IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 13, 2021, accepted July 23, 2021, date of publication July 26, 2021, date of current version August 3, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3100432

Explicit Symplectic-Precise Iteration Algorithms for Linear Quadratic Regulator and Matrix Differential Riccati Equation

HONG-YAN ZHANG[1], JIA-ZHEN LUO[2], AND YU ZHOU[1]
[1]School of Information Science and Technology, Hainan Normal University, Haikou 571158, China
[2]Sino-European Institute of Aviation Engineering (SIAE), Civil Aviation University of China, Tianjin 300300, China

Corresponding author: Hong-Yan Zhang (hongyan@hainnu.edu.cn)

This work was supported in part by the Key R & D Plan Projects in Hainan Province of China under Grant ZDYF2019010, in part by the Hainan Provincial Natural Science Foundation of China under Grant 2019RC199, and in part by the Hainan Academician Innovation Platform under Grant YSPTZX202036.

ABSTRACT Efficient, robust and precise algorithms for linear quadratic regulator (LQR) and matrix differential Riccati equation (MDRE) are essential in optimal control. However, there are lack of good algorithms for time-varying LQR problem because of the difficulty of solving the nonlinear time-varying MDRE. In this paper, we proved that the n-th order LQR problem is equivalent to n parallel 1-dim Hamiltonian systems and proposed the explicit symplectic-precise iteration method (SPIM) for solving LQR and MDRE. The explicit symplectic-precise iteration algorithms (ESPIA) designed with SPIM have three typical merits: firstly, there are no accumulative errors in the sense of long-term time which inherits from symplectic difference scheme; secondly the stiffness problem due to the inverse of matrix is avoided by the precise iteration method; and finally the algorithmic structure of ESPIA is simple and no extra assumptions are required. Systematic analysis shows that the time complexity of the symplectic algorithms for the n-th order LQR and MDRE is $\mathcal{O}(k_{max}n^3)$ where k_{max} is the iteration times specified by the time duration. Numerical examples and simulations are provided to validate the performance of the ESPIA.

INDEX TERMS Algorithm design and analysis, optimal control, Riccati equations, computational complexity.

I. INTRODUCTION

Linear Quadratic Regulator (LQR) problem has been extensively studied since 1960 when Kalman formed the basis of the linear quadratic optimal control theory [1]. The purpose of LQR problem is to find a state-feedback control law for the linear system that minimizes the integral quadratic cost functional [2]. In general the solution to LQR is expressed in terms of the optimal feedback gain matrix, which is defined by the solution to the *matrix differential Riccati equation* (MDRE) [3], [4]. Thus the LQR problem is reduced to solve the MDRE, which plays an important role in many applications, such as optimal control, estimation, communication and many others [1], [3], [5], [6]. Although the existence of DRME is discussed in [7] and [8] in details; however, deriving the closed-form or analytical solution (AS) to MDRE in an explicit form is unlikely in most situations, especially in terms of time-varying systems.

Up to now, many numerical algorithms have been proposed to solve the time-invariant and time-varying MDREs. The direct integration methods, such as classic Runge-Kutta method, linear multi-step method, etc [9], [10], treat the MDRE as the ordinary differential equation. These algorithms do not always obtain accurate results due to the stiffness problem and the errors accumulated for a long time duration. Lainiotis [11] derived a doubling partitioned numerical algorithm to solve the MDRE. Kenney *et al.* [10], [12] conducted an extensive study of algorithms for solving the MDRE in 1985, including the Chandrasekhar decomposition algorithm, Leipnik method, Davison-Maki algorithm and other modified approaches. Benner and Mena [13] proposed an efficient matrix-valued implementation of the backward

The associate editor coordinating the review of this manuscript and approving it for publication was Sun-Yuan Hsieh.

differentiation formulas for large-scale MDRE. It should be noted that most of these approaches have at least one of the following drawbacks:

- Being restricted to the linear time-invariant (LTI) system [10];
- Performing very well for time-invariant systems but badly for time-varying problems [12], [14];
- Only special cases are discussed and cannot be generalized [13], [15], [16].

In addition, there exists another kind of method, like the Anderson-Moore method and the Lyapunov equation approach [17], [18], which derives the solution to MDRE from the corresponding algebraic Riccati equation (ARE). This kind of methods works well in the reduction of computation, nevertheless, it focuses on certain conditions and lost generality.

Since the majority of real systems are time dependent, it is essential to develop an applicable method for solving the time-varying MDRE. Hu [19] proposed a method to convert the MDRE to a linear differential Hamiltonian system with two diffferent terminal conditions. However, there are some perspectives which could be pushed forward:

- The Hamiltonian equation proved is based on the terminal condition, which lacks of generality.
- The description of the Hamiltonian equation can be simplified much more and the parallel features for high performance computation are not explored.
- Only the Runge-Kutta (RK) method is discussed and its computational complexity is missing.
- The time-varying system for the general LQR problem is not discussed and there is a lack of numerical examples for the time-varying cases.
- There is a large gap between the numerical method proposed and the engineering applications since clear algorithmic description is missing.

Mceneaney [6] derived a unifying representation of the solution to the time-varying MDRE by utilizing the max-plus fundamental solution. This approach works well for stiff time-varying MDRE, yet it encounters numerical instability due to the intolerance of small errors. Zhong and Williams [20] proposed the *precise integration method* (PIM) for the stiffness problem of ordinary differential equation. Ton and Zhong [21], [22] proposed an improved precise integration method (IPIM) to calculate the MDRE efficiently with variable coefficients for time-varying systems, which results in stable and accurate solutions by using incremental technique. However, the algorithmic structure of IPIM is rather complicated and lacks beauty.

To overcome the disadvantages of the previous methods, we present the *symplectic-precise iteration method* (SPIM) for the MDRE and LQR with time-varying coefficients with a two-step strategy:

- Transforming the n-th order nonlinear MDRE into its equivalent form, i.e., n parallel 1-dim Hamiltonian canonical differential equation of n separate linear non-autonomous Hamiltonian systems, via the Heisenberg picture in quantum mechanics.
- Constructing symplectic integrator [23] by the second order time-centered Euler implicit scheme (T-CEIS) [24] and precise integration method for the equivalent Hamiltonian equations.

It should be noted that T-CEIS is a symplectic difference scheme for non-autonomous system, which has a simple algorithmic structure and preserves the symmetry and positivity of the solution to MDRE. In order to keep the round-off error small and avoid the stiffness problem, the PIM is introduced to calculate the symplectic transition operator accurately. The general advantage of symplectic method and its application in optimal control lies in the fact that it can keep the symplectic structure and avoid cumulative computational errors in the sense of long-term time [24]–[27].

The main purpose of this paper is to present explicit *symplectic-precise iteration algorithms* (ESPIA) for the LQR and MDRE. The rest of this paper is organized as follows: Section II deals with the parallel Hamiltonian equations for the LQR problem; Section III presents the SPIM and ESPIA for LQR and MDRE; Section IV discusses the time complexity of the ESPIA; Section V concerns verification and evaluation by simulation results and Section VI gives the conclusions.

II. PARALLEL TIME-VARYING HAMILTONIAN EQUATIONS FOR LQR

A. FINITE-HORIZON AND CONTINUOUS-TIME LQR

The finite-horizon and continuous-time LQR defined on the time interval $[t_0, t_f]$ is [1]

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{1}$$

with a cost functional

$$C = \frac{1}{2}\boldsymbol{x}^{\mathsf{T}}(t_f)\boldsymbol{S}\boldsymbol{x}(t_f) + \frac{1}{2}\int_{t_0}^{t_f}\left[\boldsymbol{x}^{\mathsf{T}}(t)\boldsymbol{Q}(t)\boldsymbol{x}(t) + \boldsymbol{u}^{\mathsf{T}}(t)\boldsymbol{R}(t)\boldsymbol{u}(t)\right]\mathrm{d}\,t \tag{2}$$

where $\boldsymbol{x} = \boldsymbol{x}(t) \in \mathbb{R}^{n\times 1}$ is an n-dim state vector, $\boldsymbol{A} = \boldsymbol{A}(t) \in \mathbb{R}^{n\times n}$ is the system matrix, $\boldsymbol{u} = \boldsymbol{u}(t) \in \mathbb{R}^{p\times 1}$ is a p-dim input vector, $\boldsymbol{B} = \boldsymbol{B}(t) \in \mathbb{R}^{n\times p}$ is the coefficient matrix, $\boldsymbol{S} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{Q} = \boldsymbol{Q}(t) \in \mathbb{R}^{n\times n}$ are symmetric and non-negative matrices, and $\boldsymbol{R} = \boldsymbol{R}(t) \in \mathbb{R}^{p\times p}$ is positive. With the help of Lagrange multiplier $\boldsymbol{\lambda}$ we can construct the Hamiltonian $\mathcal{H}$ as

$$\mathcal{H}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, t) = \boldsymbol{\lambda}^{\mathsf{T}}\dot{\boldsymbol{x}} + \frac{1}{2}\boldsymbol{x}^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{x} + \frac{1}{2}\boldsymbol{u}^{\mathsf{T}}\boldsymbol{R}\boldsymbol{u} \tag{3}$$

and find the following optimal input

$$\boldsymbol{u}^* = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{\lambda} \tag{4}$$

for the LQR problem [28]. According to the optimal control law, we can obtain the following linear system [29]

$$\frac{\mathrm{d}}{\mathrm{d}\,t}\begin{bmatrix}\boldsymbol{\lambda}\\ \boldsymbol{x}\end{bmatrix} = \begin{bmatrix}-\boldsymbol{A}^{\mathsf{T}} & -\boldsymbol{Q}\\ -\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}} & \boldsymbol{A}\end{bmatrix}\begin{bmatrix}\boldsymbol{\lambda}\\ \boldsymbol{x}\end{bmatrix}, \tag{5}$$

with initial and terminal conditions

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0, \quad \boldsymbol{\lambda}(t_f) = \boldsymbol{S}\boldsymbol{x}_f. \tag{6}$$

Suppose that the action of $\boldsymbol{P} : \mathbb{R} \to \mathbb{R}^{n\times n}$ on $\boldsymbol{x}(t)$ is $\boldsymbol{\lambda}(t)$, viz.,

$$\boldsymbol{\lambda}(t) = \boldsymbol{P}(t)\boldsymbol{x}(t), \tag{7}$$

then we have the famous continuous-time matrix differential Riccati equation (MDRE) and the terminal condition [2], [4]

$$\dot{\boldsymbol{P}}(t) = -\boldsymbol{P}\boldsymbol{A} - \boldsymbol{A}^{\mathsf{T}}\boldsymbol{P} - \boldsymbol{Q} + \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{P}, \tag{8}$$

$$\boldsymbol{P}(t_f) = \boldsymbol{S}. \tag{9}$$

Although the effect of t is considered in the mathematical notion, it should be remarked that it's rarely the case to deal directly with the time-varying system in engineering applications. A common practice is to regard the time-varying system as constant. Therefore, the parameter t is dropped in (8) and in the subsequent notions. There are three steps for solving the continuous time LQR problem:

a) find $\boldsymbol{P}(t)$ by solving the MDRE with terminal time condition;
b) compute $\boldsymbol{x}$ and $\boldsymbol{\lambda}$;
c) compute the optimal control input $\boldsymbol{u}^*$.

Generally, it is difficult to find the AS to MDRE due to the non-linearity and time-dependence.

B. HEISENBERG MOTION EQUATION OF LQR

We now suppose that the final state vector is $\boldsymbol{x}_f$ and take an invertible operator from the general linear transform group, i.e., $\boldsymbol{X}(t) \in \mathrm{GL}(n, \mathbb{R}) \subset \mathbb{R}^{n\times n}$, and let it act on $\boldsymbol{x}_f$, then we can get the state vector

$$\boldsymbol{x}(t) = \boldsymbol{X}(t)\boldsymbol{x}_f \tag{10}$$

by assuming that both $\boldsymbol{x}(t)$ and $\boldsymbol{x}_f$ are not zero vectors. Thus we can obtain the co-state

$$\boldsymbol{\lambda}(t) = \boldsymbol{P}(t)\boldsymbol{x}(t) = \boldsymbol{P}(t)\boldsymbol{X}(t)\boldsymbol{x}_f = \boldsymbol{Y}(t)\boldsymbol{x}_f \tag{11}$$

such that

$$\boldsymbol{P}(t) = \boldsymbol{Y}(t)\boldsymbol{X}^{-1}(t). \tag{12}$$

We remark that when $\boldsymbol{x}(t)$ becomes zero vector at some time points (or $\boldsymbol{x}(t)$ is zero everywhere), we cannot find an invertible matrix such that (10) and (12) hold. However, fortunately, we can still use the expression $\boldsymbol{Y}(t)\boldsymbol{X}^{-1}(t)$ thanks to the continuity of $\boldsymbol{X}(t)$, $\boldsymbol{Y}(t)$ and $\boldsymbol{P}(t)$.

Note that (12) is different from (7) since we can obtain $\boldsymbol{P}(t)$ directly once $\boldsymbol{X}(t)$ and $\boldsymbol{Y}(t)$ are known. Simple algebraic manipulations show that the corresponding terminal time condition for $\boldsymbol{X}(t)$ and $\boldsymbol{Y}(t)$ can be expressed by

$$\boldsymbol{Y}(t_f) = \boldsymbol{S}, \quad \boldsymbol{X}(t_f) = \boldsymbol{I}_n, \tag{13}$$

where $\boldsymbol{I}_n$ is the n-by-n identity matrix.

By analogy with the theory of quantum mechanics [30]–[32], the way that we describe the interested system with the operators $\boldsymbol{X}(t)$ and $\boldsymbol{Y}(t)$ can be called *Heisenberg picture* and its counterpart with the state $\boldsymbol{x}(t)$ and co-state $\boldsymbol{\lambda}(t)$ can be called *Schrödinger picture*. Quantum mechanics says that these two pictures are equivalent although the Schrödinger equation of quantum state (function) is different from the Heisenberg motion equation of observable (operator or matrix).

Naturally, an interesting problem rises: are there some motion equations for the operators $\boldsymbol{X}(t)$ and $\boldsymbol{Y}(t)$? Fortunately, the answer is YES! Substitute (10) and (11) into (5), we immediately have the following linear time-varying ordinary differential equation for the operators $\boldsymbol{X}(t)$ and $\boldsymbol{Y}(t)$

$$\frac{\mathrm{d}}{\mathrm{d}\,t}\begin{bmatrix}\boldsymbol{Y}\\ \boldsymbol{X}\end{bmatrix} = \begin{bmatrix}-\boldsymbol{A}^{\mathsf{T}} & -\boldsymbol{Q}\\ -\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}} & \boldsymbol{A}\end{bmatrix}\begin{bmatrix}\boldsymbol{Y}\\ \boldsymbol{X}\end{bmatrix} = \boldsymbol{\Phi}\begin{bmatrix}\boldsymbol{Y}\\ \boldsymbol{X}\end{bmatrix} \tag{14}$$

where

$$\boldsymbol{\Phi} = \begin{bmatrix}-\boldsymbol{A}^{\mathsf{T}} & -\boldsymbol{Q}\\ -\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}} & \boldsymbol{A}\end{bmatrix} = \boldsymbol{\Phi}(t) \tag{15}$$

is a time-varying matrix since $\boldsymbol{A}$, $\boldsymbol{Q}$, $\boldsymbol{B}$ and $\boldsymbol{R}$ depend on time t in general. We remark that:

- (14) implies a time-varying Hamiltonian system, which could be named with *Heisenberg motion equation of LQR since the way to obtain it is in the paradigm of methodology of quantum (matrix) mechanics by Werner Heisenberg in 1925 [31].*
- (14) is the same with that in [19] and [28] essentially but our method is more straightforward, more enlightening and more intuitive;
- the final state $\boldsymbol{x}_f$ has been removed from the two sides of the (14) since it can be an arbitrary vector in state space for a controllable system.

C. PARALLEL HAMILTONIAN SYSTEMS FOR LQR

Equation (14) can be regarded as an equivalent description of the LQR problem in Heisenberg's picture, which leads to the following theorem.

Theorem 1 (Parallel Time-varying Hamiltonian Systems): The Heisenberg motion equation of n-th order LQR is equivalent to n parallel 1-dim linear and non-autonomous Hamiltonian systems.

Proof: Let $\boldsymbol{I}_n$ be the n-th order identity matrix, $\boldsymbol{O}_n$ be the n-th zero matrix,

$$\boldsymbol{J} = \begin{bmatrix}\boldsymbol{O}_n & \boldsymbol{I}_n\\ -\boldsymbol{I}_n & \boldsymbol{O}_n\end{bmatrix} \tag{16}$$

be the $2n$-th order standard symplectic matrix and

$$\boldsymbol{K} = \begin{bmatrix}-\boldsymbol{B}(t)\boldsymbol{R}^{-1}(t)\boldsymbol{B}^{\mathsf{T}}(t) & \boldsymbol{A}(t)\\ \boldsymbol{A}^{\mathsf{T}}(t) & \boldsymbol{Q}(t)\end{bmatrix} = \boldsymbol{K}(t), \tag{17}$$

then we can deduce that

$$\boldsymbol{K} = \boldsymbol{K}^{\mathsf{T}} = \boldsymbol{J}\boldsymbol{\Phi} \tag{18}$$

since $\boldsymbol{Q} = \boldsymbol{Q}^{\mathsf{T}}$ and $\boldsymbol{R} = \boldsymbol{R}^{\mathsf{T}}$. Let $\boldsymbol{Z}$ be the column stacking of $\boldsymbol{Y}$ and $\boldsymbol{X}$, i.e.,

$$\boldsymbol{Z} = \begin{bmatrix}\boldsymbol{Y}\\ \boldsymbol{X}\end{bmatrix} = [z_1, \ldots, z_j, \ldots, z_n] \in \mathbb{R}^{2n\times n}, \tag{19}$$

where the j-th column of $\boldsymbol{Z}$ is z_j. Consequently, (14) is equivalent to

$$\frac{\mathrm{d}z_j}{\mathrm{d}t} = \boldsymbol{\Phi} z_j = \boldsymbol{J}^{-1}\boldsymbol{K}z_j, \quad j \in \{1, \ldots, n\}. \tag{20}$$

Now we let the Hamiltonian of z_j for (20) be

$$H(z_j) = \frac{1}{2} z_j^{\mathsf{T}} \boldsymbol{K} z_j, \quad j \in \{1, \ldots, n\}. \tag{21}$$

Then we immediately have

$$\boldsymbol{J}^{-1}\nabla H(z_j) = \boldsymbol{J}^{-1}\boldsymbol{K}z_j = \boldsymbol{\Phi} z_j. \tag{22}$$

Consequently, (20) shows that

$$\boxed{\frac{\mathrm{d}z_j}{\mathrm{d}t} = \boldsymbol{J}^{-1}\nabla H(z_j) = \boldsymbol{\Phi} z_j, \quad j \in \{1, \ldots, n\}} \tag{23}$$

or equivalently

$$\frac{\mathrm{d}\boldsymbol{Z}}{\mathrm{d}t} = \boldsymbol{\Phi}\boldsymbol{Z} = \boldsymbol{J}^{-1}\boldsymbol{K}\boldsymbol{Z}. \tag{24}$$

Obviously, (23) is the canonical Hamiltonian system specified by the Hamiltonian $H(z_j)$ and state vector z_j in the j-th phase space according to the definition given in Appendix . Since each $H(z_j) = \frac{1}{2} z_j^{\mathsf{T}} \boldsymbol{K} z_j = \frac{1}{2} z_j^{\mathsf{T}} \boldsymbol{K}(t) z_j$ depends on time t for column index j, (14) can be split into n parallel linear non-autonomous Hamiltonian canonical equations with identical form and well-defined terminal time condition (13), i.e.,

$$z_j(t_f) = \begin{bmatrix} \boldsymbol{S}(:, j) \\ \boldsymbol{I}_n(:, j) \end{bmatrix}, \quad 1 \le j \le n \Longleftrightarrow \boldsymbol{Z}(t_f) = \begin{bmatrix} \boldsymbol{S} \\ \boldsymbol{I}_n \end{bmatrix} \tag{25}$$

where $\boldsymbol{M}(:, j)$ is the j-th column of the matrix $\boldsymbol{M}$. ■

Although (14) can be found in various textbooks, there is no literature in which its equivalence (20) or (23) is used to design explicit symplectic algorithms for LQR and MDRE. *Theorem 1 is of significance because it shows that we can explore and construct symplectic geometric algorithm to solve the LQR problem as well as the nonlinear MDRE and preserve high precision as well as numerical stability in the sense of long-term time.* This new idea may refresh the current situation when solving the *numerical solution* (NS) to the LQR problem.

We remark that (5) is also related with the Hamiltonian canonical equation since it have the same coefficient matrix $\boldsymbol{\Phi}(t)$ as (14) does. However, there is no proper condition for determining the solution to $\boldsymbol{x}(t)$ and $\boldsymbol{\lambda}(t)$ because we just have the mixed conditions specified by (6) instead of the initial condition $[\boldsymbol{x}(t_0), \boldsymbol{\lambda}(t_0)]^{\mathsf{T}}$ or terminal time condition $[\boldsymbol{x}(t_f), \boldsymbol{\lambda}(t_f)]^{\mathsf{T}}$. Fortunately, we have the well-defined terminal time condition (13) for (14).

We also remark that this theorem is equivalent to the counterpart in [19] (See Theorem 2.2 of [19]) essentially. By comparison, there are three advantages for our proof:

- it is more simple and physics oriented;
- the symplectic method can be adopted without any extra assumption;
- the parallel feature characterized by (20) and (23) can be used for parallel computations with CUDA or GPU.

III. SYMPLECTIC-PRECISE ITERATION METHOD AND ALGORITHMS FOR LQR AND MDRE

A. SYMPLECTIC DIFFERENCE SCHEME FOR LQR PROBLEM

Since the matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{Q}, \boldsymbol{R}$ vary with time t in general, the matrix $\boldsymbol{K}$ depends on time essentially. Thus the Hamiltonian systems described by (23) or (24) must be non-autonomous systems [24], [25], [33], [34]. Therefore, the time-centered Euler implicit scheme (T-CEIS), which is a symplectic difference scheme [24], can be utilized to find the NS to (24).

Let τ be the time step, for integer $k = 0, 1, \ldots$ we denote

$$\begin{aligned}
k_{max} &= (t_f - t_0)/\tau \in \mathbb{N}, \\
t_k &= t_0 + k\tau, \quad \bar{t}_k = (t_k + t_{k+1})/2 = t_k + \tau/2, \\
z_j[k] &= z_j(t_k), \quad \bar{z}_j[k] = (z_j[k+1] + z_j[k])/2, \\
\boldsymbol{Z}[k] &= \left[z_1[k], \ldots, z_n[k]\right] = \begin{bmatrix} \boldsymbol{Y}[k] \\ \boldsymbol{X}[k] \end{bmatrix}, \\
\boldsymbol{A}[k] &= \boldsymbol{A}(\bar{t}_k), \quad \boldsymbol{B}[k] = \boldsymbol{B}(\bar{t}_k), \\
\boldsymbol{Q}[k] &= \boldsymbol{Q}(\bar{t}_k), \quad \boldsymbol{R}[k] = \boldsymbol{R}(\bar{t}_k), \\
\boldsymbol{\Phi}[k] &= \begin{bmatrix} -\boldsymbol{A}[k]^{\mathsf{T}} & -\boldsymbol{Q}[k] \\ -\boldsymbol{B}[k]\boldsymbol{R}[k]^{-1}\boldsymbol{B}[k]^{\mathsf{T}} & \boldsymbol{A}[k] \end{bmatrix} = \boldsymbol{J}^{-1}\boldsymbol{K}[k]
\end{aligned} \tag{26}$$

for the discrete counterparts of the continuous time versions. Then the T-CEIS will be

$$\frac{z_j[k+1] - z_j[k]}{\tau} = \boldsymbol{J}^{-1}\nabla H(\bar{z}_j[k], \bar{t}_k) \tag{27}$$

or equivalently

$$\frac{\boldsymbol{Z}[k+1] - \boldsymbol{Z}[k]}{\tau} = \boldsymbol{\Phi}[k]\frac{\boldsymbol{Z}[k+1] + \boldsymbol{Z}[k]}{2}. \tag{28}$$

This difference scheme will lead to the following theorem for the single step transition map.

Theorem 2: Let $\boldsymbol{I}_{2n}$ be the $2n$-by-$2n$ identity matrix. For positive time step τ, the single step transition mapping which relates $\boldsymbol{Z}[k]$ and $\boldsymbol{Z}[k+1]$ in the difference scheme (28) must be

$$\begin{aligned}
\boldsymbol{G}(\tau) &= \left[\boldsymbol{I}_{2n} - \frac{\tau}{2}\boldsymbol{\Phi}[k]\right]^{-1}\left[\boldsymbol{I}_{2n} + \frac{\tau}{2}\boldsymbol{\Phi}[k]\right] \\
&= [\boldsymbol{G}(-\tau)]^{-1}
\end{aligned} \tag{29}$$

such that $\boldsymbol{G}(\tau)$ is a symplectic matrix, i.e.,

$$\boldsymbol{G}(\tau)^{\mathsf{T}} \cdot \boldsymbol{J} \cdot \boldsymbol{G}(\tau) = \boldsymbol{J}. \tag{30}$$

Proof: The (28) can be reformulated by

$$\begin{aligned}
\boldsymbol{Z}[k+1] &= \boldsymbol{G}(\tau)\boldsymbol{Z}[k] \\
&= \left[\boldsymbol{I}_{2n} - \frac{\tau}{2}\boldsymbol{\Phi}[k]\right]^{-1}\left[\boldsymbol{I}_{2n} + \frac{\tau}{2}\boldsymbol{\Phi}[k]\right]\boldsymbol{Z}[k].
\end{aligned}$$

Let $x = \frac{\tau}{2}, \boldsymbol{C} = \boldsymbol{K}[k], \boldsymbol{F}(x) = \boldsymbol{I}_{2n} - x\boldsymbol{J}^{-1}\boldsymbol{C} = \boldsymbol{I}_{2n} - \frac{\tau}{2}\boldsymbol{\Phi}[k], \boldsymbol{V} = \boldsymbol{G}(\tau) = [\boldsymbol{F}(x)]^{-1}[\boldsymbol{F}(-x)]$. We immediately find that $\boldsymbol{G}(\tau)$ is symplectic by Lemma 4 in Appendix . Moreover, it is trivial that $\boldsymbol{G}(\tau) = [\boldsymbol{G}(-\tau)]^{-1}$. ■

Theorem 2 shows that the symplectic difference scheme for (14) will be

$$\boldsymbol{Z}[k+1] = \boldsymbol{G}(\tau)\boldsymbol{Z}[k], \quad k = 0, \ldots, k_{max} - 1 \tag{31}$$

when updating $\boldsymbol{Z}[k]$ in increasing time order or

$$\boldsymbol{Z}[k] = \boldsymbol{G}(-\tau)\boldsymbol{Z}[k+1], \quad k = k_{max} - 1, \ldots, 0 \tag{32}$$

when updating $\boldsymbol{Z}[k]$ in decreasing time order. Moreover, the $2n$-by-$2n$ matrix $\boldsymbol{G}(\pm\tau)$ can be split into four blocks with the same size (n-by-n), viz.,

$$\boldsymbol{G}(\pm\tau) = \begin{bmatrix} \boldsymbol{G}_{11}^{\pm} & \boldsymbol{G}_{12}^{\pm} \\ \boldsymbol{G}_{21}^{\pm} & \boldsymbol{G}_{22}^{\pm} \end{bmatrix} \tag{33}$$

Substitute (33) into (31) and (32), we can obtain the iterative formulas for $\boldsymbol{X}[k]$ and $\boldsymbol{Y}[k]$ since $\boldsymbol{Z}[k] = [\boldsymbol{Y}[k]^{\mathsf{T}}, \boldsymbol{X}[k]^{\mathsf{T}}]^{\mathsf{T}}$.

It should be noted that the T-CEIS method with second order precision is adopted here. For the T-CEIS method, its form is simple, its precision is high, it leads to explicit symplectic algorithms and it satisfies the requirements of control engineering well. Although higher order symplectic method can be used theoretically and there are lots of candidate high order symplectic methods in [24]. However, the computational complexity will be higher simultaneously since there are lots of matrix operations involved. It is a trade off among various performances and algorithmic structure that we choose Euler's method in the sense of balancing control theory and control engineering.

B. ITERATIVE FORMULA FOR LQR PROBLEM

The (9) for the MDRE is a terminal condition at t_f where we have to choose decreasing order for time t and take $\boldsymbol{G}(-\tau)$ to compute the state at $t < t_f$. Since $\boldsymbol{X}(t_f) = \boldsymbol{X}(t_0 + k_{max}\tau) = \boldsymbol{X}[k_{max}]$ and $\boldsymbol{Y}(t_f) = \boldsymbol{Y}(t_0 + k_{max}\tau) = \boldsymbol{Y}[k_{max}]$, we can obtain

$$\boldsymbol{X}[k_{max}] = \boldsymbol{I}_n, \quad \boldsymbol{Y}[k_{max}] = \boldsymbol{S}. \tag{34}$$

By (12), (32) and (33), the NS to the time-varying matrices $\boldsymbol{X}(t)$, $\boldsymbol{Y}(t)$ and $\boldsymbol{P}(t)$ can be expressed as follows

$$\begin{cases} \boldsymbol{Y}[k] = \boldsymbol{G}_{11}^{-}\boldsymbol{Y}[k+1] + \boldsymbol{G}_{12}^{-}\boldsymbol{X}[k+1], \\ \boldsymbol{X}[k] = \boldsymbol{G}_{21}^{-}\boldsymbol{Y}[k+1] + \boldsymbol{G}_{22}^{-}\boldsymbol{X}[k+1], \\ \boldsymbol{P}[k] = \boldsymbol{Y}[k]\boldsymbol{X}[k]^{-1}, \end{cases} \tag{35}$$

for $k = k_{max} - 1, \ldots, 1, 0$ in time decreasing order.

On the other hand, $\boldsymbol{x}(t) = \boldsymbol{X}(t)\boldsymbol{x}_f$ implies that

$$\boldsymbol{x}[0] = \boldsymbol{x}(t_0) = \boldsymbol{x}_0 = \boldsymbol{X}(t_0)\boldsymbol{x}(t_f) = \boldsymbol{X}[0]\boldsymbol{x}_f,$$

viz.,

$$\boldsymbol{x}_f = \boldsymbol{X}[0]^{-1}\boldsymbol{x}_0. \tag{36}$$

Hence the NS to $\boldsymbol{x}(t)$ must be

$$\boldsymbol{x}[k] = \boldsymbol{X}[k]\boldsymbol{X}[0]^{-1}\boldsymbol{x}_0, \quad k \in \{0, 1, \ldots, k_{max}\} \tag{37}$$

by (10) and (36). Furthermore, since the optimal control input is

$$\boldsymbol{u}^*(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{\lambda}(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{Y}(t)\boldsymbol{x}_f,$$

we can deduce that its NS must be

$$\boldsymbol{u}^*[k] = -\boldsymbol{R}[k]^{-1}\boldsymbol{B}[k]^{\mathsf{T}}\boldsymbol{Y}[k]\boldsymbol{x}_f \tag{38}$$

for $k \in \{0, 1, \ldots, k_{max}\}$. Equations (37) and (38) show that *it is not necessary to calculate the matrix* $\boldsymbol{P}(t)$ *if we only concern about the state vector* $\boldsymbol{x}(t)$ *and the optimal input* $\boldsymbol{u}^*(t)$.

C. ITERATIVE FORMULA FOR MDRE WITH INITIAL CONDITION

If we only concern about the MDRE and show no interest in the LQR problem, (9) can be replaced by the initial condition. Thus we have the following Cauchy problem

$$\begin{cases} \dot{\boldsymbol{P}}(t) &= -\boldsymbol{P}\boldsymbol{A} - \boldsymbol{A}^{\mathsf{T}}\boldsymbol{P} - \boldsymbol{Q} + \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{P}, \\ \boldsymbol{P}(t_0) &= \boldsymbol{P}_0. \end{cases} \tag{39}$$

In this case, we compute $\boldsymbol{X}[k]$, $\boldsymbol{Y}[k]$ and $\boldsymbol{P}[k]$ in time increasing order. Thus the solution to (39) will be

$$\begin{cases} \boldsymbol{Y}[k+1] &= \boldsymbol{G}_{11}^{+}\boldsymbol{Y}[k] + \boldsymbol{G}_{12}^{+}\boldsymbol{X}[k], \\ \boldsymbol{X}[k+1] &= \boldsymbol{G}_{21}^{+}\boldsymbol{Y}[k] + \boldsymbol{G}_{22}^{+}\boldsymbol{X}[k], \\ \boldsymbol{P}[k+1] &= \boldsymbol{Y}[k+1]\boldsymbol{X}[k+1]^{-1}, \end{cases} \tag{40}$$

for $k = 0, 1, \ldots, k_{max} - 1$ in time increasing order with the initial setting

$$\boldsymbol{X}[0] = \boldsymbol{I}_n, \quad \boldsymbol{Y}[0] = \boldsymbol{P}_0, \quad \boldsymbol{P}[0] = \boldsymbol{P}_0. \tag{41}$$

D. COMPUTATION OF SYMPLECTIC TRANSITION MAPPING

The key step for solving LQR problem and MDRE is to calculate the symplectic transition matrix

$$\boldsymbol{G}(\tau) = \left[\boldsymbol{I}_{2n} - \frac{\tau}{2}\boldsymbol{\Phi}[k]\right]^{-1}\left[\boldsymbol{I}_{2n} + \frac{\tau}{2}\boldsymbol{\Phi}[k]\right] \tag{42}$$

reliably with high precision. Generally, the computation of the inverse of a matrix may not be stable or robust, and we need a good algorithm for such a task. Fortunately, we can adopt the idea of the PIM [20], which is an accurate and stable method for computing the exponent of a matrix, and can avoid round-off errors.

Actually, for the small time step τ, (42) can be approximated by the following Taylor series

$$\begin{aligned} \boldsymbol{G}(\tau) &= \boldsymbol{I}_{2n} + \boldsymbol{T}_a(\tau) \\ \boldsymbol{T}_a(\tau) &= \tau\boldsymbol{\Phi}[k] + \frac{\tau^2}{2}\boldsymbol{\Phi}[k]^2 + \frac{\tau^3}{4}\boldsymbol{\Phi}[k]^3 + \mathcal{O}(\tau^4) \\ &= \boldsymbol{a} + \frac{1}{2}\boldsymbol{a}^2 + \frac{1}{4}\boldsymbol{a}^3 + \mathcal{O}(\tau^4) \\ &= \boldsymbol{a}\left[\boldsymbol{I}_{2n} + \frac{1}{2}\boldsymbol{a}\left(\boldsymbol{I}_{2n} + \frac{1}{2}\boldsymbol{a}\right)\right] + \mathcal{O}(\tau^4) \end{aligned} \tag{43}$$

where

$$\boldsymbol{a} = \tau\boldsymbol{\Phi}[k]. \tag{44}$$

If we divide the subinterval τ into 2^N extremely small intervals of equal length $\eta = \frac{\tau}{2^N}$ where N is a positive

integer suggested, then the precise computation of $\boldsymbol{G}(\tau)$ can be implemented by

$$\boldsymbol{G}(\tau) = \underbrace{\boldsymbol{G}(\eta) \circ \cdots \circ \boldsymbol{G}(\eta)}_{\text{composition of } 2^N \text{ terms}} = [\boldsymbol{G}(\eta)]^{2^N} \tag{45}$$

We can split the symplectic transition matrix $\boldsymbol{G}(\eta)$ into the identity matrix $\boldsymbol{I}_{2n}$ and the incremental part $\boldsymbol{T}_a(\eta) = \eta\boldsymbol{\Phi}[k]$, i.e.,

$$\boldsymbol{G}(\eta) = \boldsymbol{I}_{2n} + \boldsymbol{T}_a(\eta) = \boldsymbol{I}_{2n} + \eta\boldsymbol{\Phi}[k]. \tag{46}$$

Then we use an idea of storing and computing the incremental part via the following iteration (updating formula)

$$\boldsymbol{T_a} \leftarrow 2\boldsymbol{T_a} + \boldsymbol{T_a} \cdot \boldsymbol{T_a}. \tag{47}$$

Repeating (47) N times and then adding the final matrix $\boldsymbol{T_a}$ and $\boldsymbol{I}_{2n}$ by (45), we can obtain $\boldsymbol{G}(\tau)$.

It is necessary to point out that for small order n of the LQR system we can compute the symplectic transition matrix $\boldsymbol{G}(\tau)$ by (42) directly with inversion operation. For $n = 1$, we have $a = \boldsymbol{A}[k]$, $q = \boldsymbol{Q}[k]$, $b = \boldsymbol{B}[k]$, $r = \boldsymbol{R}[k]$. Thus

$$\boldsymbol{\Phi} \overset{n=1}{==} \begin{bmatrix} -a & -q \\ -\dfrac{b^2}{r} & a \end{bmatrix}, \tag{48}$$

$$\boldsymbol{I}_2 + \frac{\tau}{2}\boldsymbol{\Phi} = \begin{bmatrix} 1 - \dfrac{\tau}{2}a & -\dfrac{\tau}{2}q \\ -\dfrac{\tau}{2}\dfrac{b^2}{r} & 1 + \dfrac{\tau}{2}a \end{bmatrix}, \tag{49}$$

$$\boldsymbol{I}_2 - \frac{\tau}{2}\boldsymbol{\Phi} = \begin{bmatrix} 1 + \dfrac{\tau}{2}a & \dfrac{\tau}{2}q \\ \dfrac{\tau}{2}\dfrac{b^2}{r} & 1 - \dfrac{\tau}{2}a \end{bmatrix}. \tag{50}$$

In this case, $\boldsymbol{G}(\tau)$ is a 2-by-2 matrix, i.e.,

$$\begin{aligned} \boldsymbol{G}(\tau) &\overset{n=1}{==} \left(\boldsymbol{I}_2 - \frac{\tau}{2}\boldsymbol{\Phi}\right)^{-1}\left(\boldsymbol{I}_2 + \frac{\tau}{2}\boldsymbol{\Phi}\right) \\ &= \frac{\begin{bmatrix} \left(1 - \dfrac{\tau a}{2}\right)^2 + \dfrac{\tau^2 q b^2}{4r} & -\tau q \\ -\dfrac{\tau b^2}{r} & \dfrac{\tau^2\, b^2\, q}{4r} + \left(1 + \dfrac{\tau a}{2}\right)^2 \end{bmatrix}}{1 - \dfrac{\tau^2}{4}\left(a^2 + \dfrac{b^2 q}{r}\right)}. \end{aligned} \tag{51}$$

E. EXPLICIT SYMPLECTIC-PRECISE ITERATION ALGORITHMS FOR TIME-VARYING LQR AND MDRE

1) ALGORITHM OF SpTranMat

Algorithm 1 presents the implementation of calculating the symplectic transition matrix $\boldsymbol{G}(\tau)$ for the corresponding LQR and MDRE problems. We find that computation of Algorithm 1 needs N times of matrix multiplications and N times of scalar product in the loop. This complexity is acceptable and it avoids the matrix inverse operation in (42) for $\boldsymbol{G}(\tau)$ as well as the round-off errors. When integrating the MDRE and LQR problem, we embed Algorithm 1 into the corresponding algorithms to get robust and precise NS. The user-defined time step τ in Algorithm 1 should be set to guarantee the number of nodes $k_{max} \in \mathbb{N}$ and we set parameter $N = 10$ as default value.

Particularly, it is necessary to point out that we can directly compute $\boldsymbol{G}(\tau)$ with (51) for $n = 1$ instead of with PIM.

Algorithm 1 Compute the Symplectic Transition Matrix With the Precise Iteration Method (SpTranMat)

Input: Time-varying matrix $\boldsymbol{\Phi}[k] \in \mathbb{R}^{2n\times 2n}$ and the time step τ.
Output: Symplectic transition matrix $\boldsymbol{G}(\tau)$ at discrete time $t_k = t_0 + k\tau$.
Usage: $\boldsymbol{G}(\tau) = \text{SpTranMat}(\boldsymbol{\Phi}[k], \tau, N)$

Set $\eta = \tau/2^N$.
Compute $\boldsymbol{a} = \eta\boldsymbol{\Phi}[k]$.
$\boldsymbol{T_a} = \boldsymbol{a}\,[\boldsymbol{I}_{2n} + 0.5\boldsymbol{a}\,(\boldsymbol{I}_{2n} + 0.5\boldsymbol{a})]$.
for $(j = 0; j < N; j++)$ **do**
  $\boldsymbol{T}_a = 2\boldsymbol{T}_a + \boldsymbol{T}_a \cdot \boldsymbol{T}_a$
end for
$\boldsymbol{G}(\tau) = \boldsymbol{I}_{2n} + \boldsymbol{T}_a$;
return $< \boldsymbol{G}(\tau) >$

Algorithm 2 Compute the Solution to MDRE With Terminal Condition in LQR (SpMdreFc)

Input: Time span $[t_0, t_f]$, positive time step τ, integer N for SpTranMat, and coefficient matrices $\boldsymbol{A}(t)$, $\boldsymbol{B}(t)$, $\boldsymbol{S}$, $\boldsymbol{Q}(t)$, $\boldsymbol{R}(t)$ of the LQR problem.
Output: Discrete gain matrices of MDRE: $\{\boldsymbol{P}[k]\}_{k=0}^{k_{max}-1}$.
Usage: $< \{\boldsymbol{P}[k]\} >=$
SpMdreFc($\boldsymbol{A}(t)$, $\boldsymbol{B}(t)$, $\boldsymbol{S}$, $\boldsymbol{Q}(t)$, $\boldsymbol{R}(t)$, τ, N, t_0, t_f)

Set $k_{max} = \frac{t_f - t_0}{\tau} \in \mathbb{N}$.
Set the terminal condition:
$\boldsymbol{P}[k_{max}] = \boldsymbol{S}$,
$\boldsymbol{Y}_{old} = \boldsymbol{S}$, $\boldsymbol{X}_{old} = \boldsymbol{I}_n$, $\boldsymbol{Y}_{new} = \boldsymbol{O}$, $\boldsymbol{X}_{new} = \boldsymbol{O}$
for $(k = k_{max} - 1; k \geq 0; k--)$ **do**
  Compute $\boldsymbol{\Phi}[k] = \boldsymbol{\Phi}(t_0 + k\tau)$ via (26).
  $\boldsymbol{G}(-\tau)$ = SpTranMat($\boldsymbol{\Phi}[k]$, $-\tau$, N).
  Set $\boldsymbol{G}_{11}^-, \ldots, \boldsymbol{G}_{22}^-$ with $\boldsymbol{G}(-\tau)$ via (33).
  Compute $\boldsymbol{P}[k]$ according to (35)
    $\boldsymbol{Y}_{new} = \boldsymbol{G}_{11}^-\boldsymbol{Y}_{old} + \boldsymbol{G}_{12}^-\boldsymbol{X}_{old}$;
    $\boldsymbol{X}_{new} = \boldsymbol{G}_{21}^-\boldsymbol{Y}_{old} + \boldsymbol{G}_{22}^-\boldsymbol{X}_{old}$;
    $\boldsymbol{P}[k] = \boldsymbol{Y}_{new}\boldsymbol{X}_{new}^{-1}$.
  Update: $\boldsymbol{Y}_{old} = \boldsymbol{Y}_{new}$, $\boldsymbol{X}_{old} = \boldsymbol{X}_{new}$.
end for
return Sequence of gain matrices $< \{\boldsymbol{P}[k]\} >$.

2) ALGORITHM OF SpMdreFc

Algorithm 2 demonstrates the way to determine the solution $\boldsymbol{P}(t)$ to the MDRE in LQR problem. We have to take the inverse time order because we encounter the terminal time

Algorithm 3 Symplectic Solver for LQR: Compute the Solution to MDRE, State Vector and Optimal Control Input (SpLQR)

Input: System matrices of LQR problem: time-varying matrices $\boldsymbol{A}(t)$, $\boldsymbol{B}(t)$, $\boldsymbol{S}$, $\boldsymbol{Q}(t)$ and $\boldsymbol{R}(t)$, integer N for SpTranMat, time span $[t_0, t_f]$, positive time step τ and initial state $\boldsymbol{x}_0 = \boldsymbol{x}(t_0)$.

Output: Discrete optimal control input $\{\boldsymbol{u}[k]\}$ and state vectors $\{\boldsymbol{x}[k]\}$.

Usage: $[\{\boldsymbol{x}[k]\}, \{\boldsymbol{u}^*[k]\}] =$ SpLQR($\boldsymbol{A}(t), \boldsymbol{B}(t), \boldsymbol{S}, \boldsymbol{Q}(t), \boldsymbol{R}(t), \tau, N, t_0, t_f, \boldsymbol{x}_0$)

Set $k_{max} = \frac{t_f - t_0}{\tau} \in \mathbb{N}$.
Set the terminal time condition: $\boldsymbol{Y}[k_{max}] = \boldsymbol{S}, \boldsymbol{X}[k_{max}] = \boldsymbol{I}_n$.
for $(k = k_{max} - 1; k \geq 0; k - -)$ **do**
  Compute $\boldsymbol{\Phi}[k] = \boldsymbol{\Phi}(t_0 + k\tau)$ by (26).
  $\boldsymbol{G}(-\tau)$ = SpTranMat($\boldsymbol{\Phi}[k], -\tau, N$).
  Set $\boldsymbol{G}_{11}^-, \ldots, \boldsymbol{G}_{22}^-$ with $\boldsymbol{G}(-\tau)$ via (33).
  Compute $\boldsymbol{Y}[k]$ and $\boldsymbol{X}[k]$ by (35):

$$\boldsymbol{Y}[k] = \boldsymbol{G}_{11}^- \boldsymbol{Y}[k+1] + \boldsymbol{G}_{12}^- \boldsymbol{X}[k+1];$$
$$\boldsymbol{X}[k] = \boldsymbol{G}_{21}^- \boldsymbol{Y}[k+1] + \boldsymbol{G}_{22}^- \boldsymbol{X}[k+1].$$

end for
$\boldsymbol{x}[k_{max}] = \boldsymbol{X}[0]^{-1}\boldsymbol{x}_0$.
$\boldsymbol{u}^*[k_{max}] = -\boldsymbol{R}[k_{max}]^{-1}\boldsymbol{B}[k_{max}]^{\mathsf{T}}\boldsymbol{Y}[k_{max}]\boldsymbol{x}[k_{max}]$.
for $(k = 0; k \leq k_{max} - 1; k = ++)$ **do**
  $\boldsymbol{x}[k] = \boldsymbol{X}[k]\boldsymbol{x}[k_{max}]$.
  $\boldsymbol{u}^*[k] = -\boldsymbol{R}[k]^{-1}\boldsymbol{B}[k]^{\mathsf{T}}\boldsymbol{Y}[k]\boldsymbol{x}[k_{max}]$.
end for
return $< \{\boldsymbol{x}[k]\}, \{\boldsymbol{u}^*[k]\} >$.

condition for the MDRE. The matrices $\boldsymbol{X}[k]$ and $\boldsymbol{Y}[k]$ are not necessary to be determined completely and what we need is just the current version of $\boldsymbol{X}[k]$ and $\boldsymbol{Y}[k]$ if we only concern about the discrete version of $\boldsymbol{P}(t)$. This means that the space complexity can be reduced by tracking the current state of $\boldsymbol{X}[k]$ and $\boldsymbol{Y}[k]$. Note that each $\boldsymbol{X}[k]^{-1}$ has to be computed in $\mathbb{R}^{n\times n}$. On the other hand, the structure of the algorithm is very simple when it is compared with those mentioned in Section I.

3) ALGORITHM OF SpLQR

Algorithm 3 describes the procedure for computing the optimal control vector $\boldsymbol{u}^*(t)$ and state vector $\boldsymbol{x}(t)$. We immediately find that it is not necessary to calculate the matrix $\boldsymbol{P}(t)$ if we just need $\boldsymbol{x}(t)$ and $\boldsymbol{u}^*(t)$. This is a significant advantage over the existing algorithms which are based on solving $\boldsymbol{P}(t)$ and encounter higher computational complexity.

If we want to output $\boldsymbol{x}(t), \boldsymbol{u}^*(t)$ and $\boldsymbol{P}(t)$ at the same time, we can combine Algorithm 1 and Algorithm 3 into an extended algorithm, which is trivial and we omit it here.

4) ALGORITHM OF SpMdreIc

As a by-product, Algorithm 4 solves the time-varying MDRE with initial condition. The advantages of Algorithm 4 lie in the following facts:

- it doesn't rely on extra assumptions;
- its algorithmic structure is simple;
- accumulative computation errors can be avoided in the sense of long-term time;
- stiffness problem due to the inverse of matrix can be avoided in the sense of precise integration method [20];
- it is a second-order precision symplectic method inherited from T-CEIS [24].

Algorithm 4 Solver for MDRE With Initial Condition (SpMdreIc)

Input: Matrices $\boldsymbol{A}(t), \boldsymbol{B}(t), \boldsymbol{P}_0, \boldsymbol{Q}(t), \boldsymbol{R}(t)$ in MDRE, time span $[t_0, t_f]$, positive time step τ, integer N for SpTranMat.

Output: Sequence of matrices $\{\boldsymbol{P}[k]\}$.

Usage: $< \{\boldsymbol{P}[k]\} >=$ SpMdreIc($\boldsymbol{A}(t), \boldsymbol{B}(t), \boldsymbol{S}, \boldsymbol{Q}(t), \boldsymbol{R}(t), N, t_0, t_f$)

Set the terminal time condition: $\boldsymbol{P}[0] = \boldsymbol{P}_0, \boldsymbol{X}_{old} = \boldsymbol{I}_n, \boldsymbol{Y}_{old} = \boldsymbol{P}_0, \boldsymbol{Y}_{new} = \boldsymbol{O}, \boldsymbol{X}_{new} = \boldsymbol{O}$.
Set the integer $k_{max} = \frac{t_f - t_0}{\tau}$.
for $(k = 0; k \leq k_{max} - 1; k + +)$ **do**
  Compute $\boldsymbol{\Phi}[k] = \boldsymbol{\Phi}(t_0 + k\tau)$ by (26).
  $\boldsymbol{G}(\tau)$ = SpTranMat($\boldsymbol{\Phi}[k], \tau, N$).
  Set $\boldsymbol{G}_{11}^+, \ldots, \boldsymbol{G}_{22}^+$ with $\boldsymbol{G}(\tau)$ via (33).
  Compute $\boldsymbol{P}[k+1]$ by (40):

$$\boldsymbol{Y}_{new} = \boldsymbol{G}_{11}^+ \boldsymbol{Y}_{old} + \boldsymbol{G}_{12}^+ \boldsymbol{X}_{old};$$
$$\boldsymbol{X}_{new} = \boldsymbol{G}_{21}^+ \boldsymbol{Y}_{old} + \boldsymbol{G}_{22}^+ \boldsymbol{X}_{old};$$
$$\boldsymbol{P}[k+1] = \boldsymbol{Y}_{new}\boldsymbol{X}_{new}^{-1}.$$

  Update $\boldsymbol{Y}_{old}$ and $\boldsymbol{X}_{old}$: $\boldsymbol{Y}_{old} = \boldsymbol{Y}_{new}, \boldsymbol{X}_{old} = \boldsymbol{X}_{new}$.
end for
return $< \{\boldsymbol{P}[k]\} >$.

IV. TIME COMPLEXITY ANALYSIS OF EXPLICIT SYMPLECTIC-PRECISE ITERATION ALGORITHMS

A. TIME COMPLEXITY VECTOR OF COMPUTATION FOR BASIC OPERATIONS

Let $\mathcal{T}_*(\texttt{expr})$ and $\mathcal{T}_+(\texttt{expr})$ be the times of multiplication and addition in some operation expression `expr`. The *time complexity vector of computation* (TCVC) for `expr` is defined by

$$\mathcal{T}(\texttt{expr}) = [\mathcal{T}_*(\texttt{expr}), \mathcal{T}_+(\texttt{expr})]. \tag{52}$$

Note that we just list two components of $\mathcal{T}(\texttt{expr})$ here since for the problems and algorithms which contain massive matrix-vector operations, the multiplication and addition operations for real numbers are more fundamental and essential if they are compared each other. The *flops*, introduced by

Golub and Loan in [35], for computing `expr` will be

$$\begin{aligned}\text{flop}(\texttt{expr}) &= \|\mathcal{T}(\texttt{expr})\|_1 \\ &= \mathcal{T}_*(\texttt{expr}) + \mathcal{T}_+(\texttt{expr}). \end{aligned} \tag{53}$$

Similarly, we use $T(\texttt{expr})$ to denote the computation time for `expr`. We also use $T^*(\texttt{expr})$ and $T^+(\texttt{expr})$ to represent the computation time for the multiplications and additions involved in `expr`. Suppose that the time units for multiplication and addition are δ_1 and δ_2 respectively. Let

$$\boldsymbol{\delta} = [\delta_1, \delta_2] \tag{54}$$

be the vector of time units. Then the time for computing `expr` will be

$$\begin{aligned} T(\texttt{expr}) &= \langle \mathcal{T}(\texttt{expr}) | \boldsymbol{\delta} \rangle \\ &= \mathcal{T}_*(\texttt{expr})\delta_1 + \mathcal{T}_+(\texttt{expr})\delta_2 \end{aligned} \tag{55}$$

if only multiplication and addition are essential for the total time consumed.

For the matrix operations frequently encountered in this paper, we list the corresponding TCVC in Appendix .

B. TIME COMPLEXITY VECTOR OF COMPUTATION FOR ALGORITHM

For an algorithm named with Alg, its TCVC is defined by

$$\begin{aligned}\mathcal{T}(\text{Alg}) &= \sum_{\texttt{expr}\in\text{Alg}} \mathcal{T}(\texttt{expr}) \\ &= [\mathcal{T}_*(\text{Alg}), \mathcal{T}_+(\text{Alg})]. \end{aligned} \tag{56}$$

If there are some parameters $n, \alpha_1, \alpha_2, \ldots, \alpha_r$ for Alg, then we will take the notation

$$\mathcal{T}_{\text{Alg}}(n, \alpha_1, \ldots, \alpha_r)$$

for its TCVC. The time needed for algorithm Alg can be measured by

$$T(\text{Alg}) = \langle \mathcal{T}(\text{Alg}) | \boldsymbol{\delta} \rangle = \mathcal{T}_*(\text{Alg})\delta_1 + \mathcal{T}_+(\text{Alg})\delta_2 \tag{57}$$

theoretically with acceptable accuracy.

1) TCVC OF SpTranMat

With the help of Table 5 in Appendix , for $2n$-by-$2n$ real matrices $\boldsymbol{a}, \boldsymbol{M}_{2n}$ and $\boldsymbol{\Lambda}_{2n} = \text{diag}(\lambda_1, \ldots, \lambda_{2n})$, we can deduce that

$$\begin{aligned}&\mathcal{T}\left(\boldsymbol{a} + \frac{1}{2}\boldsymbol{a}^2 + \frac{1}{4}\boldsymbol{a}^3\right) \\ &= \mathcal{T}\left(\boldsymbol{a}\left[\boldsymbol{I} + \frac{1}{2}\boldsymbol{a}\left(\boldsymbol{I} + \frac{1}{2}\boldsymbol{a}\right)\right]\right) \\ &= 2\,\mathcal{T}(0.5\boldsymbol{a}) + 2\,\mathcal{T}(\boldsymbol{\Lambda}_{2n} + \boldsymbol{M}_{2n}) + 2\,\mathcal{T}(\boldsymbol{M}_{2n}\boldsymbol{M}_{2n}) \\ &= 2[4n^2, 0] + 2[0, 2n] + 2[8n^3, (2n-1)4n^2] \\ &= [16n^3 + 8n^2, 16n^3 - 8n^2 + 4n] \end{aligned} \tag{58}$$

Similarly, for $\boldsymbol{T}_a \in \mathbb{R}^{2n\times 2n}$ we have

$$\begin{aligned}&\mathcal{T}(2\boldsymbol{T}_a + \boldsymbol{T}_a \cdot \boldsymbol{T}_a) \\ &= \mathcal{T}(2\boldsymbol{M}_{2n}) + \mathcal{T}(\boldsymbol{M}_{2n}\boldsymbol{M}_{2n}) + \mathcal{T}(\boldsymbol{M}_{2n} + \boldsymbol{M}_{2n}) \\ &= [4n^2, 0] + [8n^3, (2n-1)4n^2] + [0, 4n^2] \\ &= [8n^3 + 4n^2, 8n^3] \end{aligned}$$

TABLE 1. Time complexity analysis of algorithm SpTranMat.

Step	$\mathcal{T}(\texttt{expr}) = [\mathcal{T}_*(\texttt{expr}), \mathcal{T}_+(\texttt{expr})]$	Remark
1	—	Ignorable time
2	$[4n^2, 0]$	$\mathcal{T}(\lambda \boldsymbol{M}_{2n})$
3	$[16n^3 + 8n^2, 16n^3 - 8n^2 + 4n]$	See (58)
4	—	Ignorable time
5	$[(8n^3 + 4n^2), 8n^3]$	$\mathcal{T}(\boldsymbol{M}_{2n} \cdot \boldsymbol{M}_{2n}) + \mathcal{T}(\boldsymbol{M}_{2n} + \boldsymbol{M}_{2n})$
6	—	N times the TCVC in Step 5
7	$[0, 2n]$	$\mathcal{T}(\boldsymbol{\Lambda}_{2n} + \boldsymbol{M}_{2n})$
8	—	Ignorable time

Table 1 demonstrates the TCVC for each step of Algorithm 1(i.e., Algorithm SpTranMat). By summarizing the TCVC for all of the steps, we find that the TCVC of Algorithm 1 is

$$\begin{aligned}&\mathcal{T}_{\text{SpTranMat}}(n, N) \\ &= \sum_{\texttt{expr}\in\text{SpTranMat}} \mathcal{T}(\texttt{expr}) \\ &= [(16+8N)n^3 + (12+4N)n^2, (16+8N)n^3 - 8n^2 + 6n] \\ &= (16+8N)n^3[1,1] + n^2[12+4N, -8] + n[0, 6] \end{aligned} \tag{59}$$

The time consumed will be

$$\begin{aligned}&T_{\text{SpTranMat}}(n, N) \\ &= \langle \mathcal{T}_{\text{SpTranMat}}(n, N) | \boldsymbol{\delta} \rangle \\ &= (16+8N)(\delta_1 + \delta_2)n^3 \\ &\quad + ((12+4N)\delta_1 - 8\delta_2)n^2 + 6\delta_2 n \\ &= \mathcal{O}(Nn^3). \end{aligned} \tag{60}$$

We remark that the assignment operation $\eta = \tau/2^N$ can be implemented with the *shift operation* in the C/C++ programming language. If we take the default parameter $N = 10$, then

$$\mathcal{T}_{\text{SpTranMat}}(n, 10) = 96n^3[1,1] + n^2[52, -8] + n[0, 6]$$

and

$$\begin{aligned}&T_{\text{SpTranMat}}(n, 10) \\ &= \langle \mathcal{T}_{\text{SpTranMat}}(n, 10) | \boldsymbol{\delta} \rangle \\ &= 96(\delta_1 + \delta_2)n^3 + (52\delta_1 - 8\delta_2)n^2 + 6\delta_2 n \\ &= \mathcal{O}(n^3). \end{aligned} \tag{61}$$

On the other hand, if we take other algorithms for computing the matrix $\boldsymbol{G}(\tau)$ defined in (29), the time computational complexity will still be $\mathcal{O}((2n)^3) = \mathcal{O}(n^3)$. Although the Algorithm SpTranMat based on PIM have the same time computational complexity, it is stable and can avoid round-off errors.

For $n = 1$, we can find that

$$\begin{aligned}\mathcal{T}(\text{SpTranMat}) &\overset{n=1}{==} [30, 6], \\ T_{\text{SpTranMat}} &\overset{n=1}{==} 30\delta_1 + 6\delta_2 \end{aligned} \tag{62}$$

with the help of (51).

2) TCVC OF SpMdreFc

Table 2 gives the TCVC for each step of Algorithm 2(i.e., Algorithm SpMdreFc). Thus the TCVC of Algorithm 2 is

$$\begin{aligned}
&\mathcal{T}_{\text{SpMdreFc}}(n, N, k_{max}) \\
&\quad = \sum_{\text{expr}\in\text{SpMdreFc}} \mathcal{T}\,(\text{expr}) \\
&\quad = k_{max}\cdot\left\{\mathcal{T}_{\text{SpTranMat}}\,(n, N)\right. \\
&\qquad \left. +\ [8n^3+3n^2, 8n^3-5n^2] + 2\,\mathcal{T}\,(\boldsymbol{M}_n^{-1})\right\} \\
&\quad = 2k_{max}\cdot\mathcal{T}\,(\boldsymbol{M}_n^{-1}) \\
&\qquad +k_{max}[(24+8N)n^3+(15+4N)n^2, \\
&\qquad \times(24+8N)n^3-13n^2+6n]. \qquad (63)
\end{aligned}$$

Therefore,

$$\begin{aligned}
T_{\text{SpMdreFc}}(n, N, k_{max}) &= \langle\mathcal{T}_{\text{SpMdreFc}}(n, N)|\boldsymbol{\delta}\rangle \\
&= \mathcal{O}(k_{max}Nn^3) \qquad (64)
\end{aligned}$$

since $\langle\mathcal{T}\,(\boldsymbol{M}_n^{-1})|\boldsymbol{\delta}\rangle = \mathcal{O}(n^3)$. If we take the default parameter $N = 10$, then

$$\begin{aligned}
&\mathcal{T}_{\text{SpMdreFc}}(n, 10, k_{max}) \\
&\quad = 2k_{max}\cdot\mathcal{T}\,(\boldsymbol{M}_n^{-1}) \\
&\qquad +k_{max}[104n^3+55n^2, 104n^3-13n^2+6n] \\
&\quad = \mathcal{O}(k_{max}n^3).
\end{aligned}$$

3) TCVC OF SpLQR

Table 3 gives the TCVC for each step of Algorithm 3(i.e., Algorithm SpMdreFc). Therefore, the time complexity vector of Algorithm 3 is

$$\begin{aligned}
&\mathcal{T}_{\text{SpLQR}}(n, N, k_{max}) \\
&\quad = \sum_{\text{expr}\in\text{SpimLQR}} \mathcal{T}\,(\text{expr}) \\
&\quad = k_{max}\cdot\left\{\mathcal{T}\,(\boldsymbol{M}_n^{-1}) + \mathcal{T}_{\text{SpTranMat}}\,(n, N)\right. \\
&\qquad \left. +\,[6n^3+3n^2, 6n^3-4n^2]\right\} \\
&\qquad +2\,\mathcal{T}\,(\boldsymbol{M}_n^{-1}) + [5n^2, 5n^2-4n] \\
&\qquad +k_{max}\left\{\mathcal{T}\,(\boldsymbol{M}_n^{-1}) + [5n^2, 5n^2-4n]\right\} \\
&\quad = (2k_{max}+2)\,\mathcal{T}\,(\boldsymbol{M}_n^{-1}) + n^3(8N+22)k_{max}[1, 1] \\
&\qquad +n^2[4k_{max}N+20k_{max}+5, k_{max}-3] \\
&\qquad +n[0, 2-4k_{max}] \\
&\quad = \mathcal{O}(Nk_{max}n^3). \qquad (65)
\end{aligned}$$

If we take the default parameter $N = 10$, then

$$\begin{aligned}
&\mathcal{T}_{\text{SpLQR}}(n, 10, k_{max}) \\
&\quad = (2k_{max}+2)\,\mathcal{T}\,(\boldsymbol{M}_n^{-1}) + 102k_{max}n^3[1, 1] \\
&\qquad +n^2[60k_{max}+5, k_{max}-3] + n[0, 2-4k_{max}] \\
&\quad = \mathcal{O}(k_{max}n^3).
\end{aligned}$$

4) TCVC OF SpMdreIc

The TCVC of SpMdreIc is almost the same with that of SpMdreFc. It is easy to show that

$$\begin{aligned}
T_{\text{SpMdreIc}}(n, N, k_{max}) &= \langle\mathcal{T}_{\text{SpMdreIc}}(n, N, k_{max})|\boldsymbol{\delta}\rangle \\
&= \mathcal{O}(Nk_{max}n^3). \qquad (66)
\end{aligned}$$

Similarly, for $N = 10$ we have

$$\mathcal{T}_{\text{SpMdreIc}}(n, 10, k_{max}) = \mathcal{O}(k_{max}n^3).$$

V. VERIFICATION AND EVALUATION

In order to evaluate the performance of our ESPIA for LQR and MDRE, we constructed typical examples with AS and compared the NS obtained by our algorithms and other popular methods. Theoretically, the proposed ESPIA are applicable to not only time-invariant but also time-varying systems. We put more attention on the time-invariant case because it's possible to obtain AS and analyze both the accuracy and stability. For the time-varying case, we use the asymptotic solution and MATLAB Simulink to verify our algorithms.

For the purpose of analyzing the performance of our numerical algorithms, it is necessary to compare the NS and the corresponding AS. For each fixed time step τ and time interval $[t_0, t_f]$, the absolute error between the NS $p_i^{NS}(t|\tau)$ and the AS $p_i^{AS}(t)$ is

$$\left|E^i(t|\tau)\right| = \left|p_i^{AS}(t) - p_i^{NS}(t|\tau)\right|, \quad i\in\{1, 2, 3\}. \qquad (67)$$

Furthermore, the maximum error for the i-th entry is defined by

$$E_{max}^i(\tau) = \max_{t\in[t_0,t_f]}\left|p_i^{AS}(t) - p_i^{NS}(t|\tau)\right| \qquad (68)$$

and the maximum error for all of the entries is defined by

$$E_{max}(\tau) = \max_{1\le i\le 3} E_{max}^i(\tau). \qquad (69)$$

When t_0, t_f and τ are specified, sufficiently small maximum error means acceptable precision.

A. EXPERIMENTS ON TIME-INVARIANT SYSTEM

Consider the LQR problem on time interval $[0, t_f]$ with the following configuration of parameters

$$\begin{aligned}
&\boldsymbol{A} = \begin{bmatrix}0 & 1\\ -1 & 1\end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix}0\\ 1\end{bmatrix}, \\
&\boldsymbol{S} = \begin{bmatrix}2 & 0\\ 0 & 1\end{bmatrix}, \quad \boldsymbol{Q} = \begin{bmatrix}3 & 0\\ 0 & 1\end{bmatrix}, \ \boldsymbol{R} = [1]. \qquad (70)
\end{aligned}$$

Then the Riccati equation comes out from (8). Note that $\boldsymbol{P}(t)$ is a symmetric matrix with three unknowns $p_1(t), p_2(t)$ and $p_3(t)$:

$$\boldsymbol{P}(t) = \begin{bmatrix}p_1(t) & p_2(t)\\ p_2(t) & p_3(t)\end{bmatrix}. \qquad (71)$$

Substitute $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{Q}$ and $\boldsymbol{R}$ into (8), then the MDRE can be reduced to

$$\begin{bmatrix}\dot{p}_1\\ \dot{p}_2\\ \dot{p}_3\end{bmatrix} = \begin{bmatrix}p_2^2+2p_2-3\\ p_3-p_2-p_1+p_2p_3\\ p_3^2-2p_3-2p_2-1\end{bmatrix}. \qquad (72)$$

TABLE 2. Time analysis of algorithm SpMdreFc.

Step	$\mathcal{T}(\text{expr}) = [\mathcal{T}_*(\text{expr}), \mathcal{T}_+(\text{expr})]$	Remark
1	—	Ignorable although it is $[1,1]$
2	—	Ignore the time for assignment
3	—	Ignorable
4	$\mathcal{T}(\boldsymbol{M}_n^{-1}) + [2n^3+3n^2, 2n^3-2n^2]$	Find $\mathcal{T}(\boldsymbol{B}[k]\boldsymbol{R}[k]^{-1}\boldsymbol{B}[k]^{\mathsf{T}}0 + 3\mathcal{T}(-\boldsymbol{M}_n)$
5	$\mathcal{T}_{\text{SpTranMat}}(n,N)$	See (59)
6	—	Ignore the time for assignment
7.1	$[2n^3, 2n^3-n^2]$	For $\boldsymbol{Y}_{new}$: $2\mathcal{T}(\boldsymbol{M}_n\boldsymbol{M}_n) + \mathcal{T}(\boldsymbol{M}_n+\boldsymbol{M}_n)$
7.2	$[2n^3, 2n^3-n^2]$	For $\boldsymbol{X}_{new}$: $2\mathcal{T}(\boldsymbol{M}_n\boldsymbol{M}_n) + \mathcal{T}(\boldsymbol{M}_n+\boldsymbol{M}_n)$
7.3	$\mathcal{T}(\boldsymbol{M}_n^{-1}) + [n^3, (n-1)n^2]$	For $\boldsymbol{P}[k]$: $\mathcal{T}(\boldsymbol{M}_n^{-1}) + \mathcal{T}(\boldsymbol{M}_n\boldsymbol{M}_n)$
8	—	Ignore the time for assignment
9	—	K times the sum in steps $4 \sim 8$.
10	—	Ignorable

TABLE 3. Time analysis of algorithm SpLQR.

Step	$\mathcal{T}(\text{expr}) = [\mathcal{T}_*(\text{expr}), \mathcal{T}_+(\text{expr})]$	Remark
1	—	Ignorable although it is $[1,1]$
2	—	Ignore the time for assignment
3	—	Ignorable
4	$\mathcal{T}(\boldsymbol{M}_n^{-1}) + [2n^3+3n^2, 2n^3-2n^2]$	Find $\mathcal{T}(\boldsymbol{B}[k]\boldsymbol{R}[k]^{-1}\boldsymbol{B}[k]^{\mathsf{T}}) + 3\mathcal{T}(-\boldsymbol{M}_n)$ and ignore $\mathcal{T}(t_0+k\tau) = [1,1]$.
5	$\mathcal{T}_{\text{SpTranMat}}(n,N)$	See (59) and ignore $\mathcal{T}(-\tau) = [1,0]$
6	—	Ignore the time for assignment
7	$[4n^3, 4n^3-2n^2]$	$4\mathcal{T}(\boldsymbol{M}_n\boldsymbol{M}_n) + 2\mathcal{T}(\boldsymbol{M}_n+\boldsymbol{M}_n)$
8	—	k_{max} times the sum in steps $4 \sim 7$
9	$\mathcal{T}(\boldsymbol{M}_n^{-1}) + [n^2, n^2-n]$	$\mathcal{T}(\boldsymbol{X}[0]^{-1}\boldsymbol{x}[0]) = \mathcal{T}(\boldsymbol{M}_n^{-1}) + \mathcal{T}(\boldsymbol{M}_n\boldsymbol{x})$
10	$\mathcal{T}(\boldsymbol{M}_n^{-1}) + 3[n^2, n^2-n] + [n^2, 0]$	$\mathcal{T}(\boldsymbol{M}_n^{-1}) + 3\mathcal{T}(\boldsymbol{M}_n\boldsymbol{x}) + \mathcal{T}(-\boldsymbol{M}_n)$
11	—	Ignorable
12	$[n^2, n^2-n]$	$\mathcal{T}(\boldsymbol{M}_n\boldsymbol{x})$
13	$\mathcal{T}(\boldsymbol{M}_n^{-1}) + 3[n^2, n^2-n] + [n^2, 0]$	The same as in step 10
14	—	k_{max} times the sum in steps $12 \sim 13$
15	—	Ignorable

The AS to (72) is

$$\boldsymbol{P}^{AS}(\hat{t}) = [p_1^{AS}(\hat{t}), p_2^{AS}(\hat{t}), p_3^{AS}(\hat{t})]^{\mathsf{T}} = \frac{\boldsymbol{C}_1\boldsymbol{\psi}(\hat{t})}{\boldsymbol{C}_2\boldsymbol{\psi}(\hat{t})} \tag{73}$$

where $\hat{t} = t_f - t$ and

$$\boldsymbol{C}_1 = \begin{bmatrix} -15 & 20 & 45 & 14 & -12 \\ 5 & -20 & 9 & 6 & 12 \\ -5 & 20 & 27 & -10 & 0 \end{bmatrix},$$
$$\boldsymbol{C}_2 = \begin{bmatrix} 5 & 20 & 9 & -2 & -4 \end{bmatrix},$$
$$\boldsymbol{\psi}(\hat{t}) = \begin{bmatrix} 1 & \mathrm{e}^{2\hat{t}} & \mathrm{e}^{4\hat{t}} & \mathrm{e}^{2\hat{t}}\cos(2\hat{t}) & \mathrm{e}^{2\hat{t}}\sin(2\hat{t}) \end{bmatrix}^{\mathsf{T}}.$$

On the other hand, we can solve this problem with the numerical method under different configurations. By changing the terminal time t_f, computation step τ, and the parameter N in ESPIA respectively, our algorithms can be evaluated in different perspectives.

1) IMPACT OF TIME STEP SIZE τ

The LQR problem constructed above is solved with the time interval $[t_0, t_f] = [0, 5]$ and $\tau = 5.0 \times 10^{-3}$. Figure 1 shows that the absolute error $E^i(t|\tau) < 10^{-11}$ for $i \in \{1, 2, 3\}$. Moreover, $E^3_{max}(\tau) < E^2_{max}(\tau) < E^1_{max}(\tau) < 10^{-11}$ and thus $E_{max}(\tau) < 10^{-11}$ for the time step $\tau = 5.0 \times 10^{-3}$.

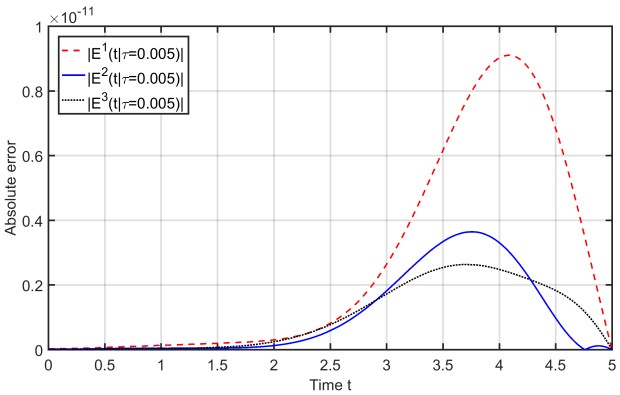

FIGURE 1. Absolute error $\left|E^i(t|\tau)\right|$ with $t \in [0, 5]$, $\tau = 5.0 \times 10^{-3}$ and $N = 10$.

The difference scheme specified by (28) or (29) has the second order accuracy [24]. We verified this fact with different values of τ ranging from 5.0×10^{-5} to 1.0×10^{-1}. Figure 2 shows the experimental result, where the maximum error $E_{max}(\tau)$ is presented by the solid blue line. We can see that the error varies linearly with the decreasing of τ and the error line is parallel to the reference curve in the logarithm coordinate system. The parallelism verifies the second-order accuracy very well. Meanwhile, the layout of the lines also implies the

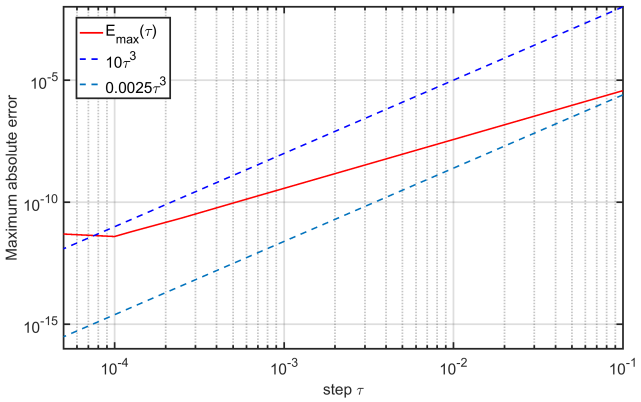

FIGURE 2. ESPIA is second order accuracy, i.e., $E_{max}(\tau) = \mathcal{O}(\tau^3) = o(\tau^2)$: $[t_0, t_f] = [0, 5]$, $N = 5$.

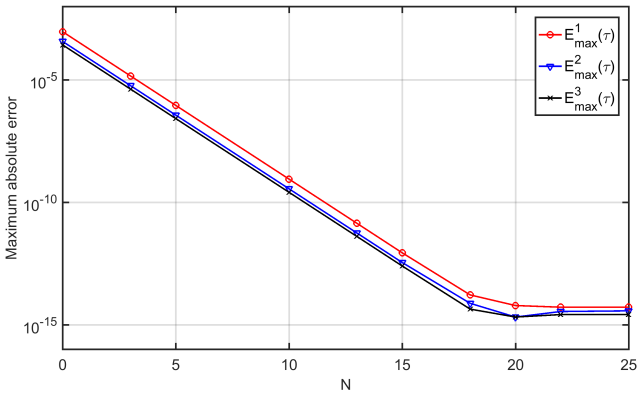

FIGURE 3. Impact of N (precise iteration parameter) on E^i_{max}: $[t_0, t_f] = [0, 5]$, $\tau = 0.05$.

accuracy level of the presented algorithm. At the right end of the curve, we remark that under this configuration for ESPIA, the maximum error $E_{max}(\tau)$ reduces to a magnitude of 10^{-6} even for a large step of $\tau = 0.1$. There is a platform at the left end of error curve, this means that the precision cannot be improved infinitely if we just reduce the computation step since τ is not the unique factor that impacts the accuracy.

2) IMPACT OF PIM PARAMETER N

Further experiments were carried out to estimate the impact of PIM. We take $\tau = 0.05$ and $[t_0, t_f] = [0, 5]$ in this group of experiments. Possible integer N is picked out in the set $\{0, 1, \ldots, 25\}$ to configure the PIM and the computation errors are recorded. The experimental results blue plotted in Figure 3. Similar to the E^i_{max}-τ curve, the E^i_{max}-N curve is composed of a slope and a platform. In the first phase, the $\log(E^i_{max})$ decreases linearly when N increases. While the error keeps stable when N is greater than a certain value. The mean error during the platform has reached the maximum precision of `double` type, i.e. $15 \sim 16$ digits if we consider the representation of data in a practical modern computer system. Usually, the typical value for N is $5 \sim 10$. Therefore, the time complexity of algorithms SpLQR, SpMdreFc and SpMdreIc will be $\mathcal{O}(k_{max} n^3)$. Moreover, the time complexity of algorithm SpTranMat will be $\mathcal{O}(n^3)$.

TABLE 4. Maximum error $E^i_{max}(\tau)$: ode45 v.s. ESPIA for $t \in [0, 5]$ and $N = 5$.

τ	E^1_{max} for p_1		E^2_{max} for p_2		E^3_{max} for p_3	
	ode45	**ESPIA**	ode45	**ESPIA**	ode45	**ESPIA**
0.100000	1.5E-03	**3.7E-06**	1.2E-03	**1.5E-06**	9.3E-04	**1.1E-06**
0.050000	1.6E-03	**9.3E-07**	1.2E-03	**3.7E-07**	9.7E-04	**2.7E-07**
0.012500	1.6E-03	**5.8E-08**	1.2E-03	**2.3E-08**	9.7E-04	**1.7E-08**
0.010000	1.6E-03	**3.7E-08**	1.2E-03	**1.5E-08**	9.7E-04	**1.1E-08**
0.005000	1.6E-03	**9.3E-09**	1.2E-03	**3.7E-09**	9.7E-04	**2.7E-09**
0.001250	1.6E-03	**5.8E-10**	1.2E-03	**2.3E-10**	9.7E-04	**1.7E-10**
0.001000	1.6E-03	**3.7E-10**	1.2E-03	**1.5E-10**	9.7E-04	**1.1E-10**
0.000500	1.6E-03	**9.3E-11**	1.2E-03	**3.7E-11**	9.7E-04	**2.7E-11**
0.000250	1.6E-03	**2.3E-11**	1.2E-03	**9.1E-12**	9.7E-04	**6.8E-12**
0.000125	6.8E-04	**6.0E-12**	3.9E-04	**2.5E-12**	5.1E-04	**1.9E-12**
0.000100	5.6E-04	**4.1E-12**	4.1E-04	**1.8E-12**	4.7E-04	**1.3E-12**

B. ESPIA VS. CLASSIC RUNGE-KUTTA METHOD

The `ode45`, the classic Runge-Kutta method, is a widely used numerical tool which is provided by MathWorks' MATLAB. We have compared the ESPIA with `ode45` to demonstrate the high accuracy of the current algorithms. As shown in Table 4, the MDRE is solved by `ode45` and ESPIA with different step configurations respectively. We counted the error by computing the error between the AS and the NS. Clearly, the accuracy of `ode45` is improved a little when we decrease the step. In spite of the smaller step, the accuracy error is limited by 4 digits. On the other hand, we can get a precision, as we discussed in Figure 2, of 12 digits when the ESPIA is adopted. Furthermore, when $\tau_{\text{SPIM}}/\tau_{\text{ode45}} = 10$, ESPIA still performs much better than `ode45` in the sense of precision.

C. EXPERIMENTS ON TIME-VARYING SYSTEM

Due to the difficulty in solving nonlinear time-varying differential equations, it's extremely difficult to find an instance which can be figured out by analytical method. Hence, we take asymptotical analysis and system simulation approach with MATLAB Simulink to compare the time-varying and time-invariant cases.

1) ASYMPTOTICAL TIME-INVARIANT CASE

The idea is to disturb the time-invariant system by adding some terms that converge to 0. In this condition, when the time keeps running, the perturbation vanishes rapidly and the solution to the time-varying should converge to the time-invariant counterpart. Consider the following time-invariant system on time interval $[0, t_f]$ which is the disturbed version of (70) by adding some vanishing terms,

$$\boldsymbol{A} = \begin{bmatrix} \frac{1}{5}\mathrm{e}^{-t} & 1 - \frac{\mathrm{e}^{-\frac{t^2}{2}}}{\sqrt{2\pi}} \\ -1 + \mathrm{e}^{-3t}\sin(t) & 1 + \mathrm{e}^{-t^2} \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$
$$\boldsymbol{S} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, \quad \boldsymbol{Q} = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}, \quad \boldsymbol{R} = \begin{bmatrix} 1 \end{bmatrix} \tag{74}$$

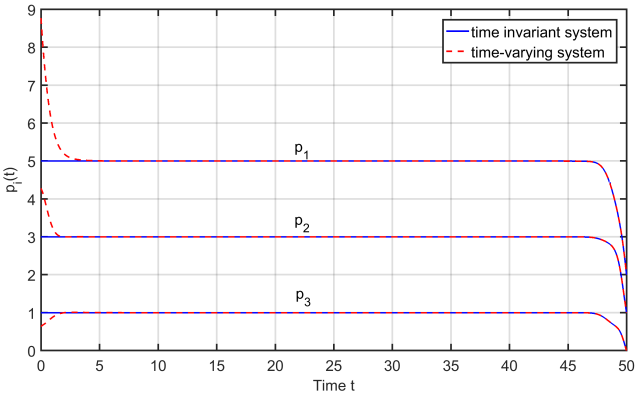

FIGURE 4. Asymptotic behavior of a time-varying LQR system with $n = 2$, $[t_0, t_f] = [0, 50]$, $\tau = 0.02$ and $N = 10$.

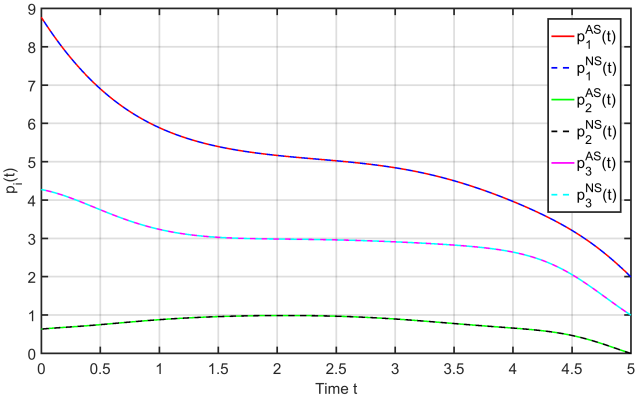

FIGURE 6. Comparison of $p_i^{AS}(t)$ by simulink and $p_i^{NS}(t)$ by ESPIA: $[t_0, t_f] = [0, 5]$, $\tau = 0.02$ and $N = 10$.

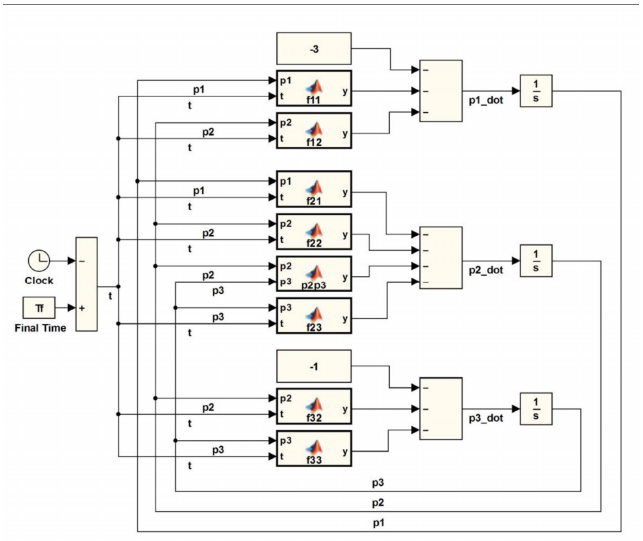

FIGURE 5. MATLAB simulink block diagram.

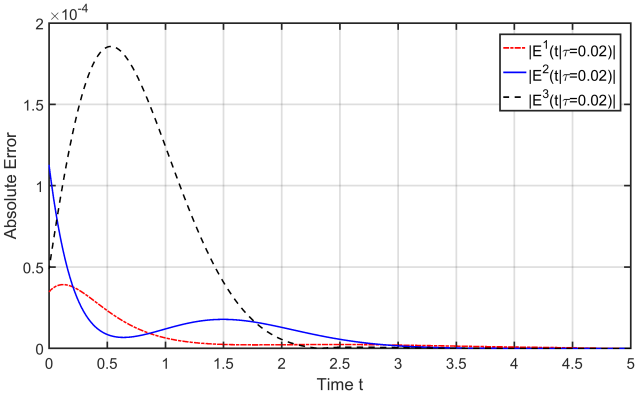

FIGURE 7. $|E^i(t|\tau)|$ between the AS by simulink and the NS by ESPIA: $[t_0, t_f] = [0, 5]$, $\tau = 0.02$ and $N = 10$.

Through the previous analysis, the solution to this time-varying system should converge to the corresponding time-invariant solution (73). Motivated by this feature, we set the terminal time $t_f = 50$ and we compared the NS and AS given by (73). The experimental result is shown in Figure 4, where the convergency relation is well explained. Moreover, the convergency indicates that the NS by the ESPIA is stable in the sense of long term time since the symplectic geometric method avoids the accumulative errors in the iterations automatically.

2) SIMULATION OF TIME-VARYING SYSTEM VIA MATLAB SIMULINK

For the assessment on the time-varying case, the NS is compared with the result given by MATLAB Simulink, which is a generally acknowledged toolbox for modeling and simulation. In this case, we still considered the asymptotically convergent system specified by (74) except $[t_0, t_f] = [0, 5]$. Considering the symmetry, the MDRE for this problem can be transformed into a third-order system, which is similar with (72). The block-diagram implemented with MATLAB Simulink is shown in Figure 5. We remark that the solution obtained by the Simulink is treated as the AS. The AS by Simulink and the NS by ESPIA are displayed in Figure 6. The differences between the two solutions are plotted in Figure 7, where the maximum absolute error $E_{max}(t|\tau = 0.02)$ between the NS and output of the MATLAB Simulink is less than 2.0×10^{-4}. This is reasonable and it is compatible with the test shown in Table 4.

VI. CONCLUSION

In this paper, we have applied the symplectic method and precise integration method to the time-varying LQR problem and have obtained some interesting and important results. Our main results are as follows:

- The n-dimensional LQR problem can be described equivalently by n non-autonomous linear Hamiltonian systems with the same Hamiltonian function via Heisenberg picture. This equivalence shows that we can not only deal with time-varying LQR problem automatically and no additional condition or assumption is needed, but also take the symplectic method to find the numeric

solutions. This equivalent description may refresh the understanding of LQR and MDRE and the corresponding methodology for seeking their numeric solutions.
- The SPIM, a combination of symplectic method and precise integration method, is presented to compute the symplectic transition matrices for the difference scheme of LQR and MDRE. Particularly, SPIM can avoid inverting matrices as well as keep high precision in the sense of long-term time at the same time.
- The ESPIA has been designed according to the T-CEIS, a symplectic difference scheme with second-order precision, and the precise iterative process. There are two essential merits for ESPIA: it is symplecitic which means there is no accumulative computational errors, and it is immune to the stiffness problem due to the inverse of matrix in the difference scheme.
- Theoretical analysis shows that it is not necessary to solve the MDRE if we only concerns about the optimal input and state vectors, which simplifies the implementation of practical LQR in engineering problems.
- The time-varying MDRE with initial conditions can be solved with ESPIA, which works well with no additional conditions.
- The time complexity of the symplectic algorithms for the n-th order LQR and MDRE are $\mathcal{O}(Nk_{max}n^3)$ where N and k_{max} are the iteration numbers for the precise iteration process and time duration. The typical value for N could be $5 \sim 10$, which results the time complexity $\mathcal{O}(k_{max}n^3)$ for the proposed algorithms for LQR and MDRE.
- A series of illustrative examples, which covering the time-invariant system, asymptotically time-invariant system and the time-varying system, verified the high accuracy and the long-term stability of ESPIA, which gains much more precision than the classic Runge-Kutta method.

It should be remarked that there are still some open problems to be considered in the future such as how to extend the developed algorithm to more complicated problems, i.e., those with non-quadratic cost functional and non-linear system equations and how to deal with possible state and control constraints.

APPENDIX A
HAMILTONIAN SYSTEM

W. R. Hamiltonian introduced the canonical differential equation

$$\frac{\mathrm{d}\,p_i}{\mathrm{d}\,t} = -\frac{\partial H}{\partial q_i}, \quad \frac{\mathrm{d}\,q_i}{\mathrm{d}\,t} = +\frac{\partial H}{\partial p_i}, \quad i \in \{1, \ldots, n\} \tag{75}$$

for problems of geometrical optics, where p_i are the generalized momentums, q_i are the generalized displacements and $H = H(p_1, \ldots, p_n, q_1, \ldots, q_n)$ is the Hamiltonian. Let $\boldsymbol{p} = [p_1, \ldots, p_n]^\mathsf{T}$, $\boldsymbol{q} = [q_1, \ldots, q_n]^\mathsf{T}$, $\boldsymbol{z} = [\boldsymbol{p}^\mathsf{T}, \boldsymbol{q}^\mathsf{T}]^\mathsf{T}$, and $\boldsymbol{I}_n$ be n-th order identity matrix, then the canonical equation is equivalent to

$$\frac{\mathrm{d}\,\boldsymbol{z}}{\mathrm{d}\,t} = \boldsymbol{J}^{-1}\nabla H(\boldsymbol{z}), \qquad \boldsymbol{J} = \begin{bmatrix} \boldsymbol{O}_n & \boldsymbol{I}_n \\ -\boldsymbol{I}_n & \boldsymbol{O}_n \end{bmatrix}, \tag{76}$$

where $\nabla H(\boldsymbol{z})$ is the gradient of $H(\boldsymbol{z})$ and $\boldsymbol{J}$ is the $2n$-th order standard symplectic matrix.

If the Hamiltonian $H(\boldsymbol{z})$ does not depend on t explicitly, the system is called an autonomous Hamiltonian system, otherwise it is called a non-autonomous system. If the Hessian matrix $\boldsymbol{K} = \left(\frac{\partial^2 H(\boldsymbol{z})}{\partial z_i \partial z_j}\right)_{2n\times 2n}$ is symmetric, the system will be called a linear Hamiltonian system.

APPENDIX B
SYMPLECTIC MATRIX

A linear transform $\boldsymbol{S} : \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ is symplectic if and only if

$$\boldsymbol{S}^\mathsf{T}\boldsymbol{J}\boldsymbol{S} = \boldsymbol{J}. \tag{77}$$

The matrix $\boldsymbol{S}$ is referred to as a symplectic matrix or symplectic transition mapping.

Lemma 3: Let $\boldsymbol{M}, \boldsymbol{N} \in \mathbb{R}^{2n\times 2n}$ and $\boldsymbol{M}$ is invertible, $\boldsymbol{S} = \boldsymbol{M}^{-1}\boldsymbol{N}$, if $\boldsymbol{M}\boldsymbol{J}\boldsymbol{M}^\mathsf{T} = \boldsymbol{N}\boldsymbol{J}\boldsymbol{N}^\mathsf{T}$, then $\boldsymbol{S} \in \mathrm{Sp}(2n, \mathbb{R}) = \left\{\boldsymbol{G} \in \mathbb{R}^{2n\times 2n} : \boldsymbol{G}^\mathsf{T}\boldsymbol{J}\boldsymbol{G} = \boldsymbol{J}\right\}$.

The proof is given in [24] and it is omitted here.

Lemma 4: Let $\boldsymbol{C} \in \mathbb{R}^{2n\times 2n}$ be symmetric, $\boldsymbol{F}(x) = \boldsymbol{I}_{2n} - x\boldsymbol{J}^{-1}\boldsymbol{C}$, the matrix $\boldsymbol{V} = [\boldsymbol{F}(x)]^{-1} \cdot [\boldsymbol{F}(-x)]$ must be a symplectic matrix for any $x \in \mathbb{R}$.

Proof: With the help of $\boldsymbol{J}^{-1} = -\boldsymbol{J} = \boldsymbol{J}^\mathsf{T}, \boldsymbol{J}^2 = -\boldsymbol{I}_{2n}$ and $\boldsymbol{C} = \boldsymbol{C}^\mathsf{T}$, we have

$$\boldsymbol{F}(x) = \boldsymbol{I}_{2n} - x\boldsymbol{J}^{-1}\boldsymbol{C} = \boldsymbol{I}_{2n} + x\boldsymbol{J}\boldsymbol{C}$$

Therefore,

$$\begin{aligned}[\boldsymbol{F}(x)]\boldsymbol{J}[\boldsymbol{F}(x)]^\mathsf{T} &= (\boldsymbol{I}_{2n} + x\boldsymbol{J}\boldsymbol{C})\boldsymbol{J}(\boldsymbol{I}_{2n} + x\boldsymbol{J}\boldsymbol{C})^\mathsf{T} \\ &= (\boldsymbol{J} + x\boldsymbol{J}\boldsymbol{C}\boldsymbol{J})(\boldsymbol{I}_{2n} - x\boldsymbol{C}\boldsymbol{J}) \\ &= \boldsymbol{J} + x^2\boldsymbol{J}\boldsymbol{C}^2 = \boldsymbol{J}(\boldsymbol{I}_{2n} + x^2\boldsymbol{C}^2) \\ &= [\boldsymbol{F}(-x)]\boldsymbol{J}[\boldsymbol{F}(-x)]^\mathsf{T}\end{aligned}$$

Let $\boldsymbol{M} = \boldsymbol{F}(x), \boldsymbol{N} = \boldsymbol{F}(-x)$, then

$$\boldsymbol{M}\boldsymbol{J}\boldsymbol{M}^\mathsf{T} = \boldsymbol{N}\boldsymbol{J}\boldsymbol{N}^\mathsf{T}.$$

Hence $\boldsymbol{V} = \boldsymbol{M}^{-1}\boldsymbol{N}$ is symplectic by Lemma 3. ■

APPENDIX C
TCVC OF MATRIX OPERATIONS

With the notations of TCVC, we can summarize the time cost of computation for some matrix operations in Table 5.

It is necessary to emphasize that the TCVC for the matrix inversion depends on the algorithm adopted, see Table 6. Petković [36] proved that generalized matrix inversion is not harder than matrix multiplication. In the introduction part of reference [35], the authors gave a complete review of the matrix multiplication algorithm development.

TABLE 5. Time cost of matrix operations.

OPERANDS	expr	$\mathcal{T}(\texttt{expr}) = [\mathcal{T}_*(\texttt{expr}), \mathcal{T}_+(\texttt{expr})]$
$\boldsymbol{A} \in \mathbb{R}^{p\times q}, \lambda \in \mathbb{R}$	$\lambda\boldsymbol{A}$	$[pq, 0]$
$\boldsymbol{A} \in \mathbb{R}^{p\times q}$	$\boldsymbol{A}^{\mathsf{T}}$	$[0, 0]$
$\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{p\times q}$	$\boldsymbol{A}+\boldsymbol{B}$	$[0, pq]$
$\boldsymbol{A} \in \mathbb{R}^{p\times p}, \boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \cdots, \lambda_p)$	$\boldsymbol{A}+\boldsymbol{\Lambda}$	$[0, p]$
$\boldsymbol{A} \in \mathbb{R}^{p\times q}, \boldsymbol{B} \in \mathbb{R}^{q\times r}$	$\boldsymbol{AB}$	$[pqr, (q-1)pr]$
$\boldsymbol{A} \in \mathbb{R}^{p\times q}, \boldsymbol{x} \in \mathbb{R}^{q\times 1}$	$\boldsymbol{Ax}$	$[pq, (q-1)p]$
$\boldsymbol{A} \in \mathbb{R}^{p\times p}, \boldsymbol{x} \in \mathbb{R}^{p\times 1}$	$\boldsymbol{Ax}$	$[p^2, (p-1)p]$
	$\boldsymbol{x}^{\mathsf{T}}\boldsymbol{A}$	$[p^2, (p-1)p]$
$\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{p\times 1}$	$\boldsymbol{x}^{\mathsf{T}}\boldsymbol{y}$	$[p, p-1]$
$\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{p\times 1}$	$\boldsymbol{xy}^{\mathsf{T}}$	$[p^2, 0]$
$\boldsymbol{A} \in \mathbb{R}^{p\times p}, \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{p\times 1}$	$\boldsymbol{x}^{\mathsf{T}}\boldsymbol{Ay}$	$[p^2+p, p^2-1]$
$\boldsymbol{A} \in \mathbb{R}^{p\times q}, \boldsymbol{B} \in \mathbb{R}^{q\times r}, \boldsymbol{x} \in \mathbb{R}^{r\times 1}$	$(\boldsymbol{AB})\boldsymbol{x}$	$[pqr+pr, pqr-p]$
	$\boldsymbol{A}(\boldsymbol{Bx})$	$[qr+pq, qr+pq-q-p]$
$\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{p\times p}, \boldsymbol{x} \in \mathbb{R}^{p\times 1}$	$(\boldsymbol{AB})\boldsymbol{x}$	$[p^3+p^2, p^3-p]$
	$\boldsymbol{A}(\boldsymbol{Bx})$	$2[p^2, p^2-p]$
$\boldsymbol{A} \in \mathbb{R}^{p\times p}, \boldsymbol{x} \in \mathbb{R}^{p\times 1}$	$\boldsymbol{A}(\cdots(\boldsymbol{Ax}))$	$n[p^2, p^2-p]$
	$(\boldsymbol{A}^n)\boldsymbol{x}$	$(n-1)[p^3, p^3-p^2]+[p^2, p^2-p]$
$\boldsymbol{A} \in \mathrm{GL}(p, \mathbb{R}) \subset \mathbb{R}^{p\times p}$	$\boldsymbol{A}^{-1}$	$\mathcal{O}(p^3)$ is enough.

TABLE 6. Time complexity for inverting real matrix $A \in \mathbf{GL}(\boldsymbol{p}, \mathbb{R})$.

MATRIX INVERSION METHOD	TIME COMPLEXITY
Gauss-Jordan elimination [35]	$\mathcal{O}(p^3)$
Strassen's algorithm [37]	$\mathcal{O}(p^{2.807})$
Coppersmith–Winograd algorithm [38]	$\mathcal{O}(p^{2.376})$
Optimized CW-like algorithms [35]	$\mathcal{O}(p^{2.3727})$

For low dimension matrix, we can find its inverse by computing the determinant and adjoint matrix. For illustration, we give the following two fundamental cases:

- For $\boldsymbol{A} \in \mathrm{GL}(2, \mathbb{R})$, we have

$$\boldsymbol{A}^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}.$$

- For $\boldsymbol{A} \in \mathrm{GL}(3, \mathbb{R})$, we have

$$\boldsymbol{A}^{-1} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}^{-1} = \frac{1}{\det(\boldsymbol{A})}\begin{bmatrix} A & D & G \\ B & E & H \\ C & F & I \end{bmatrix}$$

where $A = ei - fh$, $B = -(di - fg)$, $C = dh - eg$, $D = -(bi - ch)$, $E = ai - cg$, $F = -(ah - bg)$, $G = bf - ce$, $H = -(af - cd)$ and $I = ae - bd$.

ACKNOWLEDGMENT

The first author would like to thank Prof. Rong-Xing Zhou of Xidian University who introduced the matrix differential Riccati equation to him in 1996 for the first time when he was a freshman. The authors also thank Prof. Fei Liu of Civil Aviation University of China for her correcting some errors in this paper.

REFERENCES

[1] R. E. Kalman, "Contribution to the theory of optimal control," *Boletin Sociedad Math. Mexicana*, vol. 5, no. 2, pp. 102–119, 1960.

[2] B. D. O. Anderson and B. J. Moore, *Optimal Control: Linear Quadratic Methods*. New York, NY, USA: Dover, 2007.

[3] W. T. Reid and M. R. Chidambara, "Riccati differential equations," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, no. 9, p. 680, Sep. 1977.

[4] V. Mehrmann, *The Autonomous Linear Quadratic Control Problem: Theory and Numerical Solution* (Lecture Notes in Control and Information Sciences), vol. 163. Berlin, Germany: Springer-Verlag, 1991.

[5] T. Kailath, "Some new algorithms for recursive estimation in constant linear systems," *IEEE Trans. Inf. Theory*, vol. IT-19, no. 6, pp. 750–760, Nov. 1973.

[6] W. M. McEneaney, "A new fundamental solution for differential Riccati equations arising in control," *Automatica*, vol. 44, no. 4, pp. 920–936, Apr. 2008.

[7] V. I. H. Abou-Kandil, G. Freiling, and G. Jank, *Matrix Riccati Equations in Control and Systems Theory* (Systems & Control: Foundations & Applications). Basel, Birkhäuser: Springer-Verlag, 2003.

[8] S. Kilicaslan and S. P. Banks, "Existence of solutions of Riccati differential equations," *J. Dyn. Syst., Meas., Control*, vol. 134, no. 3, May 2012, Art. no. 031001, doi: 10.1115/1.4005496.

[9] L. Dieci, "Numerical integration of the differential Riccati equation and some related issues," *SIAM J. Numer. Anal.*, vol. 29, no. 3, pp. 781–815, Jun. 1992.

[10] C. S. Kenney and R. Leipnik, "Numerical integration of the differential matrix Riccati equation," *IEEE Trans. Autom. Control*, vol. AC-30, no. 10, pp. 962–970, Oct. 1985.

[11] D. Lainiotis, "Partitioned Ricatti solutions and integration-free doubling algorithms," *IEEE Trans. Autom. Control*, vol. AC-21, no. 5, pp. 677–689, Oct. 1976.

[12] R. B. Leipnik, "A canonical form and solution for the matrix Riccati differential equation," *J. Austral. Math. Society. Ser. B, Appl. Math.*, vol. 26, no. 3, pp. 355–361, Jan. 1985.

[13] P. Benner and H. Mena, "BDF methods for large-scale differential Riccati equations," in *Proc. 16th Int. Symp. Math. Theory Netw. Syst.*, 2004, pp. 1–12.

[14] E. J. Davison and M. Maki, "The numerical solution of the matrix Riccati differential equation," *IEEE Trans. Autom. Control*, vol. AC-18, no. 1, pp. 71–73, Feb. 1973.

[15] Y. Oshman and I. Bar-Itzhack, "Eigenfactor solution of the matrix Riccati equation–A continuous square root algorithm," *IEEE Trans. Autom. Control*, vol. AC-30, no. 10, pp. 971–978, Oct. 1985.

[16] A. Varga, "On solving periodic Riccati equations," *Numer. Linear Algebra Appl.*, vol. 15, no. 9, pp. 809–835, Nov. 2008.

[17] J. Nazarzadeh, M. Razzaghi, and K. Y. Nikravesh, "Solution of the matrix Riccati equation for the linear quadratic control problems," *Math. Comput. Model.*, vol. 27, no. 7, pp. 51–55, Apr. 1998.

[18] T. Nguyen and Z. Gajic, "Solving the matrix differential Riccati equation: A Lyapunov equation approach," *IEEE Trans. Autom. Control*, vol. 55, no. 1, pp. 191–194, Jan. 2010.

[19] G.-D. Hu, "Symplectic runge-kutta methods for the linear quadratic regulator problem," *Int. J. Math. Anal.*, vol. 1, no. 6, pp. 293–304, 2007.

[20] W. X. Zhong and F. W. Williams, "A precise time step integration method," *Proc. Inst. Mech. Eng. Mech. Eng. Sci.*, vol. 208, no. 6, pp. 427–430, Nov. 1994.
[21] S.-J. Tan and W.-X. Zhong, "Numerical solutions of linear quadratic control for time-varying systems via symplectic conservative perturbation," *Appl. Math. Mech.*, vol. 28, no. 3, pp. 277–287, Mar. 2007.
[22] Q. Gao, S.-J. Tan, W.-X. Zhong, and H.-W. Zhang, "Improved precise integration method for differential Riccati equation," *Appl. Math. Mech.*, vol. 34, no. 1, pp. 1–14, Jan. 2013.
[23] H.-Y. Zhang, Z.-H. Wang, L.-S. Zhou, Q.-N. Xue, L. Ma, and Y.-F. Niu, "Explicit symplectic geometric algorithms for quaternion kinematical differential equation," *IEEE/CAA J. Automatica Sinica*, vol. 5, no. 2, pp. 479–488, Mar. 2018.
[24] K. Feng and M. Z. Qin, *Sympletic Geometric Algorithms for Hamiltonian System*. Berlin, Germany: Springer-Verlag, 2010.
[25] K. Feng, "On difference schemes and symplectic geometry," in *Proc. Beijing Int. Symp. Differ. Geometry Differ. Equ.* Beijing, China: Science Press, Aug. 1985, pp. 42–58.
[26] H. Peng, X. Wang, S. Zhang, and B. Chen, "An iterative symplectic pseudospectral method to solve nonlinear state-delayed optimal control problems," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 48, pp. 95–114, Jul. 2017.
[27] X. Wang, H. Peng, S. Zhang, B. Chen, and W. Zhong, "A symplectic pseudospectral method for nonlinear optimal control problems with inequality constraints," *ISA Trans.*, vol. 68, pp. 335–352, May 2017.
[28] J. Frank and S. Zhuk, "Symplectic Möbius integrators for LQ optimal control problems," in *Proc. IEEE Conf. Decis. Control*, Dec. 2014, pp. 6377–6382.
[29] M. Sanz-Serna and M. P. Calvo, *Numerical Hamiltonian Problem*. London, U.K.: Chapman & Hall, 1994.
[30] P. A. M. Dirac, *The Principles of Quantum Mechanics* (The International Series of Monographs on Physics). London, U.K.: Clarendon Press, 1958.
[31] B. L. V. D. Waerden, *Sources of Quantum Mechanics*. Mineola, NY, USA: Dover, 1968.
[32] B. d'Espagnat, "Heisenberg picture and reality," *Found. Phys.*, vol. 22, no. 12, pp. 1495–1504, Dec. 1992.
[33] K. Feng, *The Hamiltonian Way for Computing Hamiltonian Dynamics*. Dordrecht, The Netherlands: Springer, 1991, pp. 17–35, doi: 10.1007/978-94-009-1908-2_3.
[34] V. I. Arnold, *Mathematical Methods Classical Mechnaics* (Graduate Texts in Mathematics), vol. 60. Berlin, Germany: Springer, 1989.
[35] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 4th ed. Baltimore, MD, USA: Johns Hopkins Univ. Press, Jan. 2013.
[36] M. D. Petković and P. S. Stanimirović, "Generalized matrix inversion is not harder than matrix multiplication," *J. Comput. Appl. Math.*, vol. 230, no. 1, pp. 270–282, 2009.
[37] V. Strassen, "Gaussian elimination is not optimal," *Numer. Math.*, vol. 13, no. 4, pp. 354–356, 1969.
[38] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," *J. Symbolic Comput.*, vol. 9, no. 3, pp. 251–280, 1990.

HONG-YAN ZHANG received the B.S. and M.S. degrees in applied physics and telecommunication engineering from Xidian University, China, in 2000 and 2003, respectively, and the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, in 2011. He is currently with the School of Information Science and Technology, Hainan Normal University, China. His research interests include symplectic method, computer vision, and data analysis.

JIA-ZHEN LUO received the B.S. and M.S. degrees in aviation engineering from the Sino-European Institute of Aviation Engineering (SIAE), Civil Aviation University of China (CAUC), in 2015 and 2018, respectively. He is currently an Engineer of vision algorithms. His research interests include computer vision, robotics, and their applications.

YU ZHOU received the Ph.D. degree in electronic engineering from the University of Newcastle upon Tyne, in 2008. He is currently a Research Associate of electronic engineering with Hainan Normal University. His research interests include circuit design, logic synthesis, and CAD tools for asynchronous circuits and systems.