# Solving static output feedback problems by direct search optimization

Didier Henrion

May 10, 2006

**Abstract**

Direct search methods are local optimization algorithms maximizing a possibly nonsmooth and nonconvex function using its values only (no first-order gradient or second-order Hessian information). More than one decade ago, N. J. Higham implemented some simple Matlab routines for direct search optimization, investigating questions on stability and accuracy of numerical algorithms in matrix computations. The purpose of this note is to report numerical experiments showing that with these routines a non-expert user can solve at moderate cost most of the static output feedback design problems available in the problem library COMPl$_e$ib.

**Index Terms**

Static output feedback, nonconvex optimization, direct search methods, computer-aided control system design.

## I. Introduction

In [3], N. J. Higham proposed to use *direct search methods* to address experimentally standard issues of numerical analysis, such as numerical stability of a given algorithm, bounds on growth factor in Gaussian elimination or quality evaluation for condition estimates. A direct search method is a local optimization heuristic that attempts to maximize (or minimize) an unconstrained function using values only, without making any convexity, smoothness or even continuity assumption regarding the function, see [7] for a nice survey. Direct search methods do not claim global or local optimality of the computed solution (since derivatives are not calculated) but in many cases they can prove useful, especially for nonconvex and/or

D. Henrion is with LAAS-CNRS, 7 Avenue du Colonel Roche, 31077 Toulouse, France. He is also with the Department of Control Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, Technická 2, 166 27 Prague, Czech Republic.

nonsmooth objective functions. In the convex and/or smooth case, they can however hardly compete with more sophisticated methods of nonlinear programming using first-order and second-order information, such as interior-point methods.

Following [3], N. J. Higham implemented three multivariate direct search maximization functions in his Matrix Computation Toolbox for Matlab:

- `mdsmax`: multidirectional search method
- `adsmax`: alternating directions method
- `nmsmax`: Nelder-Mead simplex method

see [4] for more information. Individual M-file codes do not exceed 200 lines each. Direct search methods are also described in [5, Chapter 26].

Control problems, in their simplest formulations, are generally *nonconvex and nonsmooth*. For example, the problem of stabilizing a continuous-time linear system with a linear controller of fixed order can be formulated as a *spectral abscissa*[1] *minimization* problem on a square matrix whose entries are affine functions of the controller parameters. The spectral abscissa is typically a nonconvex and nonsmooth function of the controller parameters. A well-known example of spectral abscissa optimization problem is the static output feedback (SOF) problem:

**SOF problem:** Given matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathrm{R}^{p\times n}$, find a matrix $K \in \mathrm{R}^{m\times p}$ such that the spectral abscissa of matrix $A + BKC$ is strictly negative.

Control engineers generally symmetrize the spectral abscissa problem into a nonlinear (typically bilinear) matrix inequality problem by introducing a positive definite multiplier matrix, or Lyapunov matrix. In the case of SOF, the bilinear matrix inequality (BMI) becomes $(A + BKC)^T P + P(A + BKC) \preceq -I_n,\ P = P^T \succeq I_n$ where $\preceq$, $\succeq$, $I_n$ respectively stand for negative semidefinite, positive semidefinite and the identity matrix. Note that this symmetrization does not remove nonconvexity or nonsmoothness, but it has been popular in the control community because of its decoupled structure: when the Lyapunov matrix $P$ is fixed, the BMI becomes linear (LMI) in $K$, and vice-versa. An LMI problem is convex, and it can be solved via efficient off-the-shelf implementations of primal-dual interior-point algorithms, see [9] for a user-friendly Matlab interface with links to all the public-domain implementations. The main limitation of this symmetrization approach for spectral abscissa problem seems to be the introduction via Lyapunov matrix $P$ of $n(n+1)/2$ additional

[1]The spectral abscissa of a matrix is the maximum real part of its eigenvalues.

variables in the decision problem, whereas a control engineer is primarily interested only in the $mp$ entries of the feedback matrix $K$. Since interior-point algorithms are *second-order methods* (most of the computational burden occurs when forming and storing the Hessian) it is critical to optimize over the smallest possible set of decision variables[2]

Alternatively, spectral abscissa minimization can be addressed directly with nonconvex nonsmooth optimization techniques. The recently developed Matlab package HIFOO [2] uses an implementation of a hybrid algorithm mixing quasi-Newton BFGS, bundling and gradient sampling, which all use gradient information only. In other words, they are *first-order methods*.

In this note we report numerical attempts of solving the SOF problem using direct search methods, which are *zeroth-order methods* as recalled above, in contrast with gradient or interior-point methods. The SOF problem data are extracted from the publicly available database COMPl$_e$ib gathering many practically meaningful instances from the technical literature [8].

Note that this note focuses only on the SOF *stabilization* problem, in the sense that we are not optimizing performance criteria such as closed-loop decay rate, $H_2$ or $H_\infty$ norms.

## II. Numerical experiments

We use only the first three input arguments of functions `adsmax`, `mdsmax`, `nmsmax`, which are:

- a pointer `fun` to the function to be maximized
- an initial guess `x0` on the maximizer
- a vector `stopit` describing the stopping criterion. The iteration is terminated when either:
  - the relative increase in function value between successive iterations is less than or equal to `stopit(1)`, or
  - `stopit(2)` function evaluations have been performed, or
  - a function value equals or exceeds `stopit(3)`.

To solve the SOF problem, we used the following parameters:

- `K0 = zeros(size(B,2),size(C,1))`

[2]Note that there is now a public-domain code called PENBMI for (locally) solving BMI problems, an implementation of a penalty-barrier-multiplier method which also works for (globally) solving convex LMI problems [6].

- `stopit(1) = 1e-3`
- `stopit(2) = inf`
- `stopit(3) = 1e-6`
- `fun = @(K) -max(real(eig(A+B*K*C)))`

where the last command uses Matlab's function handle syntax, see `help function_handle`.

A typical call of function `adsmax` to solve the SOF problem `AC13` from COMPl$_e$ib would be as follows:

```
>> % Retrieve A,B,C data
>> [A,B1,B,C1,C] = compleib('AC13');
>> % Initial guess
>> K0 = zeros(size(B,2),size(C,1));
>> % Parameter vector
>> stopit = [1e-3 inf 1e-6];
>> % Function to maximize
>> fun = @(K) -max(real(eig(A+B*K*C)))
>> % Call adsmax
>> K = adsmax(fun,K0,stopit)
f(x0) = -5.7993e-001
Iter  1  (nf =  1)
Comp.=1, steps=19, f=-8.2901e-002 (85.7%)
Comp.=2, steps=16, f=-3.2160e-002 (61.2%)
Comp.=3, steps=16, f=-2.9911e-002 (7.0%)
Comp.=4, steps= 8, f=-1.7325e-002 (42.1%)
Comp.=5, steps=10, f=5.0315e-003*
Exceeded target...quitting
K =
  -26.2144   -3.6029     0     0
  -85.8993  -14.4115     0     0
  281.4750         0     0     0
>> % Check SOF gain
>> max(real(eig(A+B*K*C)))
ans =
```

| name | $n$ | $m$ | $p$ | adsmax | mdsmax | nmsmax |
|---|---|---|---|---|---|---|
| AC1 | 5 | 3 | 3 | 0.08 | 0.13 | 0.08 |
| AC2 | 5 | 3 | 3 | 0.08 | 0.05 | 0.02 |
| AC3 | 5 | 2 | 4 | 0.00 | 0.02 | 0.00 |
| AC4 | 4 | 1 | 2 | 0.00 | 0.00 | 0.08 |
| AC6 | 7 | 2 | 4 | 0.06 | 0.08 | 0.08 |
| AC7 | 9 | 1 | 2 | 0.08 | 0.06 | 0.08 |
| AC8 | 9 | 1 | 5 | 0.11 | 0.13 | 0.05 |
| AC9 | 10 | 4 | 5 | 0.13 | 0.08 | 0.08 |
| AC10 | 55 | 2 | 2 | 0.47 | * | 0.64 |
| AC11 | 5 | 2 | 4 | 0.08 | 0.05 | 0.02 |
| AC12 | 4 | 3 | 4 | 0.11 | 0.02 | 0.05 |
| AC13 | 28 | 3 | 4 | 0.23 | 0.92 | 0.42 |
| AC14 | 40 | 3 | 4 | 0.23 | 0.42 | 0.44 |
| AC15 | 4 | 2 | 3 | 0.03 | 0.02 | 0.00 |
| AC16 | 4 | 2 | 4 | 0.03 | 0.00 | 0.02 |
| AC17 | 4 | 1 | 2 | 0.03 | 0.00 | 0.00 |
| AC18 | 10 | 2 | 2 | 0.05 | * | 0.08 |

TABLE I

AIRCRAFT MODELS

```
-0.0050
```

In the tables below we report the performance of the three direct search functions on the COMPl$_e$ib SOF problems, for the default parameter tunings described above. Reported are CPU times in seconds, stars (*) meaning that the algorithm failed to solve the SOF problem. We used Matlab Release 14 Service Pack 1 on a PC equipped with MS Windows 2000 and an Intel Pentium 1.60GHz CPU with 512MB RAM.

## III. DISCUSSION

We can see that most of the SOF problems could be solved in a fraction of seconds with direct search methods initialized with the zero feedback matrix. This seems to be surprising, given the amount of effort dedicated by researchers these last decades to design efficient algorithms to solve SOF problems. The most simple and basic methods seem to be amongst the most efficient ones. Morever, the methods can be easily tuned, they are very robust (absolutely no numerical troubles were encountered when running the SOF benchmark), and

| name | $n$ | $m$ | $p$ | adsmax | mdsmax | nmsmax |
|---|---|---|---|---|---|---|
| HE1 | 4 | 2 | 1 | 0.05 | 0.00 | 0.00 |
| HE2 | 4 | 2 | 2 | 0.00 | 0.00 | 0.03 |
| HE3 | 8 | 4 | 6 | 0.30 | 0.13 | 0.05 |
| HE4 | 8 | 4 | 6 | 0.33 | 0.19 | 0.13 |
| HE5 | 8 | 4 | 2 | 0.13 | 0.14 | 0.14 |
| HE6 | 20 | 4 | 6 | 0.42 | 0.11 | 0.11 |
| HE7 | 20 | 4 | 6 | 0.41 | 0.13 | 0.09 |

TABLE II

HELICOPTER MODELS

| name | $n$ | $m$ | $p$ | adsmax | mdsmax | nmsmax |
|---|---|---|---|---|---|---|
| JE1 | 30 | 3 | 5 | 0.05 | 0.05 | 0.08 |
| JE2 | 21 | 3 | 3 | * | 0.14 | 0.05 |
| JE3 | 24 | 3 | 6 | * | 0.22 | 0.08 |
| REA1 | 4 | 2 | 3 | 0.05 | 0.05 | 0.08 |
| REA2 | 4 | 2 | 2 | 0.06 | 0.00 | 0.02 |
| REA3 | 12 | 1 | 3 | 0.03 | 0.02 | 0.00 |

TABLE III

JET ENGINE (JE) AND REACTOR (REA) MODELS

| name | $n$ | $m$ | $p$ | adsmax | mdsmax | nmsmax |
|---|---|---|---|---|---|---|
| DIS1 | 8 | 4 | 4 | 0.00 | 0.02 | 0.03 |
| DIS2 | 3 | 2 | 2 | 0.06 | 0.02 | 0.00 |
| DIS3 | 6 | 4 | 4 | 0.03 | 0.02 | 0.00 |
| DIS4 | 6 | 4 | 6 | * | 0.09 | 0.17 |
| DIS5 | 4 | 2 | 2 | * | * | 0.16 |
| EB1 | 10 | 1 | 1 | 0.03 | 0.00 | 0.00 |
| EB2 | 10 | 1 | 1 | 0.00 | 0.00 | 0.00 |
| EB3 | 10 | 1 | 1 | 0.03 | 0.00 | 0.02 |
| EB4 | 20 | 1 | 1 | 0.06 | 0.00 | 0.00 |
| EB5 | 40 | 1 | 1 | 0.06 | 0.02 | 0.00 |
| EB6 | 160 | 1 | 1 | 0.23 | 0.38 | 0.34 |

TABLE IV

DECENTRALIZED INTERCONNECTED SYSTEMS (DIS) AND EULER-BERNOULLI BEAMS (EB)

| name | $n$ | $m$ | $p$ | adsmax | mdsmax | nmsmax |
|---|---|---|---|---|---|---|
| TG1 | 10 | 2 | 2 | 0.02 | 0.00 | 0.00 |
| AGS | 12 | 2 | 2 | 0.03 | 0.00 | 0.02 |
| WEC1 | 10 | 3 | 4 | 0.06 | 0.00 | 0.05 |
| WEC2 | 10 | 3 | 4 | 0.03 | 0.00 | 0.05 |
| WEC3 | 10 | 3 | 4 | 0.05 | 0.06 | 0.03 |
| HF1 | 130 | 1 | 2 | 0.17 | 0.17 | 0.17 |
| BDT1 | 11 | 3 | 3 | 0.03 | 0.05 | 0.03 |
| BDT2 | 82 | 4 | 4 | 0.50 | 0.34 | 0.34 |
| MFP | 4 | 3 | 2 | 0.00 | 0.02 | 0.00 |
| UWV | 8 | 2 | 2 | 0.06 | 0.00 | 0.00 |
| IH | 21 | 11 | 10 | * | * | 0.16 |
| CSE1 | 20 | 2 | 10 | 0.00 | 0.08 | 0.06 |
| CSE2 | 60 | 2 | 30 | 0.08 | 0.64 | 0.63 |
| PAS | 5 | 1 | 3 | * | 0.05 | 0.02 |
| TF1 | 7 | 2 | 4 | * | 0.13 | 0.08 |
| TF2 | 7 | 2 | 3 | * | 0.02 | 0.05 |
| TF3 | 7 | 2 | 3 | * | * | * |
| PSM | 7 | 2 | 3 | 0.03 | 0.00 | 0.03 |
| TL | 256 | 2 | 2 | 1.56 | 2.84 | 2.83 |
| CDP | 120 | 2 | 2 | 0.17 | 0.19 | 0.20 |

TABLE V

VARIOUS PHYSICAL SYSTEMS

publicly available. No specific knowledge of the algorithm is required, and a non-expert user has just to follow the syntax outlined in the previous section to obtain quickly meaningful results.

E. Prempain remarked that systems in COMPl$_e$ib which are already open-loop stable should be removed from the list, since the initial guess $K = 0$ obviously solves the SOF problem. He also pointed out that some of the systems are not continuous-time SOF stabilizable, e.g. NN3, NN10, NN12. Hopefully they will be removed or updated in a next release of COMPl$_e$ib.

When an algorithm fails with zero initial condition, most of the time it is successful with another (e.g. random) initial condition. Another heuristic which proves generally efficient consists in optimizing with one algorithm until no progress is possible, and then using the resulting point to initialize another algorithm, as suggested originally in [3]. Specific numerical examples are not described here, but the interested reader can easily experiment

| name | $n$ | $m$ | $p$ | adsmax | mdsmax | nmsmax |
|---|---|---|---|---|---|---|
| NN1 | 3 | 1 | 2 | * | 0.23 | 0.08 |
| NN2 | 2 | 1 | 1 | 0.03 | 0.00 | 0.00 |
| NN3 | 4 | 1 | 1 | * | * | * |
| NN4 | 4 | 2 | 3 | 0.06 | 0.00 | 0.00 |
| NN5 | 7 | 1 | 2 | 0.06 | 0.06 | 0.00 |
| NN6 | 9 | 1 | 4 | 0.27 | * | * |
| NN7 | 9 | 1 | 4 | 0.23 | * | * |
| NN8 | 3 | 2 | 2 | 0.02 | 0.00 | 0.00 |
| NN9 | 5 | 3 | 2 | 0.17 | 0.13 | 0.06 |
| NN10 | 8 | 3 | 3 | * | * | * |
| NN11 | 16 | 3 | 5 | 0.09 | 0.02 | 0.00 |
| NN12 | 6 | 2 | 2 | * | * | * |
| NN13 | 6 | 2 | 2 | * | 0.08 | 0.17 |
| NN14 | 6 | 2 | 2 | * | 0.08 | 0.17 |
| NN15 | 3 | 2 | 2 | 0.03 | 0.02 | 0.00 |
| NN16 | 8 | 4 | 4 | 0.03 | 0.03 | 0.08 |
| NN17 | 3 | 2 | 1 | 0.08 | 0.02 | 0.00 |
| NN18 | 1006 | 1 | 1 | 73.53 | 34.14 | 34.19 |

TABLE VI

ACADEMIC TEST PROBLEMS

himself.

As pointed out to us by N. J. Higham, the good performance of direct search methods on SOF problems is also related to the fact that we only require to find *some* matrix for which the spectral condition is true, not the global optimum of a function. Similarly, in [3], only a large function value was required, not necessarily the global maximum.

However, generally speaking, the good performance of simple direct search algorithms on SOF problems is not well understood. It could be due to some geometric properties inherent to SOF problems. More specifically, visual inspection reveals that 6 out of the 7 two-dimensional ($mp = 2$) SOF problems of COMPl$_e$ib are actually convex problems when formulated directly in the feedback matrix parameter space. Convexity is definitely a favorable property for optimization algorithms. It would be instructive to check out the amount of non-convexity of SOF problems in COMPl$_e$ib, measured e.g. as the relative volume of a set that has to be changed for the set to become convex. The only convexity checking method we are aware of is a randomized algorithm proposed recently in [10].

Note finally that we restricted our attention to SOF *stabilization* only, in the sense that no attempt was made to optimize closed-loop performance criteria relevant to control engineering. Sophisticated modifications of direct search routines dealing with potential nonsmoothness of these criteria (spectral abscissa, $H_\infty$ norm) have been proposed recently in [1]. Note however that no public domain software has been provided by these authors and hence it is difficult to reproduce the computational experiments described there. See [2] for a publicly available Matlab implementation of nonsmooth nonconvex optimization algorithms with reproducible numerical experiments.

## ACKNOWLEDGMENTS

This work was funded in part by projects No. 102/05/0011 and No. 102/06/0652 of the Grant Agency of the Czech Republic and project No. ME 698/2003 of the Ministry of Education of the Czech Republic. Comments by D. Arzelier, N. J. Higham, M. L. Overton, E. Prempain, V. Torczon and an anynomous reviewer are gratefully acknowledged.

## REFERENCES

[1] P. Apkarian, D. Noll. Controller design via nonsmooth multidirectional search. SIAM J. Control Opt. 44(6):1923-1949, 2005.

[2] J. V. Burke, D. Henrion, A. S. Lewis, M. L. Overton. HIFOO - A Matlab package for fixed-order controller design and $H_\infty$ optimization, to be presented at the IFAC Symposium on Robust Control Design, Toulouse, July 2006. See `www.cs.nyu.edu/overton/software/hifoo`

[3] N. J. Higham. Optimization by direct search in matrix computations. SIAM J. Matrix Anal. Appl. 14(2):317–333, April 1993.

[4] N. J. Higham. The Matrix Computation Toolbox. Version 1.2 released in September 2002. See `www.ma.man.ac.uk/~higham/mctoolbox`

[5] N. J. Higham. Accuracy and Stability of Numerical Algorithms. Second Edition, SIAM, Philadelphia, 2002.

[6] M. Kočvara, M. Stingl. PENBMI. Version 2.0 released in 2004. See `www.penopt.com`

[7] T. Kolda, R. Lewis, V. Torczon. Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods. SIAM Review, 45(3):385–482, 2003.

[8] F. Leibfritz. COMPl$_e$ib: constraint matrix optimization problem library - a collection of test examples for nonlinear semidefinite programs, control system design and related problems. Technical Report, University ot Trier, Germany, 2004. See `www.complib.de`

[9] J. Löfberg. YALMIP. Version 3.0 released in 2005. See `control.ee.ethz.ch/~joloef`

[10] L. Rademacher, S. Vempala. Testing geometric convexity. Dept. Math. MIT, Massachusetts, 2004.