INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING
Volume 10, Number 4, Pages 972–991

FAST OPTIMAL $\mathcal{H}_2$ MODEL REDUCTION ALGORITHMS BASED ON GRASSMANN MANIFOLD OPTIMIZATION

YUESHENG XU AND TAISHAN ZENG*

Abstract. The optimal $\mathcal{H}_2$ model reduction is an important tool in studying dynamical systems of a large order and their numerical simulation. We formulate the reduction problem as a minimization problem over the Grassmann manifold. This allows us to develop a fast gradient flow algorithm suitable for large-scale optimal $\mathcal{H}_2$ model reduction problems. The proposed algorithm converges globally and the resulting reduced system preserves stability of the original system. Furthermore, based on the fast gradient flow algorithm, we propose a sequentially quadratic approximation algorithm which converges faster and guarantees the global convergence. Numerical examples are presented to demonstrate the approximation accuracy and the computational efficiency of the proposed algorithms.

Key words. $\mathcal{H}_2$ approximation, gradient flow, Grassmann manifold, model reduction, MIMO system, stability, large-scale sparse system.

1. Introduction

Model reduction, which approximates a linear dynamic system of a higher order by a system of a significantly lower order, received considerable attention in recent years. This problem is important for many applications, such as design of large scale integration chips, analysis and design of micro electro mechanical system devices, weather prediction and control of partial differential equations. There are many existing methods that produce lower-order systems from given high-order systems. Most of these methods fall into two categories. The first category is projection-based methods, such as the Krylov subspace (moment matching) methods, the balanced-truncation method, and proper orthogonal decomposition (POD) methods. The second category is optimization-based methods such as the Hankel optimal model reduction [6] and the $\mathcal{H}_2$ optimal model reduction. For nice reviews of model reduction for large-scale dynamical systems, the readers are referred to [3, 7, 28].

The present paper concerns the optimal $\mathcal{H}_2$ norm model reduction problem, which has been studied by many investigators, see, for instance [4, 9, 13, 15, 22, 23, 24, 25, 26] and the references cited therein. Most of the existing algorithms are not suitable for the reduction of large-scale systems. Algorithms proposed in [9, 15, 22] require solving large-scale Lyapunov equations at each iteration step, making them computationally expensive for large scale systems. Gradient flow algorithm [26] requires computing the exponential of a large-scale matrix at each iteration step which is computationally expensive. Interpolation-based algorithms were proposed in [4, 13, 23, 25] for solving large-scale $\mathcal{H}_2$ optimal model reduction problems. A

Received by the editors December 12, 2012 and, in revised form, February 17, 2013.
2000 *Mathematics Subject Classification.* 65M10, 65M15, 65N10, 65N15.
*Corresponding author.
Supported in part by the US National Science Foundation under grant DMS-1115523, by the Natural Science Foundation of China under grants 11071286, 91130009, by Guangdong Provincial Government of China through the "Computational Science Innovative Research Team" program, and this work was supported in part by the Natural Science Foundation of China under grant 11101163.

drawback of interpolation-based algorithms is that they often cannot guarantee convergence and the stability of the reduced model.

The main purpose of this paper is to develop fast algorithms suitable for sparse systems of large scales, with the resulting reduced system preserving stability of the original system. Specifically, we treat the optimal $\mathcal{H}_2$ model reduction problem as a minimization problem on the Grassmann manifold. We propose a fast gradient flow algorithm based on the geodesic of the Grassmann manifold which is suitable for large sparse multi-input multi-output (MIMO) systems. The geodesic equation of the Grassmann manifold is easy to compute, which makes it possible to reduce the computational cost. The new algorithm avoids computing the matrix exponential and only involves computation with matrices of small size. We also reformulate the partial derivatives of the cost function to be more computationally effective. The proposed algorithm has nice properties that starting from any initial orthogonal matrix, the iterations remain on the manifold. The convergence of the algorithm is guaranteed. We also derive sufficient conditions for the algorithms to preserve the stability. Based on approximating the cost function by a quadratic function, we propose an algorithm with faster convergence. By combining with the gradient flow method, the global convergence is guaranteed. The two Grassmann manifold-based algorithms proposed in this paper have low computational cost and are suitable for large-scale sparse systems. Since they do not rely on the assumption that the target system has simple poles, they remain robust when the target system has multiple poles.

This paper is organized in eight sections. In Section 2, we introduce the optimal $\mathcal{H}_2$ model reduction problem minimizing over the Grassmann manifold. In Section 3, we describe the gradient flow on the Grassmann manifold for solving the $\mathcal{H}_2$ optimal model reduction problem. Section 4 centers at the development of fast numerical algorithms for the gradient flow on the Grassmann manifold and provides complexity analysis. In Section 5, we compare the proposed fast gradient flow algorithm with the existing gradient flow algorithm. In Section 6, an algorithm with faster convergence is proposed. Section 7 is devoted to a presentation of numerical results of the proposed algorithms. Finally, in Section 8, we draw a conclusion.

2. Optimal $\mathcal{H}_2$ Model Reduction

We describe in this section the optimal $\mathcal{H}_2$ model reduction problem.

For given matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$ and $C \in \mathbb{R}^{q\times n}$, we consider the linear dynamical systems described by

$$\frac{dx}{dt} = Ax(t) + Bu(t), \tag{1}$$

$$y(t) = Cx(t), \tag{2}$$

where $t \geq 0$ is the time variable, $u \in \mathbb{R}^p$ is the input, $y \in \mathbb{R}^q$ is the output and $x \in \mathbb{R}^n$ is the state of the system. Here, n is the system order, p and q are the number of system inputs and outputs, respectively. The linear system described by equations (1) and (2) is uniquely determined by the state space realization (A, B, C) and the initial condition $x(t_0) = x_0$. In this paper, we assume that the numbers of input and output of the full order system described by equations (1) and (2) are small, that is, $q \ll n$ and $p \ll n$. Moreover, we assume that the matrix A is sparse.

If the system order n is too big, it is not computationally efficient to solve various control problems. We need to construct a reduced order system to approximate the full order system, with preserving certain system properties such as stability and

passivity. In this paper, we will construct reduced order models via projection. The basic idea is to project the system's state space of dimension n to a space of lower dimension m with $m \ll n$. Specifically, the projection based reduction scheme involves selecting a matrix $U \in \mathbb{R}^{n\times m}$ such that $U^TU = I$. Letting $A_m := U^TAU$, $B_m := U^TB$ and $C_m := CU$, we then obtain a lower order system through projection

$$\frac{dx_m}{dt} = A_m x_m(t) + B_m u(t), \tag{3}$$

$$y_m(t) = C_m x_m(t). \tag{4}$$

A dynamical system can be reformulated via its transfer function. For a vector-valued function f, its Laplace transform is defined by

$$F(s) := \mathcal{L}\{f(t)\} = \int_0^{+\infty} f(t)e^{-st}dt, \quad s \in \mathbb{C},$$

where $\mathbb{C}$ denotes the set of all complex numbers. Let $\hat{v}$ denote the Laplace transform of a function v. If the initial condition $x(0) = x_0 = 0$, then

$$\hat{y}(s) = C(sI - A)^{-1}B\hat{u}(s), \quad s \in \mathbb{C},$$

where I denotes the identity matrix. The function

$$G(s) = C(sI - A)^{-1}B, \quad s \in \mathbb{C}$$

is called the *transfer function* of the full order system (A, B, C). Likewise, the transfer function of the reduced system described by equations (3) and (4) is given by

$$G_m(s) = C_m(sI - A_m)^{-1}B_m = CU(sI - U^TAU)^{-1}U^TB. \tag{5}$$

A transfer function G is called *stable* if the eigenvalues of A have strictly negative real parts. If G is unstable, its $\mathcal{H}_2$ norm is defined to be $+\infty$. Otherwise its square $\mathcal{H}_2$ norm is defined as the trace of a matrix integral (cf. [28]) :

$$\|G\|_2^2 = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \mathrm{trace}\{G(i\omega)^*G(i\omega)\}d\omega,$$

where $i = \sqrt{-1}$ is the imaginary unit. We introduce the error of the two transfer functions by setting

$$G_e(s) := G(s) - G_m(s), \quad s \in \mathbb{C}$$

and clearly, we have that

$$G_e(s) = C(sI - A)^{-1}B - CU(sI - U^TAU)^{-1}U^TB. \tag{6}$$

The cost function is defined by

$$J(U) := \|G_e\|_2^2. \tag{7}$$

In this paper, if G or G_m is unstable, we will define $J(U)$ to be $+\infty$.

Given a stable system with state space realization (A, B, C), $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times p}$, $C \in \mathbb{R}^{q\times n}$, the optimal $\mathcal{H}_2$ model reduction problem can be formulated as a minimization problem

$$\min_{U^TU=I,\, U\in\mathbb{R}^{n\times m}} J(U). \tag{8}$$

It is interesting to point out that problem (8) can be reformulated as a minimization problem over a smooth manifold. In [26], problem (8) is rewritten as a minimization

problem on the Stiefel manifold. For an integer m with $1 \leq m \leq n$, the Stiefel manifold $\mathrm{St}(m,n)$ is defined by

$$\mathrm{St}(m,n) := \{U \in \mathbb{R}^{n\times m} | U^T U = I\}$$

and problem (8) can be rewritten as

$$\min_{U\in \mathrm{St}(m,n)} J(U). \tag{9}$$

In this paper, we shall show that problem (8) can be rewritten as a minimization problem over the Grassmann manifold. We now recall the notion of the Grassmann manifold (cf. [8]). For an integer m with $1 \leq m \leq n$, the real Grassmann manifold $\mathrm{Gr}(m,n)$ is defined as the set of all m-dimensional real linear subspaces of $\mathbb{R}^n$. From the definition of the Grassmann manifold, a point in the Grassmann manifold $\mathrm{Gr}(m,n)$ is an m-dimensional linear subspace. An orthogonal basis of the linear subspace can be stored as an $n \times m$ matrix in the Stiefel manifold. The Grassmann manifold can be thought as a quotient space of the Stiefel manifold. This can be explained as follows. Two points U_x, $U_y \in \mathrm{St}(m,n)$ are defined to be equivalent if the columns of U_x and U_y span the same m dimensional subspace, denoted as $U_x \equiv U_y$. For $m > 0$, the orthogonal group O_m is the group of $m \times m$ orthogonal matrices. It is clear that $U_x \equiv U_y$ if and only if there exists an orthogonal matrix $Q_m \in O_m$ such that $U_x Q_m = U_y$. The equivalent class $[U]$ for a point $U \in \mathrm{St}(m,n)$ is defined to be

$$[U] := \{UQ_m | Q_m \in O_m\}.$$

It is clear that there is a one-to-one correspondence between a point in the Grassmann manifold $\mathrm{Gr}(m,n)$ and an equivalent classes of $\mathrm{St}(m,n)$. Thus, the Grassmann manifold can be written as

$$\mathrm{Gr}(m,n) = \mathrm{St}(m,n)/O_m,$$

(cf. [8]). When performing computation on the Grassmann manifold, we will use the matrix $U \in \mathrm{St}(m,n)$ to represent an entire equivalent class $[U]$, the subspace spanned by the columns of U. For more details of the Grassmann manifold, we refer to [2, 8].

For optimal $\mathcal{H}_2$ model reduction problem (9), it is easy to see from the definition (7) of $J(U)$ and equation (6) that $J(U) = J(UQ_m)$, for any orthogonal matrix Q_m. In other words, the cost function $J(U)$ depends only on the subspaces spanned by the columns of U. This means that the cost function $J(U)$ is a function on the Grassmann manifold $\mathrm{Gr}(m,n)$. Therefore, the optimal $\mathcal{H}_2$ model reduction problem (8) can be reformulated as the following minimization problem over the Grassmann manifold

$$\min_{[U]\in \mathrm{Gr}(m,n)} J(U). \tag{10}$$

By utilizing the geometry of the Grassmann manifold, a fast gradient flow algorithm can be developed. It will be described in the next section.

3. Gradient Flow on the Grassmann Manifold

In this section, we describe the gradient flow method for solving the optimization problem (10). To this end, we first present an explicit formula for the cost function $J(U)$ and then we derive the partial derivatives of the cost function $J(U)$. Finally, we propose a gradient flow method on the Grassmann manifold.

To solve the optimal $\mathcal{H}_2$ model reduction problem, it is beneficial to have an explicit formula for $J(U)$. To reformulate the function G_e in a simpler form, we introduce the following three matrices

$$A_e = \begin{pmatrix} A & 0 \\ 0 & U^T A U \end{pmatrix},\ B_e = \begin{pmatrix} B \\ U^T B \end{pmatrix},\ C_e = \begin{pmatrix} C & -CU \end{pmatrix}.$$

The function G_e defined in (6) is then rewritten as

$$G_e(s) = C_e(sI - A_e)^{-1}B_e, \quad s \in \mathbb{C}.$$

Actually, matrices A_e, B_e and C_e define an error system with realization (A_e, B_e, C_e). The function G_e is called the transfer function of the error system. The following Lyapunov equations

$$A_e E_c + E_c A_e^T + B_e B_e^T = 0, \tag{11}$$

$$A_e^T E_o + E_o A_e + C_e^T C_e = 0, \tag{12}$$

can be solved to obtain E_c and E_o, the controllability and observability gramians of the error system. Partitioning the controllability gramian E_c and observability gramian E_o into

$$E_c = \begin{pmatrix} \Sigma_c & X \\ X^T & P \end{pmatrix}, E_o = \begin{pmatrix} \Sigma_o & Y \\ Y^T & Q \end{pmatrix},$$
$$\Sigma_c, \Sigma_o \in \mathbb{R}^{n\times n} \quad \text{and} \quad P, Q \in \mathbb{R}^{m\times m},$$

the Lyapunov equations (11) and (12) are equivalent to the following Sylvester equations

$$A\Sigma_c + \Sigma_c A^T + BB^T = 0, \tag{13}$$

$$A^T\Sigma_o + \Sigma_o A + C^T C = 0, \tag{14}$$

$$U^T AUP + PU^T A^T U + U^T BB^T U = 0, \tag{15}$$

$$U^T A^T UQ + QU^T AU + U^T C^T CU = 0, \tag{16}$$

$$AX + XU^T A^T U + BB^T U = 0, \tag{17}$$

$$A^T Y + YU^T AU - C^T CU = 0. \tag{18}$$

From equations (13) and (14), we know that Σ_c and Σ_o are the controllability and observability gramians of the system with the realization (A, B, C), respectively. If G and G_m are stable, it is a standard fact (cf. [28]) that the cost function $J(U)$ can be expressed in terms of the controllability gramian E_c,

$$\begin{aligned} J(U) &= \operatorname{trace}(C_e E_c C_e^T) \\ &= \operatorname{trace}[C^T C(\Sigma_c + UPU^T - 2XU^T)], \end{aligned} \tag{19}$$

or equivalently in terms of observability gramian E_o,

$$\begin{aligned} J(U) &= \operatorname{trace}(B_e^T E_o B_e) \\ &= \operatorname{trace}[BB^T(\Sigma_o + UQU^T + 2YU^T)]. \end{aligned} \tag{20}$$

We define the $n \times m$ matrix J_U as the partial derivatives of the cost function $J(U)$ with respect to the entries of $U \in \mathbb{R}^{n\times m}$, that is,

$$(J_U)_{ij} = \frac{\partial J}{\partial U_{ij}}, \quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, m.$$

The derivation of partial derivatives J_U can be found in [26].

Lemma 3.1. *Suppose that P, Q, X and Y are the solutions of equations* (15), (16), (17) *and* (18) *for given matrices A, B, C and $U \in \mathbb{R}^{n\times m}$. Let*

$$\begin{aligned} R :=& (-C^TC + A^TUY^T)X + (C^TCU + A^TUQ)P \\ &+ (BB^T + AUX^T)Y + (BB^TU + AUP)Q. \end{aligned} \tag{21}$$

Then, the partial derivatives of $J(U)$ at U is given by $J_U = 2R$.

It is easy to see that R defined in (21) can be rewritten as

$$\begin{aligned} R =& A^T[U(Y^TX + QP)] + A[U(X^TY + PQ)] \\ &+ C^T[C(-X + UP)] + B[B^T(Y + UQ)]. \end{aligned} \tag{22}$$

As we will see later, it will be more numerically efficient to use formula (22) for computing R.

We now present the gradient flow on the Grassmann manifold for solving problem (10). Following [8], the tangent space $\mathrm{T}_{[\mathrm{U}]}\mathrm{Gr}(m,n)$ at $[U] \in \mathrm{Gr}(m,n)$ is given by

$$\mathrm{T}_{[\mathrm{U}]}\mathrm{Gr}(m,n) = \{\xi \in \mathbb{R}^{n\times m} | U^T\xi = 0\}.$$

The Grassmann manifold becomes a Riemannian manifold by endowing $\mathrm{T}_{[\mathrm{U}]}\mathrm{Gr}(m,n)$ with the inner product

$$\langle \xi, \eta \rangle := 2\mathrm{trace}(\xi^T\eta) \quad \text{for} \quad \xi, \eta \in \mathrm{T}_{[\mathrm{U}]}\mathrm{Gr}(m,n).$$

From Lemma 3.1, the partial derivative of the cost function $J(U)$ at $[U] \in \mathrm{Gr}(m,n)$ is given by $J_U = 2R$. The gradient of J at the point $[U] \in \mathrm{Gr}(m,n)$ is defined as the tangent vector $\nabla J \in \mathrm{T}_{[\mathrm{U}]}\mathrm{Gr}(m,n)$ such that

$$\mathrm{trace}((J_U)^T\xi) = \langle \nabla J, \xi \rangle = 2\mathrm{trace}(\nabla J^T\xi) \tag{23}$$

for all tangent vector ξ at $[U]$. Solving (23) for ∇J such that $U^T\nabla J = 0$, the gradient of the cost function J at $[U] \in \mathrm{Gr}(m,n)$ is

$$\nabla J = R - UU^TR.$$

The gradient flow on the Grassmann manifold is

$$\frac{dU}{dt} = -\nabla J(U(t)) = U(t)U(t)^TR(t) - R(t). \tag{24}$$

We remark that the gradient flow (24) enjoys the same form of the gradient flow method on the Stiefel manifold [26]. A numerical algorithm for solving the gradient flow (24) will be presented in the next section.

4. A Numerical Algorithm for the Gradient Flow on the Grassmann Manifold

In this section, we develop a fast numerical algorithm for solving the gradient flow problem (24) and analyze its computational complexity.

The fast gradient flow algorithm to be proposed is based on gradient flow (24). It performs a series of decent steps, each taking along a geodesic, at the direction being the negative gradient of the cost function.

We first describe how the direction of the gradient flow is computed. To this end, we compute the partial derivatives of the cost function $J(U)$. Let (A, B, C) be the state space realization of the full order system. For $k = 0, 1, \ldots$, we denote by U_k the projection matrix for model reduction attaining by the fast gradient

flow algorithm at the k-th step. We compute matrices $P_k \in \mathbb{R}^{m\times m}$, $Q_k \in \mathbb{R}^{m\times m}$, $X_k \in \mathbb{R}^{n\times m}$ and $Y_k \in \mathbb{R}^{n\times m}$ by solving the following Sylvester equations,

$$U_k^T A U_k P_k + P_k U_k^T A^T U_k + U_k^T B B^T U_k = 0, \tag{25}$$

$$U_k^T A^T U_k Q_k + Q_k U_k^T A U_k + U_k^T C^T C U_k = 0, \tag{26}$$

$$A X_k + X_k U_k^T A^T U_k + B B^T U_k = 0, \tag{27}$$

$$A^T Y_k + Y_k U_k^T A U_k - C^T C U_k = 0. \tag{28}$$

We then compute R_k by the formula

$$\begin{aligned} R_k =& A^T[U_k(Y_k^T X_k + Q_k P_k)] + A[U_k(X_k^T Y_k + P_k Q_k)] \\ &+ C^T[C(-X_k + U_k P_k)] + B[B^T(Y_k + U_k Q_k)]. \end{aligned} \tag{29}$$

From Lemma 3.1, the partial derivatives J_{U_k} at the point U_k are given by $J_{U_k} = 2R_k$. The gradient of the cost function J at $[U_k] \in \mathrm{Gr}(m,n)$ is thus given by

$$\nabla J(U_k) = R_k - U_k(U_k^T R_k).$$

The direction of the gradient flow at U_k is the negative gradient direction, i.e.,

$$F_k = -\nabla J(U_k).$$

Next, we construct the orthogonal matrix U_{k+1} when U_k is available. The proposed gradient flow algorithm constructs U_{k+1} by performing a decent step along the geodesic at the direction F_k. To introduce the geodesic on the Grassmann manifold, we define the matrix functions sin and cos by the convergent power series,

$$\sin(Z) := \sum_{i=0}^{\infty} \frac{(-1)^i}{(2i+1)!} Z^{2i+1}, \quad \cos(Z) := \sum_{i=0}^{\infty} \frac{(-1)^i}{(2i)!} Z^{2i}, \tag{30}$$

respectively, where $Z \in \mathbb{R}^{m\times m}$. A geodesic is the curve of shortest length between two points on a manifold. The geodesic on the Grassmann manifold at the point $U \in \mathrm{St}(m,n)$, with the direction $F \in \mathbb{R}^{n\times m}$, is given by (cf. [8])

$$\mathcal{U}(t) = UV\cos(t\Lambda)V^T + W\sin(t\Lambda)V^T, \quad t \in \mathbb{R}, \tag{31}$$

where $W\Lambda V^T$ is the singular value decomposition of F, $W \in \mathbb{R}^{n\times m}$, $\Lambda \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{m\times m}$. It is easily verified that

$$\mathcal{U}(t)^T \mathcal{U}(t) = I \quad \text{for} \quad t \in \mathbb{R}.$$

Suppose that the singular value decomposition of F_k is

$$F_k = W_k \Lambda_k V_k^T,$$

where

$$V_k^T V_k = I \quad \text{and} \quad \Lambda_k = \mathrm{diag}(\lambda_{1k}, \lambda_{2k}, \ldots, \lambda_{mk}).$$

Thus, the geodesic of the Grassmann manifold $\mathrm{Gr}(m,n)$ at the point U_k, with the direction F_k, is given by

$$\mathcal{U}_k(t) = U_k V_k \cos(t\Lambda_k)V_k^T + W_k \sin(t\Lambda_k)V_k^T. \tag{32}$$

By choosing a suitable time step $t_k \geq 0$, we can construct U_{k+1} by

$$U_{k+1} = U_k V_k \cos(t_k\Lambda_k)V_k^T + W_k \sin(t_k\Lambda_k)V_k^T. \tag{33}$$

Note that U_{k+1} is an orthogonal matrix for any t_k.

We summarize below the fast algorithm for computing the gradient flow.

Algorithm 1 describes a fast gradient flow algorithm (FGFA) for the optimal $\mathcal{H}_2$ model reduction problem. One nice thing about this algorithm is that it has the

Input: System realization (A, B, C), the size m of the reduced order system.
Output: Reduced model realization (A_m, B_m, C_m).

Choose a matrix $U_0 \in \mathbb{R}^{n\times m}$ such that $U_0^T U_0 = I$ and set $k = 0$;
for $k = 0, 1, \ldots, N-1$ **do**
  Compute P_k, Q_k, X_k and Y_k by solving equations (25), (26), (27) and (28);
  Compute R_k using formula (29);
  Compute the gradient $\nabla J(U_k) = R_k - U_k(U_k^T R_k)$;
  Set the new search direction $F_k = -\nabla J(U_k)$;
  Compute the singular value decomposition of $F_k = W_k \Lambda_k V_k^T$;
  Minimize $J(\mathcal{U}_k(t))$ over $t \geq 0$ where

$$\mathcal{U}_k(t) = U_k V_k \cos(t\Lambda_k) V_k^T + W_k \sin(t\Lambda_k) V_k^T. \tag{34}$$

  Set $t_k = t_{\min}$ and $U_{k+1} = \mathcal{U}_k(t_k)$;
end
Get the projection matrix $U = U_N$;
$A_m = U^T A U$; $B_m = U^T B$; $C_m = CU$;

Algorithm 1: A fast gradient flow algorithm (FGFA).

ability to generate a sequence of orthogonal matrices from any orthogonal matrix $U_0 \in \mathbb{R}^{n\times m}$ for any step-size.

In the algorithm FGFA, we need to choose a suitable step length t_k by solving the minimization problem (34). In practical computation, we can use the inexact line search methods to compute the step length t_k. A popular inexact line search condition is the Armijo condition. A step size $t_k^A = \beta^j \gamma$ is called Armijo step size if j is the smallest nonnegative integer such that the following inequality holds

$$J(U_k) - J(\mathcal{U}_k(\beta^j \gamma)) \geq -\alpha \beta^j \gamma \langle \nabla J(U_k), F_k \rangle \tag{35}$$

for some given constants $\beta, \alpha \in (0, 1)$, $\gamma > 0$, where F_k is the search direction. There are other methods that select a suitable search step size, such as adaptive step size selection strategy proposed in [26] and the accelerated line search method [2].

We denote the transfer function of the reduced model by

$$G_m^{(k)}(s) = C_m^{(k)}(sI - A_m^{(k)})^{-1} B_m^{(k)}, \quad k = 0, 1, \ldots,$$

where

$$A_m^{(k)} := U_k^T A U_k, \ B_m^{(k)} := U_k^T B, \ C_m^{(k)} := C U_k.$$

The following theorem shows that the algorithm FGFA guarantees stability and is globally convergent.

Theorem 4.1. *For a stable system with realization (A, B, C), let $\{U_k\}$ be an infinite sequence generated by algorithm FGFA. If t_k is the Armijo step size and the transfer function $G_m^{(0)}$ is stable, then $G_m^{(k)}$ is stable for each k with*

$$J(U_{k+1}) \leq J(U_k), \quad k = 0, 1, 2, \ldots.$$

Proof. Since t_k is the Armijo step size, we have that

$$J(U_{k+1}) = J(\mathcal{U}_k(t_k)) \leq J(U_k) + \alpha \beta^j \gamma \langle \nabla J(U_k), F_k \rangle,$$

where $\beta, \alpha \in (0, 1), \gamma > 0$. Noting that $F_k = -\nabla J(U_k)$, we observe that $\langle \nabla J(U_k), F_k \rangle \leq 0$. Then

$$J(U_{k+1}) \leq J(U_k), \quad k = 0, 1, \ldots.$$

From the definition of $J(U)$, it follows from that stability of G that

$$J(U_k) := \|G - G_m^{(k)}\|_2^2$$

will be equal to ∞ if and only if $G_m^{(k)}$ is unstable. From the assumption that $G_m^{(0)}$ is stable, we have that $J(U_k) \leq J(U_0) < \infty$. Thus, $G_m^{(k)}$ is also stable for each $k = 0, 1, \ldots$. □

Theorem 4.2. *Under the assumption of Theorem 4.1, if $A+A^T$ is negative definite, then the algorithm FGFA is globally convergent in the sense that, for any initial projection matrix U_0,*

$$\lim_{k\to\infty} \|\nabla J(U_k)\| = 0.$$

Proof. Since $A + A^T$ is negative definite, it follows that the eigenvalues of $U^T AU$ have strictly negative real parts for any orthogonal matrix $U \in \mathbb{R}^{n\times m}$. This implies that the transfer function of reduced order model G_m is stable. It follows that $J(U)$ is a smooth function on the Grassmann manifold. The global convergence of Algorithm FGFA can be proved by applying the convergent analysis of the linear search method on a general Riemann manifold [2]. □

We remark that an adaptive step size selection strategy [26] and the accelerated line search method [2] guarantee the global convergence of gradient flow algorithm.

In Algorithm 1, we are required to choose an initial orthogonal matrix U_0. There are several different possibilities for choosing such a matrix. For example, denoting by $\mathrm{Colsp}(U_0)$ the column space of U_0, we can choose U_0 such that

$$\mathrm{Colsp}(U_0) = \mathcal{K}_m(A^{-1}, B),$$

where $\mathcal{K}_m(A^{-1}, B)$ is a block Krylov space (cf. [11])

$$\mathcal{K}_m(A^{-1}, B) := \mathrm{span}\{B, A^{-1}B, \ldots, A^{-m+1}B\},$$

$A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times p}$. We can also choose $\mathrm{Colsp}(U_0) = D_m(A, B, C)$, m dominant subspaces as in [19]. The dominant subspaces are the union of dominant eigen-spaces of the controllability and observability gramians to produce an approximate balanced-truncation reduced-order model (cf. [19]).

At each iterative step of algorithm FGFA, we need to solve Sylvester equations (25), (26), (27) and (28) to compute P_k, Q_k, X_k and Y_k. Since both $U_k^T AU_k$ and $U_k^T A^T U_k$ are $m \times m$ matrices and m is a small integer, we use the standard solvers (cf. [5]) to solve the Sylvester equations (25) and (26). To solve equations (27) and (28) efficiently, we employ the algorithm described in [3]. The main idea of the algorithm is to first compute the Schur decompositions of $U_k^T AU_k$ and $U_k^T A^T U_k$, and then solve m large sparse linear equations of the special type

$$(A - \eta I)x = b, \quad \eta \notin \sigma(A), b \in \mathbb{R}^n, \tag{36}$$

to obtain the solution, where $\sigma(A)$ is the set of all eigenvalues of A.

Next, we turn to analyzing the computational complexity of the fast gradient flow algorithm FGFA. The computational complexity will be measured by number of floating point multiplications. The following theorem presents an estimate of the computational complexity for the fast gradient flow algorithm FGFA. Let N_s denote the maximum number of search steps for solving the minimization problem (34) by the inexact line search method. By N we denote the maximum number of iterations for the algorithm FGFA. Let $\sigma(A)$ denote the set of all eigenvalues of A and $\mathcal{N}(A)$ the number of nonzero entries of A.

Theorem 4.3. *Let (A, B, C) be the system realization of a stable system. Suppose that $A \in \mathbb{R}^{n\times n}$ is a sparse matrix. If the linear equation (36) is solved by using the GMRES(r) algorithm or the like, which requires $O(rn+\mathcal{N}(A))$ multiplications, where r is a fixed integer, then the computational complexity of the algorithm FGFA is $O(N(nmr + m\mathcal{N}(A) + N_s nm^2 + nmp + nmq))$.*

Proof. For $k = 0, 1, \ldots$, at the k-th step of iterations, the computational cost of the algorithm FGFA comes from three parts.

Part one is the cost for solving the Sylvester equation (25), (26), (27) and (28). Since $U_k^T AU_k$ and $U_k^T A^T U_k$ are $m \times m$ matrices, it requires $O(m^3)$ number of multiplications to solve (25) and (26). To solve equation (27) and (28), we need first to compute a Schur decomposition of an $m \times m$ matrix, and then to solve large sparse linear equations (36) m times. Computing the Schur decomposition requires $O(m^3)$ floating point multiplications. By assumption, solving the large sparse linear equation (36) requires $O(rn + \mathcal{N}(A))$ floating point multiplications. Therefore the computational cost for solving (27) and (28) is $O(nmr + m\mathcal{N}(A) + m^3)$.

Part two is the cost for computing the search directions. It is straightforward to see that computing R_k needs $O(m\mathcal{N}(A)+nm^2+nmp+nmq)$ number of multiplications for given A, B, C and U_k. Thus, it requires $O(m\mathcal{N}(A) + nm^2 + nmp + nmq)$ number of multiplications to compute the search direction F_k for the algorithm FGFA.

The last part is the cost for solving the minimization problem (34) to obtain the solution $t_{\min}$ by the (inexact) line search method. In each search step, we need to compute geodesic (32) for some $t \geq 0$. Since the cost for computing the geodesic is $O(nm^2)$ and the maximum number of search steps is N_s, the cost for solving the minimization problem (34) by the inexact line search method is $O(N_s nm^2)$.

Therefore, for each iteration the algorithm FGFA requires $O(nmr + m\mathcal{N}(A) + N_s nm^2+nmp+nmq)$ number of floating point multiplications. Since the maximum number of iterations of the algorithm FGFA is N, we obtain the desired overall computational complexity for the algorithm. □

We remark that the restarted GMRES algorithm, i.e., GMRES(r) algorithm requires $O(rn+\mathcal{N}(A))$ floating point multiplications to solve the large sparse linear equation (36), (cf. [21]).

In most practical cases, the order m of the reduced model is much smaller than the order n of the original model. For a fixed m, if the full order system is sparse, the computational cost of the algorithm FGFA grows *linearly* as the dimension of the full order system.

5. Comparison with the Existing Gradient Flow Algorithm

In this section, we first review the existing gradient flow algorithm (we shall call it the GFA algorithm) on the Stiefel manifold proposed in [26]. We then show that under certain conditions, the gradient flow algorithm on the Grassmann manifold proposed in this paper and the GFA algorithm generate the same sequence of orthogonal matrices. We compare the numerical efficiency of the two algorithms and show that the proposed algorithm has computational advantage over the GFA algorithm.

We first review the gradient flow approach proposed in [26] for the minimization problem (9). We then review the numerical algorithm proposed by the same authors for the gradient flow on the Stiefel manifold.

Let P, Q, X and Y be the solutions of Sylvester equations (15), (16), (17) and (18) for given state space realization (A, B, C) and U. We define

$$\begin{aligned}\widetilde{R} :=&(-C^TC + A^TUY^T)X + (C^TCU + A^TUQ)P \\ &+ (BB^T + AUX^T)Y + (BB^TU + AUP)Q.\end{aligned} \tag{37}$$

The gradient of the cost function $J(U)$ at the point $U \in \mathrm{St}(m, n)$ is $\nabla J = \widetilde{R} - UU^T\widetilde{R}$. The gradient flow method on the Stiefel manifold is given by

$$\frac{dU}{dt} = -\nabla J(U(t)) = (U(t)U(t)^T - I)\widetilde{R}(t). \tag{38}$$

It is easy to see that R defined in (22) and $\widetilde{R}$ defined in (37) are equal for given A, B, C and U. As a result, the gradient flow (24) and the gradient flow (38) are equal. However, their major difference is that the gradient flow (24) is defined on the Grassmann manifold while the gradient flow (38) is defined on the Stiefel manifold.

To derive the numerical algorithm for gradient flow (38), the following lemma from [26] which shows the symmetry of matrix $U^T\widetilde{R}$ is useful.

Lemma 5.1. *Suppose that P, Q, X and Y are the solutions of Sylvester equations* (15), (16), (17) *and* (18) *for given matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$, $C \in \mathbb{R}^{q\times n}$ and $U \in \mathrm{St}(m, n)$. If $\widetilde{R}$ is defined by* (37), *then*

$$U^T\widetilde{R} = -Y^TAX - QU^TAUP - X^TA^TY - PU^TA^TUQ, \tag{39}$$

and $U^T\widetilde{R}$ is a symmetric matrix.

In passing, we indicate that there is a typo in paper [26] where $U^T\widetilde{R}$ was mistakenly written as $U^T\widetilde{R} = Y^TAX + QU^TAUP + X^TA^TY + PU^TA^TUQ$.

Since $U^T\widetilde{R}$ is symmetric, the gradient of the cost function $J(U)$ is given by

$$\nabla J = \widetilde{R} - UU^T\widetilde{R} = \widetilde{R} - U\widetilde{R}^TU.$$

Let $\Gamma := U\widetilde{R}^T - \widetilde{R}U^T$, $\Gamma \in \mathbb{R}^{n\times n}$. The gradient flow (38) is reformulated in [26] as

$$\frac{dU}{dt} = -\nabla J(U(t)) = \Gamma(t)U(t). \tag{40}$$

We next outline the numerical algorithm presented in [26] for solving the gradient flow algorithm (38). At the k-th step, suppose that matrix U_k is available, we want to construct U_{k+1} from U_k. Matrices $P_k \in \mathbb{R}^{m\times m}$, $Q_k \in \mathbb{R}^{m\times m}$, $X_k \in \mathbb{R}^{n\times m}$, $Y_k \in \mathbb{R}^{n\times m}$ are the solutions of Sylvester equations (25), (26), (27) and (28) for given state space realization (A, B, C) and U_k. Suppose that $\widetilde{R}_k \in \mathbb{R}^{n\times m}$ is defined by

$$\begin{aligned}\widetilde{R}_k :=&(-C^TC + A^TU_kY_k^T)X_k + (C^TCU_k + A^TU_kQ_k)P_k \\ &+ (BB^T + AU_kX_k^T)Y_k + (BB^TU_k + AU_kP_k)Q_k.\end{aligned} \tag{41}$$

Let $\Gamma_k \in \mathbb{R}^{n\times n}$ be defined by

$$\Gamma_k := U_k\widetilde{R}_k^T - \widetilde{R}_kU_k^T. \tag{42}$$

For any matrix $Z \in \mathbb{R}^{n\times n}$, we recall the matrix exponential by the convergent series

$$\mathrm{e}^Z := \sum_{i=0}^{\infty} \frac{1}{i!}Z^i. \tag{43}$$

For given $t_k \in \mathbb{R}$, $U_k \in \mathrm{St}(m,n)$ and Γ_k, the gradient flow on the Stiefel manifold proposed in [26] is defined as follows

$$U_{k+1} = e^{t_k \Gamma_k} U_k, \quad k = 0, 1, \ldots. \tag{44}$$

We next show that the gradient flow (33) proposed in this paper and the gradient flow on the Stiefel manifold (44) are equivalent in the sense that the two algorithms generate the same sequence of orthogonal matrices under certain conditions. To this end, we first introduce a useful lemma. We recall the matrix function sinc defined by a convergent series.

Definition 5.2. *For a matrix $Z \in \mathbb{R}^{n\times n}$, the matrix function* sinc *is defined by the convergent infinite sums:*

$$\mathrm{sinc}(Z) := \sum_{i=0}^{\infty} \frac{(-1)^i}{(2i+1)!} Z^{2i}, \quad Z \in \mathbb{R}^{n\times n}.$$

Lemma 5.3. *Assume $N \in \mathbb{R}^{n\times n}$, $\widetilde{U} \in \mathbb{R}^{n\times m}$. If there exists a matrix $M \in \mathbb{R}^{m\times m}$ such that*

$$N^2 \widetilde{U} = -\widetilde{U} M^2, \tag{45}$$

then

$$e^{tN}\widetilde{U} = \widetilde{U}\cos(tM) + tN\widetilde{U}\mathrm{sinc}(tM), \quad t \in \mathbb{R}.$$

Proof. Write the power series of e^{tN} in two series according to the even and odd powers

$$e^{tN} = \sum_{i=0}^{\infty} \frac{1}{(2i)!}(tN)^{2i} + \sum_{i=0}^{\infty} \frac{1}{(2i+1)!}(tN)^{2i+1}.$$

Repeatedly using hypothesis (45), we obtain for each positive integer i that

$$N^{2i}\widetilde{U} = (-1)^i \widetilde{U} M^{2i}.$$

Combining this formula with the last equation, we observe that

$$e^{tN}\widetilde{U} = \widetilde{U}\sum_{i=0}^{\infty} \frac{(-1)^i}{(2i)!}(tM)^{2i} + tN\widetilde{U}\sum_{i=0}^{\infty} \frac{(-1)^i}{(2i+1)!}(tM)^{2i}.$$

Using the definition of matrix functions cos and sinc, we obtain the desired result. □

Lemma 5.3 shows that the matrix exponential $e^{tN}\widetilde{U}$, which is expensive in computation, can be implemented by a significantly less expensive method. Another interesting method of efficiently computing the matrix exponential was presented in [12].

Lemma 5.4. *Suppose that P, Q, X and Y are the solutions of Sylvester equations* (15), (16), (17) *and* (18) *for given matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$, $C \in \mathbb{R}^{q\times n}$ and $U \in \mathrm{St}(m,n)$. Let R and $\widetilde{R}$ be defined respectively by* (22) *and* (37). *Let $\Gamma := U\widetilde{R}^T - \widetilde{R}U^T$ and $F := U(U^T R) - R$. If $W\Lambda V^T$ is the singular value decomposition of F, with $W \in \mathbb{R}^{n\times m}$, $\Lambda \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{m\times m}$, then*

$$e^{t\Gamma}U = UV\cos(t\Lambda)V^T + W\sin(t\Lambda)V^T, \quad t \in \mathbb{R}.$$

Proof. By Lemma 5.1, we know that $U^T\widetilde{R}$ is symmetric. Therefore, $U^T R = U^T\widetilde{R}$ is symmetric since $R = \widetilde{R}$. It is easy to see that $F = \Gamma U$. Using the definition of Γ and the relation that $R = \widetilde{R}$, we obtain that

$$\Gamma^2 U = -U(R^T R - R^T U R^T U). \tag{46}$$

Since $F = U(U^T R) - R$, again using the symmetry of $U^T R$, we have that

$$F^T F = R^T R - R^T U R^T U. \tag{47}$$

Combining (46) and (47), we observe that $\Gamma^2 U = -U(F^T F)$.

Letting $H := V\Lambda V^T$, since $W\Lambda V^T$ is the singular value decomposition of F, there holds $H^2 = F^T F$. Thus, we have $\Gamma^2 U = -UH^2$. By Lemma 5.3, we can get

$$e^{t\Gamma} U = U\cos(tH) + tF\mathrm{sinc}(tH). \tag{48}$$

Using the facts that $H = V\Lambda V^T$ and $V^T V = I$, we have that

$$\cos(tH) = V\cos(t\Lambda)V^T \quad \text{and} \quad \mathrm{sinc}(tH) = V\mathrm{sinc}(t\Lambda)V^T.$$

Substituting these equations into (48) yields the formula

$$e^{t\Gamma} U = UV\cos(t\Lambda)V^T + tFV\mathrm{sinc}(t\Lambda)V^T.$$

Using the singular decomposition of F and noting $Z\mathrm{sinc}(Z) = \sin Z$, we obtain the desired equation. □

Lemma 5.4 leads to the following result.

Theorem 5.5. *If the same state space realization (A, B, C), initial orthogonal matrix U_0 and the step size t_k, $k = 0, 1, \ldots$, are used, then the gradient flow algorithm (33) on the Grassmann manifold and the gradient flow algorithm (44) on the Stiefel manifold generate the same sequence of orthogonal matrices U_k, $k = 1, 2, \ldots$.*

Proof. This theorem follows directly from Lemma 5.4 at each iterative step. □

We now begin the comparison of the computational complexity of the two gradient flow algorithms. Under the same state space realization and the initial conditions, the comparison of the computational complexity is mainly on the following two respects:

(1) **The comparison of the computational complexity for computing R and $\widetilde{R}$.** It is easy to see that the computational costs for computing R and $\widetilde{R}$ are $O(m\mathcal{N}(A) + nm^2 + nmp + nmq)$ and $O(n^2 m + n^2 p + n^2 q)$, respectively. Clearly, if A is a sparse matrix (for example, $\mathcal{N}(A) = O(n)$), computing R requires much less computational costs than computing $\widetilde{R}$.

(2) **The comparison of computational complexity for computing gradient flows** (33) **and** (44)**.** The complexity for computing the geodesic (33) is $O(nm^2)$. The algorithm (44) requires computing the matrix exponential of a matrix at each iterative step. Such computation is very costly. It needs at least $O(n^3)$ or even $O(n^4)$ floating point multiplications to compute the matrix exponential for general matrices (cf. [17]).

From the above discussion, we know that the computational complexity for solving the gradient flow (44) is at least $O(n^3)$ while the complexity for computing the gradient flow (33) is $O(m\mathcal{N}(A) + N_s nm^2 + nmp + nmq)$, depending on the sparsity $\mathcal{N}(A)$ of the state space realization. In most cases, the order m of the reduced model is much smaller than the order n of the original model. Therefore, the fast gradient flow algorithm (33) is much more numerically efficient than algorithm (44). In summary, the proposed algorithm generates the same matrix sequence as the existing gradient flow algorithm does, with much less computational effort.

6. Sequentially Quadratic Approximation

In this section, we propose a globally convergent algorithm which converges faster. Since the cost function $J(U)$ is a nonlinear and implicit function of U, it seems not easy to apply the superlinear convergent methods such as the Newton algorithm [27], the trust region method [2], and the BFGS algorithm [20] directly to the optimal $\mathcal{H}_2$ model reduction problem. In order to construct an algorithm that converges faster, we approximate the cost function locally by a quadratic function at each iterative step, and then search along the geodesic of the Grassmann manifold at the search direction constructed by the approximate quadratic function. The combination with the fast gradient flow algorithm makes it globally convergent.

We describe the construction of the quadratic function which approximates the cost function at each iterative step. For $k = 0, 1, \ldots$, we denote by U_k the projection matrix at the k-th step. Matrices P_k and X_k are computed by solving the Sylvester equations (25) and (27). We construct a quadratic function $\widetilde{J}$ of U

$$\widetilde{J}(U) = \mathrm{trace}[C^T C(\Sigma_c + U P_k U^T - 2X_k U^T)] \tag{49}$$

to approximate the cost function J by fixing P with P_k and X with X_k. Solving the minimization problem

$$\min_{U^T U = I,\, U \in \mathbb{R}^{n\times m}} \widetilde{J}(U) \tag{50}$$

will give a good estimate of the solution of the original minimization problem (10).

We can construct a search direction by solving the minimization problem (50). If $\widetilde{U}_{k+1}$ satisfies $\widetilde{U}_{k+1} P_k = X_k$, then the partial derivatives $\widetilde{J}_{\widetilde{U}_{k+1}} = 0$. Let $\widetilde{U}_{k+1} = X_k P_k^{-1}$ if P_k is invertible. Since $\widetilde{U}_{k+1}$ is not an orthogonal matrix in general, we can not use $\widetilde{U}_{k+1}$ directly to produce a reduced model. However, we can utilize the difference $\widetilde{U}_{k+1} - U_k$ to construct the search direction. We project $\widetilde{U}_{k+1} - U_k$ to tangent space $\mathrm{T}_{[\mathrm{U_k}]}\mathrm{Gr}(m, n)$ of the point $[U_k] \in \mathrm{Gr}(m, n)$, that is

$$\begin{aligned} \Delta_k &= \Pi(\widetilde{U}_{k+1} - U_k) \\ &= (\widetilde{U}_{k+1} - U_k) - U_k[U_k^T(\widetilde{U}_{k+1} - U_k)] \\ &= \widetilde{U}_{k+1} - U_k(U_k^T \widetilde{U}_{k+1}). \end{aligned}$$

Here $\Pi = (I - U_k U_k^T)$ denotes the projection onto the tangent space $\mathrm{T}_{[\mathrm{U_k}]}\mathrm{Gr}(m, n)$ of the Grassmann manifold (cf. [8]).

Next, we discuss how a globally convergent algorithm is constructed by combining the search direction Δ_k and the negative gradient direction $-\nabla J(U_k)$. In order to construct a globally convergent algorithm, some restrictions need to be imposed on the search direction F_k. The sequence $F_k \in \mathrm{T}_{[\mathrm{U_k}]}\mathrm{Gr}(m, n)$ is said to be gradient-related if for any subsequence of U_k that converges to a non-critical point of the cost function J, the corresponding subsequence F_k is bounded and satisfies

$$\lim_{k\to\infty} \sup_{k\in\mathcal{K}} \langle \nabla J(U_k), F_k \rangle < 0.$$

For more information about the gradient-related sequence, we refer to the book [2]. It is easy for us to construct a gradient-related sequence by combining the search direction Δ_k and the negative gradient $-\nabla J(U_k)$. If for some constants $c_1 > 0$ and $c_2 \in (-1, 0)$, Δ_k satisfies

$$\|\Delta_k\| > c_1 \quad \text{and} \quad \frac{\langle \nabla J(U_k), \Delta_k \rangle}{\|\nabla J(U_k)\| \cdot \|\Delta_k\|} < c_2, \tag{51}$$

then we set the new search direction F_k to be $F_k = \Delta_k$, or else the search direction is set to be $F_k = -\nabla J(U_k)$. It is easy to prove that the sequence F_k constructed in this way is gradient-related. The condition (51) is called the gradient-related condition.

The following is the sequentially quadratic approximation algorithm (SQA).

Initialization: Choose $U_0 \in \mathbb{R}^{n\times m}$ such that $U_0^T U_0 = I$;
for $k = 0, 1, 2, \ldots, N-1$ **do**
  Compute X_k, P_k and compute $\widetilde{U}_{k+1} = X_k(P_k)^{-1}$;
  Project $\widetilde{U}_{k+1} - U_k$ to the tangent space $\mathrm{T}_{[\mathrm{U_k}]}\mathrm{Gr}(n,m)$ of $[U_k]$

$$\Delta_k = \widetilde{U}_{k+1} - U_k(U_k^T\widetilde{U}_{k+1}).$$

  if Δ_k *satisfies the gradient-related condition* (51) **then**
    set the search direction $F_k = \Delta_k$;
  else
    set the search direction $F_k = -\nabla J(U_k)$;
  end
  Compute the singular value decomposition of $F_k = W_k\Lambda_k V_k^T$;
  Minimize $J(\mathcal{U}_k(t))$ over $t \geq 0$ where

$$\mathcal{U}_k(t) = U_k V_k \cos(t\Lambda_k)V_k^T + W_k \sin(t\Lambda_k)V_k^T.$$

  Set $t_k = t_{\min}$ and $U_{k+1} = \mathcal{U}_k(t_k)$;
end

Algorithm 2: Sequentially quadratic approximation algorithm (**SQA**).

The above algorithm is numerically efficient and has low computational costs. For model reduction of many large-scale systems, this algorithm produces satisfactory result and converges quickly. The quadratic convergence is observed in numerical experiments.

The transfer function of the reduced model is denoted by

$$G_m^{(k)}(s) = C_m^{(k)}(sI - A_m^{(k)})^{-1}B_m^{(k)}, \quad k = 0, 1, \ldots,$$

where

$$A_m^{(k)} = U_k^T A U_k, \quad B_m^{(k)} = U_k^T B, \quad C_m^{(k)} = CU_k.$$

The following theorem shows that the algorithm SQA guarantees the stability of the reduced model and it is globally convergent.

Theorem 6.1. *Given a stable system with realization (A, B, C), let U_k be an infinite sequence generated by algorithm SQA. If t_k is the Armijo step size and the transfer function $G_m^{(0)}(s)$ is stable, then $G_m^{(k)}(s)$ is stable for each k with*

$$J(U_{k+1}) \leq J(U_k), \quad k = 0, 1, \ldots.$$

Proof. From the construction of the gradient-related sequece F_k, we know that $\langle \nabla J(U_k), F_k\rangle \leq 0$. Since t_k is the Armijo step size, we have that

$$\begin{aligned} J(U_{k+1}) = J(\mathcal{U}_k(t_k)) &\leq J(U_k) + \alpha\beta^j\gamma \langle \nabla J(U_k), F_k\rangle \\ &\leq J(U_k), \end{aligned}$$

where $\beta, \alpha \in (0,1)$, $\gamma > 0$. As in the discussion of Theorem 4.1, $G_m^{(k)}$ is stable for each k if $G_m^{(0)}$ is stable. □

Theorem 6.2. *Under the assumption of Theorem 6.1, if $A+A^T$ is negative definite, then the algorithm SQA is globally convergent*

$$\lim_{k\to\infty} \|\nabla J(U_k)\| = 0.$$

Proof. As in the discussion of Theorem 4.2, $J(U)$ is a smooth function on the Grassmann manifold. It is known from [2] that if the search direction is gradient-related and the search step size is the Armijo step size, then the linear search method on manifolds is globally convergent. Therefore, the global convergence of algorithm SQA is guaranteed. □

We remark that another way of constructing a quadratic function to approximate the cost function J is to fix Q with Q_k and Y and Y_k in the equation (20). The construction of search directions Δ_k is similar to the algorithm SQA.

The orthogonal projection based model reduction method is known to be easier to preserve the stability and passivity of the reduced order system (cf. [10, 11, 18]). For example, if the system realization (A, B, C) satisfies that $A + A^T$ is negative definite and $C = B^T$, then the reduced model (U^TAU, U^TB, CU) is also passive (cf. [11]). So the algorithms FGFA and SQA guarantee passivity in this case. System realizations of this kind often arise in RC or RLC interconnect circuits.

7. Numerical Experiments

We present in this section numerical experiments to confirm the approximation accuracy and computational efficiency of the algorithms introduced in this paper. All numerical programs are run by using Matlab 7. 4 (R2007a), on a PC with a Intel Xeon 3.06 GHz processor and 3.3 GBytes of RAM.

7.1. Heat transfer model. Consider the heat transfer equation in the domain $\Omega = (0, 1)^2$. The underlying parabolic differential equation is given by

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

with $u = u(t, x, y)$, $(x, y) \in \Omega$, $t \in [0, \infty)$, (cf. [14]). Suppose that the differential equation is discretized by finite differences using a uniform grid with d^2 grid points. The resulting stiffness matrix $A \in \mathbb{R}^{d^2\times d^2}$ is sparse, stable, and its bandwidth is d. Let $n = d^2$ be the system order. Let $b_1 \in \mathbb{R}^n$ be the vector with each entry equal to 1, $b_2 \in \mathbb{R}^n$ be a random vector. Let $B = [b_1, b_2]$ and $C = B^T$. The system constructed in this way is a multi-input multi-output (MIMO) system.

We apply the proposed fast gradient flow algorithm FGFA to reduce this MIMO system. We use the gradient flow algorithm GFA (44) for comparison. The algorithms FGFA and GFA start from the same state space realization (A, B, C) and use the same initial orthogonal matrix U_0 such that $\mathrm{Colsp}(U_0) = \mathrm{D}_3(A, B, C)$. We choose the same step length t_k at each iterative step for algorithms FGFA and GFA.

The numerical results are presented in Table 1. The relative $\mathcal{H}_2$ error is computed by using $\frac{\|G-G_m\|_2}{\|G\|_2}$. The estimate relative $\mathcal{H}_2$ error is computed by the relative $\mathcal{H}_2$ difference between the reduced order model G_m and a sufficiently accurate reduced model (in this example, it is a reduced model of order 100 generated by the Krylov subspace based method [3]). The CPU time is the total computing time for each algorithm measured in seconds. The last column (n, m) stands for the order of the systems, where n is the order of the full order system and m is the order of the reduced order system. We can see from Table 1 that the proposed algorithm FGFA

spends less computing time than Algorithm GFA does. Moreover, we observe that the computational time for the proposed algorithm FGFA grows linearly, while that for Algorithm GFA grows cubically. In the case $n = 25600$, Algorithm GFA fails to compute the reduced system due to its large computational cost, while the proposed algorithm FGFA still works very well.

TABLE 1. Example 1. Comparison.

	Relative error	CPU time (s)	(n, m)
GFA	4.07×10^{-3}	89	(900, 3)
FGFA	4.05×10^{-3}	7	
	Relative error	CPU time (s)	(n, m)
GFA	5.96×10^{-3}	447	(1600, 3)
FGFA	5.96×10^{-3}	11	
	Relative error	CPU time (s)	(n, m)
GFA	7.15×10^{-3}	4793	(3600, 3)
FGFA	7.15×10^{-3}	21	
	Relative error (*estimate*)	CPU time (s)	(n, m)
GFA			(25600, 3)
FGFA	6.55×10^{-4}	302	

We investigate this example further by reporting the convergence curve ($n = 900$) in Figure 1. At each step of the iteration, we compute the relative $\mathcal{H}_2$ error and plot this error vs the number of iterations. We can see from Figure 1 that the relative $\mathcal{H}_2$ error of the two algorithms decreases as the number of iterations increases. Note that algorithms FGFA and GFA obtain the same relative $\mathcal{H}_2$ error at each iteration step. This confirms that the two algorithms generate the same sequence of orthogonal matrices.

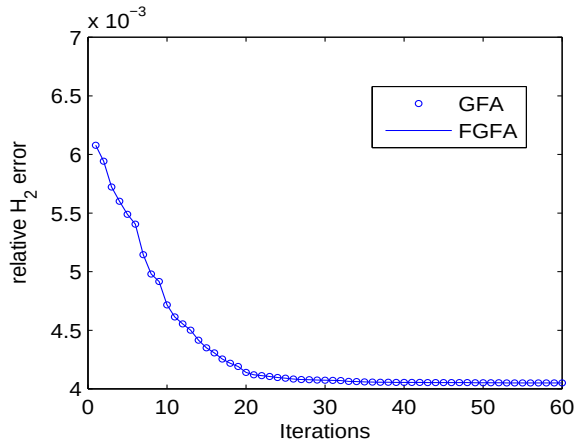

FIGURE 1. Relative error for heat transfer model, $n = 900$.

7.2. PEEC model. The PEEC model is a model of a spiral inductor, taken from Oberwolfach benchmark collection [1]. The system originally obtained from the PEEC model is a generalized symmetric SISO system of order $n = 1434$. In order to get a MIMO system, we modified the input and output of the PEEC model by adding to it random vectors. The resulting system is a symmetric MIMO system with 2 inputs and 2 outputs.

We apply the proposed sequentially quadratic approximation algorithm (SQA) to reduce this model. We use the iterative rational Krylov algorithm (IRKA) proposed in [13], the algorithm (VGA) proposed in [23] and the trust region algorithm (TRA) proposed in [4] for comparison. We reduce the order to $m = 2, 4$ with the same initial orthogonal matrix for all these algorithms. The resulting relative $\mathcal{H}_2$ errors are tabulated in Table 2. The column Iter. lists the number of iterations for reaching the relative $\mathcal{H}_2$ error. The proposed algorithm SQA guarantees the stability of the order-reduced model and converges fast. Algorithms IRKA and VGA do not guarantee stability of the order-reduced model, while algorithm TRA guarantees stability but the rate of convergence is relatively slow for this example. This conforms the numerical efficiency of the proposed algorithm SQA.

TABLE 2. Comparison of Relative $\mathcal{H}_2$ error (PEEC model)

	$m = 2$		$m = 4$		Preserving
	Relative error	Iter.	Relative error	Iter.	stability
SQA	2.99×10^{-2}	5	9.53×10^{-3}	15	Yes
IRKA	2.99×10^{-2}	6	9.53×10^{-3}	17	No
VGA	2.99×10^{-2}	6	9.53×10^{-3}	17	No
TRA	2.99×10^{-2}	24	1.21×10^{-2}	50	Yes

The convergence curves ($m = 2$) of the four algorithms are depicted in Figure 2. At each step of the iteration, we compute the relative $\mathcal{H}_2$ error and plot this error vs the number of iterations. The superlinear convergence of algorithm SQA is observed from Figure 2. This example demonstrates the effectiveness of sequentially approximating the cost function by a quadratic function.

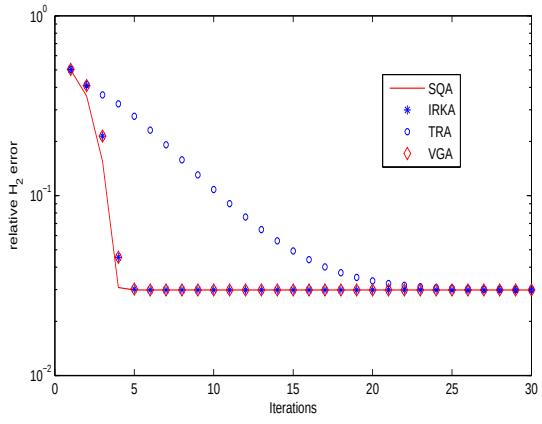

FIGURE 2. Relative $\mathcal{H}_2$ errors for PEEC model, $m = 2$.

8. Conclusion

In this paper, we reformulate the optimal $\mathcal{H}_2$ model reduction problem as a minimization problem over the Grassmann manifold. Based on the geodesic of the Grassmann manifold, we propose a fast gradient flow algorithm which is numerically effective and suitable for large-scale MIMO systems. Furthermore, based on approximating the cost function by a quadratic function at each iterative step, we propose a globally convergent algorithm which converges quickly and preserves stability of the reduced-order system. Numerical examples are presented to demonstrate the computational efficiency of the proposed algorithms.

References

[1] *Oberwolfach Model Reduction Benchmark Collection*, available online at http://simulation.uni-freiburg.de/benchmark/.
[2] P. A. Absil, R. Mahony and R. Sepulchre, Optimization Algorithms on Matrix Manifolds, Princeton, NJ: Princeton University Press, 2008.
[3] A. C. Antoulas, Approximation of Large-scale Dynamical Systems, Philadelphia, PA: SIAM, 2005.
[4] C. Beattie and S. Gugercin, A trust region method for optimal $\mathcal{H}_2$ model reduction, Proc. 48th IEEE Conf. Decision and Control (CDC 2009), Shanghai, China, Dec. 2009.
[5] R. H. Bartels and G.W. Stewart, Solution of the matrix equation AX + XB = C, Algorithm 432. Comm. ACM., vol. 15, pp. 820–826, 1972.
[6] G. Chen, C. K. Chui and Y. Yu, Optimal Hankel-norm approximation approach to model reduction of large-scale Markov chains, Int. J. Systems SCI., vol. 23, NO. 8, pp. 1289–1297, 1992.
[7] C. K. Chui and G. Chen, Discrete H^∞ Optimization, Berlin: Springer, 1st ed. 1992, 2nd ed. 1997.
[8] A. Edelman, T. A. Arias and S. T. Smith, The geometry of algorithms with orthogonality constraints, SIAM J. Matrix Anal. Appl., vol. 20, pp. 303–353, 1999.
[9] P. Fulcheriy and M. Oliviy, Matrix rational H_2 approximation: a gradient algorithm based on Schur analysis, SIAM J. Control Optim., vol. 36, pp. 2103–2127, 1998.
[10] R. W. Freund, Krylov-subspace methods for reduced-order modeling in circuit simulation, J. Comput. Appl. Math., vol. 123, pp. 395–421, 2000.
[11] R. W. Freund, Model reduction methods based on Krylov subspaces, Acta Numer., vol. 12, pp. 267–319, 2003.
[12] K. Gallivan, A. Srivastava, X. Liu and P. Van Dooren, Efficient algorithms for inferences on Grassmann manifolds, Proc. 12th IEEE Workshop on Statistical Signal Processing, St. Louis, Aug. 2003, pp. 301–304.
[13] S. Gugercin, A. C. Antoulas and C. Beattie, $\mathcal{H}_2$ model reduction for large-scale linear dynamical systems, SIAM J. Matrix Anal. Appl., vol. 30, pp. 609–638, 2008.
[14] R. Haberman, Applied partial differential equations: with Fourier series and boundary value problems, 4th Edition, New Jersey, NJ: Prentice-Hall, 2004.
[15] D. C. Hyland and D. S. Bernstein, The optimal projection equations for model reduction and the relationships among the methods of Wilson, Skelton and Moore, IEEE Trans. Automat. Contr., vol. 30, pp. 1201–1211, 1985.
[16] A. Lepschy, G. A. Mian, G. Pinato and U. Viaro, Rational L_2 approximation: a non-gradient algorithm, Proc. 30th Conf. Decis. Contr., Brighton, Dec. 1991, vol.3, pp. 2321–2323.
[17] C. Moler and C. V. Loan, Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later, SIAM Review, vol. 45, pp. 3–49, 2003.
[18] A. Odabasioglu, M. Celik and L. T. Pileggi, PRIMA: Passive reduced-order interconnect macromodeling algorithm, IEEE Trans. CAD, vol. 17 , pp. 645–654, 1998.
[19] T. Penzl, Algorithms for model reduction of large dynamical systems, Linear Alg. Appl., vol. 415, pp. 322–343, 2006.
[20] C. Qi, K. A. Gallivan and P. A. Absil, Riemannian BFGS algorithm with applications, Recent Advances in Optimization and its Applications in Engineering, pp. 183-192, 2010.
[21] Y. Saad and M.H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput., vol. 7, pp. 856–869, 1986.
[22] J. T. Spanos, M. H. Milman and D. L. Mingori, A new algorithm for L_2 optimal model reduction, Automatica., vol. 28, pp. 897–909, 1992.

[23] P. Van Dooren, K.A. Gallivan and P. A. Absil, H_2-optimal model reduction of MIMO systems, Appl. Math. Lett., vol. 21, pp. 1267–1273, 2008.
[24] P. Van Dooren, K. A. Gallivan and P. A. Absil, $\mathcal{H}_2$-Optimal model reduction with higher-order poles, SIAM J. Matrix Anal. Appl., vol. 31, pp. 2738–2753, 2010.
[25] Y. Xu and T. Zeng, Optimal $\mathcal{H}_2$ model reduction for large scale MIMO systems via tangential interpolation, Int. J. Numer. Anal. Model., vol. 8, pp. 174–188, 2011.
[26] W. Y. Yan and J. Lam, An approximate approach to $\mathcal{H}_2$ optimal model reduction, IEEE Trans. Automat. Contr., vol. 44, pp. 1341–1358, 1999.
[27] Y. Yang, Globally convergent optimization algorithms on Riemannian manifolds: uniform framework for unconstrained and constrained optimization, J. Optim. Theory Appl., pp. 132, vol. 245–265, 2007.
[28] K. Zhou, J. C. Doyle and K. Glover, Robust and Optimal Control, New Jersey, NJ: Prentice-Hall, 1996.

Department of Mathematics, Syracuse University, Syracuse, NY 13244, USA and Guangdong Province Key Lab of Computational Sciences, School of Mathematics and Computational Science, Sun Yat-sen University, Guangzhou, China, 510275.
E-mail: *yxu06@syr.edu*

School of Mathematical Sciences, South China Normal University, Guangzhou 510631, China, and School of Mathematics and Computational Sciences, Sun Yat-Sen University, Guangzhou, China, 510275.
E-mail: *zengtsh@gmail.com*