HIFOO - A MATLAB PACKAGE FOR FIXED-ORDER CONTROLLER DESIGN AND H_∞ OPTIMIZATION

J. V. Burke [1] **D. Henrion** [2] **A. S. Lewis** [3]
M. L. Overton [4]

Abstract: H_∞ controller design for linear systems is a difficult, nonconvex and typically nonsmooth (nondifferentiable) optimization problem when the order of the controller is fixed to be less than that of the open-loop plant, a typical requirement in e.g. embedded aerospace control systems. In this paper we describe a new MATLAB package called HIFOO, aimed at solving fixed-order stabilization and local optimization problems. It depends on a new hybrid algorithm for nonsmooth, nonconvex optimization based on several techniques, namely quasi-Newton updating, bundling and gradient sampling. The user may request HIFOO to optimize one of several objectives, including H_∞ norm, which requires either the Control System Toolbox for MATLAB or, for much better performance, the `linorm` function in the SLICOT package. No other external package is required, but the quadratic programming code `quadprog` from either MOSEK or the Optimization Toolbox for MATLAB is recommended. Numerical experiments on benchmark problem instances from the COMPl$_e$ib database indicate that HIFOO could be an efficient and reliable computer-aided control system design (CACSD) tool, with a potential for realistic industrial applications.

Keywords: Fixed-order Controller Design, H_∞ Control, Nonconvex Optimization, Nonsmooth Optimization, Computer-Aided Control System Design.

1. INTRODUCTION

Soon after the development of robust control theory in the 1980s, it was realized that most physically meaningful and practically relevant analysis and design problems are difficult to solve in practice and some are provably difficult in a formal sense. See the excellent survey (Blondel and Tsitsiklis, 2000) for comprehensive definitions and classifications of difficult mathematical problems arising in control.

In particular, the problem of designing a controller stabilizing a linear plant is typically difficult when the order of the controller is fixed to be less than that of the open-loop plant. Static output feedback (SOF) design is the particular case of order 0. Despite many years of intense research efforts, no polynomial-time algorithm or efficient heuristic for SOF stabilization is known. Another related difficult problem is fixed-order stabilization with optimal H_∞ performance; see (Helton and Merino, 1998) for a historical perspective.

One source of difficulty is nonconvexity. In mathematical terms, stability is equivalent to location of

[1] Department of Mathematics, University of Washington, Seattle, WA 98195, USA. Email: `burke@math.washington.edu`.
[2] LAAS-CNRS, 7 Avenue du Colonel Roche, 31077 Toulouse, France and Department of Control Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, Technická 2, 16627 Prague, Czech Republic. Email: `henrion@laas.fr`.
[3] School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14853, USA. Email: `aslewis@orie.cornell.edu`.
[4] Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA. Email: `overton@cs.nyu.edu`.

the eigenvalues of a matrix (or equivalently roots of a polynomial) in the open left half-plane. [5] When formulated in the Euclidean space of real matrices (or polynomial coefficients), this stability constraint is nonconvex. It follows that the set of stabilizing controllers is typically nonconvex (and sometimes even disconnected) in the parameter space, as illustrated in (Ackermann, 2002).

An additional difficulty is nonsmoothness. Consider, to be specific, the spectral abscissa (maximum of the real parts of the eigenvalues) of a nonsymmetric real matrix. The spectral abscissa is nonsmooth at points in matrix space where more than one real eigenvalue or conjugate pair achieve the maximum real part, and is non-Lipschitz if an eigenvalue achieving the maximum real part has multiplicity greater than one. However, the spectral abscissa is differentiable as long as only one simple eigenvalue (real or conjugate pair) achieves the maximum real part, and in this case its gradient is easily computed by means of the associated left and right eigenvectors. The difficulty is that, in almost any interesting application, nonsmoothness occurs at optimizers of the spectral abscissa of a parameterized system (Burke et al., 2003).

The same difficulties with nonconvexity and nonsmoothness (but not non-Lipschitzness) apply to other minimization objectives of interest, such as the inverse of the complex stability radius or, perhaps of most interest, H_∞ performance. These functions likewise are smooth at most, but not all, points in parameter space, and are typically *not* smooth at minimizers.

In order to overcome or address nonconvexity, researchers have developed various techniques:
- convex approximations (with polytopes, ellipsoids, linear matrix inequalities (LMI)) of nonconvex stability regions. Typically these approximations are obtained from pessimistic or conservative sufficient stability conditions, and it is difficult to evaluate how far they are from being necessary conditions;
- LMI formulations of analysis and design conditions, introducing lifting variables. This has the drawback of introducing many artificial variables, even though the original problem has only a few genuine decision variables;
- nonconvex programming (global optimization, local BMI solvers, nonsmooth optimization). Systematic, exhaustive search global optimization is typically very expensive, while purely local methods lack any guarantee of finding a global solution; see (Henrion and Šebek, 2004) for a short survey and references.

In this paper we follow the local optimization approach. We present a new MATLAB package called HIFOO (H-Infinity Fixed-Order Optimization). It uses recently developed nonsmooth, nonconvex optimization techniques that are implemented in a supporting package called HANSO (Hybrid Algorithm for Non-Smooth Optimization).

The outline of the paper is as follows. In section 2 we set the notation and recall the basics of fixed-order stabilization and H_∞ optimal control. We describe the control problems that are addressed by HIFOO. In section 3 we briefly summarize the optimization techniques that are used by the supporting package HANSO. The external software that is used is is summarized in section 4. The powerful, yet user-friendly, input/output format of HIFOO is explained in section 5. Finally in section 6 we describe some simple examples of the use of HIFOO on some benchmark problems extracted from the comprehensive database COMPl$_{\text{e}}$ib (Leibfritz, 2005), and summarize some experimental results.

2. PROBLEM STATEMENT

We consider the state-space model of a linear system

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

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the control input (actuators), $y \in \mathbb{R}^p$ is the control output (sensors), w is the performance input signal and z is the performance output signal. For simplicity we assume that there is no direct feedthrough term, i.e.

$$D_{22} = 0.$$

The open-loop system is to be controlled by another linear system

$$\left[\begin{array}{c}\dot{\hat{x}}\\ u\end{array}\right] = \left[\begin{array}{cc}\hat{A} & \hat{B}\\ \hat{C} & \hat{D}\end{array}\right] \left[\begin{array}{c}\hat{x}\\ y\end{array}\right] \tag{2}$$

called the controller, where $\hat{x} \in \mathbb{R}^{\hat{n}}$ is the controller state. Note that the controller input is the system control output y, and the controller output is the system control input u.

Combining the open-loop system (1) with the controller (2), we obtain the so-called closed-loop system

$$\left[\begin{array}{c}\dot{x}\\ \dot{\hat{x}}\\ \hline z\end{array}\right] = \left[\begin{array}{c|c}A_K & B_K\\ \hline C_K & D_K\end{array}\right] \left[\begin{array}{c}x\\ \hat{x}\\ \hline w\end{array}\right] \tag{3}$$

where

[5] We restrict the discussion to continuous-time systems.

$$\left[\begin{array}{c|c} A_K & B_K \\ \hline C_K & D_K \end{array}\right] =$$

$$\left[\begin{array}{cc|c} A & 0 & B_1 \\ 0 & 0 & 0 \\ \hline C_1 & 0 & D_{11} \end{array}\right] + \left[\begin{array}{cc} 0 & B_2 \\ I & 0 \\ \hline 0 & D_{12} \end{array}\right] \left[\begin{array}{cc} \hat{A} & \hat{B} \\ \hat{C} & \hat{D} \end{array}\right] \left[\begin{array}{cc|c} 0 & I & 0 \\ C_2 & 0 & D_{21} \end{array}\right]$$

is an affine function of the controller matrix

$$K = \left[\begin{array}{cc} \hat{A} & \hat{B} \\ \hat{C} & \hat{D} \end{array}\right].$$

The transfer function between the performance input signal w and the output signal z is given by

$$T(s) = C_K(sI - A_K)^{-1}B_K + D_K. \qquad (4)$$

The control problems addressed by HIFOO can be formulated as follows:

Fixed-order stabilization: Given open-loop system (1) and a nonnegative integer $\hat{n}$, find a controller (2) of order $\hat{n}$ such that closed-loop system (3) is stable (equivalently, all eigenvalues of A_K are in the open left half-plane). This problem is called static output feedback stabilization when $\hat{n} = 0$.

Fixed-order H_∞ optimization: Among the solutions to the fixed-order stabilization problem, find a controller that locally minimizes the H_∞ norm of the transfer function $T(s)$ defined in (4).

Fixed-order stability radius optimization: Among the solutions to the fixed-order stabilization problem, find a controller that locally maximizes the complex stability radius of the matrix A_K (equivalently, minimizes the H_∞ norm of the transfer function $(sI - A_K)^{-1}$).

Fixed-order spectral abscissa optimization: Find a controller that locally minimizes the spectral abscissa of A_K (equivalently, pushes its eigenvalues as far to the left of the imaginary axis as possible).

Stabilization is implemented by applying optimization to the spectral abscissa objective, stopping as soon as a stabilizing controller (i.e., one for which the spectral abscissa of A_K is negative) is found. Following (Burke et al., 2003), the same process is used before initiating optimization of the H_∞ norm or the complex stability radius as these quantities are not defined (or may be taken to be ∞ or 0 respectively) when A_K has an eigenvalue in the closed right half-plane.

The number of variables in the controller matrix K is often quite small (less than 50 for many relevant engineering applications).

3. OPTIMIZATION ALGORITHM

All of the objective functions mentioned in the previous section are nonconvex and nonsmooth, and it is often the case that the objectives are not differentiable at local minimizers. The supporting package HANSO is designed to locally optimize functions of this kind, using a hybrid algorithm which combines:

- a quasi-Newton algorithm (BFGS) initial phase which, rather surprisingly, typically works very effectively even in the presence of nonsmoothness when implemented with the appropriate line search, and very often provides a fast way to approximate a local minimizer, and
- a local bundle phase which attempts to verify local optimality for the best point found by BFGS, and if this does not succeed,
- a gradient sampling phase (Burke et al., 2005; Burke et al., 2006) to refine the approximation of the local minimizer, returning a rough local optimality measure.

All three of these optimization techniques share two key attributes:

- They require the use of gradients. All the optimization objective functions supported by HIFOO have gradients that are readily computable using eigenvector or singular vector information that is already obtained in the process of computing the objective function. These gradients are computed by routines in HIFOO and provided to HANSO. No effort is made to identify the exceptional points where the gradients fail to exist.
- The algorithms are not defeated by the fact that the gradients do not exist at exceptional points (typically including optimizers) or, to say the same thing somewhat more informally, they are not defeated by the discontinuities in the gradients at exceptional points. On the contrary, these algorithms exploit these discontinuities. The BFGS phase builds a highly ill-conditioned Hessian approximation matrix, and the bundle and gradient sampling final phases search for a point in parameter space for which *a convex combination of gradients at nearby points has small norm.*

4. SOFTWARE REQUIREMENTS

HIFOO runs under MATLAB. Version 0.92 is freely available for download at

`www.cs.nyu.edu/overton/software/hifoo`

and uses the following external MATLAB packages:

- HANSO (required, freeware, see HIFOO web page)
- a convex quadratic programming solver called `quadprog` (required only for the local bundle and gradient sampling phases of HANSO, which are omitted if `quadprog` is not installed), *either* from
  - MOSEK (limited freeware, `www.mosek.com`), *or*

– the MATLAB Optimization Toolbox

• an H_∞ norm computation routine (required only for optimizing the H_∞ norm and complex stability radius objectives), *either*

– the `linorm` function of the SLICOT package (commercial, `www.slicot.de`, but with a limited free availability for noncommercial use with HIFOO), *or*

– the `norm` and `ss` functions of the MATLAB Control System Toolbox

• COMPl$_e$ib (freeware, `www.compleib.de`), a database of benchmark open-loop systems in state-space format (optional).

HIFOO checks the presence of the external routines and, by default, prints informative messages about what software is being used. The function `linorm` is strongly recommended in preference to `norm` because of its much faster performance, and if both are available, `linorm` is used.

5. INPUT/OUTPUT FORMAT

HIFOO has a user-friendly interface accepting various input formats. The only required input parameter, which must be provided first, is `plant`, which may have any of the following 3 formats:

• a structure whose fields `A`, `B1`, `B2`, `C1`, `C2`, `D11`, `D12`, `D21` define the open-loop system matrices (all required only when the H_∞ norm is optimized; for other objectives the only required fields are `A`, `B2`, `C2`); the names `B` and `C` can be used instead of `B2`, `C2` respectively, *or*

• an LTI state-space system object in the Control System Toolbox `ss` format; when the H_∞ norm is optimized the partitioning of $[B_1\ B_2]$, $[C_1\ C_2]$ must be specified in the fields `InputGroup.U1`, `InputGroup.U2` and the fields `OutputGroup.Y1`, `OutputGroup.Y2`; for other objectives, if the partitioning is not specified, it is assumed that B_1 and C_1 are empty; *or*

• a string specifying the COMPl$_e$ib problem name if the plant specification is supported by the COMPl$_e$ib library.

In addition to `plant`, HIFOO accepts four optional parameters which may be provided in any order:

• `order`, a nonnegative integer specifying $\hat{n}$, the order of the controller to be computed. The default is $\hat{n} = 0$ (static output feedback).

• `obj`, a character specifying the minimization objective, as follows:

– `'h'`: the H_∞ norm of the transfer function (4) (the default);

– `'r'`: the inverse of the complex stability radius of the matrix A_K (equivalently, the H_∞ norm of the transfer function $(sI - A_K)^{-1}$); if the matrices B_1, C_1 and D_{ij} are provided they are ignored;

– `'s'`: the spectral abscissa (maximum of the real parts of the eigenvalues) of the matrix A_K; if the matrices B_1, C_1 and D_{ij} are provided they are ignored;

– `'+'`: stabilize (find a point where A_K is stable, but do not optimize the spectral abscissa); if the matrices B_1, C_1 and D_{ij} are provided they are ignored.

• `init`, an initial guess for the controller, which may be *either*

– a structure whose fields `a`, `b`, `c`, `d` give initial guesses for $\hat{A}$, $\hat{B}$, $\hat{C}$, $\hat{D}$, *or*

– an LTI state-space system object in the Control System Toolbox `ss` format. Since the optimization algorithm is local, the results are often highly dependent on a starting guess, which may be provided by the user. If the order of the initial guess is less than the desired order, the initial guess is augmented to have the desired order without increasing the objective value. Thus HIFOO can be called repeatedly to get successively better[6] controllers as the order is increased. If the order of the initial guess is greater than the desired order, the initial guess is truncated arbitrarily. By default, several initial guesses are generated randomly, and even when `init` is provided, it is supplemented with random perturbations. Because of the random initializations, successive runs of HIFOO typically produce different controllers.

• `options`, a structure with optional fields determining various optimization details; see the online help for details. Fields likely to be of most interest to users are `penalty`, for penalizing the norm of the controller to prevent it from being chosen to be too large, `barrier`, for moving H_∞ norm minimizers away from the imaginary axis into the left half-plane, `normtol` and `evaldist`, for controlling the optimization accuracy, `cpumax`, for limiting the running time of the algorithm, and `prtlevel`, which specifies the desired level of printed output.

HIFOO returns up to 3 output parameters in the following order:

• `K`, the best controller found, which is *either* a structure with fields `a`, `b`, `c`, `d` defining the matrices $\hat{A}$, $\hat{B}$, $\hat{C}$, $\hat{D}$, *or* a Control System Toolbox LTI state-space system object; the format is chosen to be compatible with that used by `init` when provided, and otherwise compatible with that used by `plant`.

• `f`, the corresponding objective function value (∞ if no stable controller was found and `obj` is 'h' or 'r'; the spectral abscissa (negative if a stable controller was found) when `obj` is '+')

• `loc`, a local optimality measure with 2 fields: `dnorm`, the norm of a vector in the convex hull of gradients of the objective at and near the final controller, and `evaldist`, the maximum distance

[6] Or, at least, no worse, assuming the externally provided function computing the H_∞ norm is continuous with respect to the data, a somewhat idealized assumption.

that any of these gradients were evaluated from the final controller; the smaller `loc.dnorm` and `loc.evaldist` are, the more confidence one may have that a local minimizer has been approximated. (When `obj` is '+', `loc` is not relevant so both fields are set to NaN.)

6. EXAMPLES

6.1 Two mass-spring system

We start with the well-known benchmark example of two masses connected with a spring (Wie and Bernstein, 1992), with (normalized) state-space matrices

$$\left[\begin{array}{c|c} A & B_2 \\ \hline C_2 & 0 \end{array}\right] = \left[\begin{array}{cccc|c} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -1 & 1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 \end{array}\right]$$

corresponding to the open-loop transfer function $(s^4+2s^2)^{-1}$ between signals u and y. Suppose we would like to place the closed-loop poles as far left in the complex plane as possible with a second-order controller, i.e. we would like to find K that minimizes the spectral abscissa of the matrix A_K with $\hat{n}=2$.

Using the Control System Toolbox as an interface, a typical HIFOO session would be as follows (we suppress the default output produced by HIFOO):

```
>> P = ss([0 0 1 0;0 0 0 1;...
 -1 1 0 0;1 -1 0 0],[0 0 1 0]',...
 [0 1 0 0],0);
>> K = hifoo(P,2,'s');
>> tf(K)
```

We find the controller

$$\frac{6.8308175s^2 - 1.8486865s - 0.28043397}{s^2 + 4.2752492s + 6.0786141}.$$

The closed-loop poles, obtained with the commands [7]

```
>> T = feedback(P,-K);
>> eig(T)
```

are placed at $-0.7073 \pm i0.2979$, $-0.7073 \pm i0.2980$, $-0.7231 \pm i0.5343$ so the achieved spectral abscissa is -0.7073. We observe the typical eigenvalue clustering phenomenon characteristic of the neighborhood of a local minimizer of the spectral abscissa (Burke et al., 2003). Note that we display the controller coefficients to 8 significant digits because clustered eigenvalues are typically very sensitive to perturbations. In other words, controllers obtained by optimizing the spectral abscissa are typically quite non-robust.

We can call HIFOO again with the controller just obtained as an initial guess:

```
>> K = hifoo(P,2,'s',K);
```

This yields another controller with an improved closed-loop spectral abscissa of -0.7380. One more run of HIFOO produces a controller

$$\frac{8.073790s^2 - 1.7330367s - 0.23544720}{s^2 + 4.5435259s + 6.7343390}$$

further pushing the spectral abscissa to -0.7572.

Solving the pole placement equation by hand, assigning all the poles to the same negative real number $-\alpha$ (a unique pole of multiplicity six), we obtain analytically $\alpha = \frac{\sqrt{15}}{5} \approx 0.7746$ and the controller

$$\frac{\frac{43}{5}s^2 - \frac{54\sqrt{15}}{125}s - \frac{27}{125}}{s^2 + \frac{6\sqrt{15}}{5}s + 7}$$

$$\approx \frac{8.6000s^2 - 1.6731s - 0.2160}{s^2 + 4.6476s + 7}.$$

We can see that HIFOO found numerically a very similar controller and that the achieved spectral abscissa was not far from the one derived analytically. In fact, subsequent work based on (Burke et al., 2006) shows that this contoller is locally optimal (Henrion and Overton, 2006).

6.2 H_∞ static output feedback design

Consider the COMPl$_e$ib aircraft model `AC2`, with 5 states, 3 control inputs and 3 control outputs:

```
>> clear P
>> [P.A,P.B1,P.B2,P.C1,P.C2,...
 P.D11,P.D12,P.D21,nx,nu,ny] = ...
 COMPleib('AC2');
>> PP = ss(P.A,[P.B1 P.B2],[P.C1;P.C2],...
 [P.D11 P.D12;P.D21 zeros(ny,nu)]);
```

First we compute a full-order H_∞ controller with the `hinfsyn` function of the Robust Control Toolbox for MATLAB (Release 14SP1)

```
>> KF = hinfsyn(ss(P.A,[P.B1 P.B2],...
 [P.C1;P.C2],[P.D11 P.D12;...
 P.D21 zeros(ny,nu)]),nu,ny);
```

The default design algorithm is based on solving an algebraic Riccati equation. With this controller, we unexpectedly obtain an unstable closed-loop system:

```
>> TF = lft(PP,KF,nu,ny);
>> max(real(eig(TF)))
ans =
```

[7] Note the negative sign in front of the controller transfer, since we have used positive feedback notation in section 2.

```
  1.4563e-011
```

An alternative algorithm based on LMI is available in `hinfsyn`:

```
>> KF = hinfsyn(ss(P.A,[P.B1 P.B2],...
 [P.C1;P.C2],[P.D11 P.D12;...
 P.D21 zeros(ny,nu)]),...
 nu,ny,'method','lmi');
>> TF = lft(PP,KF,nu,ny);
>> max(real(eig(TF)))
ans =
 -8.0716e-005
>> norm(TF,inf)
ans =
    0.1115
```

The resulting closed-loop system is now stable, and the achieved H_∞ norm is equal to 0.1115.

Now let us try to use HIFOO to synthesize an H_∞ static output feedback controller for this system

```
>> K0 = hifoo(P);   % or, hifoo('AC2')
```

and let us analyze the closed-loop system

```
>> T0 = lft(PP,K0.d,3,3);
>> max(real(eig(T0)))
ans =
 -4.2263e-004
>> norm(T0,inf)
ans =
    0.1115
```

We see that, for this example, the closed-loop H_∞ performance achieved by static output feedback is the same as the one achieved with the fifth-order controller, which is the best achievable in closed-loop.

6.3 Fixed-order H_∞ design

As a final example, consider the COMPl$_e$ib aircraft model AC4, with 4 states, 1 control input and 2 control outputs. We compute a sequence of order 0, order 1, order 2, and order 3 controllers, optimizing H_∞ performance, by:

```
>> [K,f0,loc0]=hifoo('AC4');
>> [K,f1,loc1]=hifoo('AC4',1,K);
>> [K,f2,loc2]=hifoo('AC4',2,K);
>> [K,f3,loc3]=hifoo('AC4',3,K);
```

Notice that the controller output from each call to HIFOO is input as an initial guess for optimizing the controller of one order higher. The optimal values found for order 0 (0.9355) and order 3 (0.5573) are very consistent over multiple runs, and the corresponding optimality certificates (`loc0` and `loc3`) indicate that the results are close to locally optimal. The order 2 controller typically achieves nearly the same performance as the order 3 controller, and sometimes the order 1 controller does too, though this is less consistent. By contrast, the full-order routine `hinfsyn` fails on this example.

ACKNOWLEDGMENTS

We are grateful to Bill Helton and Alexandre Megretski for insightful discussions. The SLICOT norm routine `linorm` was kindly provided by NICONET e.V. through the help of Peter Benner. Any kind of commercial use of this routine requires a license agreement with Synoptio GmbH, Berlin, see `synmath.synoptio.de`. The first, third and fourth authors are supported in part by the U.S. National Science Foundation. The second author acknowledges support by Project 102/05/0011 of the Grant Agency of the Czech Republic and Project ME 698/2003 of the Ministry of Education of the Czech Republic.

REFERENCES

Ackermann, J. (2002). *Robust control: the parameter space approach*. 2nd ed., Springer-Verlag, Berlin, Germany.

Blondel, V. D. and J. N. Tsitsiklis (2000). A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249-1274.

Burke, J. V., A. S. Lewis, M. L. Overton (2003). A nonsmooth, nonconvex optimization approach to robust stabilization by static output feedback and low-order controller. *Proc. IFAC Symp. Robust Control Design*, Milan, Italy.

Burke, J. V., A. S. Lewis and M. L. Overton (2005). A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM J. Optim.*, 15:751-779.

Burke, J. V., D. Henrion, A. S. Lewis, M. L. Overton (2006). Stabilization via nonsmooth, nonconvex optimization. To appear in *IEEE Trans. Autom. Control*.

Helton, J. W. and O. Merino (1998). *Classical control using H_∞ methods*. SIAM, Philadelphia.

Henrion, D. and M. L. Overton (2006). Maximizing the closed loop asymptotic decay rate for the two-mass-spring control problem. Submitted to *Automatica*.

Henrion, D. and M. Šebek (2004). Overcoming nonconvexity in polynomial robust control design. *Proc. Symp. Math. Theory of Networks and Systems*, Leuven, Belgium.

Leibfritz, F. (2005). COMPl$_e$ib: constraint matrix optimization problem library. Version 1.1, Univ. Trier, Germany. `www.compleib.de`.

Wie, B. and D. S. Bernstein (1992). Benchmark problems for robust control design. *AIAA J. Guidance, Control and Dynamics*, 15(5):1057–1059.