145/alsup22a.html.

[4] A. C. Antoulas. *Approximation of Large-Scale Dynamical Systems*, volume 6 of *Adv. Des. Control*. SIAM, Philadelphia, PA, 2005. doi:10.1137/1.9780898718713.

[5] E. Arian, M. Fahl, and E. W. Sachs. Managing POD models by optimization methods.

In *Proceedings of the 41st IEEE Conference on Decision and Control*, pages 3300–3305, 2002. doi:10.1109/CDC.2002.1184383.

[6] A. Asl and M. L. Overton. Behavior of limited memory BFGS when applied to nonsmooth functions and their Nesterov smoothings. In M. Al-Baali, A. Purnama, and L. Grandinetti, editors, *Numerical Analysis and Optimization*, volume 354 of *Springer Proc. Math. Stat.*, pages 25–55. Springer, Cham, 2021. doi:10.1007/978-3-030-72040-7_2.

[7] M. Behr, P. Benner, and J. Heiland. Example setups of Navier-Stokes equations with control and observation: Spatial discretization and representation via linear-quadratic matrix coefficients. e-print arXiv:1707.08711, arXiv, 2017. Mathematical Software (cs.MS). doi:10.48550/arXiv.1707.08711.

[8] P. Benner. Solving large-scale control problems. *IEEE Control Syst. Mag.*, 24(1):44–59, 2004. doi:10.1109/MCS.2004.1272745.

[9] P. Benner, A. Cohen, M. Ohlberger, and K. Willcox. *Model Reduction and Approximation: Theory and Algorithms*. Computational Science & Engineering. SIAM, Philadelphia, PA, 2017. doi:10.1137/1.9781611974829.

[10] P. Benner, S. Gugercin, and K. Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Rev.*, 57(4):483–531, 2015. doi:10.1137/130932715.

[11] P. Benner, J. Heiland, and S. W. R. Werner. A low-rank solution method for riccati equations with indefinite quadratic terms. e-print 2111.06516, arXiv, 2021. Numerical Analysis (math.NA). doi:10.48550/arXiv.2111.06516.

[12] P. Benner, J. Heiland, and S. W. R. Werner. Robust output-feedback stabilization for incompressible flows using low-dimensional $\mathcal{H}_\infty$-controllers. *Comput. Optim. Appl.*, 82(1):225–249, 2022. doi:10.1007/s10589-022-00359-x.

[13] P. Benner, J.-R. Li, and T. Penzl. Numerical solution of large-scale Lyapunov equations, Riccati equations, and linear-quadratic optimal control problems. *Numer. Lin. Alg. Appl.*, 15(9):755–777, 2008. doi:10.1002/nla.622.

[14] P. Benner and T. Mitchell. Faster and more accurate computation of the $\mathcal{H}_\infty$ norm via optimization. *SIAM J. Sci. Comput.*, 40(5):A3609–A3635, 2018. doi:10.1137/17M1137966.

[15] P. Benner, T. Mitchell, and M. L. Overton. Low-order control design using a reduced-order model with a stability constraint on the full-order model. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 3000–3005, 2018. doi:10.1109/CDC.2018.8619449.

[16] P. Benner, W. Schilders, S. Grivet-Talocia, A. Quarteroni, G. Rozza, and L. M. Silveira. *Model Order Reduction. Volume 1: System- and Data-Driven Methods and Algorithms*. De Gruyter, Berlin, Boston, 2021. doi:10.1515/9783110498967.

[17] P. Benner, W. Schilders, S. Grivet-Talocia, A. Quarteroni, G. Rozza, and L. M. Silveira. *Model Order Reduction. Volume 2: Snapshot-Based Methods and Algorithms*. De Gruyter, Berlin, Boston, 2021. doi:10.1515/9783110671490.

[18] J. M. Borwein and A. S. Lewis. *Convex Analysis and Nonlinear Optimization*. CMS Books in Mathematics. Springer, New York, NY, 2006. doi:10.1007/978-0-387-31256-9.

[19] J. V. Burke, F. E. Curtis, A. S. Lewis, M. L. Overton, and L. E. A. Simões. Gradient sampling methods for nonsmooth optimization. In A. M. Bagirov, M. Gaudioso, N. Karmitsa, M. M. Mäkelä, and S. Taheri, editors, *Numerical Nonsmooth Optimization: State of the Art Algorithms*, pages 201–225. Springer, Cham, 2020. doi:10.1007/978-3-030-34910-3_6.

[20] J. V. Burke, D. Henrion, A. S. Lewis, and M. L. Overton. HIFOO - A MATLAB package for fixed-order controller design and H_∞ optimization. *IFAC Proceedings Volumes*, 39(9):339–344, 2006. 5th IFAC Symposium on Robust Control Design. doi:10.3182/20060705-3-FR-2907.00059.

[21] J. V. Burke, D. Henrion, A. S. Lewis, and M. L. Overton. Stabilization via nonsmooth, nonconvex optimization. *IEEE Trans. Autom. Control*, 51(11):1760–1769, 2006. doi:10.1109/TAC.2006.884944.

[22] J. V. Burke, A. S. Lewis, and M. L. Overton. Two numerical methods for optimizing matrix stability. *Linear Algebra Appl.*, 351–352:117–145, 2002. doi:10.1016/S0024-3795(02)00260-4.

[23] J. V. Burke, A. S. Lewis, and M. L. Overton. A nonsmooth, nonconvex optimization approach to robust stabilization by static output feedback and low-order controllers. *IFAC Proceedings Volumes*, 36(11):175–181, 2003. 4th IFAC Symposium on Robust Control Design. doi:10.1016/S1474-6670(17)35659-8.

[24] J. V. Burke, A. S. Lewis, and M. L. Overton. A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM J. Optim.*, 15(3):751–779, 2005. doi:10.1137/030601296.

[25] A. Chaudhuri, B. Peherstorfer, and K. Willcox. Multifidelity cross-entropy estimation of conditional value-at-risk for risk-averse design optimization. In *AIAA Scitech 2020 Forum*, pages AIAA 2020–2129, 2020. doi:10.2514/6.2020-2129.

[26] K. A. Cliffe, M. B. Giles, R. Scheichl, and A. L. Teckentrup. Multilevel Monte Carlo methods and applications to elliptic PDEs with random coefficients. *Comput. Vis. Sci.*, 14(1):3, 2011. doi:10.1007/s00791-011-0160-x.

[27] F. E. Curtis, T. Mitchell, and M. L. Overton. A BFGS-SQP method for nonsmooth, nonconvex, constrained optimization and its evaluation using relative minimization profiles. *Optim. Methods Softw.*, 32(1):148–181, 2017. doi:10.1080/10556788.2016.1208749.

[28] J. Doyle, K. Glover, P. P. Khargonekar, and B. A. Francis. State-space solutions to standard $\mathcal{H}_2$ and $\mathcal{H}_\infty$ control problems. *IEEE Trans. Autom. Control*, 34(8):831–847, 1989. doi:10.1109/9.29425.

[29] M. Fahl and E. W. Sachs. Reduced order modelling approaches to PDE-constrained optimization based on proper orthogonal decomposition. In L. T. Biegler, M. Heinkenschloss, O. Ghattas, and B. Van Bloemen Waanders, editors, *Large-Scale PDE-Constrained Optimization*, volume 30 of *Lect. Notes Comput. Sci. Eng.*, pages 268–280. Springer, Berlin, Heidelberg, 2003. doi:10.1007/978-3-642-55508-4_16.

[30] C. C. Fischer, R. V. Grandhi, and P. S. Beran. Bayesian low-fidelity correction approach to multi-fidelity aerospace design. In *58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, pages AIAA 2017–0132, 2017. doi:10.2514/6.2017-0133.

[31] B. A. Francis. *A Course in $\mathcal{H}_\infty$ Control Theory*, volume 88 of *Lect. Notes Control Inf. Sci.* Springer, Berlin, Heidelberg, 1987. doi:10.1007/BFb0007371.

[32] M. B. Giles. Multilevel Monte Carlo path simulation. *Oper. Res.*, 56(3):607–617, 2008. doi:10.1287/opre.1070.0496.

[33] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, fourth edition, 2013.

[34] K. C. Kiwiel. Convergence of the gradient sampling algorithm for nonsmooth nonconvex optimization. *SIAM J. Optim.*, 18(2):379–388, 2007. doi:10.1137/050639673.

[35] B. Kramer, B. Peherstorfer, and K. Willcox. Feedback control for systems with uncertain parameters using online-adaptive reduced models. *SIAM J. Appl. Dyn. Syst.*, 16(3):1563–1586, 2017. doi:10.1137/16M1088958.

[36] F. Law, A. Cerfon, and B. Peherstorfer. Accelerating the estimation of collisionless energetic particle confinement statistics in stellarators using multifidelity Monte Carlo. *Nucl. Fusion*, 62(7):076019, 2022. doi:10.1088/1741-4326/ac4777.

[37] A. S. Lewis and M. L. Overton. Nonsmooth optimization via quasi-Newton methods. *Math. Program.*, 141(1–2):135–163, 2012. doi:10.1007/s10107-012-0514-2.

[38] A. March and K. Willcox. Constrained multifidelity optimization using model calibration. *Struct. Multidiscip. Optim.*, 46(1):93–109, 2012. doi:10.1007/s00158-011-0749-1.

[39] A. March and K. Willcox. Provably convergent multifidelity optimization algorithm not requiring high-fidelity derivatives. *AIAA J.*, 50(5):1079–1089, 2012. doi:10.2514/1.J051125.

[40] M. Millstone, M. L. Overton, D. Henrion, G. Deaconu, S. Gumussoy, and D. Arzelier. HIFOO: A MATLAB package for fixed-order $\mathcal{H}_\infty$ and $\mathcal{H}_2$ controller design (version 3.5), 2011. URL: https://cs.nyu.edu/~overton/software/hifoo/.

[41] T. Mitchell. ROSTAPACK: RObust STAbility PACKage (version 3.0), May 2022. URL: http://www.timmitchell.com/software/ROSTAPACK/.

[42] T. Mitchell and M. L. Overton. Fixed low-order controller design and H_∞ optimization for large-scale dynamical systems. *IFAC-Pap.*, 48(14):25–30, 2015. 8th IFAC Symposium on Robust Control Design ROCOND 2015. doi:10.1016/j.ifacol.2015.09.428.

[43] D. Mustafa and K. Glover. Controller reduction by $\mathcal{H}_\infty$-balanced truncation. *IEEE Trans. Autom. Control*, 36(6):668–682, 1991. doi:10.1109/9.86941.

[44] L. W. T. Ng and K. E. Willcox. Multifidelity approaches for optimization under uncertainty. *Int. J. Numer. Methods Eng.*, 100(10):746–772, 2014. doi:10.1002/nme.4761.

[45] L. W. T. Ng and K. E. Willcox. Monte Carlo information-reuse approach to aircraft conceptual design optimization under uncertainty. *J. Aircr.*, 53(2):427–438, 2016. doi:10.2514/1.C033352.

[46] G. Obinata and B. D. O. Anderson. *Model Reduction for Control System Design*. Communications and Control Engineering. Springer, London, 2001. doi:10.1007/978-1-4471-0283-0.

[47] M. L. Overton. HANSO: Hybrid Algorithm for Non-Smooth Optimization (version 3.0), 2021. URL: https://cs.nyu.edu/~overton/software/hanso/.

[48] M. L. Overton. Local minimizers of the Crouzeix ratio: a nonsmooth optimization case study. *Calcolo*, 59(1):8, 2022. doi:10.1007/s10092-021-00448-z.

[49] B. Peherstorfer, M. Gunzburger, and K. Willcox. Convergence analysis of multifidelity Monte Carlo estimation. *Numer. Math.*, 139(3):683–707, 2018. doi:10.1007/s00211-018-0945-7.

[50] B. Peherstorfer and K. Willcox. Dynamic data-driven reduced-order models. *Comput. Methods Appl. Mech. Eng.*, 291:21–41, 2015. doi:10.1016/j.cma.2015.03.018.

[51] B. Peherstorfer and K. Willcox. Online adaptive model reduction for nonlinear systems via low-rank updates. *SIAM J. Sci. Comput.*, 37(4):A2123–A2150, 2015. doi:10.1137/140989169.

[52] B. Peherstorfer, K. Willcox, and M. Gunzburger. Optimal model management for multifidelity Monte Carlo estimation. *SIAM J. Sci. Comput.*, 38(5):A3163–A3194, 2016. doi:10.1137/15M1046472.

[53] B. Peherstorfer, K. Willcox, and M. Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *SIAM Rev.*, 60(3):550–591, 2018. doi:10.1137/16M1082469.

[54] E. Qian, M. Grepl, K. Veroy, and K. Willcox. A certified trust region reduced basis approach to PDE-constrained optimization. *SIAM J. Sci. Comput.*, 39(5):S434–S460, 2017. doi:10.1137/16M1081981.

[55] A. Quarteroni and G. Rozza. *Reduced Order Methods for Modeling and Computational Reduction*, volume 9 of *MS&A – Modeling, Simulation and Applications*. Springer, Cham, 2014. doi:10.1007/978-3-319-02090-7.

[56] T. Reis and T. Stykel. A survey on model reduction of coupled systems. In W. H. A. Schilders, H. A. Van der Vorst, and J. Rommes, editors, *Model Order Reduction: Theory, Research Aspects and Applications*, volume 13 of *Mathematics in Industry*, pages 133–155. Springer, Berlin, Heidelberg, 2008. doi:10.1007/978-3-540-78841-6_7.

[57] T. D. Robinson, M. S. Eldred, K. E. Willcox, and R. Haimes. Surrogate-based optimization using multifidelity models with variable parameterization and corrected space mapping. *AIAA J.*, 46(11):2814–2822, 2012. doi:10.2514/1.36043.

[58] J. Saak. *Efficient Numerical Solution of Large Scale Algebraic Matrix Equations in PDE Control and Model Order Reduction*. Dissertation, Technische Universität Chemnitz, Germany, 2009. URL: https://nbn-resolving.org/urn:nbn:de:bsz:ch1-200901642.

[59] J. Saak, M. Köhler, and P. Benner. M-M.E.S.S. – The Matrix Equations Sparse Solvers library (version 2.1), April 2021. see also: https://www.mpi-magdeburg.mpg.de/projects/mess. doi:10.5281/zenodo.4719688.

[60] G. W. Stewart. A Krylov–Schur algorithm for large eigenproblems. *SIAM J. Matrix Anal. Appl.*, 23(3):601–614, 2001. doi:10.1137/S0895479800371529.

[61] S. W. R. Werner. Code, data and results for numerical experiments in "Multi-fidelity robust controller design with gradient sampling" (version 1.0), May 2022. doi:10.5281/zenodo.6403121.

[62] S. M. Wild and C. Shoemaker. Global convergence of radial basis function trust region derivative-free algorithms. *SIAM J. Optim.*, 21(3):761–781, 2011. doi:10.1137/09074927X.

[63] M. J. Zahr and C. Farhat. Progressive construction of a parametric reduced-order model for PDE-constrained optimization. *Int. J. Numer. Methods Eng.*, 102(5):1111–1135, 2014. doi:10.1002/nme.4770.

[64] K. Zhou and J. C. Doyle. *Essentials of Robust Control*. Prentice-Hall, Upper Saddle River, NJ, 1998.