2011 50th IEEE Conference on Decision and Control and
European Control Conference (CDC-ECC)
Orlando, FL, USA, December 12-15, 2011

Optimal static output feedback design through direct search

E. Simon

Abstract— **This paper investigates the performance of using a direct search method to design optimal Static Output Feedback (SOF) controllers for Linear Time Invariant (LTI) systems. Considering the old age of both SOF problems and direct search methods, surprisingly good performances will be obtained compared to a state-of-the-art method. The motivation is to emphasize the fact that direct search methods are too much neglected by the control community. These methods are very rich for practical purposes on a lot of complex problems unyielding to classical optimization techniques, like linear matrix inequalities, thanks to their ability to explore even non-smooth functions on non-convex feasible sets.**

Index Terms— **Optimal control, Direct search methods, Static Output Feedback, Linear Time Invariant systems**

I. Introduction

This paper considers a class of fundamental and still open problems: the design of optimal Static Output Feedback (SOF) controllers for Linear Time Invariant (LTI) systems. The approach used here is to solve benchmarks of such problems with a state-of-the art method (HIFOO [9], [5]), and compare the objective values found and computational times required with those obtained using a direct search method. The quality of these results will allow to determine if direct search methods are adequate as a powerful candidate to solve complex optimization problems in systems and control.

There are historical reasons behind the fact that direct search methods are largely overlooked in this field, which we summarize briefly as follows. Direct search methods appeared in the late fifties and quickly enjoyed a vast success amongst optimization practitioners. However since a review of W.H. Swann in 1972, direct search methods have been scorned by theoreticians because they lacked convergence guarantees [22]. It is only since the early nineties -i.e. from the PhD thesis of Virginia Torczon in 1989 [21]- that proofs of convergence started to appear for direct search methods, which then started to revive. However this was eclipsed in optimization for systems and control by Linear Matrix Inequalities (LMIs), which exploded at that time and became the standard approach still to this day.

The approach of LMIs, originally motivated by Lyapunov theory, is richer for theoretical developments. But for complex problems, for which the feasible set is typically non-convex, LMIs approaches can be inadequate and unsuccessful. Efficient iterative LMIs algorithms can be found for SOF stabilization (see e.g. [12], [8]). However these are feasibility problems and no such algorithm has been shown efficient for minimization problems, e.g. minimizing the $\mathcal{H}_2$ and $\mathcal{H}_\infty$ norms. In general, iterative LMIs algorithms proposed to solve Bilinear Matrix Inequalities (BMIs) problems do not converge toward locally optimal solutions neither in theory nor in practice (see [19] and references therein).

Note also the general large drawback of using LMIs: many additional variables are typically required to write the problem under a LMIs formulation. Most often such variables come from a Lyapunov matrix with $n(n+1)/2$ entries, where n is the number of states of not only the controller to be designed but also that of the open loop plant. Therefore, despite the polynomial time complexity of solving LMIs problems -which is however not guaranteed anymore with iterative LMIs algorithms- and the efficient solvers available, LMIs approaches may quickly lead to prohibitive computational times.

Furthermore, writing problems under a LMIs scheme can prove very strenuous and quite inflexible to adaptations or objective modifications. While the set of problems that can be brought under LMIs formulations is continuously increasing, it does not easily encompass more complex problems based on real-life objectives and constraints met in the industry.

Considering these three drawbacks, there appears a need to find better suited methods. When the gradients of the considered objective functions are available, such informations should be used (if they are not too expensive to compute). This is done within HIFOO©[9], [5] and `hinfstruct`©[2], certainly the best two methods for the problems considered in this paper (see benchmarks in [1]).

It is however interesting to investigate the performance of a direct search method, not using gradient informations, for the problem considered and see how well it fares compared to HIFOO (`hinfstruct` was not available to us at the time). If the direct search approach is found successful enough on these central problems, this gives a good indication that these methods are adequate for many current open complex problems of the field. HIFOO or `hinfstruct` are more efficient for the problems they consider thanks to the use of gradient informations, but direct search methods allow to span a very broad set of problems as long as those can be formulated as minimizing a function $f(x) : \mathbb{R}^n \longrightarrow \mathbb{R}$.

In this paper we use the Nelder-Mead algorithm (NM), restarted to improve convergence. There are other direct search methods with theoretical convergence guarantees: MDS [21], [11] on smooth functions and MADS [6] even on non-smooth functions, but this restarted NM will suffice to obtain good objective values while keeping reasonable computational times.

Emile Simon is with the Department of Mathematical Engineering, Université Catholique de Louvain, 4 avenue Georges Lemaitre, 1348 Louvain-la-Neuve, Belgium, simonemile at gmail dot com.

Contents and structure of the paper

The paper is structured as follows. First the SOF problem for LTI systems is presented in Section II, along with a brief discussion on some current methods available. Then a brief description of the NM algorithm is given in Section III, along with two modifications that improve in practice both the obtained objectives and the required computational times. Sections IV, V and VI present the results obtained for the most typical LTI optimal control objectives: stabilization, $\mathcal{H}_2$ norm and $\mathcal{H}_\infty$ norm minimizations. The LTI models used for these benchmarks are the unstable models from the COMPlib datatbase [14], which regroups many systems coming from both real applications and academic problems. Finally, the conclusions are drawn.

II. Static output feedback design

The static output feedback (SOF) is the control law $u = Ky$ with the gain matrix $K \in \mathbb{R}^{n_u \times n_y}$ that determines instantly the actuation input(s) $u \in \mathbb{R}^{n_u}$ given the measurement output(s) $y \in \mathbb{R}^{n_y}$ of a system. Designing this matrix K to ensure desired properties for the closed-loop system can be seen as the first problem in (output-feedback) control design (in [20] it is regarded as '*probably the most important open question in control engineering*'). The framework considered is that of LTI systems, where the models of the systems are fixed linear differential equations and do not vary with time. Within this setup, the models P can be written under the so-called state-space representation as:

$$P: \left\{ \begin{array}{rcl} \dot{x} &=& Ax + B_1 w + Bu \\ z &=& C_1 x + D_{11} w + D_{12} u \\ y &=& Cx + D_{21} w + D_{22} u \end{array} \right. \quad (1)$$

where $x \in \mathbb{R}^{n_x}$ is the state vector, $\dot{x}$ is the time derivative of x, $z \in \mathbb{R}^{n_z}$ the performance ouptut(s), $w \in \mathbb{R}^{n_w}$ the performance input(s), u and y defined above, and the other elements are matrices of corresponding dimensions with real entries. In this paper continuous-time systems are used but the extension to discrete-time systems is almost immediate (unlike most other methods, including HIFOO). It is assumed that $D_{22} = 0$ without loss of generality, otherwise a linear change of variables presented in [23] can be used (this is necessary to have K entering affinely the closed-loop representation). Therefore the state-space representation of the closed-loop performance channel(s) T_{wz} becomes:

$$\left(\begin{array}{cc} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{array} \right) = \left(\begin{array}{cc} A + BKC & B_1 + BKD_{21} \\ C_1 + D_{12}KC & D_{11} + D_{12}KD_{21} \end{array} \right) \quad (2)$$

This representation will be used to define the typical optimal control objectives of Sections IV, V and VI.

Current methods and results

A survey of SOF design methods is a clearly too vast topic to be treated here. For this we refer to [20] and more recently to the introduction of [4] and the references therein. Note that reduced-order design problems extend naturally from the SOF ones, by augmenting the state spaces matrices of the system. The two best methods for these problems are certainly HIFOO [9], [5] and `hinfstruct`, the last only dealing at the moment with the $\mathcal{H}_\infty$ case and appearing even faster than HIFOO [1] but for which no implementation was available to us at the time. So the method used here will be compared to HIFOO on a large series of tests, objectives and systems, both in terms of obtained objectives and required computational times.

A method that was not mentioned in [20] was pointed out surprisingly late in [10]: to use direct search (DS) methods, amongst which the NM algorithm, for the SOF problem. Note that other control design problems have been dealt with DS methods in the past but, to the best of our knowledge, not SOF problems before [10]. In [2], the use of DS methods in control design is rather harshly criticized. The discussion is actually on how often the non-smoothness of the control design problems causes DS methods to fail. In this paper, for SOF problems and using the modified version of NM, we illustrate that this very seldom happens with extensive numerical experiments and comparisons with HIFOO which is gradient-based and enjoys a convergence certificate. Note that in [2] MDS [21], [11] has been tested, but not NM.

In [10] it is outlined that DS methods can be efficient for SOF stabilization but not to what extent, or for other problems. Now that we can make comparisons with the efficient and easily available method HIFOO, we can get a clearer picture of the performances that can actually be expected. Once the good performances of NM are established, future work can deal with more complex control objectives considering that NM is extremely flexible: it admits any real-valued objective function $f(x) : \mathbb{R}^n \longrightarrow \mathbb{R}$. Furthermore, as pointed out in [10], non-expert users can easily use this optimization routine and adapt it to their needs.

We also note that with DS methods no additional variables, like Lyapunov matrices, are required. This allows to use formulations of the problems with the least variables, which can give a significant edge to direct search methods for the computational time. In practice a rule of a thumb would be to use problem formulations with less than 50 or 100 variables, depending on the computational time needed to perform one function evaluation. With more variables, only significant ones have to be selected and/or gradients informations become unavoidable to keep reasonable computational times.

III. The Nelder-Mead algorithm

This algorithm, presented in [16] in the sixties, belongs to the class of DS methods. These methods only use cost function evaluations and no gradient (first order methods, like HIFOO) or Hessian (second order methods, like interior point methods) informations. The basic ideas behind NM are briefly described here, the interested reader can refer e.g. to [16], [13], [17] for proper descriptions and details.

The first step is the generation of an initial simplex of $N + 1$ solutions (around and including the provided initial solution), where N is the number of variables of the function. The cost function is then evaluated at each of these $N + 1$ solutions and sorted from the best to the worst solution.

This initial simplex generated, the NM algorithm chooses iteratively between several possibilities (or steps) to change the shape of the simplex (eventually displacing it), trying to find better solutions. For example the basic step is that the worst solution is reflected on the other side of the simplex, in order to create a 'downhill' effect.

Different implementations of the NM algorithm can be found in the literature. Two of the most easily available are the `fminsearch` function in MATLAB©, based on [13], and the `nmsmax` function of [11] (the one used in [10]). Early tests gave us better results with the first option, thus we will consider that implementation.

The original `fminsearch` can be written as:

$$X_f = FM(F(X), X_i, \mathcal{C}_1(\epsilon_F) \bigwedge \mathcal{C}_2(\epsilon_X))$$

where X_i is the initial (e.g. random) solution, X_f is the returned solution and $F(X) : \mathbb{R}^{n_u \times n_y} \rightarrow \mathbb{R}$ is the cost function and $\mathcal{C}_1(\epsilon_F) \bigwedge \mathcal{C}_2(\epsilon_X)$ is the original stopping criterion of `fminsearch` with ($\bigwedge$ = `and`, $\bigvee$ = `or`):

$\mathcal{C}_1(\epsilon_F)$: The difference between the worst and the best objective value in the set $F(simplex)$ is less than ϵ_F.

$\mathcal{C}_2(\epsilon_X)$: The maximum coordinate difference between the best X and the other Xs in the simplex is less than ϵ_X.

It is also possible to define a maximum number of iterations and functions evaluations.

Modification of the algorithm

We have modified this implementation to overcome its main shortcomings. The new implementation is defined as:

$X_f(1) = FM(F(X), X_i, \mathcal{C}_{1r}(\epsilon_r) \bigvee \mathcal{C}_3(\epsilon_r, N))$
$i = 2$; $acc = 1$ ($> \epsilon_s$)
while $acc > \epsilon_s$ **do**
  $X_f(i) = FM(F(X), X_f(i-1), \mathcal{C}_{1rel}(\epsilon_r) \bigvee \mathcal{C}_3(\epsilon_r, N))$
  $i = i+1$; $acc = abs(abs(F(X_f(i-1))/F(X_f(i))) - 1)$
end while
return $X_f(i-1)$

The differences between the original implementation and this one are the successive restart(s) and the stopping and restarting criteria. The simplex modification strategy within `fminsearch` remains the same. The restarting criterion is $\mathcal{C}_{1rel}(\epsilon_r) \bigvee \mathcal{C}_3(\epsilon_r, N))$, where:

$\mathcal{C}_{1rel}(\epsilon_r)$: The same as $\mathcal{C}_1(\epsilon_F)$ but with relative accuracy ϵ_r and not absolute tolerance ϵ_F.

$\mathcal{C}_3(\epsilon_r, N)$: The relative difference between the current best objective and the best objective N iterations earlier is less than ϵ_r.

The stopping criteria (no further restart) is the moment when the objective obtained at the last optimization is not better than the previous one, to a relative accuracy $acc \leq \epsilon_s$.

These two important modifications are justified here.

The first essential modification is the use of local restarts of the algorithm: it is restarted from a new simplex generated around the last solution, as long as no improvements are obtained under a given (small) accuracy ϵ_s. This idea is not new: In [17] a provably convergent implementation of the NM algorithm is introduced where an additional step is used to regenerate degenerate simplexes, In [15] this mending is done by simply restarting the algorithm locally which reinitializes the simplex. In practice, this simple strategy allows to deal with the nonsmoothnesses that might cause DS methods to fail, as criticized in [2]. Maybe it could still be possible that NM 'stagnates' at 'dead points' or non-stationary solutions or 'partial solutions' (depending on the authors), but using these restarts makes it much more unlikely. Anyway, since the best we can expect is local convergence, it is clear that such algorithm must be used from several starting points (or 'global' restarts) in order to compare the different local optima obtained and choose the best (and perhaps keep looking if not enough instances of the best one have been found). By doing so, the probability of failure of convergence to a stationary point decreases further. Similarly, this idea is used in [15] where X belongs to a bounded set and the probability of convergence to a global optimum increases with the number of global restarts.

The second modification is the replacement of $\mathcal{C}_1(\epsilon_F) \bigwedge \mathcal{C}_2(\epsilon_X)$ by $\mathcal{C}_{1r}(\epsilon_r) \bigvee \mathcal{C}_3(\epsilon_r, N))$. Whereas the aim of restarting the algorithm is to avoid most of the nonstationnary solutions, the aim of this second modification is to shorten the computation time. We have noticed that the original criterion often lengthen these times uselessly. Indeed we often observed that, although an optimum has very probably been reached, a lot of time is wasted to reduce the size of the simplex to the required tolerances ϵ_F and ϵ_X. This is in particular true for problems with larger values of $F(X)$ and X because the tolerances are absolute and not relative (a mistake in our opinion). We design the criterion $\mathcal{C}_3$ that will trigger earlier in order to partly remove the often useless period where actually only the simplex size is reduced and not the objective. We leave N iterations for the algorithm to try to find an objective improvement bigger than ϵ_r, if not the algorithm is restarted if the stopping criterion is not met. We keep the condition $\mathcal{C}_1$ but normalize it to the current best objective to make it more case-independent, and use `or` instead of `and` still to shorten the computation time. We used as default values $\epsilon_s = \epsilon_r = 10^{-4}$, $N = 100$, chosen 'by hand' (this criterion makes anyway more sense than the arbitrary choice of a maximum number of iterations and/or functions evaluations of the original implementation). The values of ϵ_s, N giving the optimal trade-off between best optimum against shortest computational time differ anyway for each case.

Further modifications can be considered, but the version proposed here works well enough for our purposes. Indeed, it will already appear than this version of NM has similar performances than those of HIFOO.

IV. Stabilization

The well-known problem of SOF stabilization is to find K such that the closed loop system Σ_c given by

$$\Sigma_c : \dot{x} = (A + BKC)x = \mathbf{A}x$$

is stable, i.e. with poles in the open left-half plane. This means finding K such that the spectral abscissa (the maximum real part of the eigenvalues) of $A + BKC$ is negative. It is obtained in MATLAB with the command `max(real(eig(A+BKC)))`.

Unlike the static state feedback stabilization problem which admits a convex representation, this problem is still open and conjectured NP-hard [7], [20]. When it admits a feasible set, it can be non-convex as well as not connected (see [4] and references therein).

The models (A, B, C) used for the benchmarks are the unstable ones of the COMPLib database [14], which regroups many examples from experimental and industrial applications as well as academic examples. The stable models are not considered since K=0 trivially satisfies the problem. For each test made, both NM and HIFOO were started from the same random initial solutions K. Note that the code of HIFOO was slightly modified to perform one optimization at a time: starting from the given initial solution and not generating any other random initial solution. For each plant 100 tests were performed for which the initial points where chosen as: one K=0, 50 K=`randn`(n_u,n_y), 25 K=`rand`(n_u,n_y), 24 K=-`rand`(n_u,n_y). These MATLAB functions define each entry of the initial random matrices as follows: `randn` uses the normal distribution with mean zero and standard deviation one and `rand` uses the uniform distribution on the unit interval.

The results obtained are given for each plant in Table I[1]. The last column (seconds NM/HIFOO) gives the ratio of the time required for the 100 tests by NM on the time required by HIFOO. The previous column (NM/HIFOO) gives the number of successful stabilizations for the 100 tests by NM/ those by HIFOO.

The success rate is slightly better for NM than HIFOO: 92.3% for NM and 90.6% for HIFOO (for 63*100 tests, without the 'easy' plants HF2D). Moreover NM proved faster: it needed an average of 0.74 times the time necessary for HIFOO for each plant, 0.15 altogether[2]. For the plants HF2D Big, both techniques were very successful and NM was a bit slower: it required an average of 1.4 times the time necessary for HIFOO (but this situation can be reversed, as explained further, using ARPACK instead of LAPACK).

Note that the plants NN3 and REA4 have been removed from this list. These are not SOF stabilizable, as easily checked by plotting the spectral abscissa along the scalar K since these are SISO systems. The plants CM4,5,6$_{IS}$ are not SOF stabilizable and have been removed as well. Thorough 3D plots confirm this since these plants have $n_u * n_y = 2$.

We give some comments on the plants that proved hard(er) to SOF stabilize: AC10,IH,PAS,NN10,NN12,ROC3. The model AC10 of the Boeing 767 was SOF stabilized in 0.07s by NM and 0.4s by HIFOO, but this only worked when starting from the initial point K=0: the other random points all failed ([2] also points out the good result obtained starting from K=0, but does not discuss other initial points). IH was harder to deal with by NM, certainly because it is the problem with the most variables ($n_u * n_y = 110$). We briefly remark that by changing the stopping/restarting criteria of NM and extending the computation times (sometimes significantly) we can get the same success rate as that of HIFOO for this plant. On the other hand, the system PAS also with many variables ($n_u * n_y = 60$) was easy to deal with NM (100% success + much faster) whereas it was much more troublesome for HIFOO. Regarding NN10 and NN12, those are indeed hard considering they were deemed unfeasible in [10]. Though it does not appear on this table, further testing made HIFOO as successful as NM on NN10, i.e. less than 10% success. Finally, the plant ROC3 has not been stabilized with NM and with difficulty by HIFOO. On this we mention that using the multidirectionnal search [21], [11] along with NM (both DS methods) allows to SOF stabilize ROC3. This last possibility is too vast to be discussed here and is currently studied. Note that in [14], the plants ROC (for reduced order) are said not to be SOF stabilizable, which is a mistake.

Stabilization of large unstable plants

Table II gives the results obtained with the plants having a large number of states: $n_x \in [2025, 4489]$ (5 tests for each). HIFOO was able to SOF stabililize only the two plants with $n_x = 2025$, but not the bigger ones because of memory problems. The limit of NM is higher, it is the maximum size of a matrix whose eigenvalues can be computed (here somewhere in the 8000s). Regarding the computation times, clearly these are much longer than for smaller plants. However we have reduced these by using ARPACK's `eigs` method instead of LAPACK's `eig`. `eigs` is faster on bigger matrices (we found experimentally $n_x > 62$ on random matrices) but proves less robust (occasional numerical errors) than `eig`. The method chosen was to begin with `eigs` and revert to `eig` once an error occurred. The examples in Table II were obtained an average of 3.5 times faster than with only `eig` (especially appreciable seen the large computation times).

The average computation time for the 5 tests is given for each plant in the last column as an indication of the difficulty encountered. This version of NM, using ARPACK and a faster restarting criterion ($\epsilon_r = 10^{-2}$ instead of 10^{-4} and $N = 20$ instead of 100), stabilized faster than HIFOO the two plants HF2D6, HF2D8 (it is not shown here but this is also the case for the plants HF2D Big with $n_x \in [256, 576]$). It appears also that all these large plants were SOF stabilized by NM (but some were not 100% successful) and that the large number of states does not harm the technique's principle.

Considering the conjectured NP-hardness of the problem [7] (which theoretically renders even moderately large problems numerically untractable [20]) and comparing these results to what could be done with Lyapunov-based methods

[1]**The tables have been removed to meet the page number requirement. They are available in the originally submitted version at: `http://arxiv.org/abs/1104.5369v1`**

[2]The computer used is a HP Compaq dc7800©, processor Intel Q9300©, 2.5GHz, 3.48Go RAM, software MATLAB 2007b©

(that can hardly deal with $n_x > 100$ because of the additional $n_x(n_x+1)/2$ variables), this is excellent. Remark that an alternative method could be to reduce the size of the plant, then compute a stabilizing SOF for the reduced plant (as done in the works of [14]) and then use this solution as initial solution for the full size plant. Clearly this would work well in the current scheme but the point here was to show that the full size plants could be dealt with directly.

Minimal spectral abscissa

Tests were also performed of minimizing `max(real(eig(A+BKC)))` (i.e. maximizing the closed-loop decay rate) instead of only stabilizing (i.e. stop when a negative value is reached). The table of results is not shown due to space limitations. We only mention that around 52% of the obtained spectral abscissa were smaller with NM than HIFOO and NM was an average of 5 to 6 times faster than HIFOO. In the end we conclude that for SOF stabilization and minimal spectral abscissa, NM is faster than HIFOO and very slightly more successful.

V. $\mathcal{H}_2$ NORM

The problem of $\mathcal{H}_2$ norm minimization of T_{wz} is to find K such that $||T_{wz}||_2$ is minimal. After a stability check, returning ∞ if unstable, this norm and its gradient are computed using MATLAB's `lyap` (see e.g. [5] for details) or `norm` if an error occurs with `lyap`. The gradient information is used by HIFOO but not by NM. Since this is a true gradient (not like the subgradients that will be used for the $\mathcal{H}_\infty$ norm), HIFOO will perform better at least for the computational time. However it will appear in the results of Table III that NM gets most of the time the same optima as HIFOO, sometimes better and sometimes worse. The plants considered are those of [5] and 30 tests are performed for each plant, like in that paper. Note that there are some plants not satisfying for all D_K the equality constraint $\mathbf{D} = D_{11} + D_{12}D_K D_{21} = 0$ necessary for a finite H2 norm. These could be dealt with using the same technique as that of HIFOO: using singular value decomposition to rewrite this affine constraint in an explicit parametric form (thus with less variables so easier to deal with using DS approaches). Here we simply force $D_{11} = 0$, $D_{12} = 0$ to lighten the benchmarking work. In order to start each $\mathcal{H}_2$ minimization from the same initial solution, the stabilizing controller found by HIFOO is also used as starting point for NM (we slightly modified HIFOO to recover the stabilizing solution). Of course the stabilization computation time has been removed from the total time required by HIFOO. The results are in Table III.

As expected NM is slower than HIFOO, by an average of 10 times (for each plant, 2 times only altogether because HIFOO was very slow on NN16). However 60% of the optima obtained are similar within 1% to each other, 14% found by NM are better and 26% found by HIFOO are better. So it is reasonable to say than both methods compete in term of reached objectives values. Regarding the computationnal time, HIFOO is much better certainly thanks to the quick access to the true gradient of the objective function (a convenient feature that is typically unavailable for many problems). Note that the success rate of NM can be improved by reducing ϵ_r and/or increasing N, which also modifies (normally increases) the total computationnal time.

VI. $\mathcal{H}_\infty$ NORM

The objective considered here is to find K such that $||T_{wz}||_\infty$ is minimal (more details e.g. in [23]). The norm is computed with MATLAB's function `norm` with accuracy 10^{-10} (after a stability check, ∞ is returned when unstable). The results are to be read as in the previous section and are given in Table IV.

Now that only Clarke's subgradients are available (see e.g. [3]), HIFOO becomes much slower. For this benchmark, NM is an average of 1.32 times slower than HIFOO for each plant (without CSE2, for which NM always found a better optimum but was 98 times slower than HIFOO) and altogether NM took 0.72 times the time needed by HIFOO. Considering the optima obtained, 40% of the results obtained are similar within 1% to each other, 23% found by NM are better and 37% found by HIFOO are better.

We remark that we reran these tests on MATLAB 2010a and found about the same results and ratios. Also, for most tests for which the best objective found by NM is close but not identical to the one found by HIFOO, the same objective can be found by reducing ϵ_r, ϵ_s (which of course requires more computational time).

VII. CONCLUSIONS

This paper evaluates the performances of a direct search method, here a restarted Nelder-Mead algorithm, to design optimal Static Output Feedback controllers for LTI systems. Thorough benchmarkings (around 13000 tests) have been made with parallel comparison with one of the most numerically efficient method currently available for these problems: HIFOO. It appears that both methods are similarly successful for the considered problems: stabilization, minimization of $\mathcal{H}_2$ and $\mathcal{H}_\infty$ norms. However the computational times differ. NM is faster than HIFOO for stabilization. HIFOO is clearly faster than NM for the $\mathcal{H}_2$ norm (smooth objective function). And for the $\mathcal{H}_\infty$ norm (Clarke subdifferentiable objective function), the computational times are equivalent. Nevertheless it is quite surprising that the excellent performances of direct search methods for these problems have not been pointed out before, except very recently in [10] for SOF stabilization.

Considering the results in this paper we believe having fulfilled the objective of illustrating that direct search methods should be put back on the grid of competitive methods as a candidate of choice for complex optimal control problems. Although these methods may lead to good computational times, this is not their strong point. Gradient informations become necessary to improve the computational time, and increasingly so for larger number of variables. Their important property is their good ability to explore non-smooth objective functions on non-convex feasible sets. This ability combined

with the flexibility of dealing with any problem formulated as minimizing a function $f(x) : \mathbb{R}^n \longrightarrow \mathbb{R}$ makes such methods very useful to deal with a broad set of optimization problems in systems and control, for which classical approaches will have a hard time finding good solutions and even more so locally optimal ones.

We have seen here that NM can be used alone, but in general it can always be used as a systematic refinement for almost any other method yielding conservative results. Certainly, the idea of hybrid optimization combining DS methods with other techniques (interior point or gradient sampling or almost any method) is a very promising research direction for optimal control design (an idea already considered in [2]). Not to mention that NM has a high practical value for both academics desiring an alternative approach to their problem and for non-expert users needing a method readily available, robust, flexible and easy to use.

In the case where cost functions are quickly evaluated, it is not costly to perform many optimizations. So even if failures -i.e. convergence to non-stationary solutions- may still be possible, this can be rendered increasingly unlikely by using more different initial solutions. Therefore by obtaining more instances of the best solution found so far, one becomes increasingly sure that this solution is at least locally and possibly globally optimal. This is an idea usually encountered within global optimization methods.

The best direct search method should be MADS [6] because it has the strongest convergence properties, but this method is not so easily available as NM and has more parameters and options to choose. Clearly, the multidirectional search (MDS) [21], [11] is also a candidate of choice. Early tests gave results sometimes better with NM, sometimes with MDS. Other tests with Gauss-Newton methods were much less successful but these might probably be adapted (this idea is anyway close to what HIFOO and `hinfstruct` are already doing very well). Certainly, powerful combinations and modifications can be made (hybrid optimization as suggested above). In the literature the genetic algorithm is also sometimes used, which we rather not recommend because of its weak convergence properties and many parameters and options to choose strongly influencing its performance.

To conclude in a few words, direct search methods will be able to bring good to excellent results for many complex problems of optimizations in systems and control. This is thanks to the following facts: -these methods are able to explore non-smooth objective functions on non-convex feasible sets (in particular with MADS) -these problems typically need only a reasonable number of key variables -efficient routines are available for many objective function evaluations -the performance of computers increases exponentially with time. A second layer of work can then be made to create specialized methods for specific problems, ideally using gradient informations similarly to what is done in the excellent HIFOO and `hinfstruct`. These approaches should most often have the upper hand over e.g. iterative LMI algorithms methods to try to solve non-convex problems (see [19] for another example).

ACKNOWLEDGMENTS

The author gratefully acknowledges Vincent Wertz, Christian Ebenbauer and Pierre Apkarian for useful remarks. This research was supported by the Interuniversity Attraction Poles Programme initiated by the Belgian State, Science Policy Office and of the Network DYSCO (Dynamical Systems, Control, and Optimization).

REFERENCES

[1] D. Ankelhed, A. Helmersson and A. Hansson "A Partially Augmented Lagrangian Method for Low Order $\mathcal{H}_\infty$ Controller Synthesis using Rational Constraints," in *50th CDC-ECC*, Orlando, Dec. 2011.

[2] P. Apkarian and D. Noll, "Controller design via nonsmooth multidirectional search," *SIAM J. Control Optim.*, vol.44, 1923–1949, 2006.

[3] P. Apkarian, D. Noll and A. Rondepierre, "Mixed $\mathcal{H}_2/\mathcal{H}_\infty$ control via nonsmooth optimization," *SIAM journal of control and optimization*, vol. 47, no. 3, pp. 1516–1546, 2008.

[4] D. Arzelier, E. N. Gryazina, D. Peaucelle and B. T. Polyak, "Mixed LMI/randomized methods for static output feedback control design," in *American Control Conference*, June 2010.

[5] D. Arzelier, G. Deaconu, S. Gumussoy and D. Henrion, "$\mathcal{H}_2$ for Hifoo," in `http://arxiv.org/abs/1010.1442`

[6] C. Audet and J. Dennis, "Mesh adaptative direct search algorithms for constrained optimization," *Siam Journal of Optimization*, vol. 17, no. 1, pp. 188–217, 2006.

[7] V. Blondel and J. Tsitsiklis, "NP-hardness of some linear control design problems," *SIAM journal on control and optimization*, vol. 35, no. 6, pp. 2218–2127, 1997.

[8] L. El Ghaoui, F. Oustry and M. AitRami "A cone complementary linearization algorithm for static output-feedback and related problems" *IEEE Trans. on Automatic Control*, vol. 42, no. 8, pp. 1171–1176, 1997.

[9] S. Gumussoy, D. Henrion, M. Millstone and M. Overton, "Multiobjective robust control with Hifoo 2.0," in *Proceedings of the IFAC Symposium on Robust Control Design*, 2009.

[10] D. Henrion, "Solving static output feedback problems by direct search optimization," in *CCA, Munich*, 2006.

[11] N. J. Higham, "The Matrix Computation Toolbox," `http://www.ma.man.ac.uk/~higham/mctoolbox`.

[12] T. Iwasaki "The dual iteration for fixed-order control" *IEEE Trans. on Automatic Control*, vol. 44, no. 4, pp. 783–788, 1999.

[13] J. Lagarias, J. A. Reeds, M. H. Wright and P. E. Wright, "Convergence properties of the Nelder-Mead simplex method in low dimensions," *SIAM Journal of Optimization*, vol. 9, no. 1, pp. 112–147, 1998.

[14] F. Leibfritz, "Complib: Constraint matrix-optimization problem library - a collection of test examples for nonlinear semidefinite programs, control system design and related problems." University of Trier, Tech. Rep., 2004, `http://www.complib.de/`.

[15] M. Luersen and R. L. Riche, "Globalized Nelder-Mead method for engineer optimization," *Computers and structures*, vol. 82, no. (23-26), pp. 2251–2260, 2004.

[16] J. Nelder and R. Mead, "The downhill simplex method," *Computer journal*, vol. 7, pp. 308–313, 1965.

[17] C. Price, I. Coope and D. Byatt, "A convergent variant of the Nelder-Mead algorithm," *Journal of optimization theory and applications*, vol. 113, no. 1, pp. 5–19, 2002.

[18] E. Simon, P. R.-Ayerbe and C. Stoica and D. Dumur and V. Wertz, "LMIs-based coordinate descent method for solving BMIs in control design," in *18th IFAC World Congress*, August 2011. `http://hdl.handle.net/2078.1/69840`

[19] E. Simon and V. Wertz, "Direct search methods for an open problem of optimization in systems and control," submitted `http://arxiv.org/abs/1104.5183`

[20] V.L. Syrmos, C. Abdallah, P. Dorato and K. Grigoriadis, "Static output feedback: A survey," *Automatica* vol. 33, pp. 125–137, 1997.

[21] V. Torczon, "Multidirectional search: A direct search algorithm for parallel machines," Ph.D. dissertation, Rice University, 1989.

[22] M. H. Wright "Direct search methods: once scorned, now respectable" *Numerical Analysis*, vol. 344, pp. 191–208, 1995.

[23] K. Zhou, J. C. Doyle and K. Glover, *Robust and Optimal Control*. Prentice Hall, 1996.