2018 IEEE Conference on Decision and Control (CDC)
Miami Beach, FL, USA, Dec. 17-19, 2018

Low-Order Control Design using a Reduced-Order Model with a Stability Constraint on the Full-Order Model

Peter Benner[1], Tim Mitchell[1] and Michael L. Overton[2]

***Abstract*— We propose a new algorithm for designing low-order controllers for large-scale linear time-invariant (LTI) dynamical systems with input and output. While the high cost of working with large-scale systems can mostly be avoided by first applying model order reduction, this can often result in controllers which fail to stabilize the closed-loop plant of the original full-order system. By considering a modified version of the optimal H_∞ controller problem that incorporates both full- and reduced-order model data, our new method ensures stability while remaining efficient. Using a publicly available test set, we find that the controllers obtained by our method outperform those computed by HIFOO (H-Infinity Fixed-Order Optimization) when applied to reduced-order models alone.**

I. Introduction

Consider the open-loop LTI dynamical system [1]

$$\begin{bmatrix} \dot{x} \\ \hline z \\ y \end{bmatrix} = \left[\begin{array}{c|cc} A_1 & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right] \begin{bmatrix} x \\ \hline w \\ u \end{bmatrix} \tag{1}$$

where the state $x \in \mathbb{R}^{n_x}$, the physical (control) input $u \in \mathbb{R}^{n_u}$, the performance input $w \in \mathbb{R}^{n_w}$, the physical (measured) output $y \in \mathbb{R}^{n_y}$, the performance output $z \in \mathbb{R}^{n_z}$, and the matrices are real valued with compatible dimensions. We wish to design a controller K defining

$$\begin{bmatrix} \dot{x}_K \\ u \end{bmatrix} = K \begin{bmatrix} x_K \\ y \end{bmatrix} = \begin{bmatrix} A_K & B_K \\ C_K & D_K \end{bmatrix} \begin{bmatrix} x_K \\ y \end{bmatrix},$$

where $x_K \in \mathbb{R}^{n_K}$ is the controller state and n_K is the order of the controller, resulting in the closed-loop system:

$$\begin{bmatrix} \dot{x} \\ \dot{x}_K \\ \hline z \end{bmatrix} = \left[\begin{array}{c|c} \mathcal{A} & \mathcal{B} \\ \hline \mathcal{C} & \mathcal{D} \end{array}\right] \begin{bmatrix} x \\ x_K \\ \hline w \end{bmatrix}$$

with, assuming $D_{22} = 0$ for convenience:

$$\begin{aligned} \mathcal{A} &= \begin{bmatrix} A_1 + B_2 D_K C_2 & B_2 C_K \\ B_K C_2 & A_K \end{bmatrix} \\ \mathcal{B} &= \begin{bmatrix} B_1 + B_2 D_K D_{21} \\ B_K D_{21} \end{bmatrix} \\ \mathcal{C} &= \begin{bmatrix} C_1 + D_{12} D_K C_2 & D_{12} C_K \end{bmatrix} \\ \mathcal{D} &= \begin{bmatrix} D_{11} + D_{12} D_K D_{21} \end{bmatrix}. \end{aligned} \tag{2}$$

The number of variables in the controller K is

$$n_{\text{var}} = (n_K + n_u) \times (n_K + n_y).$$

The formulas in (2) are affine unless $D_{22} \neq 0$, in which case they involve $(I - D_K D_{22})^{-1}$ [1, p. 446]; the condition $\|D_K D_{22}\|_2 < 1$ ensures that this last matrix is well defined. However, as the test problems we use here all have $D_{22} = 0$, we can forgo this additional requirement.

Now consider the closed-loop transfer matrix function

$$G(s) = \mathcal{C}(sI - \mathcal{A})^{-1}\mathcal{B} + \mathcal{D}, \tag{3}$$

which maps the performance input w to the performance output z. The optimal H_∞ controller is

$$K_\star = \underset{A_K, B_K, C_K, D_K}{\arg\min} \|G\|_{H_\infty}, \tag{4}$$

where the H_∞ norm is defined as:

$$\|G\|_{H_\infty} = \sup_{\lambda \in \mathbb{C},\, \Re\lambda \geq 0} \|G(\lambda)\|_2. \tag{5}$$

Problem (4) is a particularly challenging nonsmooth, nonconvex optimization problem. Furthermore, since the H_∞ norm is infinite whenever $\mathcal{A}$ is unstable, (4) is actually a *constrained* optimization problem, with the implicit stability constraint $\alpha(\mathcal{A}) < 0$, where mapping $\alpha : \mathbb{R}^{n \times n} \to \mathbb{R}$ denotes the spectral abscissa:

$$\alpha(M) = \max\{\Re\lambda : \det(\lambda I - M) = 0\}$$

of a matrix M. While the characterization of all stabilizing controllers is known [2], [3], this set, the feasible region of (4), is typically nonconvex. Worse, although the H_∞ norm objective function in (4) is at least locally Lipschitz, it is nevertheless typically nonsmooth at minimizers, and the spectral abscissa stability constraint is not only nonsmooth but not even locally Lipschitz, specifically when a rightmost eigenvalue λ of $\mathcal{A}$, i.e. with $\alpha(\mathcal{A}) = \Re\lambda$, has multiplicity two or more [4]. These difficult properties generally make it untenable to just hand off the responsibility of solving (4) to an off-the-shelf optimization code.

Although methods for solving (4) with a *full-order controller*, i.e. when $n_K = n_x$, are well known, it is often preferred to design *low-order controllers* where $n_K \ll n_x$ since these controllers must be realized physically. Choosing $n_K < n_x$ makes (4) even more difficult to solve but, despite the aforementioned challenging properties of the objective and stability constraint functions, the two current state-of-the-art methods for low-order H_∞ controller design are nevertheless both optimization-based, taking advantage of specialized techniques to handle nonsmooth functions. The first of these methods, the open-source toolbox HIFOO [5], [6], was released in 2006 and is based in part on [7]. The second, the proprietary `hinfstruct` [8], was first introduced

[1] Max Planck Institute for Dynamics of Complex Technical Systems Magdeburg, {benner,mitchell}@mpi-magdeburg.mpg.de. Supported in part by the RTG 2297/1, "Mathematical Complexity Reduction - MathCoRe", funded by Deutsche Forschungsgemeinschaft.

[2] Courant Institute of Mathematical Sciences, New York University, overton@cims.nyu.edu. Supported in part by the U.S. National Science Foundation grant DMS-1620083.

in MATLAB R2010b and is based in part on [9], [10]. While neither HIFOO nor `hinfstruct` can guarantee returning a global minimizer of (4), i.e. $K_\star$, they are efficient and typically return feasible, locally-optimal controllers which are sufficient in actual applications. Furthermore, as these optimization-based methods typically have at most $\mathcal{O}(n_{\mathrm{var}}^2)$ work complexity per iteration, they are typically much faster than linear matrix inequality (LMI) based approaches, which generally require at least *quartic* work complexity per iteration with respect to n_{var} [11, Section 2.4.4].

II. Controller Design for Large-scale Systems

The main computational limiting factor in designing controllers for large-scale systems is that just computing the H_∞ norm of (3) has $\mathcal{O}((n_x + n_K)^3)$ work complexity per iteration, since the known algorithms (e.g. the method of [12], [13], which is implemented in the MATLAB function `getPeakGain`) require computing all eigenvalues of a sequence of Hamiltonian matrices of order $2(n_x + n_K)$. As such, neither HIFOO nor `hinfstruct` is practical for large-scale use, at least not directly. One possible alternative is to first apply reduced-order modeling (e.g. [14], [15], [16], [17]) to make n_x significantly smaller and then instead design a controller with respect to that smaller system. While this can alleviate much of the computational burden of working with the original full-order model (FOM), designing a controller for a reduced-order model (ROM) may ultimately result in an unstable closed-loop plant for the FOM. Although there exist methods for producing low-order controllers which do guarantee closed-loop stability of the FOM, in particular frequency-weighted balancing techniques that perform a combined plant-and-controller reduction [18], these generally assume that it is computationally feasible to work with FOMs and full- or high-order controllers. The general philosophy of these methods is that dimension reductions should be done as late as possible to preserve accuracy. Hence, such techniques are more aimed at systems of at most moderate size, where it is still reasonably tractable to compute a full-order controller but it is nevertheless preferable for engineering considerations to implement a low-order representation.

To our knowledge, the first and currently only method for designing low-order controllers of truly large-scale systems is that of [19], which is an experimental extension of HIFOO called HIFOOS (where the S denotes *sparse*). In order to circumvent the high cost of *computing* the H_∞ norm, HIFOOS instead leverages the recent introduction of scalable methods for *approximating* the H_∞ norm, specifically the hybrid-expansion-contraction (HEC) algorithm of [20], to directly design controllers of FOMs. HEC guarantees a lower bound on $\|G\|_{H_\infty}$ and, under reasonable assumptions, converges to stationary points of (5), which are typically locally or globally optimal. As shown in [19, Table 2], this new approach consistently led to stable closed-loop systems for the FOMs, while using HIFOO to design controllers only with respect to the ROMs resulted in 5 of 12 controllers that failed to stabilize the original FOMs. Nevertheless, it was noted that approximating $\|G\|_{H_\infty}$ of the FOM could sometimes lead to inconsistencies from one controller K to another, effectively implying that the function being optimized was discontinuous in K. Since the line search in the optimization method assumes the underlying function is continuous, encountering such discontinuities sometimes caused the design of the controller to halt prematurely.

In this paper, we propose a new method for designing controllers of large-scale systems, a hybrid between HIFOO and HIFOOS, where controllers are designed with respect to the H_∞ performance of the closed-loop plant of a ROM but subject to a stability constraint on the closed-loop system for the associated FOM. To that end, our approach entails solving a modified version of the optimal H_∞ norm controller problem (4), one that incorporates both ROM and FOM data. In order to effectively compute solutions to this modified problem, we also propose a new algorithm that, unlike the earlier HIFOO and HIFOOS, explicitly uses the stability constraints to influence the search direction computation. Our approach is made practical by the recent introduction of a fast method for nonsmooth, nonconvex, constrained optimization.

III. A Modified Formulation

In order to ensure closed-loop stability of the FOM, we propose solving:

$$\operatorname*{arg\,min}_{A_K,B_K,C_K,D_K} \quad \|G_{\mathrm{r}}\|_{L_\infty} \quad \text{subject to} \tag{6a}$$

$$\alpha(\mathcal{A}_{\mathrm{r}}) < 0, \tag{6b}$$

$$\alpha(\mathcal{A}_{\mathrm{f}}) < 0, \tag{6c}$$

where G_{r} is the transfer function (3) for matrix quadruple $(\mathcal{A}_{\mathrm{r}}, \mathcal{B}_{\mathrm{r}}, \mathcal{C}_{\mathrm{r}}, \mathcal{D}_{\mathrm{r}})$ defined by the matrices of (2) *built using the ROM matrices of* (1), $\mathcal{A}_{\mathrm{f}}$ is the matrix $\mathcal{A}$ also given in (2) but *built using the FOM matrices of* (1) and where instead of the H_∞ norm in the objective, we use the L_∞ norm:

$$\|G\|_{L_\infty} = \sup_{\omega \geq 0} \|G(\jmath\omega)\|_2. \tag{7}$$

Note that

$$\|G_{\mathrm{r}}\|_{H_\infty} = \{\|G_{\mathrm{r}}\|_{L_\infty} \text{ if } \alpha(\mathcal{A}_{\mathrm{r}}) < 0; \ \infty \text{ otherwise}\}.$$

There are potential benefits in solving (6) in lieu of the version given by (4). First, by construction, a feasible solution of (6) must be a controller which minimizes the ROM H_∞ performance while simultaneously stabilizing the FOM closed-loop plant, which corrects the aforementioned deficiency of designing controllers of FOMs using only ROM data. Second, even though (6) without the new FOM stability constraint (6c) would be mathematically equivalent to solving (4) for the ROM model, explicitly providing both ROM and FOM stability constraints to a solver may allow it to compute search directions that better minimize the L_∞ norm while *simultaneously* helping to retain stability.

In contrast, in order to solve (4) for a sufficiently small model G_{r}, HIFOO attempts to compute a controller by successively solving two nonsmooth *but unconstrained* optimization problems, using HANSO [21] as the underlying solver (which does not support constraints). This two-phase algorithm first reduces $\alpha(\mathcal{A}_{\mathrm{r}})$ by varying the controller K until $\alpha(\mathcal{A}_{\mathrm{r}}) < 0$ and then, having obtained a stabilizing controller for which $\|G_{\mathrm{r}}\|_{H_\infty}$ is at least finite, proceeds to minimize $\|G_{\mathrm{r}}\|_{H_\infty}$, using that stabilizing controller as a starting point. A potential downside of this *implicitly-constrained* approach, and in contrast to our new *explicitly-constrained* formulation, is that during the second phase, the search direction is computed *without regard for the need to retain stability*. Instead, when a trial value of the controller K results in $\alpha(\mathcal{A}_{\mathrm{r}})$ being nonnegative, so that the value of $\|G_{\mathrm{r}}\|_{H_\infty}$ would be ∞, the line search in HANSO must reject this point and instead will just decrease the step length. While this strategy guarantees that the line search always returns a new iterate with a finite H_∞ norm value, it can lead to slow progress as it may result in small step sizes.

Attempting to solve (6) with the explicit stability constraints presents a new and perhaps surprising challenge, one which will motivate our new controller-design algorithm. Assuming that the one or more eigenvalues attaining $\alpha(\mathcal{A}_{\mathrm{r}})$ are always controllable and observable, (6) has the special property that its objective function is infinite on the boundary of the feasible and infeasible sets and nowhere else. Thus, it is possible for a solver to accept a step which decreases $\|G_{\mathrm{r}}\|_{L_\infty}$ but nonetheless corresponds to a destabilizing controller. While giving a solver such increased freedom, allowing it to also explore the infeasible region and accept steps there, could improve the overall performance, this can be problematic in the case of (6). Once a solver has accepted an infeasible point, its next step will often be to reduce the violation, in order to return to the feasible region. However, as the solver searches for a point which reduces the violation, i.e. a step that moves a rightmost eigenvalue of $\mathcal{A}_{\mathrm{r}}$ in the right half-plane closer to the imaginary axis, $\|G_{\mathrm{r}}\|_{L_\infty}$ increases rapidly. This can make it exceptionally difficult for a solver to return to the feasible region since such steps will typically not satisfy the line search criteria, leading to step lengths being shortened. Essentially, (6) itself acts like a log-barrier method but with the *antagonistic goal of maintaining infeasibility* once it has been encountered!

IV. A New Algorithm

Before we present our new algorithm, we first directly extend the unconstrained approach of HIFOO, which should also be considered for comparative purposes, as solving (6) is mathematically equivalent to minimizing

$$F(K) = \begin{cases} \|G_{\mathrm{r}}\|_{L_\infty} & \text{if } \max\{\alpha(\mathcal{A}_{\mathrm{r}}), \alpha(\mathcal{A}_{\mathrm{f}})\} < 0 \\ \infty & \text{otherwise.} \end{cases} \tag{8}$$

Algorithm 1. Use a solver in unconstrained mode to first:

Stabilize: by minimizing $\max\{\alpha(\mathcal{A}_{\mathrm{r}}), \alpha(\mathcal{A}_{\mathrm{f}})\}$ until a feasible point for the constraints in (6b) and (6c) is found, and then

Optimize: by switching to a second unconstrained phase that directly minimizes the function $F(K)$ given in (8).

Termination takes place if a maximum iteration limit is exceeded in either phase, or if the solver, using its own termination criteria, determines that an approximate stationarity condition is satisfied in the optimization phase.

Like HIFOO, Alg. 1 may suffer from poor performance due to the search directions in its "optimize" phase being computed without explicit knowledge of the stability constraints. While this search direction deficiency could be addressed by explicitly providing (6) to an appropriate constrained optimization code, this does not lead to a reliable method (as verified in our own experiments not reported here), precisely because of the aforementioned difficultly of (6) inherently acting like a log-barrier against returning to feasibility. Thus, we propose a new alternative.

Algorithm 2. Repeat the following *in a loop*:

(A) use a solver in *unconstrained mode* to minimize $\max\{\alpha(\mathcal{A}_{\mathrm{r}}), \alpha(\mathcal{A}_{\mathrm{f}})\}$, quitting when a feasible point for the constraints (6b) and (6c) is found, and continuing to:

(B) use a solver in *constrained mode* to solve (6), quitting when an iterate is generated for which either (6b) or (6c) is violated, and returning to step (A).

Termination takes place if a cumulative iteration limit is exceeded in either (A) or (B), or, if the solver, using its own termination criteria, determines that an approximate stationarity condition is satisfied in (B). Since the restabilizations done in (A) may increase $\|G_{\mathrm{r}}\|_{L_\infty}$, Alg. 2 simply keeps track of the best controller encountered for returning to the user when the computation is finished.

Remark 1: Note that Alg. 2 is not just Alg. 1 done in a loop since, unlike the "optimize" phase of Alg. 1, the (B) phase of Alg. 2 uses the explicit stability constraint information to influence the search directions.

V. A Fast and Reliable Solver for Alg. 2

We now address the issue of the practicality of our new Alg. 2 being predicated on the existence of an appropriate solver for nonsmooth, nonconvex, constrained optimization. Fortunately, and in contrast to when HIFOO was first released in 2006, two such solvers are now publicly available. Both take advantage of the fact that most nonsmooth functions of interest are differentiable almost everywhere and can thus make use of gradients on each iteration; all locally Lipschitz functions (e.g. the L_∞ norm) and semi-algebraic functions (e.g. the spectral abscissa) satisfy this condition. Although the objective and constraints are often *not* differentiable at stationary points, these points are not normally encountered by optimization methods except in the limit.

The first of these new solvers, SQP-GS, based on the work of [22] and released in 2012, is to our knowledge the only solver for such general nonsmooth, nonconvex, constrained optimization with provable convergence guarantees. However, it relies upon gradient sampling, which must evaluate $\mathcal{O}(n_{\mathrm{var}})$ nearby gradients on *every iteration*, and the proof assumes that the nonsmooth functions are locally Lipschitz, which is not the case for the spectral abscissa.

Motivated by the high cost of SQP-GS, [23] proposed a much faster alternative in 2017. This BFGS-SQP method, implemented in the open-source GRANSO: GRadient-based Algorithm for Non-Smooth Optimization [24], combines quasi-Newton (specifically BFGS) updating with sequential quadratic programming (SQP) with *steering*, which aims to either retain feasibility at every step or, when necessary, promote progress back towards the feasible region. For nonsmooth, unconstrained optimization, BFGS has proven to be very effective within HIFOO, and as discussed at length in [25], BFGS seems to rarely if ever converge to non-stationary points. The addition of SQP with steering allows GRANSO to handle both smooth and nonsmooth constraints.

Extensive experimental results comparing these two solvers on a suite of challenging static output feedback control design problems involving multiple plants were reported in [23], with GRANSO exhibiting very good results compared to SQP-GS, while also being about 10-30 times faster. Meanwhile, the two other solvers in the comparison, included as "off-the-shelf" baselines since they were not specifically designed for nonsmooth optimization, were not competitive. In almost all cases, the objective and constraint functions were nonsmooth, and in many cases even non-locally-Lipschitz, at the computed solutions. In addition to the much faster running times, the controllers found by GRANSO were often among the best obtained from all the solvers, both in terms of reduction in the optimization objective and constraint satisfaction.

As indicated by its name, GRANSO needs access to the gradients of the L_∞ norm objective and the stability constraints. The L_∞ norm is differentiable with gradient coinciding with the gradient of the largest singular value γ of the transfer function at the frequency $\tilde{\omega}$ maximizing $\|G(\mathrm{j}\omega)\|_2$, provided γ is simple and that the maximizer attaining γ is unique (up to symmetry). The formulas for its gradient are well known and involve the corresponding right and left singular vectors for γ. Likewise the spectral abscissa α is differentiable at a matrix M provided its eigenvalue λ with largest real part is unique or part of a unique complex conjugate pair of eigenvalues and that λ is simple. The formula for the gradient of the spectral abscissa is also well known and involves the corresponding right and left eigenvectors for the eigenvalue λ. In HIFOO, gradients are computed via forming the matrix derivatives of (2) but for large-scale systems, this is too expensive in terms of storage and computation. To overcome such inefficient scaling properties, HIFOOS instead obtained the gradients of $\alpha(\mathcal{A}_\mathrm{f})$ with respect to the controller variable K via differentiating inner products defined for the matrices of (2) [19, Section 3]; we adopt the same approach.

VI. Evaluation

To assess Alg. 1 and Alg. 2, both implemented using GRANSO, we applied them to large-scale 2D heat flow problems from $COMPl_eib$ v1.1 [26]. We chose the same HF2Dx FOM plus medium-scale ROM pairs used to evaluatet HIFOOS in [19]. See Table I for the list of problems and their dimensions; for all, we computed order 10 controllers. Experiments were done in MATLAB R2017a, using an Intel Core i7-6700 and 16 GB of RAM.

TABLE I: Test Set Summary

Problem	n_x (FOM)	n_x (ROM)	n_w	n_u	n_z	n_y
HF2D1	3796	316	3798	2	3796	3
HF2D2	3796	316	3798	2	3796	3
HF2D5	4489	289	4491	2	4489	4
HF2D6	2025	289	2027	2	2025	4
HF2D9	3481	484	3483	2	3481	2
HF2D_CD1	3600	256	3602	2	3600	2
HF2D_CD2	3600	256	3602	2	3600	2
HF2D_CD3	4096	324	4098	2	4096	2
HF2D_IS1	4489	361	4491	2	4489	4
HF2D_IS2	4489	361	4491	2	4489	4
HF2D_IS3	3600	256	3602	2	3600	2
HF2D_IS4	3600	256	3602	2	3600	2

We used getPeakGain to compute $\|G_\mathrm{r}\|_{L_\infty}$, setting its tolerance to $\delta_\mathrm{high} = 10^{-14}$ since using its default of 10^{-2} or even $\delta_\mathrm{low} = 10^{-7}$, the tolerance used in HIFOO, can sometimes lead to numerical problems during optimization, which assumes gradients are computed accurately. The issue is that the error in the computed gradient of $\|G_\mathrm{r}\|_{L_\infty}$, even when it is well defined, can be significantly higher than the error in $\|G_\mathrm{r}\|_{L_\infty}$ suggested by the tolerance value.

The spectral abscissas of $\mathcal{A}_\mathrm{r}$ and $\mathcal{A}_\mathrm{f}$ were respectively computed via eig and eigs. Assuming A_1 from the FOM is sparse, as is generally the case, $\mathcal{A}_\mathrm{f}$ can be cheaply applied as a matrix-vector operator, which is all that is needed to use eigs. Thus, it is relatively efficient to assess the stability of fairly large systems for which computing the L_∞ norm is out of reach. While eigs is not absolutely guaranteed to return a globally rightmost eigenvalue of $\mathcal{A}_\mathrm{f}$, it is generally quite reliable in practice. Indeed, it was fully sufficient, with appropriate parameter choices, for use in HIFOOS; see [19] for details. To compute the gradient of $\alpha(\mathcal{A}_\mathrm{f})$, it is necessary to call eigs twice, once for $\mathcal{A}_\mathrm{f}$ and once for its transpose, to obtain both the right and left eigenvectors corresponding to the eigenvalue with largest real part.

In Tables II and III, the two columns with subheader "R+F" under "Alg. 1" and "Alg. 2" respectively refer to Algorithms 1 and 2 as specified above. For comparative purposes, we also ran versions of these algorithms that omitted the stability constraint (6c) on the FOM; the corresponding columns with subheader "R only" refer to these variants.

Note that the "R only" version of Alg. 1 is effectively the same as that used in existing versions of HIFOO but, to account for implementation differences, we also included HIFOO v3.5 (using HANSO v2.2) in our comparison. For consistency with GRANSO, we disabled HANSO's expensive gradient sampling phase and set parameters opts.normtol and opts.evaldist both to 10^{-6}. We ran HIFOO twice, once with its default $\delta_\mathrm{low} = 10^{-7}$ tolerance for computing the H_∞ norm and then again with the tighter $\delta_\mathrm{high} = 10^{-14}$ used in our new code.

For consistency, we randomly generated a starting controller for each problem so that all methods/variants would be initialized identically and, for HIFOO, disabled the additional

TABLE II: Final Values of $F(K)$.

Final values of $F(K)$ in eq. (8)						
	HIFOO v3.5		Alg. 1		Alg. 2	
Problem	δ_{low}	δ_{high}	R only	R+F	R only	R+F
HF2D1	**6511.1**	6512.8	6519.4	6519.4	22928.5	9718.7
HF2D2	5609.6	5598.8	**5597.0**	6292.7	5600.6	9635.9
HF2D5	17716.1	17403.3	17317.2	39890.5	17292.6	**16847.2**
HF2D6	7400.4	7406.0	7397.7	7383.0	**7370.0**	7392.6
HF2D9	60.3	60.3	**60.3**	60.3	60.3	60.3
HF2D_CD1	∞	∞	∞	47.4	∞	**4.6**
HF2D_CD2	∞	∞	∞	48.9	∞	**48.9**
HF2D_CD3	∞	∞	∞	37.5	∞	**4.9**
HF2D_IS1	46428.8	44777.5	44247.1	1734082.9	42779.3	**42771.5**
HF2D_IS2	10342.4	10336.5	10309.6	10327.5	**10206.7**	10462.9
HF2D_IS3	∞	∞	∞	259.3	∞	**8.5**
HF2D_IS4	∞	∞	∞	23.9	∞	**6.9**
Relative differences from the best $F(K)$ values (in bold above)						
HF2D1	—	0.000	0.001	0.001	2.521	0.493
HF2D2	0.002	0.000	—	0.124	0.001	0.722
HF2D5	0.052	0.033	0.028	1.368	0.026	—
HF2D6	0.004	0.005	0.004	0.002	—	0.003
HF2D9	0.000	0.000	—	0.000	0.000	0.000
HF2D_CD1	∞	∞	∞	9.297	∞	—
HF2D_CD2	∞	∞	∞	0.000	∞	—
HF2D_CD3	∞	∞	∞	6.694	∞	—
HF2D_IS1	0.086	0.047	0.034	39.543	0.000	—
HF2D_IS2	0.013	0.013	0.010	0.012	—	0.025
HF2D_IS3	∞	∞	∞	29.524	∞	—
HF2D_IS4	∞	∞	∞	2.453	∞	—

TABLE III: Wall-Clock Running Times.

Wall-clock running times (seconds)						
	HIFOO v3.5		Alg. 1		Alg. 2	
Problem	δ_{low}	δ_{high}	R only	R+F	R only	R+F
HF2D1	5866	7065	7167	10556	10264	11409
HF2D2	2373	5931	5546	463	6205	16034
HF2D5	3675	6579	6468	737	3045	10896
HF2D6	4781	5755	5547	6175	6458	6075
HF2D9	451	431	738	523	989	763
HF2D_CD1	2708	3368	3387	73	1130	7872
HF2D_CD2	2741	3184	3519	92	4977	137
HF2D_CD3	6040	6684	7337	295	7990	11752
HF2D_IS1	7915	10838	11988	164	15056	14458
HF2D_IS2	8068	10286	10203	14450	17930	22717
HF2D_IS3	1071	2151	1295	114	108	2023
HF2D_IS4	1178	1653	1485	185	1359	5101
Running times relative to HIFOO v3.5 (δ_{low})						
HF2D1	1	1.20	1.22	1.80	1.75	1.94
HF2D2	1	2.50	2.34	0.19	2.61	6.76
HF2D5	1	1.79	1.76	0.20	0.83	2.96
HF2D6	1	1.20	1.16	1.29	1.35	1.27
HF2D9	1	0.96	1.64	1.16	2.19	1.69
HF2D_CD1	1	1.24	1.25	0.03	0.42	2.91
HF2D_CD2	1	1.16	1.28	0.03	1.82	0.05
HF2D_CD3	1	1.11	1.21	0.05	1.32	1.95
HF2D_IS1	1	1.37	1.51	0.02	1.90	1.83
HF2D_IS2	1	1.27	1.26	1.79	2.22	2.82
HF2D_IS3	1	2.01	1.21	0.11	0.10	1.89
HF2D_IS4	1	1.40	1.26	0.16	1.15	4.33

three random initializations it attempts by default.

Table II reports the best (lowest) values of $F(K)$ (see (8)) obtained by each method; for each problem, the best value across methods is shown in bold. Recall that $F(K)$ is ∞ if the controller K fails to stabilize *both* the ROM *and* the FOM, regardless of whether or not K was obtained using any FOM information. For every ROM-only method (HIFOO and the "R only" variants of Alg. 1 and Alg. 2), we see that ∞ is reported for 5 out of the 12 problems in the corresponding columns of Table II. In each case, the respective method's computed controller failed to stabilize the FOM closed-loop systems, indeed confirming that designing controllers for FOMs, using only ROM information, can often result in complete failure. In contrast, both methods that explicitly impose the FOM stability constraint (the "R+F" methods) *always* succeeded in simultaneously stabilizing the ROMs and the FOMs, and hence in Table II, these two ROM-FOM hybrid methods have finite values of $F(K)$ reported for all 12 test problems. Furthermore, Alg. 2 ("R+F") succeeded in finding the best value of $F(K)$ on 7 of the 12 problems, while on another three, the values it found were only slightly higher than those obtained by the best methods. This observation is made easier by viewing the bottom half of the table, which shows the relative differences of the $F(K)$ values from the best value for each problem, with dashes indicating that the relevant method was in fact the best. A value of 0.000 means that the relative difference was below our reporting limit of 0.001.

The top half of Table III reports the total wall-clock running time (in seconds) for each problem-method pair. The bottom half reports the ratio of the running times relative to the running times of HIFOO v3.5 (δ_{low}), with values higher than one indicating how many times slower a method was compared to HIFOO while values less than one indicate the opposite. Even though the FOM orders were typically about 10 times the corresponding ROM orders, the running times for Alg. 2 ("R+F") ranged from as little as 0.05 to at most 6.76 times the running times of HIFOO, (using only ROM data and the less demanding δ_{low}). When HIFOO was also run with δ_{high}, Alg. 2 ("R+F") was at most 3.09 slower.

Figure 1 shows representative examples of the evolution of the $\|G_{\text{r}}\|_{L_\infty}$ values computed by Alg. 2 ("R+F") as a function of the iteration count, for problems HF2D_CD1 and HF2D_IS3. Only the (B) iterations in Alg. 2 are shown as the stabilization iterations in (A) are typically few in number and relatively less costly. The quantity $\|G_{\text{r}}\|_{L_\infty}$ is steadily reduced in (B) until an infeasible point is reached, at which point the stabilization phase in (A) typically increases $\|G_{\text{r}}\|_{L_\infty}$, sometimes significantly. The usual trend, however, is for $\|G_{\text{r}}\|_{L_\infty}$ to be consistently reduced over a sequence of (A) and (B) iterations, until either:

- a cumulative total of 1000 iterations in the (B) phases is reached, as with problem HF2D_CD1, or, occasionally,
- GRANSO determines that an approximate stationarity condition has been satisfied (see [23] for details), as with problem HF2D_IS3.

As expected, the results for Alg. 1 ("R only") are quite similar to those of HIFOO, as they differ only in implemen-

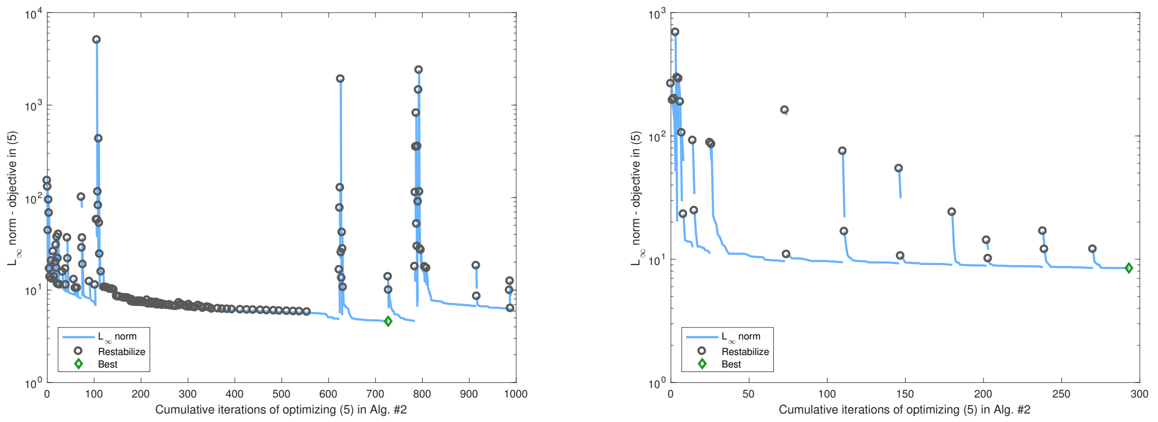

Fig. 1: Problems `HF2D_CD1` (left) and `HF2D_IS3` (right).

tation details. However, while "R only" controllers obtained by Alg. 1 and Alg. 2 were also fairly similar, the superiority of Alg. 2 is clear in the "R+F" setting. Indeed, the controllers found by Alg. 2 ("R+F") yielded values of $F(K)$ that were on average 8.35 times smaller than those of Alg. 1 ("R+F").

VII. Conclusion

We have presented an effective new algorithm, Alg. 2 ("R+F"), for designing low-order controllers of large-scale systems, which leverages recent advances in nonsmooth, constrained optimization to simultaneously use data from full- and reduced-order models. Unlike earlier methods, our scalable approach avoids issues stemming from approximating instead of computing the H_∞ norm (as was done in HIFOOS) while still ensuring stability of the full-order models (unlike existing versions of HIFOO applied to reduced-order models only).

References

[1] K. Zhou, J. C. Doyle, and K. Glover, *Robust and Optimal Control*. Upper Saddle River, NJ: Prentice-Hall, 1996.

[2] D. Youla, H. Jabr, and J. Bongiorno, "Modern Wiener-Hopf design of optimal controllers–Part II: The multivariable case," *IEEE Trans. Autom. Control*, vol. 21, no. 3, pp. 319–338, 1976.

[3] V. Kučera, "Stability of discrete linear feedback systems," *IFAC Proceedings Volumes*, vol. 8, no. 1, Part 1, pp. 573–578, 1975, 6th IFAC World Congress (IFAC 1975) - Part 1: Theory, Boston/Cambridge, MA, USA, August 24-30, 1975.

[4] J. V. Burke and M. L. Overton, "Variational analysis of non-Lipschitz spectral functions," *Math. Program.*, vol. 90, no. 2, Ser. A, pp. 317–352, 2001.

[5] M. L. Overton, "HIFOO (H-Infinity Fixed-Order Optimization)," https://www.cs.nyu.edu/~overton/software/hifoo/, 2006, see also [6].

[6] 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*, vol. 39, no. 9, pp. 339–344, 2006, 5th IFAC Symposium on Robust Control Design ROCOND 2006.

[7] ——, "Stabilization via nonsmooth, nonconvex optimization," *IEEE Trans. Autom. Control*, vol. 51, no. 11, pp. 1760–1769, 2006.

[8] "HINFSTRUCT," https://www.mathworks.com/help/robust/ref/hinfstruct.html.

[9] P. Apkarian and D. Noll, "Controller design via nonsmooth multidirectional search," *SIAM J. Control Optim.*, vol. 44, no. 6, pp. 1923–1949, 2006.

[10] ——, "Nonsmooth H_∞ synthesis," *IEEE Trans. Autom. Control*, vol. 51, no. 1, pp. 71–86, 2006.

[11] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear matrix inequalities in system and control theory*, ser. Stud. Appl. Math. Philadelphia, PA: SIAM Publications, 1994, vol. 15.

[12] S. Boyd and V. Balakrishnan, "A regularity result for the singular values of a transfer matrix and a quadratically convergent algorithm for computing its L_∞-norm," *Syst. Cont. Lett.*, vol. 15, pp. 1–7, 1990.

[13] N. A. Bruinsma and M. Steinbuch, "A fast algorithm to compute the H_∞-norm of a transfer function matrix," *Syst. Cont. Lett.*, vol. 14, no. 4, pp. 287–293, 1990.

[14] A. C. Antoulas, *Approximation of Large-Scale Dynamical Systems*, ser. Adv. Des. Control. Philadelphia, PA: SIAM Publications, 2005, vol. 6.

[15] U. Baur, P. Benner, and L. Feng, "Model order reduction for linear and nonlinear systems: A system-theoretic perspective," *Arch. Comput. Methods Eng.*, vol. 21, no. 4, pp. 331–358, 2014.

[16] P. Benner, V. Mehrmann, and D. C. Sorensen, *Dimension Reduction of Large-Scale Systems*, ser. Lect. Notes Comput. Sci. Eng. Springer-Verlag, Berlin/Heidelberg, Germany, 2005, vol. 45.

[17] P. Benner, A. Cohen, M. Ohlberger, and K. Willcox, Eds., *Model Reduction and Approximation: Theory and Algorithms*, ser. Computational Science & Engineering. Philadelphia, PA: SIAM Publications, 2017.

[18] G. Obinata and B. D. O. Anderson, *Model Reduction for Control System Design*, ser. Comm. Control Eng. London, UK: Springer-Verlag, 2001.

[19] T. Mitchell and M. L. Overton, "Fixed low-order controller design and H_∞ optimization for large-scale dynamical systems," *IFAC-PapersOnLine*, vol. 48, no. 14, pp. 25–30, 2015, 8th IFAC Symposium on Robust Control Design ROCOND 2015.

[20] ——, "Hybrid expansion-contraction: a robust scaleable method for approximating the H_∞ norm," *IMA J. Numer. Anal.*, vol. 36, no. 3, pp. 985–1014, 2016.

[21] M. L. Overton, "HANSO (Hybrid Algorithm for Non-Smooth Optimization)," https://cs.nyu.edu/~overton/software/hanso/.

[22] F. E. Curtis and M. L. Overton, "A sequential quadratic programming algorithm for nonconvex, nonsmooth constrained optimization," *SIAM J. Optim.*, vol. 22, no. 2, pp. 474–500, 2012.

[23] 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.*, vol. 32, no. 1, pp. 148–181, 2017.

[24] T. Mitchell, "GRANSO: GRadient-based Algorithm for Non-Smooth Optimization," http://timmitchell.com/software/GRANSO, see also [23].

[25] A. S. Lewis and M. L. Overton, "Nonsmooth optimization via quasi-Newton methods," *Math. Program.*, vol. 141, no. 1–2, Ser. A, pp. 135–163, 2013.

[26] F. Leibfritz, "$COMPl_eib$: *CO*nstrained *M*atrix-optimization *P*roblem *l*ibrary," 2004. [Online]. Available: http://www.compleib.de