

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

Peter Benner\*      Tim Mitchell\*      Michael L. Overton†

Feb. 28, 2018

## Abstract

We consider low-order controller design for large-scale linear time-invariant dynamical systems with inputs and outputs. Model order reduction is a popular technique, but controllers designed for reduced-order models may result in unstable closed-loop plants when applied to the full-order system. We introduce a new method to design a fixed-order controller by minimizing the  $L_\infty$  norm of a reduced-order closed-loop transfer matrix function subject to stability constraints on the closed-loop systems for both the reduced-order and the full-order models. Since the optimization objective and the constraints are all nonsmooth and nonconvex we use a sequential quadratic programming method based on quasi-Newton updating that is intended for this problem class, available in the open-source software package GRANSO. Using a publicly available test set, the controllers obtained by the new method are compared with those computed by the HIFOO (H-Infinity Fixed-Order Optimization) toolbox applied to a reduced-order model alone, which frequently fail to stabilize the closed-loop system for the associated full-order model.

## 1 Introduction

As the size of dynamical systems continues to grow rapidly, reduced-order modeling [Ant05, BBF14, BMS05, BCOW17] has become essential. However, straightforward control design using reduced-order models may result in unstable closed-loop plants for the full-order model. Though there exist methods guaranteeing closed-loop stability for the latter, in particular frequency-weighted balancing techniques that perform a combined plant-and-controller reduction [OA01], these techniques are often challenging from a computational perspective. In this paper we consider designing *low-order* controllers via  $H_\infty$  optimization applied to the closed-loop system for a *reduced-order* model (ROM), subject to a stability constraint on the closed-loop system for the associated *full-order* model (FOM).

Consider the open-loop linear time invariant dynamical system [ZDG96]

$$\begin{bmatrix} \dot{x} \\ z \\ y \end{bmatrix} = \begin{bmatrix} A_1 & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix} \begin{bmatrix} x \\ w \\ u \end{bmatrix} \quad (1)$$

---

This work was supported in part by the Research Training Group RTG 2297/1, “Mathematical Complexity Reduction - MathCoRe” in Magdeburg, funded by Deutsche Forschungsgemeinschaft, and in part by the U.S. National Science Foundation under grant DMS-1620083

\*Peter Benner and Tim Mitchell are with the Max Planck Institute for Dynamics of Complex Technical Systems, 39106 Magdeburg, Germany [benner@mpi-magdeburg.mpg.de](mailto:benner@mpi-magdeburg.mpg.de), [mitchell@mpi-magdeburg.mpg.de](mailto:mitchell@mpi-magdeburg.mpg.de)

†Michael L. Overton is with the Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA [overton@cims.nyu.edu](mailto:overton@cims.nyu.edu)

where  $x \in \mathbb{R}^{n_x}$  contains the states,  $u \in \mathbb{R}^{n_u}$  is the physical (control) input,  $w \in \mathbb{R}^{n_w}$  is the performance input,  $y \in \mathbb{R}^{n_y}$  is the physical (measured) output,  $z \in \mathbb{R}^{n_z}$  is the performance output, and the matrices are real 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:

$$\begin{bmatrix} \dot{x} \\ \dot{x}_K \\ z \end{bmatrix} = \begin{bmatrix} \mathcal{A} & \mathcal{B} \\ \mathcal{C} & \mathcal{D} \end{bmatrix} \begin{bmatrix} x \\ x_K \\ w \end{bmatrix}$$

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

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

When  $D_{22}$  is nonzero, the formulas for  $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$  are not affine, as they involve  $(I - D_K D_{22})^{-1}$  [ZDG96, p. 446]; the condition  $\|D_K D_{22}\|_2 < 1$  ensures that this last matrix is well defined. However, since  $D_{22}$  is zero in all problems in our test set described below, this constraint will not play a role in this paper.

The closed-loop transfer matrix function

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

maps the performance input  $w$  to the performance output  $z$ .

## 2 Low-Order Controller Design

Let  $\alpha : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}$  denote the spectral abscissa, defined for a matrix  $M$  by

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

A key requirement of a controller  $K$  is that the closed-loop system is stable, i.e.,  $\alpha(\mathcal{A}) < 0$ . The  $L_\infty$  and  $H_\infty$  norms of the transfer matrix function are defined by

$$\|G\|_{L_\infty} = \sup_{\omega \geq 0} \|G(\mathbf{i}\omega)\|_2 \tag{4}$$

and

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

Note that the  $L_\infty$  norm is finite as long as  $\mathcal{A}$  has no eigenvalues on the imaginary axis while the  $H_\infty$  norm is finite provided that the eigenvalues of  $\mathcal{A}$  are all in the open left half-plane. The standard method for computing  $\|G\|_{L_\infty}$  is the Boyd-Balakrishnan-Bruinsma-Steinbuch (BBBS) algorithm [BB90, BS90], implemented in the MATLAB function `getPeakGain`. The BBBS method converges quadratically to a global maximizer of (4) but it involves computing the eigenvalues of a sequence of Hamiltonian matrices of order  $2(n_x + n_K)$ ; the algorithm thus requires  $O((n_x + n_K)^3)$  operations per iteration. Computing the eigenvalues of  $\mathcal{A}$  then determines  $\alpha(\mathcal{A})$  and hence whether  $\|G\|_{H_\infty}$  equals  $\|G\|_{L_\infty}$  or  $\infty$ ; checking stability via the MATLAB function

`eig` is generally at least an order of magnitude faster than using the BBBS method to compute  $\|G\|_{L_\infty}$ .

If  $n_K = n_x$ , methods to compute a controller that minimizes  $\|G\|_{H_\infty}$  are well known. However, it is often desirable to design a *low-order* controller with  $n_K \ll n_x$ ; note that low-order here refers to the size of the controller  $K$ , not to whether  $n_x$ , the dimension of the state space, has been reduced. The open-source toolbox HIFOO [Ove06, BHLO06a], dating from 2006 and based in part on [BHLO06b], addresses this problem by employing nonsmooth unconstrained optimization techniques to minimize  $\|G\|_{H_\infty}$  over the controller variable  $K$ . Similarly, the MATLAB code `hinfstruct` [hin] introduced in release R2010b and based in part on [AN06a, AN06b], also optimizes  $\|G\|_{H_\infty}$  over the controller variable  $K$ . As the `hinfstruct` code is proprietary, we focus here on HIFOO, whose code can be readily modified. HIFOO employs a two-phase algorithm, first reducing  $\alpha(\mathcal{A})$  by varying the controller  $K$  until  $\alpha(\mathcal{A}) < 0$ . Then, having obtained a stabilizing controller for which  $\|G\|_{H_\infty}$  is finite, HIFOO proceeds to the second phase, locally minimizing  $\|G\|_{H_\infty}$ , using that stabilizing controller as a starting point. HIFOO relies on the nonsmooth unconstrained optimization code HANSO [Ove] in both phases. During the second phase, if a trial value of the controller  $K$  results in  $\alpha(\mathcal{A})$  being nonnegative, the value of  $\|G\|_{H_\infty}$  is  $\infty$  by definition, so the line search in HANSO rejects this point and decreases the step length. This strategy guarantees that all iterates accepted by the line search result in finite  $H_\infty$  norm values. From its inception, the motivation for HIFOO was ease of use, so that an engineer could easily call it to design low-order controllers without any need to understand details of the optimization method on which it relies. References to many applications where HIFOO has been used appear in [MO15].

However, neither HIFOO nor `hinfstruct` is practical for large-scale systems because of the high cost of computing  $\|G\|_{L_\infty}$  and hence  $\|G\|_{H_\infty}$ . For this reason, Mitchell and Overton [MO15] introduced a new experimental code HIFOOS that optimizes an *approximation* to  $\|G\|_{H_\infty}$  for a large sparse system, using a recently proposed, scalable algorithm called hybrid-expansion-contraction (HEC) [MO16]. The HEC algorithm is guaranteed to find a lower bound for  $\|G\|_{H_\infty}$  and, under reasonable assumptions, converges to a stationary point (usually at least a local maximizer) of (4). In [MO15, Table 2], this new approach consistently led to stable closed-loop systems for the large-scale system, contrasting with the ROM-only-based HIFOO approach, which resulted in 5 of 12 controllers that failed to stabilize the original full-order models. However, approximating  $\|G\|_{H_\infty}$  for the large-scale system sometimes led to inconsistencies from one controller  $K$  to another, effectively implying that the function being optimized was discontinuous in  $K$ . The result was that optimization would not infrequently halt prematurely due to failure in the line search, which assumes the function is continuous.

### 3 The New Formulation

To eliminate this difficulty, we take a different course in this paper, designing a low-order controller by optimizing  $\|G\|_{H_\infty}$  for a ROM while ensuring the stability of the closed-loop system for the FOM. More specifically, we consider the optimization problem

$$\min_{A_K, B_K, C_K, D_K} \|G_r\|_{L_\infty} \quad \text{subject to} \quad (5a)$$

$$\alpha(\mathcal{A}_r) < 0, \quad (5b)$$

$$\alpha(\mathcal{A}_f) < 0, \quad (5c)$$

where  $G_r$  is the transfer function (3) for matrix quadruple  $(\mathcal{A}_r, \mathcal{B}_r, \mathcal{C}_r, \mathcal{D}_r)$  defined by the matrices of (2) *built using the ROM matrices of* (1), and  $\mathcal{A}_f$  is the matrix  $\mathcal{A}$  also given in (2) but *built using the FOM matrices of* (1).

The key idea of the new formulation is that while the optimization objective in (5a) is the ROM transfer function norm  $\|G_r\|_{L_\infty}$ , stability of the FOM closed-loop system, specified by

inequality constraint (5c), is a requirement for the design of the controller. As we discuss below, assessing the stability of the FOM closed-loop system via computing the spectral abscissa of  $\mathcal{A}_f$  can be done much more efficiently than the computation of the FOM transfer function norm  $\|G_f\|_{L_\infty}$ . Furthermore, by also specifying stability of the ROM closed-loop system as an explicit inequality constraint, given by (5b), the optimization objective may now be chosen as the  $L_\infty$  norm of  $G_r$  instead of the  $H_\infty$  norm. Note that solving (5) is mathematically equivalent to minimizing

$$F(K) = \begin{cases} \|G_r\|_{L_\infty} & \text{if } \max\{\alpha(\mathcal{A}_r), \alpha(\mathcal{A}_f)\} < 0 \\ \infty & \text{otherwise.} \end{cases} \quad (6)$$

Although we use the formulation (5) in the algorithmic development below, it is the final computed values of  $F(K)$  that we will report in our evaluation of the algorithms.

The constrained optimization problem given by (5) has objective and constraint functions that are all continuous, though none are convex or smooth. The  $L_\infty$  norm function in (5a) is locally Lipschitz, but the spectral abscissa function in (5b)–(5c) is not locally Lipschitz at a matrix  $M$  with an eigenvalue  $\lambda$  with  $\text{Re } \lambda = \alpha(M)$  that has multiplicity two or more. Since standard constrained optimization software packages are not intended for cases where the optimization objective or any of its constraints is nonsmooth, we use a recently introduced sequential quadratic programming method based on quasi-Newton (BFGS) updating [CMO17] that is intended for this problem class. This method is implemented in the open-source software package GRANSO (GRAdient-based Algorithm for Non-Smooth Optimization) [Mit]. Extensive experimental results on a suite of challenging static output feedback control design problems involving multiple plants were reported in [CMO17], exhibiting very good results compared with three other methods. In almost all cases, the objective and constraint functions were nonsmooth, and in many cases even non-locally-Lipschitz, at the approximate solutions computed by GRANSO. Compared to approximations found by the other methods, those obtained by GRANSO were often better both in terms of reduction in the optimization objective and constraint satisfaction, and also in terms of running time.

As its name suggests, in order for GRANSO to be applied to an optimization problem it needs access to the gradients of the relevant objective and constraint functions. The philosophy underlying the use of BFGS for nonsmooth optimization [CMO17, LO13] is that, since locally Lipschitz functions such as the  $L_\infty$  norm and semi-algebraic functions such as the spectral abscissa are differentiable almost everywhere, it makes sense to compute gradients at optimization iterates. Although the objective and constraints are often *not* differentiable at stationary points, these points are not normally encountered by the optimization method except in the limit. A remarkable observation concerning the use of BFGS on nonsmooth problems is that the method rarely if ever converges to non-stationary points, as discussed at length in [LO13]. Naturally, two gradients evaluated at two nearby points may be very different, but it is precisely this property that is exploited by the BFGS updating. The  $L_\infty$  norm is differentiable with gradient coinciding with the gradient of the largest singular value  $\gamma$  of the transfer function at the frequency  $\tilde{\omega}$  maximizing  $\|G(i\omega)\|_2$ , provided the maximizer is unique and the largest singular value  $\gamma$  is simple. The formulas for its gradient are well known and involve the corresponding right and left singular vectors for  $\gamma$ . Likewise the spectral abscissa  $\alpha$  is differentiable at a matrix  $M$  provided its eigenvalue  $\lambda$  with largest real part is unique or part of a unique complex conjugate pair of eigenvalues and that  $\lambda$  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  $\lambda$ . HIFOO's gradient calculations are done via forming the matrix derivatives of (2) but for large-scale systems, this is expensive in terms of storage and computation. To overcome such inefficient scaling properties, HIFOOS instead obtained the gradients of  $\alpha(\mathcal{A}_f)$  with respect to the controller variable  $K$  via differentiating inner products defined for the matrices of (2) [MO15, Section 3]; we adopt the same approach here.

The method used by the optimization code GRANSO allows the use of infeasible points, i.e., values of the controller  $K$  for which either or both of the stability constraints (5b) and (5c) on the

ROM and the FOM respectively are violated. However, we found that this worked poorly, because once an infeasible point was generated, moving towards the feasible region typically resulted in substantial increase in the optimization objective  $\|G_r\|_{L_\infty}$ , often preventing the algorithm, which weighs information from the objective and constraints together using a penalty function, from finding another feasible point. So, instead we used two alternative methods.

**Algorithm 1.** Use GRANSO in unconstrained mode to first:

**Stabilize:** by minimizing  $\max\{\alpha(\mathcal{A}_r), \alpha(\mathcal{A}_f)\}$  until a feasible point for the constraints in (5b) and (5c) is found, and then

**Optimize:** by switching to a second unconstrained phase that directly minimizes the function  $F(K)$  given in (6) using GRANSO.

Termination takes place if a maximum iteration limit is exceeded in either phase, or if, using its default options, GRANSO determines that an approximate stationarity condition is satisfied in the optimization phase.

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

- (A) use GRANSO in *unconstrained mode* to minimize  $\max\{\alpha(\mathcal{A}_r), \alpha(\mathcal{A}_f)\}$ , quitting when a feasible point for the constraints (5b) and (5c) is found, and continuing to:
- (B) use GRANSO in *constrained mode* to solve (5), quitting when an iterate is generated for which either (5b) or (5c) is violated, and returning to step (A).

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

**Remark 1** *The stabilize-then-optimize approach of Alg. 1 is effectively identical to that used in current versions of HIFOO except that (a) the new algorithm uses the GRANSO optimization code instead of the older code HANSO, and (b) existing versions of HIFOO do not use the stability constraint (5c) for the FOM. As in HIFOO, the line search in the second “optimize” phase of Alg. 1 ensures stability is maintained by rejecting any points where the minimization objective  $F(K)$  is infinite due to a stability constraint being violated.*

**Remark 2** *Note that Alg. 2 is not just Alg. 1 done in a loop. The “optimize” phase of Alg. 1 simply rejects controllers  $K$  that destabilize either the ROM or FOM (since  $F(K)$  is infinite for such  $K$ ) while the (B) phase of Alg. 2 takes advantage of the explicit stability constraints in attempt to produce better search directions, i.e., ones which simultaneously minimize the  $L_\infty$  norm and maintain stability.*

## 4 Evaluation

In order to assess the ability of our new methods to find controllers that minimize the  $H_\infty$  norm of the ROMs as much as possible while also stabilizing the FOMs, we applied it to a selection of large-scale 2D heat flow problems from version 1.1 of *COMPl<sub>e</sub>ib* [Lei04]. We chose the same twelve HF2Dx FOM examples used in the evaluation of HIFOOS [MO15] because *COMPl<sub>e</sub>ib* already provides corresponding medium-scale ROMs for these examples. See Table 1 for the list of problems chosen and their full and reduced orders. For all problems, we elected to compute order 10 controllers.

We used `getPeakGain` to compute  $\|G_r\|_{L_\infty}$ , setting its tolerance to  $\delta_{\text{high}} = 10^{-14}$  because we have observed that its default tolerance of  $10^{-2}$  often returns insufficiently accurate results, and that even  $\delta_{\text{low}} = 10^{-7}$ , the tolerance used by HIFOO for its  $H_\infty$  norm calculations, can

Table 1: 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

also sometimes lead to numerical problems. This issue of numerical accuracy is important since gradients play a key role in the optimization procedure, and a given tolerance sometimes results in a computed gradient of  $\|G_{\text{r}}\|_{L_\infty}$  that is significantly more inaccurate than the tolerance might suggest, even when the gradient is well defined at the point.

The spectral abscissa of  $\mathcal{A}_{\text{f}}$  is computed by a call to `eig`, while the spectral abscissa of  $\mathcal{A}_{\text{f}}$  is computed using `eigs` which accesses the matrix  $\mathcal{A}_{\text{f}}$  only via matrix-vector products. Assuming the original full-order version of  $A_1$  is sparse, as is generally the case,  $\mathcal{A}_{\text{f}}$  can be cheaply applied as a matrix-vector operator. Thus, by using `eigs`, it is relatively efficient to assess the stability of fairly large systems for which computing the  $L_\infty$  norm is out of reach. Although `eigs` is not absolutely guaranteed to return the eigenvalues of  $\mathcal{A}_{\text{f}}$  with largest real part, it is generally quite reliable in practice. Indeed, it was fully sufficient, with appropriate parameter choices, for the aforementioned experiments validating HIFOOS; see [MO15] for details. It is necessary to call `eigs` twice, once for  $\mathcal{A}_{\text{f}}$  and once for its transpose, to obtain both the right and left eigenvectors corresponding to the eigenvalue with largest real part, as these are needed to compute the gradient of the spectral abscissa, as noted above.

In Tables 2 and 3 below, the columns labeled “Alg. 1” with subheading “R+F” and those labeled “Alg. 2” with subheading “R+F” refer to Algorithms 1 and 2 specified above. For comparative purposes, we also ran versions of these algorithms that omitted the stability constraint (5c) on the FOM; the columns labeled “Alg. 1” with subheading “R only” and those labeled “Alg. 2” with subheading “R only” refer to these experiments. Alg. 1 without the FOM stability constraint is effectively the same as the algorithm used in existing versions of HIFOO but, to account for differences in implementations, we also ran version 3.5 of HIFOO (using version 2.2 of HANSO) to produce controllers using only the ROM data. 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  $\delta_{\text{low}} = 10^{-7}$  tolerance for the BBBS method and again with the tighter  $\delta_{\text{high}} = 10^{-14}$  tolerance we used for `getPeakGain` in our new code.

For each problem, we randomly generated an initial controller so that all methods/variants in our comparison would be initialized at the same point. Since by default, HIFOO also attempts optimization from three automatically randomly-generated controllers (determined by the positive integer parameter `opts.nrand`), we disabled these additional starting points via a slight modification to the HIFOO code so that it would allow `opts.nrand := 0`.

All numerical experiments were implemented and run using MATLAB R2017a on a workstation with an Intel Core i7-6700 (4 Cores @ 3.4 GHz) and 16 GB memory.

The top half of Table 3 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

Table 2: Final Values of  $F(K)$ .

Final values of $F(K)$ in eq. (6)						
Problem	HIFOO v3.5		Alg. 1		Alg. 2	
	$\delta_{\text{low}}$	$\delta_{\text{high}}$	R only	R+F	R only	R+F
HF2D1	<b>6511.1</b>	6512.8	6519.4	6519.4	22928.5	9718.7
HF2D2	5609.6	5598.8	<b>5597.0</b>	6292.7	5600.6	9635.9
HF2D5	17716.1	17403.3	17317.2	39890.5	17292.6	<b>16847.2</b>
HF2D6	7400.4	7406.0	7397.7	7383.0	<b>7370.0</b>	7392.6
HF2D9	60.3	60.3	<b>60.3</b>	60.3	60.3	60.3
HF2D_CD1	$\infty$	$\infty$	$\infty$	47.4	$\infty$	<b>4.6</b>
HF2D_CD2	$\infty$	$\infty$	$\infty$	48.9	$\infty$	<b>48.9</b>
HF2D_CD3	$\infty$	$\infty$	$\infty$	37.5	$\infty$	<b>4.9</b>
HF2D_IS1	46428.8	44777.5	44247.1	1734082.9	42779.3	<b>42771.5</b>
HF2D_IS2	10342.4	10336.5	10309.6	10327.5	<b>10206.7</b>	10462.9
HF2D_IS3	$\infty$	$\infty$	$\infty$	259.3	$\infty$	<b>8.5</b>
HF2D_IS4	$\infty$	$\infty$	$\infty$	23.9	$\infty$	<b>6.9</b>
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	$\infty$	$\infty$	$\infty$	9.297	$\infty$	—
HF2D_CD2	$\infty$	$\infty$	$\infty$	0.000	$\infty$	—
HF2D_CD3	$\infty$	$\infty$	$\infty$	6.694	$\infty$	—
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	$\infty$	$\infty$	$\infty$	29.524	$\infty$	—
HF2D_IS4	$\infty$	$\infty$	$\infty$	2.453	$\infty$	—

of HIFOO v3.5 ( $\delta_{\text{low}}$ ), with values higher than one indicating how many times slower a method was compared to HIFOO while values less than one indicate the opposite. We see that 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 (which uses only the ROM data), and less compared to the version of HIFOO using the more demanding tolerance  $\delta_{\text{high}}$  to compute  $\|G_{\text{r}}\|_{L_{\infty}}$ .

Table 2 reports the best (lowest) values of  $F(K)$  (see (6)) obtained by each method, with, for each problem, the best value obtained over all the methods shown in bold text. Recall that the value of  $F(K)$  is  $\infty$  if the controller  $K$  fails to stabilize *both* the ROM *and* the FOM, regardless of whether or not the controller  $K$  was obtained using any FOM information. For every ROM-only method in the comparison (HIFOO and the “R only” variants of Alg. 1 and Alg. 2), we see that  $\infty$  is reported in the corresponding columns of Table 2 for 5 out of the twelve problems. 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 2, these two ROM-FOM hybrid methods have finite values of  $F(K)$  reported for all twelve test problems. Furthermore, in addition to ensuring stability of the closed-loop systems for the ROMs and FOMs, Alg. 2 (“R+F”) even

Table 3: Wall-Clock Running Times.

Wall-clock running times (seconds)						
Problem	HIFOO v3.5		Alg. 1		Alg. 2	
	$\delta_{\text{low}}$	$\delta_{\text{high}}$	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 ( $\delta_{\text{low}}$ )						
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

succeeded in finding the best value of  $F(K)$  on seven of the 12 problems, while on another three, the values it found were only slightly higher than those obtained by the best methods for those three problems. 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.

Figs. 1 and 2 show representative examples of the evolution of the  $\|G_r\|_{L_\infty}$  values computed by Alg. 2 (“R+F”) as a function of the iteration count, for problems HF2D\_CD1 and HF2D\_IS3 respectively. Only the (B) iterations in Alg. 2 are shown as the stabilization iterations in (A) are relatively less costly. The quantity  $\|G_r\|_{L_\infty}$  is steadily reduced in (B) until an infeasible point is reached, at which point the stabilization phase in (A) typically increases  $\|G_r\|_{L_\infty}$ , sometimes significantly. The usual trend, however, is for  $\|G_r\|_{L_\infty}$  to be consistently reduced over a sequence of (A) and (B) iterations, until either:

- a cumulative maximum number of 1000 iterations in phase (B) is reached, as with problem HF2D\_CD1, or, occasionally,
- GRANSO determines that an approximate stationarity condition has been satisfied (see [CMO17] for details), as with problem HF2D\_IS3.

It is also worth noting that the results for Alg. 1 (“R only”) are quite similar to the results



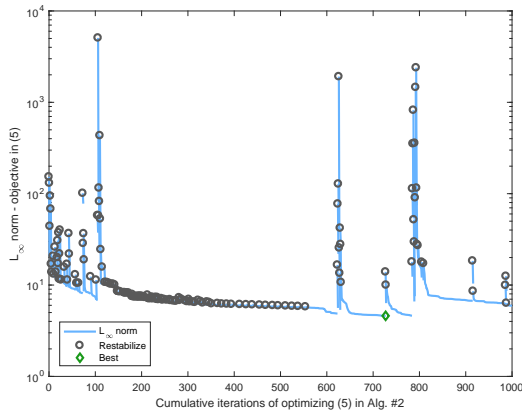


Figure 1: Problem HF2D\_CD1

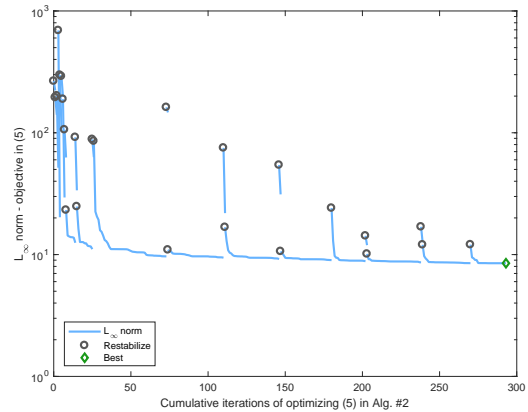


Figure 2: Problem HF2D\_IS3

for HIFOO, as expected, since these methods differ only in implementation details. Finally, in contrast to the fairly comparable results of Alg. 1 and Alg. 2 in the “R only” setting, it is clear that Alg. 2 is superior to Alg. 1 in the “R+F” setting, giving overall very satisfactory results. Indeed, the controllers found by Alg. 2 (“R+F”) yielded values of  $F(K)$  that were on average 8.35 times smaller than those obtained by Alg. 1 (“R+F”).

## 5 Conclusions

We have presented a new formulation for minimizing the  $L_\infty$  norm of the closed-loop transfer function for a reduced-order model (ROM) subject to stability constraints on the closed-loop systems for both the ROM and the full-order model (FOM). Algorithm 2 (“R+F”) was clearly effective in accomplishing this goal on the test problems that we considered, with running times that were not much slower, and sometimes faster, than the same algorithm without the FOM stability constraint, which, in consequence, often failed to stabilize the closed-loop system for the FOM.

## References

- [AN06a] P. Apkarian and D. Noll. Controller design via nonsmooth multidirectional search. *SIAM J. Control Optim.*, 44(6):1923–1949, 2006.
- [AN06b] P. Apkarian and D. Noll. Nonsmooth  $H_\infty$  synthesis. *IEEE Trans. Automat. Control*, 51(1):71–86, 2006.
- [Ant05] A. C. Antoulas. *Approximation of Large-Scale Dynamical Systems*, volume 6 of *Adv. Des. Control*. SIAM Publications, Philadelphia, PA, 2005.
- [BB90] S. Boyd and V. Balakrishnan. A regularity result for the singular values of a transfer matrix and a quadratically convergent algorithm for computing its  $L_\infty$ -norm. *Syst. Cont. Lett.*, 15:1–7, 1990.
- [BBF14] U. Baur, P. Benner, and L. Feng. Model order reduction for linear and nonlinear systems: A system-theoretic perspective. *Arch. Comput. Methods Eng.*, 21(4):331–358, 2014.

- [BCOW17] P. Benner, A. Cohen, M. Ohlberger, and K. Willcox. *Model Reduction and Approximation: Theory and Algorithms*. SIAM Publications, Philadelphia, PA, 2017. ISBN: 978-1-611974-81-2.
- [BHLO06a] J. V. Burke, D. Henrion, A. S. Lewis, and M. L. Overton. HIFOO - A MATLAB package for fixed-order controller design and  $H_\infty$  optimization. *IFAC Proceedings Volumes*, 39(9):339–344, 2006. 5th IFAC Symposium on Robust Control Design ROCOND 2006.
- [BHLO06b] J. V. Burke, D. Henrion, A. S. Lewis, and M. L. Overton. Stabilization via nonsmooth, nonconvex optimization. *IEEE Trans. Automat. Control*, 51(11):1760–1769, Nov 2006.
- [BMS05] P. Benner, V. Mehrmann, and D. C. Sorensen. *Dimension Reduction of Large-Scale Systems*, volume 45 of *Lect. Notes Comput. Sci. Eng.* Springer-Verlag, Berlin/Heidelberg, Germany, 2005.
- [BS90] N. A. Bruinsma and M. Steinbuch. A fast algorithm to compute the  $H_\infty$ -norm of a transfer function matrix. *Syst. Cont. Lett.*, 14(4):287–293, 1990.
- [CMO17] 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.
- [hin] Fixed-Structure H-infinity Synthesis with HINFSTRUCT. <https://www.mathworks.com/help/robust/examples/fixed-structure-h-infinity-synthesis-with-hinfstruct.html>.
- [Lei04] F. Leibfritz. *COMPLib: COstrained Matrix-optimization Problem library*. Technical report, Universität Trier, 2004.
- [LO13] A. S. Lewis and M. L. Overton. Nonsmooth optimization via quasi-Newton methods. *Math. Program.*, 141(1–2, Ser. A):135–163, 2013.
- [Mit] T. Mitchell. GRANSO: GRAdient-based Algorithm for Non-Smooth Optimization. <http://timmitchell.com/software/GRANSO>. See also [CMO17].
- [MO15] T. Mitchell and M. L. Overton. Fixed low-order controller design and  $H_\infty$  optimization for large-scale dynamical systems. *IFAC-PapersOnLine*, 48(14):25–30, 2015. 8th IFAC Symposium on Robust Control Design ROCOND 2015.
- [MO16] T. Mitchell and M. L. Overton. Hybrid expansion-contraction: a robust scaleable method for approximating the  $H_\infty$  norm. *IMA J. Numer. Anal.*, 36(3):985–1014, 2016.
- [OA01] G. Obinata and B. D. O. Anderson. *Model Reduction for Control System Design*. Communications and Control Engineering Series. Springer-Verlag, London, UK, 2001.
- [Ove] M. L. Overton. HANSO (Hybrid Algorithm for Non-Smooth Optimization). <https://cs.nyu.edu/~overton/software/hanso/>.
- [Ove06] M. L. Overton. HIFOO (H-Infinity Fixed-Order Optimization). <https://www.cs.nyu.edu/~overton/software/hifoo/>, 2006. See also [BHLO06a].
- [ZDG96] K. Zhou, J. C. Doyle, and K. Glover. *Robust and Optimal Control*. Prentice-Hall, Upper Saddle River, NJ, 1996.