Online balanced truncation for linear time-varying systems using continuously differentiable interpolation on Grassmann manifold*

Nguyen Thanh Son[1,2] and Pierre-Yves Gousenbourger[1] and Estelle Massart[1] and P.-A. Absil[1]

***Abstract*— We consider model order reduction of linear time-varying systems on a finite time interval using balanced truncation. A standard way to perform MOR is to first numerically integrate the associated pair of differential Lyapunov equations for the two gramians, then compute projection matrices using the square root method, and finally formulate the reduced systems at each time instant of a chosen grid. This approach is well-known for delivering good approximation, but rather costly in computation and storage requirement. Furthermore, if one needs to compute the reduced system for any new time instant that is not included in the chosen grid, the mentioned procedure must be performed again without explicitly making use of the already computed data. For dealing with such a situation, we propose to store the projection matrices corresponding to a simplified sparse time grid and to use them to recover the projection subspaces at any other time instant via curve interpolation on the Grassmann manifold. By doing this, we can avoid the repetition of solving the differential Lyapunov equations which is the most expensive step in the procedure and therefore, as shown in a numerical example, accelerate the online reduction process.**

I. Introduction

Let us consider a linear time-varying (LTV) control system of the form

$$\begin{aligned} E(t)\dot{x}(t) &= A(t)x(t) + B(t)u(t),\ x(0) = 0,\\ y(t) &= C(t)x(t), \end{aligned} \tag{1}$$

where $E(t), A(t) \in \mathbb{R}^{n\times n}, B(t) \in \mathbb{R}^{n\times m}$ and $C(t) \in \mathbb{R}^{p\times n}$ depend continuously on time $t \in [0, T]$, $x(t) \in \mathbb{R}^n$ is the state vector, $u(t) \in \mathbb{R}^m$ is the input and $y(t) \in \mathbb{R}^p$ is the output. System (1) arises as a spatially discretized model for simulation of, e.g., heat distribution with moving sources, mechanical structure with moving loads, or as the linearization of nonlinear systems, see, e.g., [1], [2]. Due to the demand for precision, the size of the state vector, also called the order of the model, is very large and therefore needs to be reduced in order to enable the simulation in reasonable time. In model order reduction (MOR), it is required to approximate (1) by a reduced order model (ROM)

$$\begin{aligned} \hat{E}(t)\dot{\hat{x}}(t) &= \hat{A}(t)\hat{x}(t) + \hat{B}(t)u(t),\ \hat{x}(0) = 0,\\ \hat{y}(t) &= \hat{C}(t)\hat{x}(t), \end{aligned} \tag{2}$$

in which $\hat{E}(t), \hat{A}(t) \in \mathbb{R}^{r\times r}, \hat{B}(t) \in \mathbb{R}^{r\times m}$ and $\hat{C}(t) \in \mathbb{R}^{p\times r}$, $\hat{x}(t) \in \mathbb{R}^r$, and $\hat{y}(t) \in \mathbb{R}^p$ with $r \ll n$. This model is chosen such that the output error of the approximation $\|\hat{y}-y\|$ is small for all inputs belonging to a given admissible set.

Approaches to MOR of LTV systems can be roughly divided into two trends. The first one is based on the Krylov subspace method and the assumption that the considered system is periodic, see, e.g., [2]–[4] and references therein. The second one, which is the concern of the present paper, is based on the balanced truncation method that was mainly proposed and developed in [5]–[8]. As many other MOR methods, it projects the full-order model (FOM) on low-dimensional subspaces of the state space. To track the dynamics of the model, these subspaces should also depend on time. Let us denote by $W(t),\ Z(t) \in \mathbb{R}^{n\times r}$ the full-rank matrices that span the left and the right projection subspaces, respectively. This means that the full order state $x(t)$ is approximated by $Z(t)\hat{x}(t)$ and the corresponding residual of the state equation in (1) is constrained to be orthogonal to the subspace spanned by $W(t)$:

$$\begin{aligned} W(t)^\top E(t)\frac{d(Z(t)\hat{x}(t))}{dt} &= W(t)^\top A(t)Z(t)\hat{x}(t)\\ &\quad + W(t)^\top B(t)u(t), \end{aligned}$$

which is equivalent to

$$\begin{aligned} W(t)^\top E(t)Z(t)\dot{\hat{x}}(t) &= W(t)^\top (A(t)Z(t) - E(t)\dot{Z}(t))\hat{x}(t)\\ &\quad + W(t)^\top B(t)u(t). \end{aligned} \tag{3}$$

It follows that the coefficient matrices in ROM (2) are constructed as

$$\begin{aligned} &\hat{E}(t) = W(t)^\top E(t)Z(t),\ \hat{B}(t) = W(t)^\top B(t), \\ &\hat{A}(t) = W(t)^\top (A(t)Z(t) - E(t)\dot{Z}(t)),\ \hat{C}(t) = C(t)Z(t). \end{aligned} \tag{4}$$

The construction of the projection matrices $W(t)$ and $Z(t)$ using balanced truncation relies on the time-dependent reachability and observability gramians determined as the solutions of the associated differential Lyapunov matrix equations (DLEs) and the square root method which is rather computationally expensive, see, e.g., [8] and references therein. This means that $W(t)$ and $Z(t)$ must be obtained on a discrete time grid, say, $\mathcal{T}_{grid} = \{0 = t_0 < t_1 < \cdots < t_l = T\}$. They are also used to compute the reduced matrices $\hat{A}(t), \hat{E}(t), \hat{B}(t), \hat{C}(t)$ for later simulation of the ROM, which is usually required. However, if for some reason, simulation at another time grid with a different input is required, all the available data at $\mathcal{T}_{grid}$ can not be used

*This work was supported by (i) the Fonds de la Recherche Scientifique – FNRS and the Fonds Wetenschappelijk Onderzoek – Vlaanderen under EOS Project no 30468160 and (ii) "Communauté française de Belgique - Actions de Recherche Concertées" (contract ARC 14/19-060)

[1]ICTEAM Institute, Université catholique de Louvain, 1348 Louvain-la-Neuve, Belgium {thanh.son.nguyen, pierre-yves.gousenbourger, estelle.massart, pa.absil}@uclouvain.be

[2]Department of Mathematics and Informatics, Thai Nguyen University of Sciences, 25000 Thai Nguyen, Vietnam

anymore. This happens also in the case, termed as *online* or *many-query* context, in which the reduced state and/or the reduced output have to be computed at many other different time instants from those contained in the given grid. So, the question is: how can we exploit the computed data so that we can avoid solving the DLEs again? An apparently feasible solution is to store the reduced matrices $\hat{A}(t), \hat{E}(t), \hat{B}(t), \hat{C}(t)$ for all instants in $\mathcal{T}_{grid}$ and interpolate them to get the ROM at any other time t. In this case, treating t as an ordinary parameter, direct interpolation of reduced system matrices loses the physical meaning of the states of the derived systems as pointed out in [9]. It is therefore advisable to adjust the reduced matrices by an adequate state transformation before interpolating. However, formulating a state transformation based on either the singular value decomposition (SVD) of the collections of $\{W(t)\}_{t\in\mathcal{T}_{grid}}$ (and $\{Z(t)\}_{t\in\mathcal{T}_{grid}}$) as in [9] or minimization of the distances with respect to the Frobenius norm between a reference projection matrix among $\{W(t)\}_{t\in\mathcal{T}_{grid}}$ (and $\{Z(t)\}_{t\in\mathcal{T}_{grid}}$ for the right subspaces) and the others in the corresponding set as in [10] depend on the clustering of those bases. They only work effectively if the time-dependent bases approximately span the same subspace for the method in [9], or, in the case of the method in [10], if the time-dependent bases are closely transferable to a reference basis for which the selection strategy is still unclear. Note also that the approach of storing the precomputed data on a dense grid is in general unfeasible as pointed out in Section IV.

For these reasons, we propose making use of the computed projection matrices $W(t)$ and $Z(t)$ by first storing them on a simplified sparse time grid, say, $\mathcal{T}_{train} = \{\tau_0 = 0 < \tau_1 < \cdots < \tau_k = T\}$ where k is much smaller than l. This sparse grid can be chosen as a subset of $\mathcal{T}_{grid}$ if $\{W(t), Z(t)\}_{t\in\mathcal{T}_{grid}}$ have already been computed. Otherwise, we can compute $\{W(t), Z(t)\}_{t\in\mathcal{T}_{train}}$ in advance as a preparation step, so-called the *offline* step. In the *online* step, given any time instant t, we first recover the projection matrices $W(t)$ and $Z(t)$ by interpolation, and then compute the reduced matrices as in (4). By manipulating projection matrices, we conceptually work with the underlying Grassmann manifold $\mathrm{Grass}(r, n)$: the set of all r-dimensional subspaces of $\mathbb{R}^n$ [11]. Moreover, in the case of LTV systems and especially the continuous differentiability (C^1) requirement (3), $\mathrm{span}(W(t))$ and $\mathrm{span}(Z(t))$ can be represented as C^1 curves on the Grassmann manifold. These facts motivate us to make use of the curve interpolation technique on general Riemannian manifolds recently proposed in [12]. Admitting some additional error, our proposed method avoids the most expensive step of solving DLEs. It helps to save time compared to the standard MOR process and is therefore advantageous to online balanced truncation.

We organize the rest of the paper as follows. In Section II, we first briefly review the notion of Grassmann manifold with necessary formulas and then explain the curve interpolation technique on general Riemannian manifolds. In Section III, after summarizing the steps for balanced truncation of LTV systems, we present the application of the presented technique to online balanced truncation, i.e., constructing the ROM at any new time instant. We consider in Section IV the order reduction of a model for the heat distribution of a beam with moving heat source.

II. Interpolation on the Grassmann manifold

This section introduces the necessary tools for interpolation on the Grassmann manifold. We first recall basic concepts on general Riemannian manifolds and then focus on the Grassmann manifold. The theory is taken from [13] and [14]. Finally, we present the C^1-interpolation methods that we apply in this paper. These methods were recently presented in [12].

A. Riemannian manifolds

Manifolds are nonlinear spaces that can be locally approximated by a Euclidean space (named tangent space). A Riemannian manifold $\mathcal{M}$ is a manifold on which the tangent spaces are endowed with a smoothly varying inner product. We denote by $T_x\mathcal{M}$ the tangent space to $\mathcal{M}$ at x, and $\langle .,. \rangle_x$ the inner product defined on $T_x\mathcal{M}$. This inner product induces a norm $\|v\|_x$ (and so a notion of length) for any vector $v \in T_x\mathcal{M}$. The shortest path between two points x and $y \in \mathcal{M}$ is called the *geodesic* $t \mapsto g(t; x, y)$, with $g(0; x, y) = x$ and $g(1; x, y) = y$. The initial velocity $\dot{g}(t; x, y)|_{t=0} = \xi_x \in T_x\mathcal{M}$ of the geodesic can be computed via the *logarithmic map* $\log_x(\cdot) : \mathcal{M} \to T_x\mathcal{M} : y \mapsto \log_x(y) = \xi_x$. The logarithm actually maps a point $y \in \mathcal{M}$ to the tangent space $T_x\mathcal{M}$. The reverse operation is called the *exponential map* $\exp_x(\cdot) : T_x\mathcal{M} \to \mathcal{M} : \xi_x \mapsto \exp_x(\xi_x) = y$. It maps a tangent vector ξ_x from $T_x\mathcal{M}$ to $\mathcal{M}$.

The exponential and logarithm maps are two useful tools for interpolation and optimization on manifolds. Note that the geodesic $g(\cdot; x, y)$ can be computed as $g(t; x, y) = \exp_x(t\log_x(y))$. We furthermore introduce the notation $\mathrm{av}_w(x, y) = g(w; x, y)$ for the *weighted geodesic averaging*, with respect to a weight $w \in [0, 1]$.

B. The Grassmann manifold $\mathrm{Grass}(r, n)$

Let $r, n \in \mathbb{N}$ with $r < n$. The Grassmann manifold $\mathrm{Grass}(r, n)$ is the set of all r-dimensional subspaces of $\mathbb{R}^n$. Each subspace can be represented by an orthogonal basis, i.e., a rectangular matrix $X \in \mathbb{R}^{n\times r}$, such that $X^\top X = I_r$. The set of all orthogonal bases in $\mathbb{R}^n$ of r elements is called the Stiefel manifold $\mathrm{St}(r, n) = \{X \in \mathbb{R}^{n\times r} | X^\top X = I_r\}$. Observe now that the choice of the basis is not unique: any pair of matrices $X, Y \in \mathrm{St}(r, n)$ such that $X = YQ$, where $Q \in \mathcal{O}_r$, the set of $r \times r$ orthogonal matrices, span the same subspace. We say that those matrices are equivalent, and that $\mathrm{Grass}(r, n)$ is a quotient manifold of the Stiefel manifold: $\mathrm{Grass}(r, n) \simeq \mathrm{St}(r, n)/\mathcal{O}_r$. Other matrix representations were suggested for $\mathrm{Grass}(r, n)$ (see, e.g., [15]–[17]).

Assuming that $\mathrm{St}(r, n)$ is endowed with the metric inherited as a Riemannian submanifold of $\mathbb{R}^{n\times r}$, the quotient structure induces some expressions for the Riemannian logarithm and exponential. Let $X, Y \in \mathrm{Grass}(r, n)$, and let

$H \in T_X\mathrm{Grass}(r,n)$, with $H = U\Sigma V^\top$ a compact SVD. The Riemannian exponential is computed in [15]:

$$\exp_X(H) = (XV\cos(\Sigma) + U\sin(\Sigma))\,V^\top,$$

while the Riemannian logarithm is obtained in [18], [19]:

$$\log_X(Y) = V_2\Theta V_1^\top,$$

where V_1, V_2 and Θ come from the Cosine Sine (CS) decomposition:

$$\begin{bmatrix} X^\top Y \\ (I_n - XX^\top)Y \end{bmatrix} = \begin{bmatrix} V_1\cos(\Theta)S^\top \\ V_2\sin(\Theta)S^\top \end{bmatrix}.$$

C. Interpolation on manifolds

As mentioned in Section I, the continuous differentiability of the curve is needed. A piecewise-geodesic interpolation is not acceptable because the resulting curve γ will not be differentiable at the interpolation points. We summarize here two methods to compute a C^1 curve $\mathbf{B}(t)$ on a Riemannian manifold $\mathcal{M}$. The curve interpolates a set of data points $d_0,\dots,d_k \in \mathcal{M}$ associated with parameters $t_0 < \cdots < t_k$, such that $\mathbf{B}(t_i) = d_i$, $i = 0,\dots,k$. These two methods only rely on the exponential and logarithm maps: they can be applied to the Grassmann manifold using the expressions given in Section II-B.

The first method computes a curve $\mathbf{B}_{\mathrm{TS}}(t)$ with a three-step procedure: (i) map all the data points to the tangent space at a point $x \in \mathcal{M}$ via the logarithm, i.e., obtain $\tilde{d}_i = \log_x(d_i)$, $i = 0,\dots,k$; (ii) on this Euclidean space, compute the cubic spline $\tilde{\gamma}(t)$ interpolating the tangent vectors; (iii) map the curve back to the manifold thanks to the exponential map, such that $\mathbf{B}_{\mathrm{TS}}(t) = \exp_x(\tilde{\gamma}(t))$. We will refer to this method as the TS method. As shown in [12], this method has the advantage to be extremely fast, but leads to poor results when the curvature of the manifold is high.

The second method, introduced in [12], compensates the problems linked to the curvature of the manifold by blending together solutions obtained on different tangent spaces. The *blended cubic spline* $\mathbf{B}_{\mathrm{blend}}(t)$ is a piecewise curve defined as

$$\mathbf{B}_{\mathrm{blend}} : [0,k] \to \mathcal{M} : t \mapsto \mathbf{B}_{\mathrm{blend}}(t) = \mathbf{B}_i(t-i), \quad i = \lfloor t \rfloor,$$

where the pieces $\mathbf{B}_i(s) : [0,1] \to \mathcal{M}$, $s = t-i$, are weighted geodesic averages of the restriction on the interval $[i, i+1]$ of two curves $\mathbf{B}^L_{\mathrm{TS}} : [0,k] \to T_{d_i}\mathcal{M} : t \mapsto \mathbf{B}^L_{\mathrm{TS}}(t)$ and $\mathbf{B}^R_{\mathrm{TS}} : [0,k] \to T_{d_{i+1}}\mathcal{M} : t \mapsto \mathbf{B}^R_{\mathrm{TS}}(t)$ computed respectively on the tangent space at d_i and d_{i+1}. Note that these curves are obtained using the first method defined above. Choosing the weight $w(s) = 3s^2 - 2s^3$, the i^{th} piece of the blended cubic spline is thus given by

$$\mathbf{B}_i(s) = \mathrm{av}_{w(s)}\left(\mathbf{B}^R_{\mathrm{TS}}(s+i), \mathbf{B}^L_{\mathrm{TS}}(s+i)\right).$$

This method is illustrated on Fig. 1 and will be referred to as the blend method.

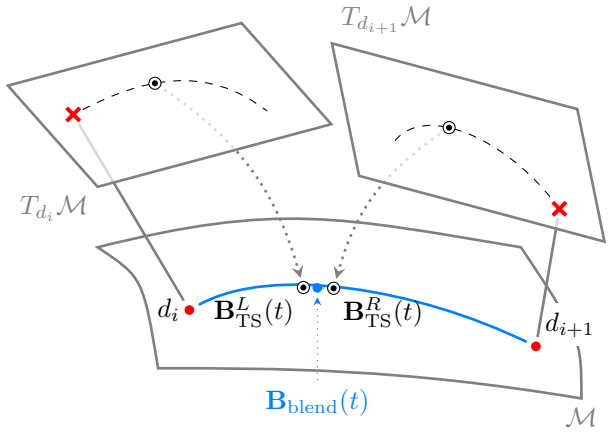

Fig. 1. The blended cubic spline $\mathbf{B}_{\mathrm{blend}}(t)$ at $t \in [i, i+1]$ is made of two curves $\mathbf{B}^L_{\mathrm{TS}}(t)$ and $\mathbf{B}^R_{\mathrm{TS}}(t)$ computed on the tangent spaces to d_i and d_{i+1}, respectively, and mapped back to $\mathcal{M}$. They are then blended together with a geodesic averaging of weight $w(s) = 3s^2 - 2s^3$, $s = t - i$.

III. Interpolation for online balanced truncation of LTV systems

A. Balanced truncation of LTV systems

We first briefly review the method presented in [8]. The construction of the projection matrices $W(t)$ and $Z(t)$ requires to solve two (DLEs):

$$\begin{aligned} E(t)\dot{P}(t)E(t)^\top &= A(t)P(t)E(t)^\top + E(t)P(t)A(t)^\top \\ &\quad + B(t)B(t)^\top,\ P(0) = 0, \end{aligned} \tag{5}$$

$$\begin{aligned} -E(t)^\top\dot{Q}(t)E(t) &= A(t)^\top Q(t)E(t) + E(t)^\top Q(t)A(t) \\ &\quad + C(t)^\top C(t),\ Q(T) = 0. \end{aligned} \tag{6}$$

for the reachability and observability gramians $P(t)$ and $Q(t)$. As DLEs are a special case of differential Riccati equations, available methods are [20]–[23], just to name a few. Once the gramians are computed, the projection matrices are constructed using the square root method [24]. That is, at each time instant t where the two gramians are precomputed, one has to compute the SVD of the product $R(t)^\top E(t)^\top L(t)$ in which $R(t)$ and $L(t)$ are respectively low rank factors approximations of $P(t)$ and $Q(t)$, and then construct the ROM based on removing the (low valuable) states corresponding to smallest singular values. We summarize the above steps in Alg. 1. By checking the computational costs, one can see that solving the DLEs is the most expensive step in this approach.

B. Interpolation of projection matrices for online BT

Suppose that the projection matrices $W(\tau_0),\dots,W(\tau_k)$, $Z(\tau_0),\dots,Z(\tau_k)$, have been computed in the offline step, with $\tau_i \in \mathcal{T}_{grid}$, $i = 0,\dots,k$. Given any time instant $t \in [0,T]$, we construct $W(t)$ and $Z(t)$ using one of the two interpolation methods presented in Section II-C. This action can be considered as a replacement for the first three steps in Alg. 1. After that, we just have to perform the last step to formulate the ROM. Our proposed approach for online balanced truncation is briefly described in Alg. 2. Note that if the goal is to simulate the ROM, which is often the case, the ROM must be constructed also at earlier instants than t that are needed for numerical integration.

Algorithm 1 Balanced truncation for LTV systems

Require: FOM $E(t), A(t), B(t), C(t)$
Ensure: ROM $\hat{E}(t), \hat{A}(t), \hat{B}(t), \hat{C}(t)$

1: Solve DLEs (5) and (6) for solutions in low rank form $P(t) = R(t)R(t)^\top$ and $Q(t) = L(t)L(t)^\top$.
2: Compute the singular value decomposition (SVD)

$$R(t)^\top E(t)^\top L(t) = [U_1(t)\ U_2(t)] \begin{bmatrix} \Sigma_1(t) & \\ & \Sigma_2(t) \end{bmatrix} \times [V_1(t)\ V_2(t)]^\top$$

with $\Sigma_1(t) = \text{diag}(\sigma_1(t), \ldots, \sigma_r(t))$ and $\Sigma_2(t) = \text{diag}(\sigma_{r+1}(t), \ldots, \sigma_n(t))$.
3: Compute the projection matrices as $W(t) = L(t)V_1(t)\Sigma_1^{-1/2}(t)$ and $Z(t) = R(t)U_1(t)\Sigma_1^{-1/2}(t)$.
4: Compute the reduced matrices as in (4).

Algorithm 2 Online balanced truncation for LTV systems

Require: FOM $E(t), A(t), B(t), C(t)$, and $\text{span}(W(t))$, $\text{span}(Z(t)) \subset \text{Grass}(r, n)$ for $t \in \mathcal{T}_{train}$
Ensure: ROM $\hat{E}(t), \hat{A}(t), \hat{B}(t), \hat{C}(t)$

1: For any $t \notin \mathcal{T}_{train}$, interpolate on $\text{Grass}(r, n)$ to get $W(t), Z(t)$
2: Compute the reduced matrices as in (4).

The derivative $\dot{Z}(t)$ in (3) is conceptually a vector on the tangent space $T_{Z(t)}\text{Grass}(r, n)$. However, computing it relates to Jacobi field, i.e., the derivative of a geodesic with respect to its end points [25], which is quite involved and therefore is not performed here. Instead, we approximate it as $d(Z(t)\hat{x}(t))/dt \approx (Z(t)\hat{x}(t) - Z(t_-)\hat{x}(t_-))/(t - t_-)$. Another way is the manifold version of finite differences, which means $\dot{Z}(t) \approx -\log_{Z(t)}(Z(t_-))/(t - t_-)$ where t_- is the left neighbor of t at which the data are available. In the numerical example later, we use the earlier.

IV. Numerical example

As a preliminary result, we consider one of the models from [8]. It is a beam of length l with a heat source that moves along. The heat distribution is modeled by a 1-D equation

$$\begin{aligned} &c\rho\frac{\partial\theta(t,z)}{\partial t} = \kappa\frac{\partial^2\theta(t,z)}{\partial z^2} + \delta(z - \xi(t))u(t), \\ &(t,z) \in (0,T] \times (0,l), \\ &\theta(t,0) = 0,\ \theta(t,l) = 0,\ t \in (0,T), \\ &\theta(0,z) = 0, z \in (0,l). \end{aligned} \tag{7}$$

In (7), $\theta(t,z)$ is the temperature, c, ρ and κ stand for the specific heat capacity, the mass density and the heat conductivity, respectively. The movement of the heat source is modeled by the Dirac delta function $\delta(z - \xi(t))$, where $\xi(t) = (lt)/T$ is the time-dependent location.

For computation, we spatially discretize (7) using the finite element method to derive a system of the form (1) with $E = E^\top$, $A = A^\top$ of order $n = 1000$. By considering the temperature at exactly the location of the heat source as the information of interest, we have that $C(t)^\top = B(t)$. The final time is set to $T = 50$ and the grid of times $\mathcal{T}_{grid}$ is made of 101 equispaced points in the interval $[0, 50]$. For simulation, the heat flux density is chosen to be constant: $u(t) = 50$ for all t. All computations are performed with MATLAB R2018a on a standard desktop using 64-bit OS Windows 10, equipped with 3.20 GHz 16 GB Intel Core i7-8700U CPU.

In Alg. 1, solving the DLEs using the backward differentiation formula (BDF) of order 2 takes 70.25 seconds. In online balanced truncation, instead of having to solve the DLEs and perform step 2 and step 3 in Alg. 1, we compute a curve on Grass(40; 1000) interpolating 18 training points (step 1 of Alg. 2) to recover the projection subspaces at the same time grid mentioned above. This step costs 2.17 seconds by the TS method and 5.01 seconds by the blend method. The other steps are performed as in the standard procedure presented in Alg. 1.

Fig. 2 and Fig. 3 compare the outputs of the FOM, of the standard ROM (computed using the 4 steps in Alg. 1) and of the ROM formulated by our procedure using the TS and the blend interpolation methods. We can see that the TS method is only accurate in a neighborhood of $t = 25$. This behavior is understandable as the tangent space chosen is based at that point. The manifold is well approximated by the tangent space in the neighborhood of the basis point of the tangent space, but not far away from it. This is due to the curvature of the manifold and was observed, for instance, in [12]. Conversely, the deviations caused by the blend method can be neglected. Fig. 4 compares the error on the states $\|x(t) - Z(t)\hat{x}(t)\|$ and on the outputs $\|y(t) - \hat{y}(t)\|$ of the standard ROM and of the ROM constructed by the proposed blend method.

Before ending this section, we would like to discuss the possibility of storing precomputed data $W(t), Z(t)$ on an extremely fine grid of time and then, in the online stage, to extract and directly use the information from it without interpolation. This approach comes at a price. Firstly, solving the DLEs on an extremely fine grid is very time consuming. Secondly, the feasibility of this suggestion must be considered in context. For the size of the presented example, it is possible. For larger systems, it is hard to achieve. This is because $W(t)$ and $Z(t)$ are dense matrices with entries in the *double* format. Therefore, they require $2 \times 8 \times n \times r$ bytes for storing. If there are L time steps, the memory needed is $16 \times L \times n \times r$ bytes. For example, if $n = 10000, r = 100, L = 2000$, the memory required is almost 30 GB which is far beyond the RAM of most personal computers. Storing data in hard disk will result in slowing down the online stage.

V. Conclusions

In this paper, we have proposed to use the blended curve interpolation technique on the Grassmann manifold to interpolate precomputed time-dependent projection subspaces for

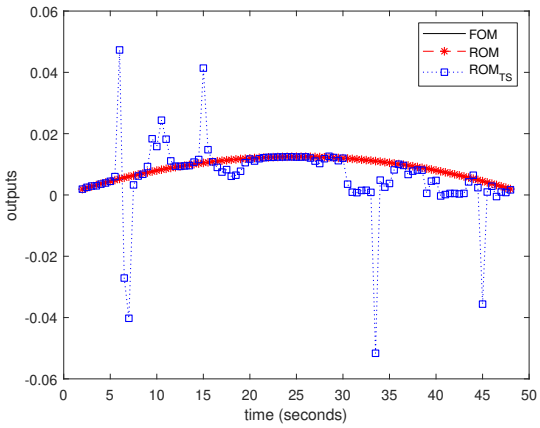

Fig. 2. Outputs of the FOM (FOM), standard ROM (ROM) and the ROM formulated by our procedure using the first interpolation method ($\mathrm{ROM}_{\mathrm{TS}}$)

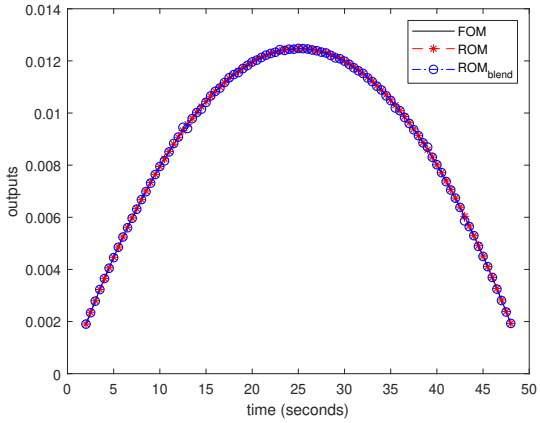

Fig. 3. Outputs of the FOM (FOM), standard ROM (ROM) and the ROM formulated by our procedure using the second interpolation method ($\mathrm{ROM}_{\mathrm{blend}}$)

model reduction of linear time-varying control systems. Provided that the offline data are available, our method allows not only time and memory savings but also the flexibility in choosing time instants for simulation of the reduced system in the online stage. Improving the method technically, e.g., using the Jacobi fields mentioned in Sec. III, and testing the method with more practical and/or larger examples are our future consideration.

ACKNOWLEDGMENT

We would like to thank the authors of [8] for sharing the MATLAB codes. Implementations of the Riemannian logarithm and exponential maps on the Grassmann manifold come from the Manopt toolbox [26].

REFERENCES

[1] S. Hein, *MPC-LQG-based Optimal Control of Nonlinear Parabolic PDEs*, Ph.D. thesis, TU Chemnitz, 2009.

[2] J.R. Philips, "Projection-based approaches for model reduction of weakly nonlinear, time-varying systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 2, pp. 171–187, 2003.

[3] J. Roychowdhury, "Reduced-order modeling of time-varying systems," *IEEE Trans. Circuits Syst. II*, vol. 46, no. 10, pp. 1273–1288, 1999.

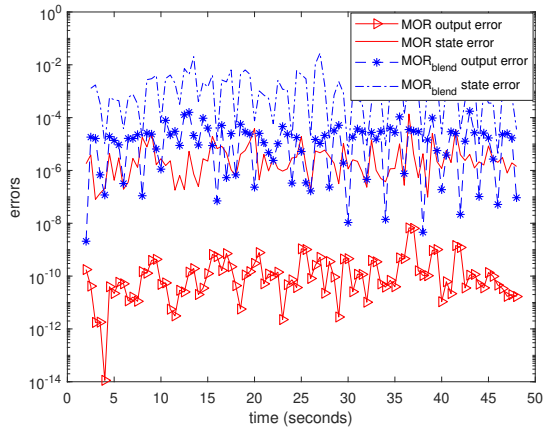

Fig. 4. Errors of the states and outputs of the standard ROM and the ROM constructed by our procedure using the second interpolation method ($\mathrm{ROM}_{\mathrm{blend}}$)

[4] M.-S. Hossain, P. Benner, "Projection-based model reduction for time-varying descriptor systems using recycled Krylov subspaces", in *Proceedings in Applied Mathematics and Mechanics*, vol. 8, no. 1, pp. 10081–10084, 2008.

[5] S. Shokoohi, L. Silverman, and P. Van Dooren, "Linear time-variable systems: balancing and model reduction," *IEEE Trans. Autom. Control*, vol. 28, no. 8, pp. 810–822, 1983.

[6] E.I. Verriest and T. Kailath, "On generalized balanced realizations," *IEEE Trans. Autom. Control*, vol. 28, no. 8, pp. 833–844, 1983.

[7] H. Sandberg and A. Rantzer, "Balanced truncation of linear time-varying systems," *IEEE Trans. Autom. Control*, vol. 49, no. 2, pp. 217–229, 2004.

[8] N. Lang, J. Saak, and T. Stykel, "Balanced truncation model reduction for linear time-varying systems," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 22, no. 4, pp. 267–281, 2016.

[9] H. Panzer, J. Mohring, R. Eid, and B. Lohmann, "Parametric model order reduction by matrix interpolation," *at–Automatisierungstechnik*, vol. 58 no. 4, pp. 475–484, 2010.

[10] D. Amsallem and C. Farhat. "An online method for interpolating linear parametric reducedorder models," *SIAM J. Sci. Comput.*, vol. 33, no. 5, pp. 2169–2198, 2011.

[11] D. Amsallem and C. Farhat. "Interpolation method for adapting reduced-order models and application to aeroelasticity," *AIAA Journal*, vol. 46, no. 7, pp. 1803–1813, 2008.

[12] P.-Y. Gousenbourger, E. Massart, and P.-A. Absil, "Data fitting on manifolds with composite Bézier-like curves and blended cubic splines," *J. Math. Imaging Vis.*, (2018). DOI:10.1007/s10851-018-0865-2

[13] B. O'Neill, *Elementary Differential Geometry*, Academic Press INC, London, 1966.

[14] P.-A. Absil, R. Mahony and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*, Princeton University Press, Princeton, NJ, 2008.

[15] A. Edelman, T.A. Arias and S.T. Smith, "The geometry of algorithms with orthogonality constraints," *SIAM J. Matrix Anal. Appl.*, vol. 20, no. 2, pp. 303–353, 1998.

[16] P.-A. Absil, R. Mahony, and R. Sepulchre, "Riemannian geometry of Grassmann manifolds with a view on algorithmic computation," *Acta Appl. Math.*, vol. 80, no. 2, pp. 199–220, 2004.

[17] P. Turaga, A. Veeraraghavan, A. Srivastava, and R. Chellappa, "Statistical computations on Grassmann and Stiefel manifolds for image and video-based recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 11, 2273–2286, 2011.

[18] Q. Rentmeesters, "A gradient method for geodesic data fitting on some symmetric Riemannian manifolds," in *2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, 12-15 Dec. 2011, Orlando, FL, USA, pp. 7141–7146, 2012.

[19] K.A. Gallivan, A. Srivastava, X. Liu and P. Van Dooren, "Efficient algorithms for inferences on Grassmann manifolds," in *IEEE Workshop on Statistical Signal Processing*, 28 Sept.-1 Oct. 2003, St. Louis, MO, USA, pp. 315–318, 2004.

[20] C. Choi and A.J. Laub, "Efficient matrix-valued algorithms for solving stiff Riccati differential equations," *IEEE Trans. Autom. Control*, vol. 35, no. 7, pp. 770–776, 1990.

[21] P. Benner and H. Mena, "Rosenbrock methods for solving Riccati differential equations," *IEEE Trans. Autom. Control*, vol. 58, no. 11, pp. 2950–2957, 2013.

[22] H. Mena, A. Ostermann, L.-M. Pfurtscheller, and C. Piazzola, "Numerical low-rank approximation of matrix differential equations," *Journal of Computational and Applied Mathematics*, vol. 340, no. 1 , pp. 602–614, 2018.

[23] T. Stillfjord, "Adaptive high-order splitting schemes for large-scale differential Riccati equations," *Numerical Algorithms*, vol. 78, no. 4, pp. 1129–1151, 2018.

[24] M.S. Tombs and I. Postlethwaite, "Truncated balanced realization of a stable nonminimal state-space system," *Int. J. Control.*, vol. 46, no. 4, pp. 1319–1330, 1987

[25] J.M. Lee, *Riemannian Manifolds: An Introduction to Curvature*, Graduate Texts in Mathematics, Springer Verlag, New-York, 1997.

[26] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre, "Manopt, a Matlab Toolbox for Optimization on Manifolds", *Journal of Machine Learning Research*, vol. 15, pp. 1455-1459, 2014. `http://www.manopt.org`.