algorithms

MDPI

Article

On Application of the Ray-Shooting Method for LQR via Static-Output-Feedback

Yossi Peretz

Department of Computer Sciences, Lev Academic Center, Jerusalem College of Technology, P.O. Box 16031, 93721 Jerusalem, Israel; yosip@g.jct.ac.il; Tel.: +972-2-675-1016; Fax: +972-2-675-1046

Received: 29 October 2017; Accepted: 4 January 2018; Published: 16 January 2018

Abstract: In this article we suggest a randomized algorithm for the LQR (Linear Quadratic Regulator) optimal-control problem via static-output-feedback. The suggested algorithm is based on the recently introduced randomized optimization method called the Ray-Shooting Method that efficiently solves the global minimization problem of continuous functions over compact non-convex unconnected regions. The algorithm presented here is a randomized algorithm with a proof of convergence in probability. Its practical implementation has good performance in terms of the quality of controllers obtained and the percentage of success.

Keywords: optimal control; state-space models; randomized algorithms; continuous-time systems

1. Introduction

Let a continuous-time system be given by

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) \end{cases} \tag{1}$$

where $A \in \mathbb{R}^{p\times p}, B \in \mathbb{R}^{p\times q}$, $C \in \mathbb{R}^{r\times p}$ and x, u, y are the state, the input and the measurement, respectively. We assume that (A, B) and $\left(A^T, C^T\right)$ are controllable (see [1] for a new reduction from the case where (A, B) and $\left(A^T, C^T\right)$ are only stabilizable). Let

$$J(x_0, u) := \int_0^\infty \left(x(t)^T Qx(t) + u(t)^T Ru(t)\right) dt, \tag{2}$$

denote the cost functional, where $Q > 0$ and $R \geq 0$. Assuming that $x_0 = x(0)$ is given, the LQR problem is to attenuate the disturbance x_0 using minimal control cost, i.e., to design a regulation input $u(t)$ that minimizes $J(x_0, u)$, subject to the constraints given by (1). Let $u = -Ky$ be the static-output-feedback (SOF) with the closed-loop matrix $A_{c\ell}(K) := A - BKC$. Let $\mathbb{C}_-$ denote the left-half plane, let $\alpha > 0$ and let $\mathbb{C}_\alpha$ denote the set of all $z \in \mathbb{C}$ with $\Re(z) \leq -\alpha$, where $\Re(z)$ is the real part of z. Let $\mathcal{S}^{q\times r}$ denote the set of all matrices $K \in \mathbb{R}^{q\times r}$, such that $A_{c\ell}$ is stable, i.e., $\sigma(A_{c\ell}) \subset \mathbb{C}_-$ (where $\sigma(A_{c\ell})$ is the spectrum of $A_{c\ell}$). By $\mathcal{S}_\alpha^{q\times r}$, we denote the set of all matrices $K \in \mathbb{R}^{q\times r}$, such that $\sigma(A_{c\ell}) \subset \mathbb{C}_\alpha$. In this case, we say that $A_{c\ell}$ is α-stable. Below, we will occasionally write $\mathcal{S}_\alpha$ instead of $\mathcal{S}_\alpha^{q\times r}$, when it is clear what the size of the related matrices is.

Let $K \in \mathcal{S}_\alpha^{q\times r}$ be given. Substitution of $u = -Ky = -KCx$ into (2) gives

$$J(x_0, K) := \int_0^\infty x(t)^T \left(Q + C^T K^T RKC\right) x(t)\, dt. \tag{3}$$

Since $Q + C^T K^T RKC > 0$ and since $A_{c\ell}(K)$ is stable, it follows that the Lyapunov equation

$$A_{c\ell}(K)^T P + P A_{c\ell}(K) = -\left(Q + C^T K^T R K C\right) \tag{4}$$

has a unique solution $P > 0$, given by

$$P = -mat\left(\left(I_p \otimes A_{c\ell}(K)^T + A_{c\ell}(K)^T \otimes I_p\right)^{-1} \cdot vec\left(Q + C^T K^T R K C\right)\right), \tag{5}$$

where *vec* puts all the columns of the given matrix in a single column and *mat* is the inverse of *vec*. Let us denote the solution (5) as $P(K)$. Substitution of (4) into (3) and noting that $\dot{x}(t) = A_{c\ell}(K)\, x(t)$ with $A_{c\ell}(K)$ stable, leads to

$$\begin{aligned} J(x_0, K) &= -\int_0^\infty x(t)^T \left(A_{c\ell}(K)^T P(K) + P(K) A_{c\ell}(K)\right) x(t)\, dt \\ &= -\int_0^\infty \frac{d}{dt}\left(x(t)^T P(K)\, x(t)\right) dt = x_0^T P(K)\, x_0. \end{aligned}$$

Thus, we look for $K \in \mathcal{S}_\alpha^{q\times r}$ that minimizes the functional $J(x_0, K) = x_0^T P(K)\, x_0$. When x_0 is unknown, we seek $K \in \mathcal{S}_\alpha^{q\times r}$ for which

$$\sigma_{max}(K) := \max\left(\sigma\left(P(K)\right)\right) \tag{6}$$

is minimal. In this case, we get a robust LQR via SOF, in the sense that it minimizes $J(x_0, K)$ for the worst possible (unknown) x_0. Note that

$$x_0^T P(K)\, x_0 = \left\|P(K)^{\frac{1}{2}} x_0\right\|^2 \le \left\|P(K)^{\frac{1}{2}}\right\|^2 \|x_0\|^2 = \|P(K)\|\, \|x_0\|^2 = \sigma_{max}(K)\, \|x_0\|^2,$$

and that there exists $x_0 \neq 0$ for which equality holds. Therefore $\frac{J(x_0,K)}{\|x_0\|^2} \le \sigma_{max}(K)$, where equality holds in the worst case.

Note that the functionals $J(x_0, K)$ and $\sigma_{max}(K)$ are generally not convex since their domain of definition $\mathcal{S}^{q\times r}$ (and therefore $\mathcal{S}_\alpha^{q\times r}$) is generally non-convex. Necessary conditions for optimality were given as three quadratic matrix equations in [2–5]. Necessary and sufficient conditions for optimality, based on linear matrix inequalities (LMI), were given in [6–8]. However, algorithms based on these formulations are generally not guaranteed to converge, seemingly because of the non-convexity of the coupled matrix equations or inequalities, and when they converge, it is to a local optimum only.

The application of SOFs in LQRs is appealing for several reasons: they are reliable and cheap, and their implementation is simple and direct. Moreover, the long-term memory of dynamic-feedbacks is useless for systems subject to random disturbances, to fast dynamic loadings or to impulses, and the application of state-feedbacks is not always possible, due to unavailability of full-state measurements (see [9], for example). On the other hand, in practical applications, the entries of the needed SOFs are bounded, and since the problem of SOFs with interval constrained entries is NP-hard (see [10,11]), one cannot expect the existence of a deterministic polynomial-time algorithm to solve this problem. Randomized algorithms are thus natural solutions to this problem. The probabilistic and randomized methods for the constrained SOF problem and robust stabilization via SOFs (among other hard problems) are discussed in [12–15]. The Ray-Shooting Method was recently introduced in [16], where it was utilized to derive the Ray-Shooting (RS) randomized algorithm for the minimal-gain SOF problem with regional pole-assignment, where the region can be non-convex and unconnected. For a survey of the SOF problem see [17] and for a recent survey of the robust SOF problem see [18].

The contribution of this research is as follows:

1. The suggested algorithm is based on the Ray-Shooting Method (see [16]), which, as opposed to smooth optimization methods, has the potential of finding a global optimum of continuous functions over compact non-convex and unconnected regions.
2. The suggested algorithm has a proof of convergence (in probability) and explicit complexity.
3. Experience with the algorithm shows good quality of controllers, high percent of success and good run-time for real-life systems. Thus, the suggested practical algorithm efficiently solves the problem of LQR via SOF.
4. The algorithm does not need to solve any Riccati equations and thus can be applied to large systems.
5. The suggested algorithm is one of the few that deals with LQR via SOF and has the ability to deal with discrete-time systems under the same formulation.

The reminder of the article is organized as follows:

In Section 2, we introduce the practical randomized algorithm for the problem of LQR via SOF. In Section 3, we give the results of the algorithm for some real-life systems and we compare its performance with the performance of a well known algorithm that has a proof of convergence to local minimum (under some reasonable assumptions). Finally, in Section 4 we conclude with some remarks.

2. The Practical Algorithm for the Problem of LQR via SOF

Assume that $K^{(0)} \in int\left(\mathcal{S}_\alpha\right)$ was found by the RS algorithm (see [16]) or by any other method (see [19–21]). Let $h > 0$ and let $U^{(0)}$ be a unit vector w.r.t. the Frobenius norm, i.e., $\left\|U^{(0)}\right\|_F = 1$. Let $L^{(0)} = K^{(0)} + h \cdot U^{(0)}$ and let $\mathcal{L}$ be the hyperplane defined by $L^{(0)} + V$, where $\left\langle V, U^{(0)} \right\rangle_F = 0$. Let $r_\infty > 0$ and let $\mathcal{R}_\infty$ denote the set of all $F \in \mathcal{L}$, such that $\left\|F - L^{(0)}\right\|_F \leq r_\infty$. Let $\mathcal{R}_\infty\left(\epsilon\right) = \mathcal{R}_\infty + \overline{\mathbb{B}\left(0, \epsilon\right)}$, where $\overline{\mathbb{B}\left(0, \epsilon\right)}$ denotes the closed ball centered at 0 with radius ϵ ($0 < \epsilon \leq \frac{1}{2}$), with respect to the Frobenius norm on $\mathbb{R}^{q \times r}$. Let $\mathcal{D}^{(0)} = CH\left(K^{(0)}, \mathcal{R}_\infty\left(\epsilon\right)\right)$ denote the convex-hull of the vertex $K^{(0)}$ with the basis $\mathcal{R}_\infty\left(\epsilon\right)$. Let $\mathcal{S}_\alpha^{(0)} = \mathcal{S}_\alpha \cap \mathcal{D}^{(0)}$ and note that $\mathcal{S}_\alpha^{(0)}$ is compact (but generally not convex). We wish to minimize the continuous function $\sigma_{max}\left(K\right)$ (or the continuous function $J\left(x_0, K\right)$, when x_0 is known) over the compact set $\mathcal{S}_\alpha \cap \overline{\mathbb{B}\left(K^{(0)}, h\right)}$. Let K_* denote a point in $\mathcal{S}_\alpha \cap \overline{\mathbb{B}\left(K^{(0)}, h\right)}$ where the minimum of $\sigma_{max}\left(K\right)$ is accepted. Obviously, $K_* \in \mathcal{D}^{(0)}$, for some direction $U^{(0)}$, as above.

The suggested algorithm in Algorithm 1 works as follows:

We start with a point $K^{(0)} \in int\left(\mathcal{S}_\alpha\right)$, found by the RS algorithm.

Assuming that $K_* \in \mathcal{D}^{(0)}$, the inner-loop ($j = 1, \ldots, n$) uses the Ray-Shooting Method in order to find an approximation of the global minimum of the function $\sigma_{max}\left(K\right)$ over $\mathcal{S}_\alpha^{(0)}$—the portion of $\mathcal{S}_\alpha$ bounded in the cone $\mathcal{D}^{(0)}$. The proof of convergence in probability of the inner-loop and its complexity (under the above mentioned assumption) can be found in [16] (see also [22]). In the inner-loop, we choose a search direction by choosing a point F in $\mathcal{R}_\infty\left(\epsilon\right)$—the base of the cone $\mathcal{D}^{(0)}$. Next, in the most inner-loop ($k = 1, \ldots, s$) we scan the ray $K\left(t\right) := \left(1 - t\right) K^{(0)} + tF$ and record the best controller on it. Repeating this a sufficient number of times (as is given in (7) and in the discussion right after), we reach K_* (or an ϵ- neighborhood of it) with high probability, under the assumption that $K_* \in \mathcal{D}^{(0)}$.

The outer-loop ($i = 1, \ldots, m$) is used as a substitution for restarting the RS algorithm again and again, by taking $K^{(\text{best})}$ as the new vertex of the search cone instead of $K^{(0)}$ and by choosing a different direction $U^{(0)}$. The choice of a different direction is made as a backup to the case where the above mentioned assumption didn't hold in the previous iterations (see Remark 1 below). The replacement of $K^{(0)}$ by $K^{(\text{best})}$ can be considered as a heuristic step, which is made instead of running the RS algorithm many times in order to generate "the best starting point", which is relevant only if we actually evaluate $\sigma_{max}\left(K\right)$ on each such point and take the point with the best value as the best starting point. Since we,

in any case, evaluate $\sigma_{max}(K)$ in the main algorithm, we could avoid the repeated execution of the RS algorithm. The outer-loop is similar to what is done in the Hide-And-Seek algorithm (see [23,24]). The convergence in probability of the Hide-And-Seek algorithm can be found in [25].

Remark 1. *The volume of* $\overline{\mathbb{B}\left(K^{(0)},h\right)}$ *is given by* $\frac{\pi^{\ell/2}}{\Gamma(\ell/2+1)}\cdot h^{\ell}$ *where* $\ell := qr$ *and* Γ *is the known* Γ*-function. The volume of* $\mathcal{D}^{(0)}$ *is given approximately (and exactly when* $\epsilon = 0$*) by* $\frac{h}{\ell}\cdot\frac{\pi^{(\ell-1)/2}}{\Gamma((\ell-1)/2+1)}\cdot r_{\infty}^{\ell-1}$*. Thus, by taking* $r_{\infty} = h$*, the portion of* $\overline{\mathbb{B}\left(K^{(0)},h\right)}$ *covered by* $\mathcal{D}^{(0)}$ *(i.e., the probability that* $K_* \in \mathcal{D}^{(0)}$*) is given by* $\frac{\Gamma(\ell/2+1)}{\ell\cdot\sqrt{\pi}\cdot\Gamma((\ell-1)/2+1)}$*. Let* Θ *denote the known relation between functions* $f,g:\mathbb{N}\to R$ *defined by* $f(n)=\Theta(g(n))$ *if and only if* $\lim_{n\to\infty}\frac{f(n)}{g(n)}=1$*. Since* $\Gamma(\ell/2+1)=\Theta\left(\sqrt{2\pi}e^{-\ell/2}\left(\frac{\ell}{2}\right)^{\frac{\ell+1}{2}}\right)$ *and since* $\left(\frac{\ell-1}{\ell}\right)^{\ell/2}\to e^{-1/2}$ *when* $\ell\to+\infty$*, it follows that*

$$\frac{\Gamma(\ell/2+1)}{\ell\cdot\sqrt{\pi}\cdot\Gamma((\ell-1)/2+1)}=\Theta\left(\frac{1}{e\cdot\sqrt{2\pi\ell}}\right).$$

Therefore, by taking $m=\left\lceil e\cdot\sqrt{2\pi\ell}\right\rceil$ *iterations in the outer-loop, we have* $K_*\in\mathcal{D}^{(0)}$ *almost surely. Specifically, when* $\ell\geq 12$*, we suggest taking* $m=2\ell$ *and* $\ell\times\ell$ *orthogonal matrix* $U=\left[\begin{array}{ccccc} u_1 & u_2 & \cdots & u_{\ell}\end{array}\right]$*, and to take the directions* $U_j^{(0)}=\pm mat\left(u_j\right), j=1,\ldots,\ell$ *in the outer-loop.*

The complexity of the suggested practical algorithm measured as the number of its arithmetic operations is given as follows:

- computing the matrix $P(K(t))$ as in (5) takes $O\left(\left(p^2\right)^3\right)=O\left(p^6\right)$, since the dominant operation is the inversion of the $p^2\times p^2$ matrix there.
- checking $K(t)\in\mathcal{S}_{\alpha}$ by checking the α-stability of $A_{cl}(K(t))$ (as well as computing $\sigma(K(t))$), takes $O\left(p^3\right)$ for computing the characteristic polynomial of $A_{c\ell}(K(t))$ (of $P(K(t))$, respectively) and $O\left(p\log_2^2(p)\left(\log_2^2(p)+\log_2^2(b)\right)\right)$ for computing approximations $\tilde{\lambda}_j$ for all the eigenvalues $\lambda_j, j=1,\ldots,p$ of $A_{c\ell}(K(t))$ (of $P(K(t))$, respectively), with accuracy $\left|\tilde{\lambda}_j-\lambda_j\right|<2^{2-\frac{b}{p}}$ where $b\geq p\log_2(p)$. The approximated eigenvalues can be computed to the accuracy $2^{2-\frac{b}{p}}=\epsilon$, with $b=\left(2+\log_2\left(\frac{1}{\epsilon}\right)\right)p$, by the algorithm of V. Y. Pan (see [26]). We end up with $O\left(p^3\right)$ for these operations.
- computing uniformly distributed $q\times r$ matrix takes $O\left(\max(q,r)^3\right)$ operations.

We therefore have a total complexity of $O\left(mns\left(\max(q,r)^3+p^6\right)\right)$.
Let a closed ϵ-neighborhood of K_* in $D^{(0)}$ be defined by

$$\mathcal{S}_{\alpha}^{(0)}(\epsilon)=\left\{K\in\mathcal{S}_{\alpha}^{(0)}\ \middle|\ \sigma(K)\leq\sigma(K_*)+\epsilon\right\}.$$

Let the idealized algorithm be the algorithm that samples the search space $D^{(0)}$ until hitting $\mathcal{S}_{\alpha}^{(0)}(\epsilon)$, where the sampling is according to a general p.d.f. g and a related generator G. For $0<\beta<1$, the number of iterations needed to guarantee a probability of at least $1-\beta$ to hit $\mathcal{S}_{\alpha}^{(0)}(\epsilon)$ is given by

$$\left\lceil\frac{M_g Vol\left(D^{(0)}\right)}{m_g Vol\left(\mathcal{S}_{\alpha}^{(0)}(\epsilon)\right)}\left|\ln(\beta)\right|\right\rceil, \tag{7}$$

where Voℓ denotes the volume of the related set and M_g, m_g are the essential-supremum and essential-infimum of g over $D^{(0)}$, respectively (see [16]). Similarly to what is done in [16], one can show that the last is $O\left(\frac{|\ln(\beta)|hM_g}{\epsilon m_g}\left(\frac{r_\infty}{r_\epsilon}\right)^{qr}\right)$, where r_ϵ is a radius of a ball of a basis of a cone with height ϵ and vertex K_* that has a volume that equals to $Vol\left(\mathcal{S}_\alpha^{(0)}(\epsilon)\right)$. This results in an exponential number of iterations, but if we restrict the input of the algorithm to systems with q, r satisfying $q \leq q_0, r \leq r_0$ when q_0, r_0 are fixed, then, the number of iterations would be $O\left(\frac{|\ln(\beta)|hM_g}{\epsilon m_g}\left(\frac{r_\infty}{r_\epsilon}\right)^{q_0 r_0}\right)$, i.e., polynomial in $\left(\frac{r_\infty}{r_\epsilon}\right)$—which can be considered as the true size of the problem (for a fixed p, q, r). In this sense we can say that the algorithm is efficient. The total number of arithmetic operations of the idealized algorithm that guarantees a probability at least $1-\beta$ to hit $\mathcal{S}_\alpha^{(0)}(\epsilon)$ is therefore given by $O\left(\frac{|\ln(\beta)|h}{\epsilon}\left(\frac{r_\infty}{r_\epsilon}\right)^{q_0 r_0}\left(\max(q,r)^3 + p^6\right)\right)$, since sampling points according to the uniform distribution g (and therefore $m_g = M_g = 1$) and the related generator G, takes $O\left(\max(q,r)^3\right)$.

For the sake of comparison, which will be presented in the next section, we bring here, in Algorithm 2, the algorithm of D. Moerder and A. Calise (see [5]) adjusted to our formulation of the problem, which we call: the MC Algorithm. To the best of our knowledge, this is the best algorithm for LQR via SOF published so far.

Algorithm 1: The practical randomized algorithm for the LQR via static-output-feedback (SOF) problem.

Input: $0 < \epsilon \leq \frac{1}{2}, \alpha, h, r_\infty > 0$, integers: $m, n, s > 0$,
controllable pairs (A, B) and $\left(A^T, C^T\right)$,
matrices $Q > 0, R \geq 0$ and $K^{(0)} \in int\left(\mathcal{S}_\alpha\right)$.
Output: $K \in \mathcal{S}_\alpha$ close as possible to K_*.

1. compute $P\left(K^{(0)}\right)$ as in (5)
2. $P^{(\text{best})} \leftarrow P\left(K^{(0)}\right)$
3. $\sigma_{max}^{(\text{best})} \leftarrow \max\left(\sigma\left(P^{(\text{best})}\right)\right)$
4. for $i = 1$ to m do
4.1. choose $U^{(0)}$ such that $\left\|U^{(0)}\right\|_F = 1$,
uniformly at random
4.2. let $L^{(0)} \leftarrow K^{(0)} + h \cdot U^{(0)}$
4.3. for $j = 1$ to n do
4.3.1. choose $F \in \mathcal{R}_\infty(\epsilon)$ uniformly at random
4.3.1.1. for $k = 1$ to s do
4.3.1.1.1. $t \leftarrow \frac{k}{s}$
4.3.1.1.2. $K(t) \leftarrow (1-t)K^{(0)} + tF$
4.3.1.1.3. if $K(t) \in \mathcal{S}_\alpha$ then
4.3.1.1.3.1. compute $P(K(t))$ as in (5)
4.3.1.1.3.2. $\sigma_{max}(K(t)) \leftarrow \max(\sigma(P(K(t))))$
4.3.1.1.3.3. if $\left(\sigma_{max}(K(t)) < \sigma_{max}^{(\text{best})}\right)$ then
4.3.1.1.3.3.1. $K^{(\text{best})} \leftarrow K(t)$
4.3.1.1.3.3.2. $P^{(\text{best})} \leftarrow P(K(t))$
4.3.1.1.3.3.3. $\sigma_{max}^{(\text{best})} \leftarrow \sigma_{max}(K(t))$
4.4. $K^{(0)} \leftarrow K^{(\text{best})}$
5. return $K^{(\text{best})}, P^{(\text{best})}, \sigma_{max}^{(\text{best})}$

Algorithm 2: The MC Algorithm.

Input: $0 < \epsilon, \alpha$, integers: $m, s > 0$,
controllable pairs (A, B) and $\left(A^T, C^T\right)$,
matrices $Q > 0, R > 0$ and $K_0 \in int\left(\mathcal{S}_\alpha\right)$.
Output: $K \in \mathcal{S}_\alpha$ close as possible to K_*.

1. $j \leftarrow 0$
2. $A_0 \leftarrow A - BK_0C$
3. $P_0 \leftarrow -mat\begin{pmatrix} \left(I_p \otimes A_0^T + A_0^T \otimes I_p\right)^{-1} \cdot \\ vec\left(Q + C^T K_0^T R K_0 C\right) \end{pmatrix}$
4. $S_0 \leftarrow -mat\left(\left(I_p \otimes A_0 + A_0 \otimes I_p\right)^{-1} \cdot vec\left(I_p\right)\right)$
5. $\sigma_{max}\left(K_0\right) \leftarrow \max\left(\sigma\left(P_0\right)\right)$
6. $\Delta K_0 \leftarrow R^{-1}B^T P_0 S_0 C^T \left(CS_0C\right)^{-1} - K_0$
7. $flag \leftarrow 0$
8. for $k = 1$ to s do
8.1. $t \leftarrow \frac{k}{s}$
8.2. $K\left(t\right) \leftarrow \left(1 - t\right) K_0 + t\Delta K_0$
8.3. if $K\left(t\right) \in \mathcal{S}_\alpha$ then
8.3.1. $A\left(t\right) \leftarrow A - BK\left(t\right) C$
8.3.2. $P\left(t\right) \leftarrow -mat\begin{pmatrix} \left(I_p \otimes A\left(t\right)^T + A\left(t\right)^T \otimes I_p\right)^{-1} \cdot \\ vec\left(Q + C^T K\left(t\right)^T R K\left(t\right) C\right) \end{pmatrix}$
8.3.3. $S\left(t\right) \leftarrow -mat\left(\left(I_p \otimes A\left(t\right) + A\left(t\right) \otimes I_p\right)^{-1} \cdot vec\left(I_p\right)\right)$
8.3.4. $\sigma_{max}\left(K\left(t\right)\right) \leftarrow \max\left(\sigma\left(P\left(t\right)\right)\right)$
8.3.5. if $\sigma_{max}\left(K\left(t\right)\right) < \sigma_{max}\left(K_0\right)$ then
8.3.5.1. $K_1 \leftarrow K\left(t\right)$
8.3.5.2. $A_1 \leftarrow A - BK_1C$
8.3.5.3. $P_1 \leftarrow P\left(t\right)$
8.3.5.4. $S_1 \leftarrow S\left(t\right)$
8.3.5.5. $\sigma_{max}\left(K_1\right) \leftarrow \sigma_{max}\left(K\left(t\right)\right)$
8.3.5.6. $flag \leftarrow 1$
9. if $flag == 1$ then
9.1. while $\left(\left|\sigma_{max}\left(K_{j+1}\right) - \sigma_{max}\left(K_j\right)\right| \geq \epsilon\right)$ and $\left(j < m\right)$ do
9.1.1. $\Delta K_j \leftarrow R^{-1}B^T P_j S_j C^T \left(CS_jC\right)^{-1} - K_j$
9.1.2. for $k = 1$ to s do
9.1.2.1. $t \leftarrow \frac{k}{s}$
9.1.2.2. $K\left(t\right) \leftarrow \left(1 - t\right) K_j + t\Delta K_j$
9.1.2.3. if $K\left(t\right) \in \mathcal{S}_\alpha$ then
9.1.2.3.1. $A\left(t\right) \leftarrow A - BK\left(t\right) C$
9.1.2.3.2. $P\left(t\right) \leftarrow -mat\begin{pmatrix} \left(I_p \otimes A\left(t\right)^T + A\left(t\right)^T \otimes I_p\right)^{-1} \cdot \\ vec\left(Q + C^T K\left(t\right)^T R K\left(t\right) C\right) \end{pmatrix}$
9.1.2.3.3. $S\left(t\right) \leftarrow -mat\left(\left(I_p \otimes A\left(t\right) + A\left(t\right) \otimes I_p\right)^{-1} \cdot vec\left(I_p\right)\right)$
9.1.2.3.4. $\sigma_{max}\left(K\left(t\right)\right) \leftarrow \max\left(\sigma\left(P\left(t\right)\right)\right)$
9.1.2.3.5. if $\sigma_{max}\left(K\left(t\right)\right) < \sigma_{max}\left(K_j\right)$ then
9.1.2.3.5.1. $K_{j+1} \leftarrow K\left(t\right)$
9.1.2.3.5.2. $A_{j+1} \leftarrow A - BK_{j+1}C$
9.1.2.3.5.3. $P_{j+1} \leftarrow P\left(t\right)$
9.1.2.3.5.4. $S_{j+1} \leftarrow S\left(t\right)$
9.1.2.3.5.5. $\sigma_{max}\left(K_{j+1}\right) \leftarrow \sigma_{max}\left(K\left(t\right)\right)$
9.1.2.3.5.6. $j \leftarrow j + 1$
10. return $K^{(\text{best})} \leftarrow K_j, A_j, P_j, S_j, \sigma_{max}^{(\text{best})} \leftarrow \sigma_{max}\left(K^{(best)}\right)$

3. Experiments

In the following experiments we applied the Algorithm 1 and Algorithm 2, on systems taken from the liberaries [27–29]. We took only the systems with controllable (A,B), $\left(A^T,C^T\right)$ pairs, for which the RS algorithm succeeded in finding SOFs (see [16], Table 8, p. 231). In order to initialize the MC Algorithm, we also used the RS algorithm to find a starting α-stabilizing static-output-feedback, for known optimal value for α. In all the experiments for the suggested algorithm we used $m=100$, $n=100$, $s=100$, $h=100$, $r_\infty=100$, $\epsilon=10^{-16}$, and for the MC Algorithm we used $m=10{,}000$, $s=100$ (in order to get the same number of 10^6 overall iterations and the same number $s=100$ of iterations for the local search). In every case, we took $Q=I_p$, $R=I_q$. The Stability Margin column of Table 1 relates to $\alpha>0$ for which the real part of any eigenvalue of the closed-loop is less than or equal to $-\alpha$. The values of α in Table 1 relates to the largest α for which the RS algorithm succeeded in finding $K^{(0)}$. The reason is that if λ is an eigenvalue of $A_{c\ell}(K)$ with corresponding eigenvector v then, (4) implies

$$\Re(\lambda)=-\frac{v^*\left(Q+C^TK^TRKC\right)v}{2v^*P(K)v}\leq-\frac{v^*Qv}{2v^*P(K)v}.$$

It follows that minimizing $\sigma_{max}(K)$ results in a larger abscissa. Thus, it is worth searching for a starting point $K^{(0)}$ that maximizes the abscissa α. This can be done efficiently by running a binary search on the abscissa and using the RS algorithm as an oracle. Note that RS CPU time appearing in the third column of Table 1 relates to running the RS algorithm for known optimal value of the abscissa. The RS algorithm is sufficiently fast also for this purpose, but other methods such as the HIFOO algorithm (see [19]) can be applied for this purpose as well.

Let $\sigma_{max}(F)$ denote the functional (6) for the system $\left(A,B,I_p\right)$, where $A-BF$ is stable, i.e., $F\in\mathcal{S}^{q\times p}$. Let $P(F)$ denote the unique solution of (4) for the system $\left(A,B,I_p\right)$ with F as above. Let $\sigma_{max}(K)$ denote the functional (4) for the system (A,B,C) with $K\in\mathcal{S}^{q\times r}$ and related Lyapunov matrix $P=P(K)$. Now, if $A-BKC$ is stable for some K then, $A-BF$ is stable for $F=KC$ (but there might exist a F such that $A-BF$ is stable, but which cannot be defined as KC for some $q\times r$ matrix K). Therefore

$$\sigma_{max}(F_*)=\min_{F\in\mathcal{S}^{q\times p}}\sigma_{max}(F)\leq\min_{K\in\mathcal{S}_\alpha^{q\times r}\cap\overline{\mathbb{B}\left(K^{(0)},h\right)}}\sigma_{max}(K)=\sigma_{max}(K_*),\tag{8}$$

where F_* is an optimal LQR state-feedback controller for the system $\left(A,B,I_p\right)$. We conclude that $\sigma_{max}(F_*)\leq\sigma_{max}(K_*)\leq\sigma_{max}\left(K^{(\text{best})}\right)$. Thus, $\sigma_{max}(F_*)$ is a lower-bound for $\sigma_{max}\left(K^{(\text{best})}\right)$ and can serve as a good estimator for it (as is evidently seen from Table 1 in many cases) in order to stop the algorithm earlier, since $\sigma_{max}(F_*)$ can be calculated in advance.

The fourth column of Table 1 represents $\sigma_{max}\left(K^{(0)}\right)$. The fifth column stands for $\sigma_{max}\left(K^{(\text{best})}\right)$, where the number in the parentheses is the relative improvement over $\sigma_{max}\left(K^{(0)}\right)$, in percent. The sixth column is for $\sigma_{max}(F_*)$ and the seventh column is for the CPU time of the suggested algorithm, given in seconds. The eighth column stands for $\sigma_{max}\left(K^{(\text{best})}\right)$ found by the MC Algorithm and finally, the ninth column stands for the CPU time of the MC Algorithm.

Regarding the suggested algorithm, note that the relative improvement of $\sigma_{max}^{(\text{best})}$ over $\sigma_{max}^{(0)}$ is at least 1000% for the systems: AC6, AC11, AC12, DIS4, REA1, TF1, NN9, NN15 and NN17 (i.e., in 9 out of 29 systems). The AC12 should be noted, for which the improvement is $7.55\cdot10^{17}$%! Note also that for the (13 out of 29) systems: AC1, AC2, AC6, AC11, AC12, AC15, HE3, DIS4, REA1, REA2, HF2D10, HF2D11 and NN15, the value of $\sigma_{max}^{(\text{best})}$ for (A,B,C) is very close to the value of $\sigma_{max}(F_*)$ for (A,B).

Table 1. Results of the suggested randomized algorithm for LQR via SOF compared with the results of the MC Algorithm.

System	Stability Margin	RS CPU Time	$\sigma_{max}^{(0)}$ for (A, B, C)	$\sigma_{max}^{(best)}$ for (A, B, C) Suggested Algorithm	σ_{max} (F_*) for (A, B)	Suggested Algorithm CPU Time	$\sigma_{max}^{(best)}$ for (A, B, C) MC Algorithm	MC Algorithm CPU Time
AC1	0.1	0.0625	20.9727	13.3987 (56.52%)	13.0686	43.1718	16.2315	0.1562
AC2	0.1	0.0312	20.9727	13.4217 (56.25%)	13.0686	42.3906	16.2315	0.2500
AC5	0.875	0.0625	2.2821×10^6	2.2208×10^6 (2.76%)	8.4264×10^5	56.8125	2.0608×10^6	0.1093
AC6	0.875	0.1093	463.8502	6.2413 (7.33×10^3%)	5.9721	43.8125	91.3276	0.0625
AC11	0.1	0.1093	1.0661×10^3	7.4244 (1.42×10^4%)	5.8648	31.6406	1.0661×10^3	0.0625
AC12	0.1	$<10^{-4}$	4.1640×10^{19}	5.5149×10^3 (7.55×10^{17}%)	2.7690×10^3	46.3437	2.9950×10^3	3.0625
AC15	0.25	$<10^{-4}$	112.0500	106.1312 (5.57%)	104.8458	42.0000	111.9781	0.1562
HE1	0.1	$<10^{-4}$	26.7550	9.1169 (1.93×10^2%)	2.9961	44.8125	12.9412	0.1250
HE3	0.25	0.0312	944.3921	632.1294 (49.39%)	611.8468	100.7812	668.0894	0.4218
HE4	0.01	0.0625	4.7440×10^3	474.7121 (8.99×10^2%)	229.9171	152.9531	4.7440×10^3	$<10^{-4}$
ROC1	10^{-16}	0.0468	1.4211×10^4	1.1368×10^4 (25%)	1.1207×10^3	81.2500	1.4211×10^4	0.0937
ROC4	10^{-16}	0.1875	1.7688×10^4	8.6513×10^3 (1.04×10^2%)	8.5454×10^2	79.7968	1.7688×10^4	0.0468
DIS4	1	$<10^{-4}$	3.2208×10^7	1.7529 (1.83×10^9%)	1.7504	68.2812	2.0166	0.7031
DIS5	0.1	0.0312	3.9985×10^5	2.3800×10^5 (68%)	9.0756×10^4	109.8906	2.8304×10^5	0.3593
REA1	0.75	0.0468	9.3478×10^5	2.2790 (4.10×10^7%)	2.2265	58.3906	2.7385	0.5156
REA2	0.1	0.0312	9.8349	2.2640 (3.34×10^2%)	2.2443	69.9843	2.7770	0.3125
TMD	0.05	0.0312	45.0958	27.2080 (65.74%)	16.7680	37.2031	27.6023	0.8281
TF1	10^{-16}	0.1093	9.1632×10^3	193.2880 (4.64×10^3%)	58.1296	51.3750	9.1632×10^3	0.0312
HF2D10	10^{-16}	0.0312	1.8090	1.4032 (28.91%)	1.3832	118.2656	1.4269	2.0781
HF2D11	10^{-16}	0.0312	0.4315	0.3699 (16.65%)	0.3676	154.0312	0.3784	1
NN1	10^{-16}	0.0312	7.4631×10^3	1.6144×10^3 (3.62×10^2%)	106.7801	50.4375	2.3561×10^3	0.2500
NN5	0.01	0.0468	3.9123×10^4	9.6741×10^3 (3.04×10^2%)	2.8787×10^3	129.2812	9.8102×10^3	0.3125
NN9	0.01	0.0312	4.0577×10^3	295.6797 (1.27×10^3%)	21.2937	40.6093	3.7349×10^3	0.0937
NN12	0.01	0.0937	934.5127	236.4467 (2.95×10^2%)	30.3714	35.3281	934.5127	0.0625
NN13	0.1	0.0312	6.4782	1.7094 (2.66×10^2%)	0.6299	33.9218	1.8055	0.4531
NN14	0.1	0.0312	4.8907	1.6664 (1.93×10^2%)	0.6299	33.4687	1.7922	0.1718
NN15	0.01	$<10^{-4}$	1.3309×10^5	387.3778 (3.42×10^4%)	386.5741	94.2031	387.4183	0.3281
NN16	0.1	0.1093	35.9487	6.0864 (4.90×10^2%)	2.3276	45.0468	5.9982	0.7343
NN17	0.1	0.0312	2.6404×10^3	36.7664 (7.08×10^3%)	3.1308	28.3281	666.7218	0.0937

Regarding the comparison with the MC Algorithm, we conclude that the MC algorithm actually failed in finding any improvement of $\sigma_{max}^{(\text{best})}$ over $\sigma_{max}^{(0)}$ for the systems: AC11, HE4, ROC1, ROC2, TF1 and NN12. The suggested algorithm performs significantly better regarding this key of performance for the systems: AC6, AC11, AC15, HE3, HE4, TF1, ROC4, NN9 and NN12. The suggested algorithm performs slightly better for the systems: AC1, AC2, DIS4, DIS5, HE1, HF2D10, HF2D11, REA1, REA2, ROC1, TMD, NN1, NN5 and NN13. Finally, the MC algorithm performs slightly better only for the systems: AC5, AC12 and NN16. We conclude that the suggested algorithm outperforms the MC Algorithm regarding the above-mentioned key of performance, although the MC Algorithm outperforms the suggested algorithm in terms of CPU time.

The MC Algorithm seems to perform better locally, while the suggested algorithm seems to perform better globally. Thus, practically, the best approach would be to apply the suggested algorithm in order to find a close neighborhood of a global minimum and then to apply the MC Algorithm on the result, for the local optimization.

4. Concluding Remarks

For a discrete-time system

$$\begin{cases} x_{k+1} = Ax_k + Bu_k \\ y_k = Cx_k \end{cases} \tag{9}$$

and cost functional

$$J(x_0, u) := \sum_{k=0}^{\infty} \left(x_k^T Q x_k + u_k^T R x_k \right), \tag{10}$$

let $u_k = -Ky_k$ be the SOF, and let $A_{c\ell}(K) := A - BKC$ be the closed-loop matrix. Let $\mathbb{D}$ denote the open unit-disk, let $0 < \alpha < 1$ and let $\mathbb{D}_\alpha$ denote the set of all $z \in \mathbb{D}$ with $|z| \leq 1 - \alpha$ (where $|z|$ is the absolute value of z). Let $\mathcal{S}^{q\times r}$ denote the set of all matrices $K \in \mathbb{R}^{q\times r}$ such that $\sigma(A_{c\ell}) \subset \mathbb{D}$ (i.e., stable in the discrete-time sense), and let $\mathcal{S}_\alpha^{q\times r}$ denote the set of all matrices $K \in \mathbb{R}^{q\times r}$ such that $\sigma(A_{c\ell}) \subset \mathbb{D}_\alpha$. In this case we say that $A_{c\ell}$ is α-stable. Let $K \in \mathcal{S}_\alpha^{q\times r}$ be given. Substitution of $u_k = -Ky_k = -KCx_k$ into (10) yields

$$J(x_0, K) := \sum_{k=0}^{\infty} x_k^T \left(Q + C^T K^T R K C \right) x_k. \tag{11}$$

Since $Q + C^T K^T R K C > 0$ and since $A_{c\ell}(K)$ is stable, it follows that the Stein equation

$$P - A_{c\ell}(K)^T P A_{c\ell}(K) = Q + C^T K^T R K C \tag{12}$$

has a unique solution $P > 0$, given by

$$P(K) = mat\left(\left(I_p \otimes I_p - A_{c\ell}(K)^T \otimes A_{c\ell}(K)^T \right)^{-1} \cdot vec\left(Q + C^T K^T R K C \right) \right).$$

Substitution of (12) into (11) and noting that $x_k = A_{c\ell}(K)^k x_0$ with $A_{c\ell}(K)$ stable, leads to

$$\begin{aligned} J(x_0, K) &= \sum_{k=0}^{\infty} x_k^T \left(P - A_{c\ell}(K)^T P A_{c\ell}(K) \right) x_k \\ &= \sum_{k=0}^{\infty} x_0^T A_{c\ell}(K)^{Tk} \left(P - A_{c\ell}(K)^T P A_{c\ell}(K) \right) A_{c\ell}(K)^k x_0 = x_0^T P(K) x_0. \end{aligned}$$

Thus, we look for $K \in \mathcal{S}_\alpha^{q\times r}$ that minimizes the functional $J(x_0, K) = x_0^T P(K) x_0$, and when x_0 is unknown, we seek $K \in \mathcal{S}_\alpha^{q\times r}$ for which $\sigma_{max}(K) := \max(\sigma(P(K)))$ is minimal. Similarly to the continuous-time case, we have $\frac{J(x_0,K)}{\|x_0\|^2} \leq \sigma_{max}(K)$ with equality in the worst case.

Finally, if λ is an eigenvalue of $A_{c\ell}(K)$ and v is a corresponding eigenvector then (12) yields $1-|\lambda|^2 = \frac{v^*\left(Q+C^TK^TRKC\right)v}{v^*P(K)v} \geq \frac{v^*Qv}{v^*P(K)v}$. Therefore $|\lambda|^2 \leq 1-\frac{v^*Qv}{v^*P(K)v}$, and thus, minimizing $\sigma_{max}(K)$ results in larger abscissa. Now, the suggested algorithm can be readily applied to discrete-time systems. As to the MC Algorithm, we are not aware of any discrete-time analogue of it.

We conclude that the Ray-Shooting Method is a powerful tool, since it practically solves the problem of LQR via SOF, for real-life systems. The suggested algorithm has good performance, and is proved to converge in probability. The suggested method can similarly handle the problem of LQR via SOF for discrete-time systems. Obviously, this enlarges the scope and usability of the Ray-Shooting Method and we expect to receive more results in this direction.

Acknowledgments: I wish to dedicate this article to the memory of my beloved father Pinhas Peretz and to the memory of my beloved wife and friend Rivka Rimonda Peretz.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Peretz, Y. A characterization of all the static stabilizing controllers for LTI systems. *Linear Algebra Its Appl.* **2012**, *437*, 525–548.
2. Johnson, T.; Atahns, M. On the design of optimal dynamic compansators for linear constant systems. *IEEE Trans. Autom. Control* **1970**, *15*, 658–660.
3. Levine, W.; Athans, M. On the determination of the optimal constant output feedback gains for linear multivariables systems. *IEEE Trans. Autom. Control* **1970**, *15*, 44–48.
4. Levine, W.; Johnson, T.L.; Athans, M. Optimal limited state variable feedback controllers for linear systems. *IEEE Trans. Autom. Control* **1971**, *16*, 785–793.
5. Moerder, D.; Calise, A. Convergence of numerical algorithm for calculating optimal output feedback gains. *IEEE Trans. Autom. Control* **1985**, *30*, 900–903.
6. Iwasaki, T.; Skelton, R. All controllers for the general H_∞ control problem: LMI existance conditions and stste space formulas. *Automatica* **1994**, *30*, 1307–1317.
7. Iwasaki, T.; Skelton, R.E. Linear quadratic suboptimal control with static output feedback. *Syst. Control Lett.* **1994**, *23*, 421–430.
8. Peres, P.L.D.; Geromel, J.; de Souza, S. Optimal H_2 control by output feedback. In Proceedings of the 32nd IEEE Conference on Decision and Control, San Antonio, TX, USA, 15–17 December 1993; pp. 102–107.
9. Camino, J.F.; Zampieri, D.E.; Peres, P.L.D. Design of A Vehicular Suspension Controller by Static Output Feedback. In Proceedings of the American Control Conference, San Diego, CA, USA, 2–4 June 1999.
10. Blondel, V.; Tsitsiklis, J.N. NP-hardness of some linear control design problems. *SIAM J. Control Optim.* **1997**, *35*, 2118–2127.
11. Nemirovskii, A. Several NP-hard problems arising in robust stability analysis. *Math. Control Signals Syst.* **1993**, *6*, 99–105.
12. Arzelier, D.; Gryazina, E.N.; Peaucelle, D.; Polyak, B.T. Mixed LMI/randomized methods for static output feedback control. In Proceedings of the American Control Conference, Baltimore, MD, USA, 30 June–2 July 2010; pp. 4683–4688.
13. Tempo, R.; Calafiore, G.; Dabbene, F. *Randomized Algorithms for Analysis and Control of Uncertain Systems*; Springer: London, UK, 2005.
14. Tempo, R.; Ishii, H. Monte Carlo and Las Vegas Randomized Algorithms for Systems and Control. *Eur. J. Control* **2007**, *13*, 189–203.
15. Vidyasagar, M.; Blondel, V.D. Probabilistic solutions to some NP-hard matrix problems. *Automatica* **2001**, *37*, 1397–1405.
16. Peretz, Y. A randomized approximation algorithm for the minimal-norm static-output-feedback problem. *Automatica* **2016**, *63*, 221–234.
17. Syrmos, V.L.; Abdallah, C.; Dorato, P.; Grigoradis, K. Static Output Feedback: A Survey. *Automatica* **1997**, *33*, 125–137.
18. Sadabadi, M.S.; Peaucelle, D. From static output feedback to structured robust static output feedback: A survey. *Annu. Rev. Control* **2016**, *42*, 11–26.

19. Gumussoy, S.; Henrion, D.; Millstone, M.; Overton, M.L. Multiobjective Robust Control with HIFOO 2.0. In Proceedings of the IFAC Symposium on Robust Control Design, Haifa, Israel, 16–18 June 2009.
20. Yang, K.; Orsi, R. Generalized pole placement via static output feedback: A methodology based on projections. *Automatica* **2006**, *42*, 2143–2150.
21. Henrion, D.; Loefberg, J.; Kočvara M.; Stingl, M. Solving Polynomial static output feedback problems with PENBMI. In Proceedings of the IEEE Conference on Decision and Control, Sevilla, Spain, 15 December 2005.
22. Peretz, Y. On applications of the Ray-Shooting method for structured and structured-sparse static-output-feedbacks. *Int. J. Syst. Sci.* **2017**, *48*, 1902–1913.
23. Zabinsky, Z.B. *Stochastic Adaptive Search for Global Optimization*; Kluer Academic Publishers: Dordrecht, The Netherlands, 2003.
24. Romeijn, H.E.; Smith, R.L. Simulated Annealing for Constrained Global Optimization. *J. Glob. Optim.* **1994**, *5*, 101–126.
25. Bélisle, C.J.P. Convergence Theorems for a Class of Simulated Annealing Algorithms on $\mathbb{R}^d$. *J. Appl. Probab.* **1992**, *29*, 885–895.
26. Pan, V.Y. Univariate polynomials: Nearly optimal algorithms for numerical factorization and root-finding. *J. Symb. Comput.* **2002**, *33*, 701–733.
27. Leibfritz, F. *COMPl$_e$ib: Constrained Matrix-Optimization Problem Library—A Collection of Test Examples for Nonlinear Semidefinite Programs, Control System Design and Related Problems*; Technical Report; Department of Methematics, University of Trier: Trier, Germany, 2003.
28. Leibfritz, F. *Description of the Benchmark Examples in COMPl$_e$ib 1.0*; Technical Report; Department of Methematics, University of Trier: Trier, Germany, 2003.
29. Leibfritz, F.; Lipinski, W. *COMPl$_e$ib 1.0—User Manual and Quick Reference*; Technical Report; Department of Methematics, University of Trier: Trier, Germany, 2004.