A continuation like Algorithm for Static Output Feedback Stabilization

P.O. Shirley\textsuperscript{a*}, J.M. Lemos\textsuperscript{b}

\textsuperscript{a}INESC-ID/Universidade Aberta, R. Escola Politécnica 147, 1269-001 LISBOA, Portugal

\textsuperscript{b}INESC-ID/Instituto Superior Técnico, R. Alves Redol 9, 1000-029 LISBOA, Portugal

An algorithm to find a stabilizing controller with any order and structure for continuous linear time invariant systems is presented. The control framework is based on generalized systems, numerical minimization of a modified $H_2$ cost functional and a static output feedback controller. The underlying strategy consists of solving a series of optimization problems on the original system with its closed loop pole map shifted to the left half complex plane by an amount specified by the algorithm, until a stabilizing controller is found for the original, non shifted system.

Keywords: Optimal control; $H_2$; Numerical methods; Static controllers.

1. Introduction

As computing power is becoming more and more available to desktop personal computers, numerical optimization based methods for controller synthesis are becoming more attractive and viewed as a viable alternative of current usage. An advantage of these methods is the extra flexibility in specifying the control objectives and the structure of the desired controller, since they are not constrained by the type of optimal solution attached to the problem formulation, such as full order controllers in LQG problems. A problem shared by all numerical based methods \cite{4} is that they require a stabilizing controller as a starting point to the numerical optimization procedure. Such an initial controller can be easy to find in simple problems, but can be difficult or very time consuming for moderate and complex problems. This paper presents an algorithm that aims to contribute to solve this problem, providing a full automated procedure to find a stabilizing custom controller. As the procedure is based on the $H_2$ cost functional, its natural outcome is a local optimal $H_2$ controller with the specified order and structure.

The structure of the paper is as follows. Section 2 presents the $H_2$ control setup used. Section 3 presents the algorithm and its inner workings. Section 4 shows through an example the application of the method and section 5 presents the conclusions.

2. The Control Setup

Consider the control setup shown in Figure 1, where $w \in \mathbb{R}^{n_w}$ is a disturbance input (such as tracking signals, measurable and non measurable disturbances), $u \in \mathbb{R}^{n_u}$ is the control input, $z \in \mathbb{R}^{n_z}$ is the controlled output (such as error and control signals) and $y \in \mathbb{R}^{n_y}$ is the measured output. The generalized plant $G_0$ includes the system to be controlled and the additional elements to generate the signals involved. It is assumed that $G_0$ does not have direct terms from $w$ to $z$ and $y$. The $F$ block is a matrix gain and implements the Static Output Feedback (SOF) \cite{3}.
control law \( u = Fy \). This law is not a limitation since a fixed order control problem can be recasted as a SOF control problem [2].

The generalized system \( G_0 \) admits the following state space representation
\[
G_0 \equiv \begin{cases} \dot{x} = A x + B_1 w + B_2 u \\ y = C_1 x + D_{12} u \end{cases}
\] (1)

The closed loop system when \( u = Fy \) is given by
\[
G_0(F) \equiv \begin{cases} \dot{x} = A_F x + B_1 w \\ y = C_1 x \\ z = C_2 x \end{cases}
\] (2)

\( A_F = A + B_2 FC_2, \quad C_1 F = C_1 + D_{12} FC_2 \)

Define the matrices \( W = \sum_{i=1}^{n_w} w_i w_i' = W' \geq 0 \), \( Q = Q' \geq 0 \), \( \delta w_i \) as \( w_i \delta(t) \), and the vector norm
\[
\|v(t)\|_{2Q_\alpha}^2 = \int_0^{+\infty} e^{-2\alpha \tau} v'(\tau) Q v(\tau) d\tau
\]
with \( \alpha \geq 0 \) a parameter. The cost functional is the \( H_2 \) type norm
\[
J(F, \alpha) = \sum_{i=1}^{n_w} \|G_0(F) \delta w_i\|_{2Q_\alpha}^2 = \text{tr}(B_1' X B_1 W)
\] (3)
\[= \text{tr}(C_1 F Y C_1' F Q) \] (4)

where \( \text{tr}(\cdot) \) is the trace operator and the matrices \( X \) and \( Y \) are the solutions of the “shifted” continuous time algebraic Lyapunov equations
\[
(A_F - \alpha I)'X + X(A_F - \alpha I) + C_1' F Q C_1 F = 0
\]
\[
(A_F - \alpha I) Y + Y(A_F - \alpha I)' + B_1 W B_1' = 0
\]
The \( \alpha \) parameter implements a \( H_2 \) “\( \alpha \)-shifted” norm. If a system has a transfer function \( G(s) \), its \( H_2 \) \( \alpha \)-shifted norm \( \|G(s)\|_{2\alpha} \) is related with the standard \( H_2 \) norm by
\[
\|G(s)\|_{2\alpha} = \|G(s + \alpha)\|_2
\] (5)
that is, the pole-zero map of \( G(s + \alpha) \) in the complex plane is equal to the one of \( G(s) \) but shifted to the left or to the right by an amount \( |\alpha| \), respectively as \( \alpha > 0 \) or \( \alpha < 0 \). For \( \alpha = 0 \), the cost functional (3) correspond to the standard \( H_2 \) norm. The \( \alpha \) parameter provides a tool to deal with unstable systems. If an initial controller \( F_0 \) is not stabilizing, the idea is to start with a stabilizing \( \alpha \) and manipulate it while the numerical optimization proceeds in order to reach \( \alpha = 0 \). The next section describes this strategy in detail.

For solving optimization problems with \( J(F, \alpha) \) with numerical techniques, the first order derivatives with respect to \( F \), in matricial form, are given by [5]
\[
\frac{\partial J(F, \alpha)}{\partial F} = 2(D_{12} Q C_1 F + B_2' X Y C_2') = 0
\] (6)

3. The Algorithm

Let \( \theta \) be a vector that collects the relevant coefficients in \( F \), that is, the variables to be optimized. Then, the closed loop system (2) writes as \( G(\theta) \), its system matrix as \( A(\theta) \) and the cost functional (3) as \( J(\theta, \alpha) \). For each value of \( \alpha \), define the optimization problem,
\[
\theta^*(\alpha) = \arg \min_{\theta} J(\theta, \alpha)
\] (7)
The stabilization problem consists of finding a vector \( \theta \), if one exists, such that \( G(\theta) \) is stable, that is, \( \lambda[A(\theta)] < 0 \), where \( \lambda(\cdot) \) represents the maximum real part of the eigenvalues. Consider the following definitions,

**Definition 1** Define \( S_0(\alpha) \) as the set of all values of \( \theta \) that stabilizes the matrix \( A(\theta) - \alpha I \),
\[
S_0(\alpha) = \{ \theta : \lambda[A(\theta) - \alpha I] < 0 \}
\] (8)

**Definition 2** Define \( S_{(\theta, \alpha)} \) as the set of all pairs \((\theta, \alpha)\) for \( \alpha \geq 0 \) such that \( \theta \in S_0(\alpha) \),
\[
S_{(\theta, \alpha)} = \{ (\theta, \alpha) : \alpha \geq 0 \land \theta \in S_0(\alpha) \}
\] (9)

and the following implications,
\[
\alpha_1 > \alpha_2 \Rightarrow \lambda[A(\theta) - \alpha_1 I] < \lambda[A(\theta) - \alpha_2 I]
\] (10)
\[
\alpha_1 > \alpha_2 \Rightarrow S_{\theta(\alpha_2)} \subseteq S_{\theta(\alpha_1)}
\] (11)
Suppose one wants to solve the optimization problem (8) with \( \alpha = 0 \) and a stabilizing value for \( \theta \) is not available, so one has to start the optimization from a non stabilizing initial point...
\( \theta_0 \notin S_0(0) \). Let \( \alpha_0 > 0 \) be such that \( \theta_0 \in S_0(\alpha_0) \) and suppose that exists \( \theta \in S_0(0) \). Then the line parameterized by \( \alpha \), defined by \((\theta(\alpha), \alpha)\) for \( \theta(\alpha) = \theta \) and \( 0 \leq \alpha \leq \alpha_0 \) belongs to \( S_{(\theta, \alpha)} \), that is, there exists at least one continuous line in \( S_{(\theta, \alpha)} \) parameterized by \( \alpha \) between the stability regions \((S_0(\alpha_0), \alpha_0)\) and \((S_0(0), 0)\). This property is a direct consequence of (12), and means that \( S_0(\alpha) \) is a non decreasing hipervolume with \( \alpha \). To solve the stabilization problem, one tries to find a trajectory \((\theta^*(\alpha), \alpha)\) parameterized by \( \alpha \) between \((S_0(\alpha_0), \alpha_0)\) and \((S_0(0), 0)\) based on (8). The main difficulty with this strategy is that for some intermediate value \( 0 < \tilde{\alpha} < \alpha_0 \), the set \( S_0(\tilde{\alpha}) \) can not be connected, resulting from the union of several disjoint subsets,

\[
S_0(\tilde{\alpha}) = \bigcup_i S^i_0(\tilde{\alpha})
\]

separated by instability regions. Then, it could not exist a continuous line in \( S_{(\theta, \alpha)} \) for \( 0 \leq \alpha \leq \tilde{\alpha} \) between \((S^1_0(\tilde{\alpha}), \tilde{\alpha})\) and \((S_0(0), 0)\) for some of the subsets \( S^i_0(\tilde{\alpha}) \). Figure 2 shows this concepts for an hypothetical example with a scalar \( \theta \). The \( S_{(\theta, \alpha)} \) region is delimited by the function

\[
\max(0, \tilde{\lambda}[A(\theta)])
\]

agreeing with (12). The set \( S_0(\tilde{\alpha}) \) is given by the union of 3 subsets. In each subset \( S^i_0(\tilde{\alpha}) \) there exists at least one local minimum of \( J(\theta, \tilde{\alpha}) \), as each subset is bounded by an instability region or extends itself to the infinite. These local minimums guarantee at least the existence of the 3 trajectories \((\theta^{i*}(\alpha), \alpha)\) indicated with dash lines. These could extend themselves for greater values of \( \alpha \) or extinguish. Note that in this example it does not exists a continuous line in \( S_{(\theta, \alpha)} \) for \( 0 \leq \alpha \leq \tilde{\alpha} \) between \((S^1_0(\tilde{\alpha}), \tilde{\alpha})\) and \((S_0(0), 0)\). If in the optimization problem (8) one only requires a local minimum and this happens to be found in \( S^1_0(\tilde{\alpha}) \), then the trajectory defined by \((\theta^{i*}(\alpha), \alpha)\) for decreasing values of \( \alpha \) will not end in \((S_0(0), 0)\), but in a point \((\theta^{i*}(\alpha^*), \alpha^*)\) with \( \alpha^* > 0 \) such that \( \theta^{i*}(\alpha^*) \) is a local minimum of \( \tilde{\lambda}[A(\theta)] \). If on the other end, one requires a global minimum for (8), then one can guarantee that the path defined by \((\theta^*(\alpha), \alpha)\) will end in \((S_0(0), 0)\), because jumps between the sets \( S_0^i(\tilde{\alpha}) \) can occur, depending on the localization of the global minimum. However, finding a global minimum is only possible for very simple cases, since \( S_0(\alpha) \) is unknown and probably unbounded.

The algorithm presented in this section employs only local minimums of (8), establishing a path \((\theta^*(\alpha), \alpha)\) between \((S_0(\alpha_0), \alpha_0)\) and \((S_0(0), 0)\), possibly using values \( \alpha > \alpha_0 \), through the “net” of trajectories constituted by all local minimums of (8) in \( S_{(\theta, \alpha)} \) for \( \alpha \geq 0 \).

The algorithm is heuristic in its nature and goes through the branch \((\theta^*(\alpha), \alpha)\) starting from the point \((\theta^*(\alpha_0), \alpha_0)\) in the decreasing direction. If convergence is detected at a point \((\theta^*(\alpha^*), \alpha^*)\) with \( \alpha^* > 0 \) the algorithm reverses the direction of the trajectory going trough the same branch but in the increasing direction of \( \alpha \), until the branch extinguish and a transition to another branch is detected. The direction is changed to decreasing again and verified if the new branch ends in \((S_0(0), 0)\) and so on. With this kind of algorithm, the solution of (8) for \( \alpha = 0 \) is just the last step and is included in the algorithm.

The following function \( \phi(\lambda, \varepsilon) \) is most useful for the algorithm implementation,

\[
\phi(\lambda, \varepsilon) = \begin{cases} 
0 & \lambda < 0 \\
\lambda + \varepsilon & \lambda \geq 0
\end{cases}
\]

If \( \theta \) is a non stabilizing vector, then

\[
\alpha = \phi(\tilde{\lambda}[A(\theta)], \varepsilon) \Rightarrow \tilde{\lambda}[A(\theta)] - \alpha I = -\varepsilon
\]

which means that \( \varepsilon \) is the distance that \( \tilde{\lambda}[A(\theta)] - \alpha I \) is placed from the imaginary axis by the action of \( \alpha \) and can be seen as a stability margin. If \( \theta \) is a stabilizing vector, one gets \( \alpha = 0 \). Figure 4 shows the algorithm in pseudo-code. The
DOWN mode is associated in going through a branch \((\theta^*(\alpha), \alpha)\) in the decreasing direction of \(\alpha\) and the UP mode in the increasing direction of \(\alpha\). To go through a branch, one alternates between the computation of \(\alpha_n\) and the optimization with respect to \(\theta\),

\[
\alpha_{n+1} = \arg \min_{\theta} J(\theta, \alpha_n)
\]

using \(\theta_n\) as a starting point, since \(J(\theta_n, \alpha_n) < +\infty\) by construction. Depending how the values of \(\{\varepsilon_n\}\) are chosen, one can go down or up through a branch. Figure 3 shows the values of \(\varepsilon\) to be implemented.

As one only goes up through a branch after having gone down, the initial value of \(\alpha\) is stored in \(\alpha_s\) in order to be reused as a starting point for going up if needed.

Last but not least, the algorithm relies in the detection of two fundamental conditions. For the DOWN mode, convergence detection of \(\lambda_n\) can be implemented by the test \(\{\lambda_n - \lambda_{n-1} \leq \beta_1 \land \|\theta_n - \theta_{n-1}\| < \beta_2\}\). Practical values are \(\beta_1 \approx 10\varepsilon_{\text{min}}\) and \(\beta_2 \in [10^{-8} ; 10^{-6}]\). For the UP mode, it is not clear how to detect a branch transition. The option made is to detect \(\lambda_n < \lambda_{n-1}\), which is interpreted as passing by an extreme of \(\lambda[A(\theta)]\), that is, a jump from a set \(S^b_0(\alpha)\) to another (see also figure 2).

**Algorithm**

1. Given \(\theta_0\), choose \(a > 1\), \(0 < \varepsilon_{\text{min}} < 1\), \(0 < \beta < \varepsilon_{\text{min}}\), \(\alpha_{\text{max}}\)
2. \(mode = \text{DOWN}, \varepsilon_0 = \varepsilon_{\text{min}}, \lambda_0 = \lambda[A(\theta_0)]\), \(\alpha_0 = \phi(\lambda_0, \varepsilon_0), \alpha_s = \alpha_0, n = 1\)
3. Solve the optimization problem \(\theta_n = \arg \min_{\theta} J(\theta, \alpha_{n-1})\) using \(\theta_{n-1}\) as a starting point
4. \(\lambda_n = \lambda[A(\theta_n)]\)
5. If \(\lambda_n \geq 0\)
   - If \(mode = \text{DOWN}\)
     - If \(\lambda_n\) converges
       - \(mode = \text{UP}, \varepsilon_n = a \cdot (\alpha_s - \lambda_n)\)
     - else
       - \(\varepsilon_n = \varepsilon_{\text{min}}\)
   - else
     - If branch transition occurs
       - \(mode = \text{DOWN}, \alpha_s = \alpha_{n-1}, \varepsilon_n = \varepsilon_{\text{min}}\)
     - else
       - \(\varepsilon_n = a \cdot (\alpha_{n-1} - \lambda_n)\)
   - else
     - \(\varepsilon_n = 0\)
6. \(\alpha_n = \phi(\lambda_n, \varepsilon_n)\)
7. If \(|\alpha_n - \alpha_{n-1}| \leq \beta \lor \alpha_n > \alpha_{\text{max}}\), finnish, else \(n = n + 1\), goto step 3

Figure 4. Algorithm for solving the optimization problem (8) with \(\alpha = 0\) starting from any initial point, stabilizing or not.
4. Example

This example shows the application of the algorithm described in the previous section for the manipulation of the $\alpha$ parameter in order to solve the stabilization problem.

Consider the system described by the block diagram of figure 5, where $P$ is defined by the transfer function,

$$P(s) = \frac{(s - 2)^2 + 5^2}{(s - 12)^3}$$

(18)

The objective is to tune the $f$ parameter in order to minimize the system impulse response given by the cost functional

$$J(f, \alpha) = \int_{0}^{+\infty} e^{-2\alpha \tau} [c_1^2(\tau, f) + \rho u_0^2(\tau, f)] d\tau$$

(19)

The optimal value of $f$ is given by the minimization problem

$$f^*(\alpha) = \arg \min_{f} J(f, \alpha)$$

(20)

for $\alpha = 0$. The control setup of figure 1 is given with $F = f$ and the generalized system $G_0$ is obtained reorganizing the block diagram of figure 5 into the one in figure 6, where $u = \delta_r$, $u = u_p$, $z = [e_r \ u_p]'$ and $y = e_r$. The cost functional (19) is defined with $Q = \begin{bmatrix} 1 & 0 \\ 0 & \rho \end{bmatrix}$, $\rho = 10^{-6}$ and $W = 1$ in (3).

Figure 7 shows the root-locus for the closed loop of $P$ with $f$, which is stabilizable for $f \in [59.59; 103.29]$. In order to illustrate the behavior of the algorithm, the non-stabilizing initial value $f = 0.1$ is chosen. Figure 8 is the analog of figure 2, showing the stabilizing region $S(f, \alpha)$, which is delimited by the curve $\alpha(f) = \max(0, \lambda(A(f)))$, where $A(f)$ is the dynamic system matrix of the closed loop system. For this example, associated with the minimums of $J(f, \alpha)$ are two trajectories ($f^{1*}(\alpha), \alpha$) and ($f^{2*}(\alpha), \alpha$), where the former extinguish for $\alpha \geq 16.68$. Figure 9 shows the evolution of the algorithm in the $(f, \alpha)$ plane. The optimization is started from the initial point $f_0 = 0.1$, marked as the point A. An horizontal movement corresponds to the optimization of $J(f, \alpha)$ with respect to $f$ with constant $\alpha$. A vertical movement corresponds to a change in $\alpha$ as computed by the algorithm. The algorithm starts to go down the branch ($f^{1*}(\alpha), \alpha$) until convergence is detected at point B, which is not a stabilizing point. The algorithm proceeds on the same branch but in the up direction starting on point C with a value of $\alpha$ greater than in point A, which was memorized. In point D a branch transition is detected and a new descent is started, now in the branch ($f^{2*}(\alpha), \alpha$), which leads to the desired stabilizing
solution $(f^2, 0)$ marked by point E. Figure 10 shows details of the final steps of the optimization procedure. The final point value is $f^* = 103.066$.

Figure 10. Zoom of the final steps of the optimization.

Figure 11 shows the evolution of the cost functional $J(f, \alpha)$ and of the parameter $\alpha$, where the index $n$ means a line search performed by the numeric minimization algorithm BFGS [1]. Note that a reduction in the $\alpha$ parameter causes an increase of the cost functional (less exponential weight) and that an increase in the $\alpha$ parameter or a minimization causes a decrease in the cost functional.

A minimization of the cost functional with constant $\alpha$ requires several numeric iterations, however, while $\alpha > 0$ an exact minimization is not necessary, since the objective is only to be able to reduce the value of $\alpha$.

5. Conclusions

Numerical optimization based methods for controller synthesis need a stabilizing controller as a starting point. An algorithm to find such a stabilizing controller has been presented. Together with a method for reduced order controllers synthesis, the developed methodology allows to find a stabilizing controller with any order and structure without requiring any initial guess about the controller parameters. The procedure can be fully automated leaving only to the user the task to specify the system to be controlled and the desired type of controller.

As is clear from the given example, most of the computational effort is spent in finding a stabilizing controller. However, in a project of an optimal controller, this type of effort is typically made only once. Final optimizations of the controller can be made with relatively small effort. In a complex problem where the knowledge about the system, the intuition or the classical linear control synthesis methods are all unable to find an initial stabilizing custom controller, this algorithm reveals itself most useful. Experiments made by the authors with more complex systems, including multivariable ones, confirm these assertions.

REFERENCES