

QUADRATIC ALTERNATING DIRECTION IMPLICIT ITERATION FOR THE FAST SOLUTION OF ALGEBRAIC RICCATI EQUATIONS

Ngai Wong

Department of EEE
The University of Hong Kong
Pokfulam Road, Hong Kong
Email: nwong@eee.hku.hk

Venkataramanan Balakrishnan

School of ECE
Purdue University
West Lafayette, IN, USA
Email: ragu@ecn.purdue.edu

ABSTRACT

Algebraic Riccati equations (AREs) spread over many branches of signal processing and system design problems. Solution of large scale AREs, however, can be computationally prohibitive. This paper introduces a novel second order extension to the alternating direction implicit (ADI) iteration, called quadratic ADI or *QADI*, for the efficient solution of an ARE. QADI is simple to code and exhibits fast convergence. A Cholesky factor variant of QADI, called *CFQADI*, further accelerates computation by exploiting low rank matrices commonly found in physical system modeling. Application examples show remarkable efficiency and scalability of the QADI algorithms over conventional ARE solvers.

1. INTRODUCTION

The solution of an algebraic Riccati equation (ARE) plays an important role in many engineering and scientific applications, and has attracted substantial research [1–8]. Prominent applications of AREs include Kalman filtering, linear quadratic regulator (LQR), and optimal controller design [9, 10]. Recently, VLSI backend design tools are also applying balanced stochastic truncation (BST) for passivity-preserving model order reduction and fast simulation of VLSI circuits [8, 11, 12], wherein a pair of high order AREs need to be solved. Solution of an ARE, however, can be computationally intensive even for medium orders. Generally, there are continuous time and discrete time AREs, denoted by CARE and DARE, respectively. Transformation exists that converts one type into another [4]. This paper focuses on the CARE setting (henceforth simply called ARE). An ARE takes the form

$$A^T X + X A \pm X B B^T X + C^T C = 0 \quad (1)$$

where $A \in \mathbf{R}^{n \times n}$ is assumed stable, $B \in \mathbf{R}^{n \times m}$, and $C \in \mathbf{R}^{p \times n}$. Certain formulations would render a definite matrix in between B and B^T (C^T and C), which can then be factorized and absorbed into B (C), thus (1) is assumed without loss of generality. Different applications produce different signs before $X B B^T X$, and the unified representation in (1) will be adopted throughout this paper. When it is necessary to distinguish the difference, we will refer to them as (1+) or (1-) according to the sign in front of $X B B^T X$. In particular, in (1+), we further assume

$\sup \bar{\sigma}(C(j\omega - A)^{-1}B) < 1, \forall \omega \in \mathbf{R}$, where $\bar{\sigma}(\circ)$ stands for the maximum singular value. This extra condition in (1+) is equivalent to the H_∞ norm of the state space (A, B, C) being less than one [10], or, when A, B , and C are obtained from certain system matrices (see e.g., [12]), the original system is passive. Using $M > 0$ ($M \geq 0$) to denote a positive definite (positive semidefinite) matrix M , and defining $\text{spec}(\circ)$ to be the spectrum of a matrix, and \mathbf{C}^- to be the open left half plane, the above assumptions guarantee the existence of a unique *stabilizing solution*, $X(\in \mathbf{R}^{n \times n}) \geq 0$, that solves (1) and satisfies $\text{spec}(A \pm B B^T X) \subset \mathbf{C}^-$.

Almost all existing ARE solvers work with the *Hamiltonian matrix*, namely,

$$H = \begin{bmatrix} A & \pm B B^T \\ -C^T C & -A^T \end{bmatrix}. \quad (2)$$

The stabilizing solution is then found by identifying the invariant subspace of (2) associated with the set of stable eigenvalues $\text{spec}(H) \cap \mathbf{C}^-$. Representative algorithms are eigenvector and Schur vector methods, matrix sign function method, multishift algorithm etc. (see [1–5] and references therein). Alternatively, the Newton method is a non-Hamiltonian approach that treats (1) as a system of nonlinear equations and computes the solution by solving a linear matrix equation, called Lyapunov equation, in each iteration step [6–9]. This approach has high numerical accuracy, but is mainly used for final refinement because it requires a good initial condition (not always available) to guarantee convergence.

On the other hand, in the context of large scale Lyapunov equations (linear counterparts of AREs), alternating direction implicit (ADI) [13] provides an efficient way to iteratively compute or approximate the solution. Recent results also feature a Cholesky factor (CF) ADI variant that directly obtains the square-root solution [6, 14]. This enables exploitation of low rank and/or sparse matrices to speed up computation and reduce memory requirement. When Newton method is employed for solving an ARE, such CF ADI scheme (called CF-ADI in [14]) may also be used in the Lyapunov equation in each Newton step, thus indirectly forming a square-root solution to an ARE [6, 8]. A natural question is whether there exists an ADI-like algorithm, possibly with a CF variant, that directly solves an ARE. Fortunately, the answer is yes.

The main contribution of this paper is the generalization of ADI to a second order version, called quadratic ADI or *QADI*, that efficiently solves a (large scale) ARE. Well-posedness and convergence of QADI are analytically proven. As in ADI, QADI features a CF variant, called *CFQADI*, that operates on square-root

This work was supported in part by the Hong Kong Research Grants Council, the University Research Committee of The University of Hong Kong, and the National Science Foundation of the United States of America under Grant No. ECS-0200320.

iterates. Compared to Hamiltonian-based algorithms, this non-Hamiltonian QADI approach allows direct exploitation of low rank and/or sparse system matrices. It has better scalability and is more favorable toward progressive solution of large scale AREs. Section 2 of this paper revises the basics of ADI. Section 3 presents QADI and CFQADI. A combined proof for their well-posedness and convergence is also given. Numerical experiments in Section 4 demonstrate the superiority of the QADI approach over conventional solvers. Finally, Section 5 draws the conclusion.

2. BASICS OF ADI

This section gives a brief account of the alternating direction implicit (ADI) method [13, 14]. Key results necessary for later sections are presented. Specifically, ADI attempts to solve the Lyapunov equation

$$A^T W + W A + C^T C = 0 \quad (3)$$

where the matrix dimensions are consistent with those in (1). We assume $\text{spec}(A) \subset \mathbf{C}^-$ so there exists a $W (\in \mathbf{R}^{n \times n}) \geq 0$ that solves (3). The original ADI consists of two ‘‘half-steps’’ in each iteration

$$(A^T + p_j I) W_{j-\frac{1}{2}}^T = -C^T C - W_{j-\frac{1}{2}}^T (A - p_j I) \quad (4a)$$

$$(A^T + p_j I) W_j = -C^T C - W_{j-\frac{1}{2}} (A - p_j I) \quad (4b)$$

where $W_0 = 0$ and the shift parameters $p_j \in \mathbf{C}^-$ ($j = 1, 2, \dots$) appear as real numbers or conjugate pairs. For compactness we define $S_j = (A + p_j I)^{-1}$ and $T_j = (A - p_j I)$. A useful fact is that for any integers m and n , the multiplication among S_m, T_n , and A is commutative, and similarly for S_m^T, T_n^T , and A^T . From (4) we have

$$W_j = - \sum_{i=1}^j 2p_i \left(\prod_{k=1}^{i-1} S_k^T T_k^T \right) S_i^T C^T C S_i \left(\prod_{k=1}^{i-1} T_k S_k \right) \quad (5)$$

so all W_j s are symmetric. In [14] it is shown that the ordering of shift parameters in (5) is immaterial. Combining (3) and (4),

$$W - W_j = \left(\prod_{k=1}^j S_k^T T_k^T \right) W \left(\prod_{k=1}^j T_k S_k \right). \quad (6)$$

To achieve the best convergence in, say, L runs of (4), p_j s are chosen according to the minimax problem

$$\min_{\{p_1, p_2, \dots, p_L\}} \left(\max_{\lambda_i \in \text{spec}(A)} \left| \prod_{j=1}^L \frac{p_j - \lambda_i}{p_j + \lambda_i} \right| \right). \quad (7)$$

Moreover, from (6) we can derive a ‘‘residual error’’ for the Lyapunov operator evaluated at W_j , namely,

$$A^T W_j + W_j A + C^T C = \left(\prod_{k=1}^j S_k^T T_k^T \right) C^T C \left(\prod_{k=1}^j T_k S_k \right) \quad (8)$$

where it is straightforward to see that the norm of the right hand side of (8) converges to zero as j approaches infinity, with a speed dependent on the (approximate) solution of (7).

3. QUADRATIC ADI

The following second order generalization of ADI, called *quadratic ADI* or *QADI*, is proposed for solving (1):

$$(A^T \pm X_{j-1}^T B B^T + p_j I) X_{j-\frac{1}{2}}^T = -C^T C - X_{j-1}^T (A - p_j I) \quad (9a)$$

$$(A^T \pm X_{j-\frac{1}{2}}^T B B^T + p_j I) X_j = -C^T C - X_{j-\frac{1}{2}} (A - p_j I) \quad (9b)$$

where $X_0 = 0$ and $p_j \in \mathbf{C}^-$, $j = 1, 2, \dots$, are either real or conjugate pairs. It can immediately be seen that (9) reduces to (4) when $B = 0$. For ease of illustration we will assume, for the rest of the paper, all p_j s are negative real. However, all qualitative results hold for conjugate pairs if we combine two runs of (9) into one such that all quantities remain real. It will be shown that, as in ADI, X_j converges to X when j tends to infinity. One may also merge the two half-steps in (9) into one. Again, using the definitions $S_j = (A + p_j I)^{-1}$ and $T_j = (A - p_j I)$, it can be carefully shown that

$$X_j = M_{11} + M_{12} X_{j-1} (I - M_{22} X_{j-1})^{-1} M_{12}^T \quad (10)$$

where

$$M_{11} = -2p_j S_j^T C^T (I \mp C S_j B B^T S_j^T C^T)^{-1} C S_j \quad (11a)$$

$$M_{12} = I - 2p_j S_j^T (I \mp C^T C S_j B B^T S_j^T)^{-1} \quad (11b)$$

$$M_{22} = \mp 2p_j S_j B (I \mp B^T S_j^T C^T C S_j B)^{-1} B^T S_j^T. \quad (11c)$$

For the (1-) case, the matrix inverses in (11a)-(11c) are easily seen to be well-defined. For the (1+) case, invertibility follows from the assumptions in (1) which ensure $\bar{\sigma}(C S_j B) < 1$. Well-posedness of the inverse in (10) will be shown in Section 3.2. Also, it can be seen that a symmetric X_{j-1} implies a symmetric X_j . Since $X_0 = 0$, all X_j s are symmetric.

3.1. Cholesky Factor Variant

As in ADI, in the presence of low rank matrices B and C , it is desirable for QADI to work with the Cholesky factor (CF) or square-root iterate Z_j where $X_j = Z_j Z_j^T$. Utilizing (10) and (11), we formulate a CF variant of QADI called *CFQADI*. In particular, setting $Z_0 = 0$ and for $j = 1, 2, \dots$,

$$1. M_{11}^{\frac{1}{2}} = \sqrt{-2p_j} S_j^T C^T (I \mp C S_j B B^T S_j^T C^T)^{-\frac{1}{2}} \quad (12a)$$

$$2. M_{12} = I - 2p_j S_j^T (I \mp C^T C S_j B B^T S_j^T)^{-1} \quad (12b)$$

$$3. M_{22} = \mp 2p_j S_j B (I \mp B^T S_j^T C^T C S_j B)^{-1} B^T S_j^T \quad (12c)$$

$$4. Z_j = [M_{11}^{\frac{1}{2}} \quad M_{12} Z_{j-1} (I - Z_{j-1}^T M_{22} Z_{j-1})^{-\frac{1}{2}}]. \quad (12d)$$

It can be observed that in each run of CFQADI, i.e., for each j , the number of columns in Z_j grows by the number of rows in C . The iteration is continued until the update $\|Z_j Z_j^T - Z_{j-1} Z_{j-1}^T\|$ is smaller than a preset tolerance. When we have low rank B and C , matrix inversion lemma can further be utilized in (12a)-(12c) to reduce arithmetics. In summary, in case of low rank system matrices, CFQADI provides significant computational and memory savings (only low rank factors are stored). Moreover, symmetry of X_j is perfectly preserved by reconstruction from Z_j .

3.2. Well-Posedness and Convergence

Here we prove the well-posedness and convergence of the basic QADI in (10). The properties carry over to CFQADI since it is mathematically equivalent. Define $\tilde{A} = A \pm B B^T X$, and with the assumption of a stabilizing solution X to (1), we have $\text{spec}(\tilde{A}) \subset \mathbf{C}^-$. Let $\tilde{S}_j = (\tilde{A} + p_j I)^{-1}$ and $\tilde{T}_j = (\tilde{A} - p_j I)$, the key of the

proof is to rewrite (9a) and (9b) by the knowledge of (1), namely,

$$X - X_{j-\frac{1}{2}} = -\tilde{T}_j^T (X - X_{j-1}) \left(I \mp \tilde{S}_j B B^T (X - X_{j-1}) \right)^{-1} \tilde{S}_j \quad (13a)$$

$$X - X_j = -\tilde{S}_j^T (X - X_{j-\frac{1}{2}}) \left(I \mp B B^T \tilde{S}_j^T (X - X_{j-\frac{1}{2}}) \right)^{-1} \tilde{T}_j. \quad (13b)$$

Substituting (13a) into (13b) we get

$$X - X_j = \tilde{S}_j^T \tilde{T}_j^T (X - X_{j-1}) \left(I \mp 2p_j \tilde{S}_j B B^T \tilde{S}_j^T (X - X_{j-1}) \right)^{-1} \tilde{T}_j \tilde{S}_j \quad (14)$$

which is equivalent to (10). Applying (14) recursively to itself and noting $X_0 = 0$, we get

$$X - X_j = \Pi_j^T X (I \pm \Omega_j X)^{-1} \Pi_j \quad (15)$$

where

$$\Pi_j = \left(\prod_{k=1}^j \tilde{T}_k \tilde{S}_k \right) = \left(\prod_{k=1}^j \tilde{S}_k \tilde{T}_k \right)$$

$$\Omega_j = - \sum_{i=1}^j 2p_i \left(\prod_{k=1}^{i-1} \tilde{S}_k \tilde{T}_k \right) \tilde{S}_i B B^T \tilde{S}_i^T \left(\prod_{k=1}^{i-1} \tilde{T}_k^T \tilde{S}_k^T \right).$$

Therefore, the QADI described by (10) is well-defined if and only if the inverse in (15) is well-defined for every j , $j = 1, 2, \dots$. An important observation is that $\Omega_j \geq 0$ is exactly the j th iterate of the ADI solution (c.f. (5)) to the Lyapunov equation

$$\tilde{A} \Omega + \Omega \tilde{A}^T + B B^T = 0. \quad (16)$$

That is, we have $\Omega \geq \Omega_j \geq 0$. For the (1+) case, the inverse in (15), i.e., $(I + \Omega_j X)^{-1}$, is obviously valid as the product of two positive semidefinite matrices contains only non-negative eigenvalues. For the (1-) case, it can be shown that $\bar{\sigma}(X^{1/2} \Omega X^{1/2}) < 1$ which in turn implies $(I - \Omega_j X)^{-1}$ is valid for all j s. Lastly, convergence of QADI (and therefore that of CFQADI) can readily be deduced from (15), namely,

$$\|X - X_j\| \leq K_1 \left\| \Pi_j^T X \Pi_j \right\| \leq K_2 \left(\max_{\lambda_i \in \text{spec}(\tilde{A})} \left| \prod_j \frac{p_j - \lambda_i}{p_j + \lambda_i} \right| \right)^2 \quad (17)$$

where K_1, K_2 are positive lumped constants. This shows that, as in ADI, QADI exhibits (super-)linear convergence. Apparently, the convergence rate depends on p_j s whose choice is analogous to the minimax problem in ADI, c.f. (7). The major difference is that the shift parameters p_j , $j = 1, 2, \dots, L$, should now be chosen with respect to the spectrum of \tilde{A} .

4. NUMERICAL EXAMPLES

We study the solution of AREs stemming from different engineering applications. The focus here is on the speed of different solvers operating on the same ARE, rather than the origination or formulation of these AREs. The QADI in (10) and CFQADI in (12) are coded in MATLAB m-files (ordinary text files) and executed, without compilation, in the MATLAB R14 environment. They are compared against the MATLAB subroutine *aresolv* with the *schur*

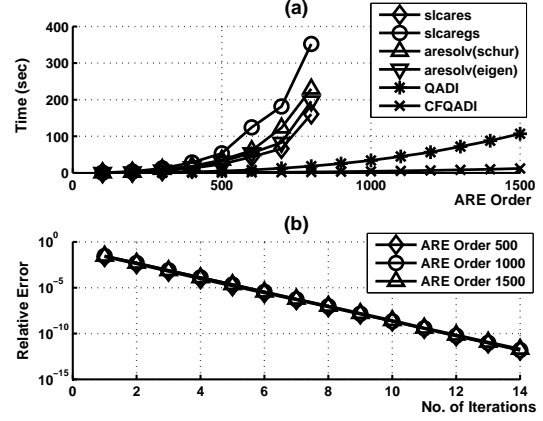


Fig. 1. (a) CPU time for solving an ARE of type (1+); (b) convergence of X_j to the stabilizing X at several ARE orders.

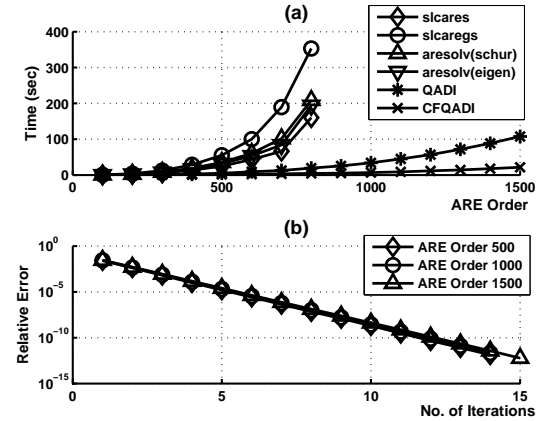


Fig. 2. (a) CPU time for solving an ARE of type (1-); (b) convergence of X_j to the stabilizing X at several ARE orders.

and *eigen* flags enabled successively. The former option implements the Schur vector method, while the latter uses the eigenvector method [2]. The other two solvers, *slcares* (Schur vector method) and *slcaregs* (generalized Schur vector method), are invoked from the SLICOT library [5]. These are prebuilt Fortran 77 subroutines written with numerically reliable, robust, and efficient algorithms. For fairness, ARE solutions from QADI and CFQADI are computed to the same or better accuracies than those from other solvers. The experiments were done on a 3GHz desktop with 1G RAM.

The first class of AREs correspond to (1+) and contain the term $+X B B^T X$. These AREs are taken from passivity-preserving model order reduction problems (e.g., in VLSI simulation [11, 12]) where B and C are both of rank one. Fig. 1(a) plots the CPU time for solving an ARE against its order. It is immediately seen that QADI and its CF variant have superior speed and scalability over conventional solvers. Compared to the fastest Hamiltonian solver *slcares* in our example, at the order of 800, QADI is more than 8X faster and CFQADI is even more than 64X. As for *aresolv(eigen)*,

the gains are about 10X and 80X, respectively. It should be noted that the original QADI of (10) does not explicitly rely on low rank matrices, and is suitable when B and C are of high/full ranks (i.e., where CFQADI does not bring about savings). However, when low rank B and C are present, CFQADI should always be adopted to achieve memory savings and fast computation (as is obvious from the plot). Fortunately, this is often the case, say, in the modeling of large scale electrical networks with only a small number of input/output ports. To see how QADI converges, Fig. 1(b) plots the metric $\|X_j - X\|_F / \|X\|_F$ ($\|\cdot\|_F$ being the Frobenius norm of a matrix) against the number of iterations at several ARE orders. From these straight-line curves, (super-)linear convergence of QADI is obvious, which carries over to CFQADI. The almost overlapped curves also verify that QADI converges irrespective of the ARE order.

The second class of AREs correspond to (1-) and bear the term $-XBB^T X$. Origins of these AREs include Kalman filters and linear quadratic regulator problems [9]. To introduce variations, the ranks of B and C in this scenario are both set to the ARE order divided by 100. Fig. 2 shows similar observations as before. This is not surprising as the sign difference does not introduce much effect on the operations and properties of the solver algorithms. At order 800, in comparison to *slcares*, the speed gains in QADI and CFQADI are more than 8X and 36X, respectively. These results demonstrate that QADI and its CF variant, which are quadratic matrix equation solvers, inherit essentially all the desirable properties of ADI and its CF variant, which are linear matrix equation solvers, without much increase in algorithmic complexity.

Additional Remarks:

1. To the knowledge of the authors, QADI is a new (non-Hamiltonian) ARE solver. CFQADI is the first iterative scheme that directly computes the square-root solution of an ARE. This is in contrast to the “indirect” factor obtained from accumulation of CF solutions to the Lyapunov equations (in each Newton step), which intrinsically leads to a factor of high dimension [6, 8].

2. The runtime of QADI or CFQADI is mainly determined by the number of shifts. The most expensive step is the matrix inversion in finding S_j for each p_j , which takes roughly $3n^3$ flops in the general case when A is dense. If there are L shifts, the work of both algorithms is proportional to $3Ln^3$. QADI has another $O(n^3)$ component proportional to the number of iterations (which is always less than 20 in our examples), while all other operations in CFQADI are of $O(n^2)$ due to exploitation of low rank matrices. In short, work of QADI or CFQADI increases in a cubic manner, but much more slowly compared to conventional solvers. If matrix inversion can be done in $O(n^2)$ work, e.g., when A is sparse or banded, then both algorithms will reduce to $O(n^2)$ complexity.

3. QADI and CFQADI are very simple to code. Both algorithms converge with the simple initial condition $X_0 = 0$. The square-root factor in CFQADI may readily be adapted to other applications, e.g., BST model reduction, to provide further speedup.

4. For simplicity and demonstrative purpose, only a single p_j is used in the numerical experiments, which is analogous to the Smith method as a special case of ADI [6, 8]. In our examples, using $\lambda_{max}(\circ)$ and $\lambda_{min}(\circ)$ to denote the maximum- and minimum-modulus eigenvalues, we set $p \approx -\sqrt{|\lambda_{max}(\tilde{A})\lambda_{min}(\tilde{A})|} = -\sqrt{|\lambda_{max}(H)\lambda_{min}(H)|}$ [8]. Here p is approximated by simple power iterations whose impact on the overall cost is relatively insignificant. It is expected that with multiple properly chosen p_j s,

QADI and CFQADI will converge even faster despite a bigger overhead in finding p_j s and more explicit inversions. Research is being done along this line.

5. CONCLUSION

This paper has devised a quadratic extension to the alternating direction implicit (ADI) algorithm, called QADI, for the efficient solution of (large scale) algebraic Riccati equations (AREs) commonly found in signal processing and some VLSI applications. Well-posedness and convergence of QADI have been analytically proven and characterized. It has been shown that QADI also facilitates a Cholesky factor variant, called CFQADI, to exploit low rank system matrices in some ARE formulations, thus enabling further memory savings and faster computation. Application examples have confirmed the remarkable computational efficiency of QADI and CFQADI over conventional solvers.

6. REFERENCES

- [1] A. J. Laub, “A Schur method for solving algebraic Riccati equations,” *IEEE Trans. Automat. Contr.*, vol. 24, no. 6, pp. 913–921, Dec. 1979.
- [2] I. W. F. Arnold and A. J. Laub, “Generalized eigenproblem algorithms and software for algebraic Riccati equations,” *Proc. IEEE*, vol. 72, no. 12, pp. 1746–1754, Dec. 1984.
- [3] J. Gardiner and A. Laub, “A generalization of the matrix-sign-function solution for algebraic Riccati equations,” *Int. J. Control*, vol. 44, pp. 823–832, 1986.
- [4] P. Benner, A. J. Laub, and V. Mehrmann, “Benchmarks for the numerical solution of algebraic Riccati equations,” *IEEE Control Syst. Mag.*, vol. 17, no. 5, pp. 18–28, 1997.
- [5] P. Benner, V. Mehrmann, V. Sima, S. Van-Huffel, and A. Varga, “SLICOT—a subroutine library in systems and control theory,” *Appl. and Comput. Contr., Signals, and Circuits*, vol. 1, pp. 499–539, 1999.
- [6] T. Penzl, “A cyclic low rank Smith method for large sparse Lyapunov equations with applications in model reduction and optimal control,” *SIAM J. SCI. COMPUT.*, vol. 21, no. 4, pp. 1401–1418, 2000.
- [7] P. Benner and R. Byers, “An exact line search method for solving generalized continuous-time algebraic Riccati equations,” *IEEE Trans. Automat. Contr.*, vol. 43, no. 1, pp. 101–107, Jan. 1998.
- [8] N. Wong, V. Balakrishnan, C.-K. Koh, and T. S. Ng, “A fast Newton-Smith algorithm for solving algebraic Riccati equations and its application in model order reduction,” in *Proc. IEEE Conf. Acoustics, Speech, and Signal Processing*, May 2004, pp. 53–56.
- [9] V. Mehrmann, *The Autonomous Linear Quadratic Control Problem: Theory And Numerical Solution*, ser. No. 163 Lecture Notes in Control and Information Sciences. Berlin, Heidelberg: Springer-Verlag, Nov. 1991.
- [10] K. Zhou, J. C. Doyle, and K. Glover, *Robust and Optimal Control*. Upper Saddle River, NJ: Prentice-Hall, 1996.
- [11] J. R. Phillips, L. Daniel, and L. M. Silveira, “Guaranteed passive balancing transformations for model order reduction,” *IEEE Trans. Computer-Aided Design*, vol. 22, no. 8, pp. 1027–1041, Aug. 2003.
- [12] N. Wong, V. Balakrishnan, and C.-K. Koh, “Passivity-preserving model reduction via a computationally efficient project-and-balance scheme,” in *Proc. IEEE Design, Automation Conf.*, June 2004, pp. 369–374.
- [13] A. Lu and E. L. Wachspress, “Solution of Lyapunov equations by alternating direction implicit iteration,” *Computers Math. Appl.*, vol. 21, no. 9, pp. 43–58, 1991.
- [14] J. Li and J. White, “Low-rank solution of Lyapunov equations,” *SIAM Review*, vol. 46, no. 4, pp. 693–713, 2004.