Applied Mathematics, 2015, 6, 1263-1270
Published Online July 2015 in SciRes. http://www.scirp.org/journal/am
http://dx.doi.org/10.4236/am.2015.68119

On the Iterative Solution to H_∞ Control Problems

Ivan G. Ivanov[1,2], Ivelin G. Ivanov[2], Nikolay C. Netov[1]

[1]Faculty of Economics and Business Administration, Sofia University "St. Kliment Ohridski", Sofia, Bulgaria
[2]Pedagogical College Dobrich, Shoumen University, Shoumen, Bulgaria
Email: i_ivanov@feb.uni-sofia.bg, iwelin.ivanow@gmail.com, nnetoff@feb.uni-sofia.bg

Received 28 May 2015; accepted 14 July 2015; published 17 July 2015

Abstract

This paper addresses the problem for solving a Continuous-time Riccati equation with an indefinite sign of the quadratic term. Such an equation is closely related to the so called full information H_∞ control of linear time-invariant system with external disturbance. Recently, a simultaneous policy update algorithm (SPUA) for solving H_∞ control problems is proposed by Wu and Luo (*Simultaneous policy update algorithms for learning the solution of linear continuous-time H_∞ state feedback control*, *Information Sciences*, 222, 472-485, 2013). However, the crucial point of their method is to find an initial point, which ensuring the convergence of the method. We will show one example where Wu and Luo's method is not effective and it converges to an indefinite solution. Three effective methods for computing the stabilizing solution to the considered equation are investigated. Computer realizations of the presented methods are numerically compared on the computational platforms MATLAB and SCILAB.

Keywords

Continuous-Time Riccati Equation, Indefinite Sign, Iterative Computation, Stabilizing Solution

1. Introduction

The continuous-time algebraic Riccati equations and their extensions have been investigated extensively in the literature. Recently, the H_∞ control problem was solved for linear time-invariant system [1]-[3] and for stochastic systems [4]-[7].

Wu and Luo [8] have commented the iterative solution of the following continuous-time algebraic Riccati equation

$$\mathcal{R}(X) := A^{\mathrm{T}}X + XA + C^{\mathrm{T}}C + \gamma^{-2}XGG^{\mathrm{T}}X - XB\left(D^{\mathrm{T}}D\right)^{-1}B^{\mathrm{T}}X = 0. \tag{1}$$

Note that this equation has indefinite quadratic part. Assume there exists a positive semidefinite solution X^* to (1) with property that real parts of eigenvalues of $A(X) = A + \gamma^{-2}GG^{\mathrm{T}}X^* - B\left(D^{\mathrm{T}}D\right)^{-1}B^{\mathrm{T}}X^*$ are negative. Such type solution is called a stabilizing solution.

The H_∞ linear quadratic problems have been introduces by Basar and Bernhard [9] as a two-player zero sum gane. We consider a model for a a two-player zero-sum game, where the control function $u(t)$ is a minimizing player (or a controller player) of the functional $J_\gamma(u,v)$ and the disturbance function $v(t)$ is a maximizing player (or a disturbance player), where

$$J_\gamma(u,v) = \int_0^\infty \left(x^{\mathrm{T}}C^{\mathrm{T}}Cx + u^{\mathrm{T}}D^{\mathrm{T}}Du - \gamma^2 v^{\mathrm{T}}v\right)\mathrm{d}t.$$

The controller player aims to minimize the $J_\gamma(u,v)$ and the disturbance player aims to maximize the $J_\gamma(u,v)$ under a constrain of the system:

$$\mathrm{d}x(t) = Ax(t)\mathrm{d}t + Gv(t)\mathrm{d}t + Bu(t)\mathrm{d}t. \tag{2}$$

Knowing the stabilizing solution X^* to (1) we define the following functions:

$$u^* = -\left(D^{\mathrm{T}}D\right)^{-1}B^{\mathrm{T}}X^*x, \qquad v^* = \gamma^{-2}G^{\mathrm{T}}X^*x.$$

The functions $\left(u^*,v^*\right)$ have the property

$$J_\gamma\left(u^*,v\right) \le J_\gamma\left(u^*,v^*\right) \le J_\gamma\left(u,v^*\right).$$

And thus they form the equilibrium point of the two-player zero-sum game described by (2) and the functional $J_\gamma(u,v)$. This fact is well known in the literature and it can be derived using the Pontryagin's Maximum Principle for example. Moreover, the stabilizing solution is very important solution to Equation (1).

So, why we need to study the iterative equations for computing the stabilizing equation to (1)? Many researchers have investigated Riccati Equation (1) and more specially how to compute his stabilizing solution. Lanzon *et al.* [1] have proposed two effective methods. The first method constructs two matrix sequences where the first sequence converges to the stabilizing solution. The second method avoids the second matrix sequence and defines one matrix sequence which directly approximates the stabilizing solution. Later, Wu and Luo [8] have studied the same equation and the proposed method in [1]. They have commented that the second Lanzon's method (it is Algorithm 2 [8]) is not fully effective and by this reason they have introduced the new method described as Algorithm 4 in their paper [8]. Here, we consider an example where these two algorithms will be compared.

Example 1. Let us we take the following matrix coefficients to (1) (using the MATLAB notations):

$$A = [-0.0665\ 8\ 0\ 0;\ 0\ -3.663\ 3.663\ 0;\ -6.86\ 0\ -13.736\ -13.736;\ 0.6\ 0\ 0\ 0];$$

$$G = [-8;\ 0;\ 0;\ 0];\quad B = [0;\ 10;\ 15;\ 0];\quad C^{\mathrm{T}}C = \mathrm{eye}(4,4);\quad D^{\mathrm{T}}D = 1;\quad \gamma = 5.5\,.$$

We execute Algorithm 2 [8] and Algorithm 4 [8] with the initial point $X_0 = 0$. Starting Algorithm 2 in MATLAB we obtain the following stabilizing solution to (1) after 4 main iterations and (the average number of iterations for the inner loop is 7):

$$\tilde{X}_{Alg2} = \begin{pmatrix} 0.2842 & 0.1488 & -0.0128 & 0.3095 \\ 0.1488 & 0.1591 & -0.0058 & 0.1693 \\ -0.0128 & -0.0058 & 0.0295 & -0.0228 \\ 0.3095 & 0.1693 & -0.0228 & 2.0917 \end{pmatrix}.$$

And the solution computed by Algorithm 4 is

$$\tilde{X}_{Alg4} = \begin{pmatrix} 0.0834 & -0.0116 & -0.0954 & 0.0661 \\ -0.0116 & 0.0310 & -0.0718 & -0.0251 \\ -0.0954 & -0.0718 & -0.0045 & -0.1229 \\ 0.0661 & -0.0251 & -0.1229 & 1.7967 \end{pmatrix}.$$

Note that the matrix $\tilde{X}_{Alg2}$ is positive definite and $\left\|\mathcal{R}\left(\tilde{X}_{Alg2}\right)\right\| = 3.3265e-14$ while the matrix $\tilde{X}_{Alg4}$ is indefinite and $\left\|\mathcal{R}\left(\tilde{X}_{Alg4}\right)\right\| = 2.1367e-12$. In addition $\left\|\tilde{X}_{Alg2} - \tilde{X}_{Alg4}\right\| = 0.6579$. Thus these two solutions are different! Which of them is sought? We have to check whether the corresponding matrices stabilize system (2). We compute eigenvalues of $A\left(\tilde{X}_{Alg2}\right)$ and $A\left(\tilde{X}_{Alg4}\right)$. The eigenvalues of $A\left(\tilde{X}_{Alg2}\right)$ have negative real parts and the matrix $A\left(\tilde{X}_{Alg4}\right)$ has one positive eigenvalue. Thus the matrix $\tilde{X}_{Alg2}$ is the stabilizing solution to Equation (1) while the matrix $\tilde{X}_{Alg4}$ is not the stabilizing solution to (1). In addition we have execute the same example with the open software SCILAB (http://www.scilab.org/scilab/about). We apply the SCILAB's function "lyap" for Algorithm 2 and Algorithm 4. After 4 main iterations Algorithm 2 in SCILAB computes the stabilizing solution with $\left\|\mathcal{R}\left(\tilde{X}_{Alg2}\right)\right\| = 7.4000e-15$. After 18 iterations with Algorithm 4 in SCILAB we obtain the same solution $\tilde{X}_{Alg4}$ with $\left\|\mathcal{R}\left(\tilde{X}_{Alg4}\right)\right\| = 6.047e-15$. The solution is indefinite and it is not the stabilizing solution.

So, this example gives us the conclusion that the Algorithm 4 described in [8] compute only a solution to (1) and this solution is not always positive definite and this solution is not always stabilizing.

In this reason we confirm that the Lanzon's method [1] is an effective method for computing the stabilizing solution. His main essential feature is that the iterative process includes two iterative loops-the out loop and the inner loop. We extend the ideas described by Lanzon *et al.* [1] and Feng and Anderson [6] to propose iterative methods where one matrix sequence is constructed. Here we introduce additional two iterative methods which lead directly to the stabilizing solution. Our contribution is to apply two computational schemes for realization the first iterative equation. Moreover, the second iterative equation is a new method for computing the stabilizing solution to (1). We present a few examples for testing the introduced recurrence equations on the MATLAB and SCILAB computational platforms.

We write $X > Y$ or $X \ge Y$ if $X - Y$ is positive definite or $X - Y$ is positive semidefinite for any two symmetric matrices *X* and *Y*. We use some properties of positive definite and positive semidefinite matrices. A matrix *A* is said to be asymptotically stable if all the eigenvalues of *A* lie in the open left half plane.

2. Iterative Methods for Stabilizing Solution to (1)

The first method is the Lanzon's method [1] and Algorithm 2 from [8]. We present the main theorem with properties for constructing two matrix sequences of positive semidefinite matrices $\left\{X^{(k)}\right\}_{k=0}^{\infty}, \left\{Z^{(k)}\right\}_{k=0}^{\infty}$. The matrix sequences are constructed as follows. We take

$$X^{(k+1)} = X^{(k)} + Z^{(k)}, \quad \text{with } X^{(0)} = 0, \quad k = 0,1,2,\cdots \tag{3}$$

We find $Z^{(k)}$ as the stabilizing solution of the algebraic Riccati equation with definite quadratic part:

$$0 = \left(A^{(k)}\right)^{\mathrm{T}} Z^{(k)} + Z^{(k)} A^{(k)} - Z^{(k)} B \left(D^{\mathrm{T}} D\right)^{-1} \left(Z^{(k)} B\right)^{\mathrm{T}} + \mathcal{R}\left(X^{(k)}\right), \tag{4}$$

where

$$\begin{cases} F_1\left(X^{(k)}\right) = \gamma^{-2} G^{\mathrm{T}} X^{(k)} \\ F_2\left(X^{(k)}\right) = -\left(D^{\mathrm{T}} D\right)^{-1} \left(X^{(k)} B\right)^{\mathrm{T}} \\ A^{(k)} = A + G F_1\left(X^{(k)}\right) + B F_2\left(X^{(k)}\right). \end{cases}$$

The matrices $\left\{Z^{(k)}\right\}_{k=0}^{\infty}$ are stabilizing solutions for the sequence of algebraic Riccati Equations (4). We will prove that the second sequence is monotonically non-decreasing and converges to the unique stabilizing solution to set of Equation (1). We reformulate the convergence theorem introduced in [1] (Theorem 3) and we present it as sufficient conditions to existence the stabilizing solution to (1).

Theorem 1 *Assume there exist symmetric matrices* $\hat{X}$ *and* $X^{(0)}$ *such that* $R\left(X^{(0)}\right) \ge 0$ *and* $R\left(\hat{X}\right) \le 0$ *and* $0 \le X^{(0)} \le \hat{X}$, *and the pair* $\left(A^{(0)}, B\right)$ *is a stabilizable one. Then for the matrix sequences* $\left\{X^{(k)}\right\}_{k=0}^{\infty}, \left\{Z^{(k)}\right\}_{k=0}^{\infty}$ *defined by* (3), (4) *are satisfied for* $k = 0,1,2,\cdots$

1) $\left(A^{(k)}, B\right)$ is stabilizable;

2) $\mathcal{R}\left(X^{(k+1)}\right) = \gamma^{-2} Z^{(k)} G G^{\mathrm{T}} Z^{(k)} \geq 0$;

3) the matrix $\tilde{A}^{(k)} = A + G F_1\left(X^{(k)}\right) + B F_2\left(X^{(k+1)}\right)$ is asymptotically stable for $k = 0, 1, \cdots$;

4) $\hat{X} \geq X^{(k+1)} \geq X^{(k)} \geq 0$.

Proof. The proof follows the proof of Theorem 3 from [1].

Theorem 1 presents sufficient conditions for the equation $R(X) = 0$ has a solution. Such type conditions are introduced here for the considered equation $R(X) = 0$ for the first time. Theorem 1 confirms the convergence properties of iterative method (3), (4).

Further on, we consider an alternative iteration process where one matrix sequence is constructed. Consider the behaviour of the controller player ($u(t)$). Assume the controller player knows the matrix $P_u^{(k-1)}$. He wants to find $P_u^{(k)}$. Then he takes $v = \gamma^{-2} G^{\mathrm{T}} P_u^{(k-1)} x$. The system (2) becomes

$$dx(t) = \left(A + \gamma^{-2} G G^{\mathrm{T}} P_u^{(k-1)}\right) x(t) \mathrm{d}t + Bu(t) \mathrm{d}t.$$

And the functional $J_\gamma(u_k, v_k)$ is

$$J_\gamma(u_k, v_k) = \int_0^\infty \left(x^{\mathrm{T}}\left(C^{\mathrm{T}} C - \gamma^{-2} P_u^{(k-1)} G G^{\mathrm{T}} P_u^{(k-1)}\right) x + u_k^{\mathrm{T}} D^{\mathrm{T}} D u_k\right) \mathrm{d}t.$$

The corresponding Riccati equation regarding to $P_u^{(k)}$ is

$$0 = \left(A_u^{(k-1)}\right)^{\mathrm{T}} P_u^{(k)} + P_u^{(k)} A_u^{(k-1)} - P_u^{(k)} B\left(D^{\mathrm{T}} D\right)^{-1} B^{\mathrm{T}} P_u^{(k)} + Q_u^{(k-1)} \tag{5}$$

where

$$\begin{cases} A_u^{(k-1)} = A + \gamma^{-2} G G^{\mathrm{T}} P_u^{(k-1)} \\ Q_u^{(k-1)} = C^{\mathrm{T}} C - \gamma^{-2} P_u^{(k-1)} G G^{\mathrm{T}} P_u^{(k-1)}. \end{cases}$$

Based on recurrence Equation (5) we derive the following new iteration:

$$\begin{aligned} 0 = &\left(A + \gamma^{-2} G G^{\mathrm{T}} P^{(k-1)}\right)^{\mathrm{T}} P^{(k)} + P^{(k)}\left(A + \gamma^{-2} G G^{\mathrm{T}} P^{(k-1)}\right) \\ &- P^{(k)} B\left(D^{\mathrm{T}} D\right)^{-1} B^{\mathrm{T}} P^{(k)} + C^{\mathrm{T}} C - \gamma^{-2} P^{(k-1)} G G^{\mathrm{T}} P^{(k-1)}, \end{aligned} \tag{6}$$

with $P_0 = 0$. We perform iteration (6) using two recurrence approaches. The first one is to solve Equation (6) as a Riccati equation. We call this approach "(6) + care". The second one is to solve Equation (6) applying the Lyapunov iteration:

$$\begin{aligned} \mathcal{G}(Y_s) := &\left(A + \gamma^{-2} G G^{\mathrm{T}} P^{(k-1)} - B\left(D^{\mathrm{T}} D\right)^{-1} B^{\mathrm{T}} Y_{s-1}\right)^{\mathrm{T}} Y_s \\ &+ Y_s\left(A + \gamma^{-2} G G^{\mathrm{T}} P^{(k-1)} - B\left(D^{\mathrm{T}} D\right)^{-1} B^{\mathrm{T}} Y_{s-1}\right) \\ &+ C^{\mathrm{T}} C - \gamma^{-2} P^{(k-1)} G G^{\mathrm{T}} P^{(k-1)} + Y_{s-1} B\left(D^{\mathrm{T}} D\right)^{-1} B^{\mathrm{T}} Y_{s-1} = 0, \end{aligned} \tag{7}$$

with $Y_s = 0$. The matrix sequence $\{Y_s\}, s = 0, 1, \cdots$ converges to $P^{(k)}$. Iteration (7) defines the inner loop for iteration (6). We call the second approach "(6) + lyap". In fact, that is an extension of the Algorithm 2 [8].

The notation "(6) + care" means that the iteration (6) is solved as a Ricacti equation with unknown matrix $P^{(k)}$. Each solution $P^{(k)}$) of (6) is computed as a solution to Riccati Equation (6). The notation "(6) + lyap" stands for the fact that the solution $P^{(k)}$ to Equation (6) is computed as a limit of the sequence $\{Y_s\}$ and each matrix Y_s is a solution to iteration (7). Iteration (7) describes the inner loop for finding the matrix sequence $\left\{P^{(k)}\right\}$ defined by iteration (6) and it is a Lyapunov iteration for computing $P^{(k)}$.

Thus, Equation (6) can be considered as a new iteration formula. This equation constructs a new matrix sequence $\left\{P^{(k)}\right\}_{k=0}^\infty$ which converges to the stabilizing solution to (1). It is easy to see that the recurrence Equation (6) is obtained from (4) when we substitute $P^{(k)} := Z^{(k)} = X^{(k+1)} - X^{(k)}$. This fact is observed by

Praveen and Bhasin in Lemma 2 [2]. Thus $P^{(k)}$ is the stabilizing solution to (7) with $P^{(k)} > 0$, we conclude the matrix pair $\left(A+\gamma^{(-2)}GG^{\mathrm{T}}P^{(k-1)}, B\right)$ is a stabilizable pair and $C^{\mathrm{T}}C-\gamma^{(-2)}P^{(k-1)}GG^{\mathrm{T}}P^{(k-1)}$ is positive definite. This is enough to start iterative process (7) with $Y_0=\alpha I>0, (\alpha>0)$. Following Theorem 9.1.1 derived by Lancaster and Rodman [10] it is sufficient to claim iterative process (7) converges.

Further on, we extend the idea for constructing the matrix sequence $\left\{P^{(k)}\right\}_{k=0}^{\infty}$. When we put $Z^{(k)}=X^{(k+1)}-X^{(k)}$ in (4) we obtain:

$$\begin{aligned}
0 = &\left(A+GF_1\left(X^{(k)}\right)+BF_2\left(X^{(k)}\right)\right)^{\mathrm{T}} X^{(k+1)} + X^{(k+1)}\left(A+GF_1\left(X^{(k)}\right)+BF_2\left(X^{(k)}\right)\right) \\
&-\left(A+GF_1\left(X^{(k)}\right)+BF_2\left(X^{(k)}\right)\right)^{\mathrm{T}} X^{(k)} - X^{(k)}\left(A+GF_1\left(X^{(k)}\right)+BF_2\left(X^{(k)}\right)\right) \\
&+A^{\mathrm{T}}X^{(k)}+X^{(k)}A+C^{\mathrm{T}}C+\gamma^{-2}X^{(k)}GG^{\mathrm{T}}X^{(k)}-X^{(k)}B\left(D^{\mathrm{T}}D\right)^{-1}B^{\mathrm{T}}X^{(k)} \\
&-X^{(k+1)}B\left(D^{\mathrm{T}}D\right)^{-1}B^{\mathrm{T}}X^{(k+1)}+X^{(k+1)}B\left(D^{\mathrm{T}}D\right)^{-1}B^{\mathrm{T}}X^{(k)} \\
&+X^{(k)}B\left(D^{\mathrm{T}}D\right)^{-1}B^{\mathrm{T}}X^{(k+1)}-X^{(k)}B\left(D^{\mathrm{T}}D\right)^{-1}B^{\mathrm{T}}X^{(k)}.
\end{aligned}$$

Next, we extricate the term $\left(A+BF_2\left(X^{(k)}\right)\right)^{\mathrm{T}}X^{(k+1)}+X^{(k+1)}\left(A+BF_2\left(X^{(k)}\right)\right)$ and continue with some matrix manipulations. We derive

$$\begin{aligned}
0 = &\left(A+BF_2\left(X^{(k)}\right)\right)^{\mathrm{T}} X^{(k+1)} + X^{(k+1)}\left(A+BF_2\left(X^{(k)}\right)\right) \\
&+C^{\mathrm{T}}C+\gamma^{-2}X^{(k+1)}GG^{\mathrm{T}}X^{(k+1)}+\gamma^{-2}\left(X^{(k+1)}-X^{(k)}\right)GG^{\mathrm{T}}\left(X^{(k+1)}-X^{(k)}\right) \\
&+X^{(k)}B\left(D^{\mathrm{T}}D\right)^{-1}B^{\mathrm{T}}X^{(k)}-\left(X^{(k+1)}-X^{(k)}\right)B\left(D^{\mathrm{T}}D\right)^{-1}B^{\mathrm{T}}\left(X^{(k+1)}-X^{(k)}\right).
\end{aligned} \tag{8}$$

We apply the following implementation for the latest recurrence equation:

$$\begin{aligned}
&X^{(0)}=0, \quad 0=A^{\mathrm{T}}X^{(1)}+X^{(1)}A+C^{\mathrm{T}}C, \\
&0=\left(A+BF_2\left(X^{(k)}\right)\right)^{\mathrm{T}} X^{(k+1)} + X^{(k+1)}\left(A+BF_2\left(X^{(k)}\right)\right)+C^{\mathrm{T}}C+\gamma^{-2}X^{(k)}GG^{\mathrm{T}}X^{(k)} \\
&\quad+\gamma^{-2}\left(X^{(k)}-X^{(k-1)}\right)GG^{\mathrm{T}}\left(X^{(k)}-X^{(k-1)}\right)+X^{(k)}B\left(D^{\mathrm{T}}D\right)^{-1}B^{\mathrm{T}}X^{(k)} \\
&\quad-\left(X^{(k)}-X^{(k-1)}\right)B\left(D^{\mathrm{T}}D\right)^{-1}B^{\mathrm{T}}\left(X^{(k)}-X^{(k-1)}\right) \qquad k=1,2,\cdots
\end{aligned} \tag{9}$$

Our thoughts and algebraic manipulations for deriving recurrence Equation (8) show that it is equivalent to the main iterative process (3)-(4). Thus iteration (9) constructs a new matrix sequence which converges to the stabilizing solution of (1). In order to execute iteration (9) we apply the following algorithm:

1) We take $X^{(0)}=0$, and ε as a small positive number.
2) We compute $X^{(1)}$ as a solution to the equation $0=A^{\mathrm{T}}X^{(1)}+X^{(1)}A+C^{\mathrm{T}}C$.
3) For $k=1,2,3,\cdots$ we carry out.
a) Compute $\breve{A}_k=A+BF_2\left(X^{(k)}\right)$ and

$$\begin{aligned}
W_k = &C^{\mathrm{T}}C+\gamma^{-2}X^{(k)}GG^{\mathrm{T}}X^{(k)}+\gamma^{-2}\left(X^{(k)}-X^{(k-1)}\right)GG^{\mathrm{T}}\left(X^{(k)}-X^{(k-1)}\right) \\
&+X^{(k)}B\left(D^{\mathrm{T}}D\right)^{-1}B^{\mathrm{T}}X^{(k)}-\left(X^{(k)}-X^{(k-1)}\right)B\left(D^{\mathrm{T}}D\right)^{-1}B^{\mathrm{T}}\left(X^{(k)}-X^{(k-1)}\right).
\end{aligned}$$

b) Find $X^{(k+1)}$ as a solution to the Lyapunov equation $\breve{A}_k^{\mathrm{T}}X^{(k+1)}+X^{(k+1)}\breve{A}_k+W_k=0$.
c) Algorithm stops when the inequality $\left\|\mathcal{R}\left(X^{(k+1)}\right)\right\|\le\varepsilon$ holds.
4) The stabilizing solution is $X^{*}=X^{(k+1)}$.

3. Numerical Experiments

We carry out experiments for solving a continuous-time algebraic Riccati equation with an indefinite quadratic term (1). We construct two matrix sequences $\left\{X^{(k)}\right\}$ and $\left\{Z^{(k)}\right\}, k=0,1,\cdots$ for each example. The first matrix sequence is computed using the iterative process (3)-(4). Iteration (4) is a Riccati equation and $Z^{(k)}$ is its stabilizing solution. In addition, we apply two iterations (6) and (9) for computing the stabilizing solution to (1), where one matrix sequence is established. We perform iteration (6) in two ways "(6) + care" and "(6) + lyap". We are solving Riccati recurrence Equations (4) and (6) with the MATLAB procedure *care* where the flops are $81n^3$ per one iteration. The MATLAB procedure *lyab* is applied for solving (7) and (9) and the flops are $\frac{27}{2}n^3$ per one iteration.

Moreover, we have carried out experiments in the open source software SCILAB http://www.scilab.org/scilab/about. It provides a computing environment for scientific applications. There are functions for solving linear and nonlinear matrix equations. We apply the "ricc" function for solving a continuous Riccati equation and "lyap" function for a linear Lyapunov equation.

Our experiments are executed in MATLAB on a 2.20 GHz Intel (R) Core (TM) i7-4702MQ CPU computer. We use two variables *tolR* and *tol* for small positive numbers to control the accuracy of computations. We denote $Error_{\mathcal{R},k}=\left\|\mathcal{R}\left(X^{(k)}\right)\right\|$ and $Error_{\mathcal{G},s}=\left\|\mathcal{G}\left(Y_s\right)\right\|$. All iterations stop when the inequality $Error_{\mathcal{R},k_0}\le tolR$ is satisfied for some k_0. That is a practical stopping criterion. However, iteration "(6) + lyap" defines two loops-external and inner. The inner loop stops when the inequality $Error_{\mathcal{R},s_0}\le tol$ is satisfied for some integer s_0.

For our purpose we have executed hundred runs of each value of n for two family of examples. The tables report the maximal number *It* of iterations for which the inequality $Error_{\mathcal{R},It}\le tolR$ holds and the average number $avIt$ of iterations for all hundred runs of each size. In addition, the variable $avIt_L$ stands for the average number of iterations executed by (7) in order to obtain the stabilizing solution through (6) for all hundred runs of each size. For instance, the iteration "(6) + lyap" executes $It=4$ main iterations and $avIt_L=7$ for Example 1. The column "CPU" presents the CPU time for execution the corresponding iterations. In our definitions the functions randn (p, k) and sprand (q, m, 0.3) return a p-by-k matrix of pseudorandom scalar values and a q-by-m sparse matrix respectively (for more information see the MATLAB description).

Example 2. We consider a family of examples in case $n=10,\cdots,15$, where the coefficient real matrices are given as follows: A,G,B,D and *C* were constructed using the MATLAB notations:

$$m=2;\ \gamma=2.5;$$
$$A=abs\left(randn(n,n)\right)/4-4*eye(n,n);$$
$$G=randn(n,m);\qquad B=randn(n,m);$$
$$D^{\mathrm{T}}D=0.45*eye(m,m);\qquad C^{\mathrm{T}}C=0.2*eye(n,n).$$

Results from experiments in Example 2 are given in **Table 1** with $tolR=1e-7, tol=1e-5$ for all values of *n*.

Example 3. We consider a family of examples in case $n=15,\cdots,20$, where the coefficient real matrices are given as follows: A,G,B,D and *C* were constructed using the MATLAB notations:

$$m=2;\ \gamma=2.5;$$
$$A=abs\left(randn(n,n)\right)/6-4*eye(n,n);$$
$$B_1=2*randn(n,m);\qquad B_2=randn(n,m);$$
$$R=0.45*eye(m,m);\qquad Q=0.2*eye(n,n).$$

Results from experiments for Example 3 are given in **Table 2** with $tolR=1e-7, tol=1e-5$ for all values of *n*.

The application of all iterative methods shows that they achieve the same accuracy for different number of iterations. Our conclusions based on experiments are:

1) The execution the iterations (3), (4) and "(6) + care" takes almost the same CPU time (see the corresponding

Table 1. Example 2. Results from 100 runs for each value of *n*.

	(3)-(4)			(6) + lyap (Alg 2)			(6) + care			(9)		
n	*It*	*avIt*	CPU	*It*	*avIt*	CPU	*It*	*avIt*	CPU	*It*	*avIt*	CPU
The MATLAB Execution												
10	3	2.6	0.5 s	4	6.9	0.32 s	4	3.15	0.54 s	5	4.1	0.16 s
11	3	2.7	0.5 s	4	7.2	0.37 s	4	3.26	0.57 s	6	4.22	0.25 s
12	3	2.8	0.56 s	4	7.24	0.39 s	4	3.28	0.62 s	6	4.31	0.22 s
13	3	2.8	0.57 s	4	7.46	0.45 s	4	3.43	0.64 s	7	4.56	0.26 s
14	3	2.9	0.65 s	4	7.7	0.40 s	4	3.62	0.65 s	8	4.86	0.25 s
15	3	2.9	0.76 s	4	7.9	0.49 s	4	3.69	0.75 s	9	5.02	0.37 s
The SCILAB Execution												
10	3	2.6	0.4 s	4	6.9	0.36 s	4	3.10	0.45 s	5	4.0	0.29 s
11	3	2.7	0.5 s	4	7.1	0.40 s	4	3.22	0.58 s	5	4.1	0.31 s
12	3	2.7	0.56 s	4	7.2	0.45 s	4	3.27	0.64 s	6	4.2	0.35 s
13	3	2.8	0.72 s	4	7.5	0.52 s	4	3.38	0.80 s	7	4.4	0.4 s
14	3	2.9	0.77 s	4	7.7	0.55 s	4	3.64	0.94 s	7	4.7	0.45 s
15	3	2.9	0.92 s	4	8.0	0.64 s	4	3.76	1.10 s	8	5.0	0.52 s

Table 2. Example 3. Results from 100 runs for each value of *n*.

	(3)-(4)			(6) + lyap (Alg 2)			(6) + care			(9)		
n	*It*	*avIt*	CPU	*It*	*avIt*	CPU	*It*	*avIt*	CPU	*It*	*avIt*	CPU
The MATLAB Execution												
15	4	3	0.42 s	11	8.3	0.5 s	4	4.0	0.57 s	12	6.6	0.28 s
16	4	3	0.45 s	11	8.3	0.4 s	5	4.0	0.57 s	14	6.7	0.28 s
17	4	3	0.54 s	10	8.5	0.39 s	5	4.0	0.57 s	11	7.0	0.32 s
18	4	3	0.57 s	11	8.8	0.57 s	5	4.0	0.60 s	12	7.5	0.32 s
19	4	3.1	0.61 s	17	8.8	0.56 s	6	4.1	0.68 s	21	7.7	0.38 s
20	4	3.1	0.64 s	12	9.3	0.54 s	5	4.1	0.65 s	26	8.3	0.42 s
The SCILAB Execution												
15	4	3	0.68 s	9	8.1	0.44 s	5	4.0	0.82 s	10	6.2	0.37 s
16	4	3	0.72 s	12	8.5	0.47 s	5	4.0	0.87 s	10	6.5	0.38 s
17	4	3	0.96 s	11	8.5	0.54 s	4	4.0	1.12 s	12	6.7	0.44 s
18	4	3	1.03 s	11	8.7	0.58 s	4	4.0	1.20 s	11	6.9	0.49 s
19	4	3.1	1.21 s	11	8.7	0.63 s	5	4.1	1.40 s	15	7.3	0.57 s
20	4	3.1	1.29 s	12	9.0	0.67 s	7	4.2	1.53 s	25	8.0	0.70 s

columns of the tables). Note that the procedure *care* in these iterations have to be applied;

2) Iterations based on the solution of Lyapunov equations faster than the iterations based on the solution of Riccati equations;

3) The new iteration (9) is fastest than other iterative methods;

4) Comparing the MATLAB Execution and the SCILAB Execution we note the MATLAB implementations of the considered iterative methods are faster than the same executed in the SCILAB environment. However, the

SCILAB implementations achieve the same accuracy and based on the fact it is an open source software we deduce the SCILAB is an useful tool for education to master and PhD students.

The conclusions are indicated by implemented numerical simulations.

4. Conclusion

We have studied two iterative processes for finding the stabilizing solution to generalized Riccati Equations (2). We have made numerical experiments for computing this solution and we have compared the considered methods numerically. We have compared the results from the experiments in regard of number of iterations and CPU time for executing. Our numerical experiments confirm the effectiveness of proposed new method (9). It is introduced here and moreover numerical experiments show its efficiency.

Acknowledgements

The present research paper was supported in a part by the EEA Scholarship Programme BG09 Project Grant D03-91 under the European Economic Area Financial Mechanism.

References

[1] Lanzon, A., Feng, Y., Anderson, B. and Rotkowitz, M. (2008) Computing the Positive Stabilizing Solution to Algebraic Riccati Equations with an Indefinite Quadratic Term via a Recursive Method. *IEEE Transactions on Automatic Control*, **53**, 2280-2291. http://dx.doi.org/10.1109/TAC.2008.2006108

[2] Praveen, P. and Bhasin, S. (2013) Online Partially Model-Free Solution of Two-Player Zero Sum Differential Games, Preprints of the 10th IFAC International Symposium on Dynamics and Control of Process Systems. *The International Federation of Automatic Control*.

[3] Vrabie, D. and Lewis, F. (2011) Adaptive Dynamic Programming for Online Solution of a Zero-Sum Differential Game. *Journal of Control Theory and Applications*, **9**, 353-360. http://dx.doi.org/10.1007/s11768-011-0166-4

[4] Dragan, V., Freiling, G., Morozan, T. and Stoica, A.-M. (2008) Iterative Algorithms for Stabilizing Solutions of Game Theoretic Riccati Equations of Stochastic Control. *Proceedings of the* 18th *International Symposium on Mathematical Theory of Networks & Systems*. http://scholar.lib.vt.edu/MTNS/Papers/078.pdf

[5] Dragan, V. and Ivanov, I. (2011) Computation of the Stabilizing Solution of Game Theoretic Riccati Equation Arising in Stochastic H_∞ Control Problems. *Numerical Algorithms*, **57**, 357-375. http://dx.doi.org/10.1007/s11075-010-9432-7

[6] Feng, Y.T. and Anderson, B.D.O. (2010) An Iterative Algorithm to Solve State-Perturbed Stochastic Algebraic Riccati Equations in LQ Zero-Sum Games. *Systems & Control Letters*, **59**, 50-56. http://dx.doi.org/10.1016/j.sysconle.2009.11.006

[7] Zhu, H.-N., Zhang, C.-K. and Bin, N. (2012) Infinite Horizon LQ Zero-Sum Stochastic Differential Games with Markovian Jumps. *Applied Mathematics*, **3**, 1321-1326. http://dx.doi.org/10.4236/am.2012.330188

[8] Wu, H.-N. and Luo, B. (2013) Simultaneous Policy Update Algorithms for Learning the Solution of Linear Continuous-Time H_∞ State Feedback Control. *Information Sciences*, **222**, 472-485. http://dx.doi.org/10.1016/j.ins.2012.08.012

[9] Basar, T. and Bernhard, P. (1995) H_∞ Optimal Control and Related Minimax Design Problems: A Dynamic Game Approach. *Systems & Control*: *Foundations and Applications Series*, Birkhauser, Boston, 1995.

[10] Lancaster, P. and Rodman, L. (1995) Algebraic Riccati Equations. Clarendon Press, Oxford.