Bericht des Instituts für Aerodynamik und Strömungstechnik

Report of the Institute of Aerodynamics and Flow Technology

IB 124-2011/3

A Comprehensive Comparison of Various Algorithm for Efficiently Updating Singular Value Decomposition Based Reduced Order Models

R. Zimmermann

Herausgeber:
Deutsches Zentrum für Luft- und Raumfahrt e.V.
Institut für Aerodynamik und Strömungstechnik
Lilienthalplatz 7, Germany, 38108 Braunschweig

ISSN 1614-7790

Stufe der Zugänglichkeit: 1
Braunschweig, im Juni 2011

Institutsdirektor:	Verfasser:
Prof. Dr.-Ing. habil. C.-C. Rossow	Dr. rer. nat. R. Zimmermann

Abteilung: $C^2A^2S^2E$

Abteilungsleiter:

Prof. Dr.-Ing. N. Kroll

Der Bericht enthält:

16	Seiten
0	Bilder
3	Tabellen
13	Literaturstellen

Abstract

Efficiently updating an SVD-based data representation while keeping accurate track of the data mean when new observations are coming in is a common objective in many practical application scenarios. In this paper, two different SVD update algorithms capable of treating an arbitrary number of new observations are introduced following the symmetric EVD philosophy. These methods are compared to an SVD update method known from the literature. The comparison criterion of interest is the theoretical computational complexity, it being understood that the dimension of the observation vectors is much larger than the number of observations. From this point of view, a hierarchy of methods is derived, and the computational savings of the update strategies pursuing the symmetric EVD approach are demonstrated. It is exposed, how the compression level of the initial SVD model affects the performance of these algorithms and the break point where one method becomes more efficent than the other is determined. In addition, simple rules of thumb are derived for easing the choice of an algorithm valid in most practical scenarios.

1 Introduction

Singular value decomposition (SVD) based reduced order models (ROMs) apply to a large variety of scientific problems, ranging from data processing, pattern recognition and image analysis to solving partial differential equations [1, 2, 5, 6, 9, 10, 11, 13]. The method goes by different names in the various communities and is also referred to as Proper Orthogonal Decomposition (POD), Principal Component Analysis (PCA) and Karhunen-Loève Decomposition.
In many application scenarios, an existing SVD-based ROM, which has been built based on a number of observation, needs to be updated by incorporating the information provided by new snapshots of observations. In order to keep memory requirements at a feasible low level, most often the initial observations cannot be stored but only their compressed SVD-representation, so that computing an SVD of the augmented set of snapshot observations from scratch becomes not only inefficient but impossible. This is the case for adaptively improving POD-based ROMs as applied for example in aeronautical design and optimization, see [2] and references therein. While this is the application that we have in mind, the methods presented in this paper are presented in the general context of basic linear algebra.
Since in many applications the fluctuations around the mean are of interest, the snapshot data is centered by subtracting the mean vector over all snapshots before it is decomposed and possibly compressed. Hence, the update methods presented here keep accurate track of the mean of the observations.
We adopt the following terminology from the literature, see e.g. [9]. In an *incremental computation*, an existing SVD-based ROM is updated using new snapshot observations; in a *batch computation* all observations are used simultaneously to compute the SVD-based ROM.
The main contributions of this report are the following: (1) two different incremental SVD update algorithms capable of treating an arbitrary number of new observations are introduced following the symmetric eigenvalue decomposition philosophy. In addition, it is exposed how the method proposed in [3] applies to the scenario considered in this report; (2) a comparison of the three major incremental SVD update strategies is conducted w.r.t. computational complexity: Assuming that the dimension n of the snapshot data vectors exceeds by far the number of original snapshots m plus the number of additional snapshots p, i.e. $n \gg m + p$, it is shown theoretically that the EVD-based update strategies are more efficient in practical application scenarios when compared to the SVD-based approach of [3]. It turns out that the hierarchy of the EVD-based methods depends on the compression level of the SVD representation. Therefore, (3) precise threshold valuess are derived for the compression level, where one method starts to become more effcient than the other and vice versa. Based on empirical re-

sults, it is exposed that all methods share a comparable level of accuracy.
An additional advantages of the EVD-based incremental methods proposed in this paper is that they can be parallelized in a straight-forward way, requiring only parallelized matrix-vector and matrix-matrix products.
Related work: Efficiently updating the SVD has been investigated by several authors. All methods have in common that updating the SVD of an $n \times m$ matrix by adding p new columns is essentially reduced to solving an SVD (or EVD) problem of size $m+p$, here termed *the small update problem*. In [4], update modifications for $p = 1$ are investigated. In this special setting, the matrix of the small update problem is highly structured, such that pseudo-explicit formulae for the updated SVD data exist and [4] is concerned with efficiently solving the resulting small-size EVD update problem. A refinement of this method is derived in [8] and a recent work in line with these contributions is presented in [12]. The new methods proposed in this report can be considered a generelization of the method introduced in [4, §4] to adding an arbitrary number of update observations. From the point of view taken in this paper, solving the small update problem is performed by a black box function, since for our applications of interest, essentially only those parts of the update algorithms that scale in the snapshot dimension n contribute to the overall computational costs.
The only references dealing with an arbitrary number of update observations known to the author are [3] and [9]. While, at first glance, the scope of [9] looks slightly different, both approaches share the main ingredient of updating the SVD by orthogonalizing the incoming snapshots against the existing basis, which has to be achieved via Gram-Schmidt's or related methods. When transferred to the problems considered in this work, the approaches [3] and [9] essentially coincide. For additional references, we refer to the literature reviews given in [3, §5] and [9, §1].

2 SVD-based reduced order models

Let $Y_m = (W^1, ..., W^m) \in \mathbb{R}^{n\times m}$. As customary in the context POD, the columns $W^i, i = 1, ..., m$ of Y will be referred to as *snapshots*.
Let $\mathbf{1}_m = (1, ..., 1)^T \in \mathbb{R}^m$ be the vector of dimension m with all entries equal to 1. The snapshot mean vector is defined by $A_m := \frac{1}{m}\sum_{j=1}^m W^j$ and the *centered snapshot matrix* is given by

$$\bar{Y}_m = Y_m - A_m\mathbf{1}_m^T = \left(W^1 - A_m, ..., W^m - A_m\right) =: \left(\bar{W}^1, ..., \bar{W}^m\right).$$

Note that rank$(\bar{Y}) \leq m-1$, since $\bar{Y}\mathbf{1}_m = 0$.
Let $\mathbf{U}\Sigma\mathbf{V}^T = \bar{Y}_m$ be the thin SVD of the centered snapshot matrix $\bar{Y}_m$. The left hand side singular vectors $U^i \in \mathbb{R}^n, i = 1, ..., m$ are called *POD eigenmodes* or *POD modes* in short. The *relative information content (RIC)* of the first $r \leq m$ POD modes is defined as

$$\text{ric}(r) = \text{ric}(r, \Sigma^2) = \frac{\sum_{i=1}^r \sigma_i^2}{\sum_{i=1}^m \sigma_i^2}, \tag{2.1}$$

where diag$(\sigma_1, ..., \sigma_m) = \Sigma \in \mathbb{R}^{m\times m}$. Suppose that $r_m \leq m-1$ has been determined such that ric$(r_m) \geq 1-\epsilon$ for a given $\epsilon \in (0,1)$. An order-r_m representation of $\bar{Y}_m$ is obtained by discarding the small singular values $(\sigma_{r_m+1}, ..., \sigma_m)$ and removing the corresponding columns from the singular vector matrices **U** and **V**, arriving at

$$\bar{Y}_m \approx \mathbf{U}_m\Sigma_m\mathbf{V}_m^T,$$

where $\mathbf{U}_m = (U^1, ..., U^{r_m}) \in \mathbb{R}^{n\times r_m}$, $\mathbf{V}_m = (V^1, ..., V^{r_m}) \in \mathbb{R}^{m\times r_m}$,
and $\Sigma_m = \text{diag}(\sigma_1, ..., \sigma_{r_m}) \in \mathbb{R}^{r_m\times r_m}$.

Definition 2.1 (Reduced Order Model, Compression Rate). *The data set*

$$\left(\mathbf{U}_m, \Sigma_m, \mathbf{V}_m, A_m, n, m, r_m\right), \tag{2.2}$$

is called a reduced order model (ROM) *of order r_m of Y_m. The ratio $x = \frac{r_m}{m}$ is called* the compression rate.

The *SVD basis update problem* can now be formulated as follows:

Problem 2.1 (SVD basis update). *Let $Y_m = (W^1, ..., W^m) \in \mathbb{R}^{n\times m}$ and let $(\mathbf{U}_m, \Sigma_m, \mathbf{V}_m, A_m, n, m, r_m)$ be an order-r_m ROM of Y_m. Suppose that p new snapshot observations (column vectors) $(W^{m+1}, ..., W^{m+p})$ have been provided in order to enhance the ROM and let*

$$Y_{m+p} = (W^1, ..., W^m, W^{m+1}, ..., W^{m+p}) \in \mathbb{R}^{n\times(m+p)}.$$

Objective: *Compute a ROM*

$$(\mathbf{U}_{m+p}, \Sigma_{m+p}, \mathbf{V}_{m+p}, A_{m+p}, n, m+p, r_{m+p})$$

of Y_{m+p}, in particular, decompose

$$\bar{Y}_{m+p} = Y_{m+p} - A_{m+p}\mathbf{1}_{m+p}^T \approx \mathbf{U}_{m+p}\Sigma_{m+p}\mathbf{V}_{m+p}^T \tag{2.3}$$

by only using the previous-stage ROM $(\mathbf{U}_m, \Sigma_m, \mathbf{V}_m, A_m, n, m, r_m)$ and the new snapshots $(W^{m+1}, ..., W^{m+p})$ but not the previous-stage snapshot matrix Y_m.

Remark 2.2. *Alternatively, given a matrix Y, its thin singular value decomposition $Y = U\Sigma V^T$ can be obtained by solving the symmetric eigenvalue problem $Y^TY = V\Lambda V^T$ and setting $\Sigma := \sqrt{\Lambda}$ and $U = YV diag(\frac{1}{\sigma_1}, ..., \frac{1}{\sigma_r})$, where r denotes the rank of the matrix Σ.*

3 SVD basis update strategies

In this section, we introduce three different approaches for tackling Problem 2.1. In order to keep accurate track of the snapshot mean, all methods share the following data preprocessing step.

3.1 Updating the snapshot mean vector

Suppose that the previous-stage snapshot mean A_m and the new set of observations $W = (W^{m+1}, ..., W^{m+p}) \in \mathbb{R}^{n \times p}$ are given. As a first step, the new snapshots are shifted to the previous-stage center $\bar{W}^{m+i} = W^{m+i} - A_m$ for $i = 1, ..., p$ and the snapshot matrix is augmented $\left(\bar{Y}_m, \bar{W}^{m+1}, ..., \bar{W}^{m+p}\right) \approx \left(\mathbf{U}_m \Sigma_m \mathbf{V}_m^T, \bar{W}^{m+1}, ..., \bar{W}^{m+p}\right) \in \mathbb{R}^{n \times (m+p)}$.
The updated mean vector $A_{m+p} = \frac{1}{m+p} \sum_{i=1}^{m+p} W^i$ is computed as follows.
For $i = 1, ..., m+p$, it holds $W^i - A_{m+p} = W^i - A_m + A_m - A_{m+p} = \bar{W}^i + T_{m+p}$, where

$$T_{m+p} := A_m - A_{m+p} = \frac{1}{m+p} \left(pA_m - \sum_{i=m+1}^{m+p} W^i \right).$$

The updated mean value is thus $A_{m+p} = A_m - T_{m+p}$. Writing $\bar{\mathbf{W}} = \left(\bar{W}^{m+1}, ..., \bar{W}^{m+p}\right)$ the main objective (2.3) becomes

$$\text{Decompose } \bar{Y}_{m+p} = \left(\mathbf{U}_m \Sigma_m \mathbf{V}_m^T, \bar{W}\right) + T_{m+p} \mathbf{1}_{m+p}^T \approx \mathbf{U}_{m+p} \Sigma_{m+p} \mathbf{V}_{m+p}^T. \tag{3.1}$$

3.2 An EVD-based two-steps update strategy

In this section, the updating is traced back to solving an $(r+p) \times (r+p)$ EVD problem. This generalizes the rank-1 update approach introduced in [4, §4] to adding an arbitrary number of new observations. Contrary to the SVD-update technique presented in [3] (see also Section 3.3) and the merging of eigenspaces proposed in [9], an expensive computation of certain orthogonal complements via Gram-Schmidt's, [7, §5.2.7], or related methods can be avoided.
The EVD-based update algorithm relies on the following observation: Writing $\bar{\mathbf{W}} = \left(\bar{W}^{m+1}, ..., \bar{W}^{m+p}\right) \in \mathbb{R}^{n \times p}$, it holds

$$\left(\bar{Y}_m, \bar{\mathbf{W}}\right)^T \left(\bar{Y}_m, \bar{\mathbf{W}}\right) = \begin{pmatrix} \bar{Y}_m^T \bar{Y}_m & \bar{Y}_m^T \bar{\mathbf{W}} \\ \bar{\mathbf{W}}^T \bar{Y}_m & \bar{\mathbf{W}}^T \bar{\mathbf{W}} \end{pmatrix} \approx \begin{pmatrix} \mathbf{V}_m \Sigma_m^2 \mathbf{V}_m^T & \mathbf{V}_m \Sigma_m \mathbf{U}_m^T \bar{\mathbf{W}} \\ \bar{\mathbf{W}}^T \mathbf{U}_m \Sigma_m \mathbf{V}_m^T & \bar{\mathbf{W}}^T \bar{\mathbf{W}} \end{pmatrix}$$

$$= \begin{pmatrix} \mathbf{V}_m & \mathbf{0} \\ \mathbf{0} & I^{p\times p} \end{pmatrix} \begin{pmatrix} \Sigma_m^2 & \Sigma_m \mathbf{U}_m^T \bar{\mathbf{W}} \\ \bar{\mathbf{W}}^T \mathbf{U}_m \Sigma_m & \bar{\mathbf{W}}^T \bar{\mathbf{W}} \end{pmatrix} \begin{pmatrix} \mathbf{V}_m^T & \mathbf{0} \\ \mathbf{0} & I^{p\times p} \end{pmatrix}$$

Note that $\tilde{V} := \begin{pmatrix} \mathbf{V}_m & \mathbf{0} \\ \mathbf{0} & I^{p\times p} \end{pmatrix} \in \mathbb{R}^{(m+p)\times(r_m+p)}$ is not square, yet it holds $\tilde{V}^T\tilde{V} = I^{(r_m+p)\times(r_m+p)}$. Combining the above observation with Remark 2.2 and the rank-1 SVD-shift proposed in [3, §3] leads to Algorithm 3.2.

Remark 3.1 (on Algorithm 3.2). *Steps 11 to 14 are related to the shifting of the new found SVD to the updated center, for details, see [3, §3].*

3.3 An SVD-based two-steps update strategy

In this section, the update strategy proposed in [3] is being followed, which relies on a rank-p SVD update. Introducing $X = \left(\bar{Y}_m, \mathbf{0}^{n\times p}\right) \in \mathbb{R}^{n\times(m+p)}$, $B^T = (\mathbf{0}^{p\times m}, I^{p\times p}) \in \mathbb{R}^{p\times(m+p)}$, it holds

$$\left(\bar{Y}, \bar{W}\right) = X + \bar{W}B^T.$$

Given the thin SVD $\bar{Y}_m \approx \mathbf{U}_m\Sigma_m\mathbf{V}_m^T$, it holds

$$X \approx \mathbf{U}_m\Sigma_m\left(\mathbf{V}_m^T, \mathbf{0}^{r_m\times p}\right) =: \mathbf{U}_m\Sigma_m\tilde{V}^T,$$

with $\mathbf{U} \in \mathbb{R}^{n\times r_m}$, $\Sigma \in \mathbb{R}^{r_m\times r_m}$, $\tilde{V} \in \mathbb{R}^{(m+p)\times r_m}$.
In Algorithm 3.3, it is stated precisely, how this approach applies to Problem 2.1. Note that only steps 3 to 10 have to be adjusted when compared to Algorithm 3.2, while steps 1,2 and 11-14 stay unchanged.
The function orth : $\mathbb{R}^{n\times p} \to \mathbb{R}^{n\times p}, P \mapsto P_\perp$ in step 3 of Algorithm 3.3 denotes the computation of an orthonormal basis spanning the column space of the input matrix. This can be achieved via Gram-Schmidt orthogonalization or via a more robust SVD. Note that essentially the same idea has already been used in [9, §3].
As pointed out in [7, 5.2.9] Gram-Schmidt "should be used to compute orthonormal bases only when the vectors to be orthogonalized are fairly independent". While this is most certainly the case for the random examples shown in section 4, in practical application, where snapshots are e.g. given by solutions to PDEs at different parameter conditions, they are expected to be much closer to being linearly dependent.

3.4 An EVD-based one-step update strategy

In this section, the approach introduced in section 3.2 is modified such that the recentering is directly considered in setting up the small $(m+p)\times(m+p)$-EVD problem.

$$\begin{aligned} \bar{Y}_{m+p}^T\bar{Y}_{m+p} &\approx \left(\left(\mathbf{U}_m\Sigma_m\mathbf{V}_m, \bar{W}\right) + T_{m+p}\mathbf{1}_{m+p}^T\right)^T \left(\left(\mathbf{U}_m\Sigma_m\mathbf{V}_m, \bar{W}\right) + T_{m+p}\mathbf{1}_{m+p}^T\right) \\ &= \begin{pmatrix} \mathbf{V}_m\Sigma_m^2\mathbf{V}_m^T & \mathbf{V}_m\Sigma_m\mathbf{U}_m^T\bar{W} \\ \bar{W}^T\mathbf{U}_m\Sigma_m\mathbf{V}_m^T & \bar{W}^T\bar{W} \end{pmatrix} + \begin{pmatrix} \mathbf{V}_m\Sigma_m\mathbf{U}_m^T \\ \bar{W}^T \end{pmatrix} T_{m+p}\mathbf{1}_{m+p}^T \\ &\quad +\mathbf{1}_{m+p}T_{m+p}\left(\mathbf{U}_m\Sigma_m\mathbf{V}_m, \bar{W}\right) + \|T_{m+p}\|^2\mathbf{1}_{m+p}\mathbf{1}_{m+p}^T \\ &=: K \in \mathbb{R}^{(m+p)\times(m+p)}. \end{aligned} \tag{3.2}$$

Computing the large left-hand-side singular vectors from a decomposition of $K = \mathbf{Q}\Lambda\mathbf{Q}^T$ becomes more costly. The corresponding approach is detailed in Algorithm 3.4.

3.5 Pseudo batch method

Keeping in mind Remark 2.2, recomputing the updated SVD from scratch can be outlined as follows: (1) recompute $\bar{Y} \approx (\mathbf{U}_m\Sigma_m\mathbf{V}_m)$; (2) compute $X = (\bar{Y}, \bar{W}) - T_{m+p}\mathbf{1}_{m+p}^T$; (3) compute $K = X^TX$, decompose $K = \mathbf{V}_{m+p}\Lambda\mathbf{V}_{m+p}^T$; (4) for $j = 1, ..., r_{m+p}$ compute $\mathbf{U}_{m+p}^j = \frac{1}{\sqrt{\lambda_j}}X\mathbf{V}_{m+p}^j$.
In Section 4 it is exposed under negligible prerequisites that this approach is actually more costly than Algorithm 3.4.

```
ALGORITHM 3.2: POD-ROM update algorithm via EVD and SVD
Input: previous-stage ROM $(\mathbf{U}_m, \Sigma_m, \mathbf{V}_m, A_m, n, m, r_m)$
    number of new snapshots $p$, new snapshots $W^{m+1}, ..., W^{m+p} \in \mathbb{R}^n$
 1: Compute average of augmented snapshot set and average shift vector
        $$T_{m+p} := \frac{1}{m+p}\left(pA_m - \sum_{i=m+1}^{m+p} W^i\right), \quad A_{m+p} := A_m - T_{m+p}$$
 2: Compute $\bar{\mathbf{W}} := (W^{m+1} - A_m, ..., W^{m+p} - A_m) \in R^{n \times p}$.
 3: Compute $M := \Sigma_m \mathbf{U}_m^T \bar{\mathbf{W}} \in R^{r_m \times p}$.
 4: Compute $\bar{\mathbf{w}} = \bar{\mathbf{W}}^T \bar{\mathbf{W}} \in R^{p \times p}$.
 5: Decompose (EVD) $K = \begin{pmatrix} \Sigma_m^2 & M \\ M^T & \bar{\mathbf{w}} \end{pmatrix} \overset{!}{=} \tilde{Q}\tilde{\Lambda}\tilde{Q}^T \in R^{(r_m+p)\times(r_m+p)}$.
 6: Determine $\tilde{r} \le r_m + p$ such that $\mathrm{ric}(\tilde{r}) = \frac{\sum_{i=1}^{\tilde{r}} \tilde{\lambda}_i}{\sum_{i=1}^{r_m+p} \tilde{\lambda}_i} \ge 1 - \epsilon$.
 7: Discard the columns of $\tilde{Q}$ of index larger than $\tilde{r}$.
 8: Compute $\tilde{\Sigma} = \mathrm{diag}(\tilde{\sigma}_1, ..., \tilde{\sigma}_{\tilde{r}}) = \mathrm{diag}\left(\sqrt{\tilde{\lambda}_1}, ..., \sqrt{\tilde{\lambda}_{\tilde{r}}}\right) \in \mathbb{R}^{\tilde{r}\times\tilde{r}}$.
 9: Compute $\mathbf{Q} := \begin{pmatrix} \mathbf{V}_m & \mathbf{0} \\ \mathbf{0} & I^{p\times p} \end{pmatrix} \tilde{Q} \in \mathbb{R}^{(m+p)\times\tilde{r}}$
10: Compute $\mathbf{S} = \left(\mathbf{U}_m\Sigma_m, \bar{\mathbf{W}}\right)\tilde{Q}\mathrm{diag}\left(\frac{1}{\tilde{\sigma}_1}, ..., \frac{1}{\tilde{\sigma}_{\tilde{r}}}\right) \in R^{n\times\tilde{r}}$.
11: Compute
        $f := \mathbf{S}^T T_{m+p} \in \mathbb{R}^{\tilde{r}}$;  $\tilde{F} := T_{m+p} - \mathbf{S}f \in \mathbb{R}^n$;  $F = \frac{\tilde{F}}{\|\tilde{F}\|} \in \mathbb{R}^n$;
        $g := \mathbf{Q}^T \mathbf{1} \in \mathbb{R}^{\tilde{r}}$;  $\tilde{G} := \mathbf{1} - Qg \in \mathbb{R}^{m+p}$;  $G = \frac{\tilde{G}}{\|\tilde{G}\|} \in \mathbb{R}^{m+p}$;
12: Decompose (SVD)
        $$\mathbf{K} = \begin{pmatrix} \tilde{\Sigma} + fg^T & \|\tilde{G}\|f \\ \|\tilde{F}\|g^T & \|\tilde{F}\|\|\tilde{G}\| \end{pmatrix} \overset{!}{=} \tilde{U}\Sigma_{m+p}\tilde{V}^T \in \mathbb{R}^{(\tilde{r}+1)\times(\tilde{r}+1)}$$
13: Choose $r_{m+p} \le \tilde{r} + 1$ such that $\mathrm{ric}(r_{m+p}) = \frac{\sum_{i=1}^{r_{m+p}} \sigma_{m+p,i}^2}{\sum_{i=1}^{\tilde{r}+1} \sigma_{m+p,i}^2} \ge 1 - \epsilon$.
    Discard the singular values with index larger than $r_{m+p}$, as well as the corresponding columns of $\tilde{U}$ and of $\tilde{V}$.
14: Compute
        $$\mathbf{U}_{m+p} := (\mathbf{S}, F)\,\tilde{U} \in \mathbb{R}^{n\times r_{m+p}}, \quad \mathbf{V}_{m+p} := (\mathbf{Q}, G)\,\tilde{V} \in \mathbb{R}^{(m+p)\times r_{m+p}}.$$
Output: Updated ROM $(\mathbf{U}_{m+p}, \Sigma_{m+p}, \mathbf{V}_{m+p}, A_{m+p}, n, m+p, r_{m+p})$
```

```
ALGORITHM 3.3: POD-ROM update via two-steps SVD à la [3]
Input: previous-stage ROM $(\mathbf{U}_m, \Sigma_m, \mathbf{V}_m, A_m, n, m, r_m)$
    number of new snapshots $p$, new snapshots $W^{m+1}, ..., W^{m+p} \in \mathbb{R}^n$.
 1: Identical to Algorithm 3.2.
 2: Identical to Algorithm 3.2.
 3: Compute $P := \bar{W} - \mathbf{U}(\mathbf{U}^T \bar{W})$
 4: Compute $P_\perp = \text{orth}(P)$; $R_{\bar{W}} = P_\perp^T P$.
 5: Compute $K = \begin{pmatrix} \Sigma_m & \mathbf{U}^T\bar{W} \\ \mathbf{0}^T & R_{\bar{W}} \end{pmatrix} \in \mathbb{R}^{(r_m+p)\times(r_m+p)}$.
    Decompose (SVD) $K = \mathbf{U}_K \Sigma_K \mathbf{V}_K^T$.
 6: Determine $\tilde{r} \leq r_m + p$ such that $\text{ric}(\tilde{r}, \Sigma_K^2) \geq 1 - \epsilon$.
 7: Discard the columns of $\mathbf{U}_K$ and $\mathbf{V}_K$ of index larger than $\tilde{r}$.
 8: Reduce the singular value matrix $\tilde{\Sigma} := \text{diag}\,(\Sigma_K)_{ii} \in \mathbb{R}^{\tilde{r}\times\tilde{r}}$.
 9: Compute $\mathbf{Q} := \left( \begin{pmatrix} \mathbf{V}_m \\ \mathbf{0}^{p\times r} \end{pmatrix}, \begin{pmatrix} \mathbf{0}^{m\times p} \\ I^{p\times p} \end{pmatrix} \right) \mathbf{V}_K \in \mathbb{R}^{(m+p)\times\tilde{r}}$
10: Compute $\mathbf{S} = (\mathbf{U}, P_\perp)\, \mathbf{U}_K \in \mathbb{R}^{n\times\tilde{r}}$
11: Identical to Algorithm 3.2.
12: Identical to Algorithm 3.2.
13: Identical to Algorithm 3.2.
14: Identical to Algorithm 3.2.
Output: Updated ROM $(\mathbf{U}_{m+p}, \Sigma_{m+p}, \mathbf{V}_{m+p}, A_{m+p}, n, m+p, r_{m+p})$
```

```
ALGORITHM 3.4: POD-ROM update via one-step EVD
Input: previous-stage ROM $(\mathbf{U}_m, \Sigma_m, \mathbf{V}_m, A_m, n, m, r_m)$
    number of new snapshots $p$, new snapshots $W^{m+1}, ..., W^{m+p} \in \mathbb{R}^n$.
 1: Identical to Algorithm 3.2.
 2: Identical to Algorithm 3.2.
 3: Compute $K_1 := \begin{pmatrix} \mathbf{V}_m \Sigma_m^2 \mathbf{V}_m^T & \mathbf{V}_m \Sigma_m \mathbf{U}_m^T \bar{W} \\ \bar{W}^T \mathbf{U}_m \Sigma_m \mathbf{V}_m^T & \bar{W}^T \bar{W} \end{pmatrix}$
 4: Compute $K_2 := \mathbf{1}_{m+p} T_{m+p}^T \left( \mathbf{U}_m \Sigma_m \mathbf{V}_m^T, \bar{W} \right)$;
    Compute $K_3 := \|T_{m+p}\|^2 \mathbf{1}_{m+p} \mathbf{1}_{m+p}^T$.
 5: Compute $K = K_1 + K_2 + K_2^T + K_3 \in \mathbb{R}^{(m+p)\times(m+p)}$.
    Decompose (EVD) $K = \tilde{Q}\tilde{\Lambda}\tilde{Q}^T$.
 6: Determine $r_{m+p} \leq m + p$ such that $\text{ric}(r_{m+p}, \tilde{\Lambda}) \geq 1 - \epsilon$.
 7: Discard the columns of $\tilde{Q}$ of index larger than $r_{m+p}$.
 8: Compute $\Sigma_{m+p} := \text{diag}\left(\sigma_1, ..., \sigma_{r_{m+p}}\right) = \text{diag}\left(\sqrt{\tilde{\lambda}_1}, ..., \sqrt{\tilde{\lambda}_{r_{m+p}}}\right)$
 9: Set $\mathbf{V}_{m+p} := \tilde{Q}$; Compute $\text{diag}(\frac{1}{\sigma_1}, ..., \frac{1}{\sigma_{r_{m+p}}})$.
10: Compute
    $\mathbf{U}_{m+p} = \left( \left( \mathbf{U}_m \Sigma_m \mathbf{V}_m^T, \bar{\mathbf{W}} \right) + T_{m+p} \mathbf{1}_{m+p}^T \right) \tilde{Q} \text{diag}\left( \frac{1}{\tilde{\sigma}_1}, ..., \frac{1}{\tilde{\sigma}_{r_{m+p}}} \right) \in R^{n\times r_{m+p}}$.
Output: Updated ROM $(\mathbf{U}_{m+p}, \Sigma_{m+p}, \mathbf{V}_{m+p}, A_{m+p}, n, m+p, r_{m+p})$
```

4 Analysis of Computational Costs

Computational costs will be measured in floating point operations (flops) [7, §1.2.4]; for simplicity, nmp flops will be counted for the standard product of matrices $A \cdot B, A \in \mathbb{R}^{n\times m}, B \in \mathbb{R}^{m\times p}$.
Table 4.1 displays the flop count of the first three update methods presented in Section 3. The portion of the total computational costs of the pseudo batch method 3.5 that scale in the snapshot dimension n add up to

$$n\left(mr_m + (m+p) + \frac{1}{2}(m+p)(m+p+1) + (m+p)r_{m+p}\right) \text{ flops.}$$

Using Table 4.1, it follows that

$$\text{flops(Alg. pseudo batch) - flops(Alg. 3.4)} = n\left(\frac{1}{2}m^2 + (r_m - \frac{1}{2})m - (r_m - 1)p - 1\right).$$

An elementary estimate shows that Algorithm 3.4 is more efficient than the pseudo batch method, if the number of originally retained modes is larger than the number of snapshot vectors to be added $(r_m \geq p)$ or if the number of original snapshots is both larger than four and larger than the number of snapshots to be added $(m \geq p \wedge m \geq 4)$. Note that for most practical applications one of these conditions (most probably both of them) will hold true.
As can be deduced from Table 4.1, the computational costs of Algorithm 3.3 exceed those for Algorithm 3.2 by more than

$$\text{flops(Alg. 3.3) - flops(Alg. 3.2)} = \left(r_m p + \frac{1}{2}(p^2+p)\right) n + O(\text{orth}(P),$$

Table 4.1: Computational complexity of Algorithm 3.2, Algorithm 3.3 and Algorithm 3.4. For convenience, only steps depending on n are taken into account. In Alg. 3.3, step 4, the symbol $O(\text{orth})$ denotes the flop count for computing an orthonormal basis for the column space of the input matrix. The exact flop count depends on the method applied but is always in $O(p^2 n)$.

Flops	Algorithm 3.2	Algorithm 3.3	Algorithm 3.4
step 1	$(p+3)n$	$(p+3)n$	$(p+3)n$
step 2	pn	pn	pn
step 3	$r_m pn$	$(2r_m p + p)n$	$(r_m p + mp + \frac{1}{2}p(p+1))n$
step 4	$\frac{1}{2}p(p+1)n$	$p^2 n + O(\text{orth})$	$(m+p+1)n$
step 10	$(r_m + p)\tilde{r}n$	$(r_m + p)\tilde{r}n$	$(m+p+(m+p)r_{m+p})n$
step 11	$(2\tilde{r}+3)n$	$(2\tilde{r}+3)n$	$\emptyset$
step 14	$(\tilde{r}+1)r_{m+p}n$	$(\tilde{r}+1)r_{m+p}n$	$\emptyset$

where $O(\text{orth})$ denotes the flop count for computing an orthonormal basis for the column space of the argument matrix $P \in \mathbb{R}^{n\times p}$, see Section 3.3. It holds $O(\text{orth}(P)) = O(np^2)$, the precise computational effort depends on the orthogonalization scheme applied.
Next, we compare Algorithm 3.2 with Algorithm 3.4. From Table 4.1 it follows that

$$\begin{aligned}\text{flops(Alg. 3.4) - flops(Alg. 3.2)} &= n\Big(mp + 2(m+p) + (m+p)r_{m+p} \\ &\quad -(r_m + p + 2)\tilde{r} - 2 - (\tilde{r}+1)r_{m+p}\Big) =: (*)\end{aligned} \tag{4.1}$$

Further analyse will be made under the assumption that the additional snapshots are linearly independent and therefore add a maximum number of p to the rank of the SVD. The initial compression level will be expressed in relation to the original number of snapshots. More precisely

$$\text{assume: } \tilde{r} = r_m + p; \;\; r_{m+p} = r_m + p + 1; \;\; r_m = xm. \tag{4.2}$$

Thus, equation (4.1) becomes a quadratical expression in the compression rate x.

$$(*) = \left((x - 2x^2)m^2 + (2-3x)mp + (3-4x)m - p^2 - p - 3)\right) n. \tag{4.3}$$

As a consequence, Algorithm 3.2 is more efficient than Algorithm 3.4, if

$$0 < x < \frac{1}{4m}\left(m - 3p - 4 + \sqrt{p^2 + m^2 - 8p + 22m + 16mp - 8}\right) \tag{4.4}$$

A rough estimate shows that a positive compression rate $x > 0$ fulfilling the above equation exists, if $m \geq \frac{p}{2} + 3$, which is a negligible condition.
The above considerations imply that it is case dependent, whether Algorithm 3.2 is more efficient than Algorithm 3.4 or vice versa. Given a specific triple m, p, x, equation (4.4) can be used to make a precise choice between Algorithm 3.2 and Algorithm 3.4. In the next observation, the above results are summarized and simple rules of thumb valid in most practical cases are given.

Observation 1. *Algorithm 3.2 is always more efficient than Algorithm 3.3.*
Algorithm 3.4 is more efficient than the pseudo batch method (see Section 3.5), if $(r_m \geq p)$ *or* $(m \geq p \wedge m \geq 4)$.
Rules of thumb: For highly compressed models, Algorithm 3.2 is more efficient than Algorithm 3.4; for weakly compressed models, the opposite is true. More precisely:

(i) If the SVD-ROM in question features a compression rate of $x = \frac{r_m}{m} \leq \frac{1}{2}$ *and if the number of original snapshots* m *and the number of update snapshots* p *are related by* $m \geq 2p + 2$, *then Alg. 3.2 is more efficient than Alg. 3.4.*

(ii) If the SVD-ROM in question features a compression rate of $x = \frac{r_m}{m} \geq \frac{2}{3}$ *and if the number of original snapshots is larger than two,* $m \geq 2$, *then Alg. 3.4 is more efficient than Alg. 3.2.*

(iii) If the SVD-ROM in question features a compression rate of $x = \frac{r_m}{m} \geq \frac{3}{5}$ *and if the number of original snapshots* m *and the number of update snapshots* p *are related by* $m \geq \frac{5}{3}p + 5$, *then Alg. 3.4 is more efficient than Alg. 3.2.*

Proof. Proofs of the first two statements have already been indicated. *On (iii)*: Let $x \geq \frac{3}{5}$ and $m \geq \frac{5}{3}p + 5$. Consider equation (4.1) respectively (4.3). It holds

$$\begin{aligned} \frac{1}{n}(*) &= (x - 2x^2)m^2 + (2 - 3x)mp + (3 - 4x)m - p^2 - p - 3 \\ &\leq -\frac{3}{25}m^2 + \frac{1}{5}mp + \frac{3}{5}m - p^2 - p - 3 \qquad (4.5) \\ &\leq m\left(-\frac{3}{25}\left(\frac{5}{3}p + 5\right) + \frac{1}{5}p + \frac{3}{5}\right) - p^2 - p - 3 < 0. \end{aligned}$$

This shows *(iii)*. The claims *(i)* and *(ii)* are proved in an analoguous manner. □

Table 4.2 displays the computational costs and accuracy of all three update approaches when applied to snapshot matrices $Y \in \mathbb{R}^{n \times m}$ and update matrices $W \in \mathbb{R}^{n \times p}$ with fixed but random entries; here n denotes the size of the snapshots, m denotes the number of original snapshot observations and p denotes the number of update snapshots. The reconstruction error is based on the 2-norm of the matrix $\left(\bar{Y}_{m+p} - \mathbf{U}_{m+p}\Sigma_{m+p}\mathbf{V}^T_{m+p}\right)$, where the SVD-representation of $\bar{Y}_{m+p}$ is computed via the various algorithms.
In all cases, the initial and final compression level were chosen to be the maximum value of $r_m = m - 1$, respectively $r_{m+p} = m + p - 1$. As predicted by theory, Algorithm 3.4 outperforms its competitors under this conditions.
The amount of time to compute the initial SVD has not been considered.

Using the same notation as introduced above, Table 4.3 displays the computational costs and accuracy of all three update approaches when applied to a random snapshot matrix of dimensions $100,000 \times 500$. Here, the original SVD was truncated by force, disregarding its true rank. The initial and final compression level were chosen to be $r_m = 0.4m$, respectively $r_{m+p} = r_m + 100$. In this test case, again in line with theory, Algorithm 3.2 is the most efficient.

Remark 4.1 (A note on parallelization). *All subroutines in algorithms 3.2 and 3.4 that scale in the snapshot dimension n consist of simple arithmetics or matrix-matrix products. Think of matrices $A \in \mathbb{R}^{n \times m}$ and $B \in \mathbb{R}^{n \times p}$ as divided into sub-blocks $A^T = \left(A_1^T, ..., A_q^T\right), B^T = \left(B_1^T, ..., B_q^T\right)$ with $A_i \in \mathbb{R}^{n_i \times m}, B_i \in \mathbb{R}^{n_i \times q}$ and $\sum_{i=1}^q n_i = n$, then $A^T B = \sum_{i=1}^q A_i^T B_i$. This indicates an obvious way for parallel computations.*
On the other hand, Algorithm 3.3 involves computing an orthogonal basis for a given column space, which is not as straight-forward to do in parallel.

Table 4.2: Accuracy and CPU time of the three update approaches when applied to random matrices and an initial SVD featuring a compression level of 100%.

$(n, m, p) = (100.000, 100, 50)$			
Method	Reconstruction error	t(steps 3,4) (sec.)	t(total) (sec.)
Alg. 3.2	$1.923e-12$	0.1540	0.6895
Alg. 3.3, SVD-orth	$2.211e-12$	0.7720	1.300
Alg. 3.3, QR-orth	$2.375e-12$	0.8073	1.343
Alg. 3.4	$1.874e-12$	0.2366	0.5321
$(n, m, p) = (100.000, 500, 100)$			
Method	Reconstruction error	t(steps 3,4) (sec.)	t(total) (sec.)
Alg. 3.2	$2.342e-12$	0.3683	6.316
Alg. 3.3, SVD-orth	$2.504e-12$	3.311	7.884
Alg. 3.3, QR-orth	$2.333e-12$	3.178	8.145
Alg. 3.4	$2.279e-12$	1.888	4.424
$(n, m, p) = (2.000.000, 20, 5)$			
Method	Reconstruction error	t(steps 3,4) (sec.)	t(total) (sec.)
Alg. 3.2	$3.709e-11$	0.09098	1.023
Alg. 3.3, SVD-orth	$5.320e-11$	0.6542	1.520
Alg. 3.3, QR-orth	$3.724e-11$	0.6927	1.612
Alg. 3.4	$3.666e-11$	0.4325	0.9761

Table 4.3: Accuracy and CPU time of the three update approaches when applied to a random matrix with the initial SVD featuring a compression level of 40%.

$(n, m, p) = (100.000, 500, 100)$			
compression level $r_m = 200$, $r_{m+p} = 300$, $x = \frac{r_m}{m} = 0.4$			
Method	Reconstruction error	t(steps 3,4)	t(total) (sec.)
Alg. 3.2	92.21	0.1818	1.934
Alg. 3.3, SVD-orth	92.21	2.700	4.184
Alg. 3.3, QR-orth	92.21	2.746	4.233
Alg. 3.4	93.41	1.882	3.545

5 Summary and final Remarks

It has been demonstrated that the update strategies of algorithms 3.2 and 3.4, which follow the symmetric EVD approach, are more efficent than Algorithm 3.3, which relies on a rank-p SVD update.. It is well-known that computing an SVD of a matrix A via the symmetric EVD of $A^T A$ may lead to a loss of information [7, §5.3.2]. While Algorithm 3.3 does not suffer from this effect, the orthogonalization of the incoming snapshots against the existing basis required in this approach incorporates additional numerical inaccuracy. In order to keep track of the data mean, Algorithm 3.2 necessitates to successively solve a small EVD and a small SVD and Algorithm 3.3 necessitates to solve two small SVD problems whereas only one such problem accrues for Algorithm 3.4.
For the examples presented in section 4, all methods shared a very similar level of accuracy. A theoretical error analysis is beyond the scope of this work.
Theory as well as the examples suggest that for practical impelentations, an automated switching between method 3.2 and method 3.4 based on the compression level and equation (4.4) is advantageous.

Bibliography

[1] P. Astrid, S. Weiland, K. Wilcox, and T. Backx. Missing points estimation in models described by proper orthogonal decomposition. *IEEE Transactions on Automatic Control*, 53(10):2237–2251, 2007.

[2] T. Braconnier, M. Ferrier, J.-C. Jouhaud, M. Montagnac, and P. Sagaut. Towards an adaptive pod/svd surrogate model for aeronautic design. *Computers & Fluids*, 40(1):195 – 209, 2011.

[3] M. Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications*, 415:20–30, 2006.

[4] J. R. Bunch and C. P. Nielsen. Updating the singular value decomposition. *Numer. Math.*, 31:111–129, 1978.

[5] S. Chandrasekaran, B. S. Manjunath, Y. F. Wang, J. Winkeler, and H. Zhang. An eigenspace update algorithm for image analysis. *Graphical Models and Image Processing*, 59(5), 1997.

[6] R. Everson and L. Sirovich. Karhunnen-loève procedure for gappy data. *Journal of the Optical Society of America*, 12(8):1657–1664, 1995.

[7] G .H. Golub and C. F. Van Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore – London, 3 edition, 1996.

[8] M. Gu and S. C. Eisenstat. A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem. *SIAM J. Matrix Anal. Appl.*, 15:1266–1276, October 1994.

[9] P. Hall, D. Marshall, and R. Martin. Merging and splitting eigenspace models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9):1042–1049, 2000.

[10] R. Pinnau. Model reduction via proper orthogonal decomposition. In W. H. A. Schilders, H. A. Van der Vorst, and J. Rommes, editors, *Model Order Reduction: Theory, Research Aspects and Applications*, volume 13 of *Springer Series Mathematics in Industry*, pages 95–109. Springer, 2008.

[11] M. Rathinam and L. R. Petzold. A new look on proper orthogonal decomposition. *SIAM Journal on Numerical Analysis*, 41(5):1893–1925, 2003.

[12] P. Stange. On the efficient update of the singular value decomposition. *PAMM*, 8(1):10827–10828, 2008.

[13] R. Zimmermann and S. Görtz. Non-linear pod-based reduced order models for steady turbulent aerodynamics. In *RAeS Conference 2010 Applied Aerodynamics: Capabilities and Future Requirements*, Bristol, UK, July 2010.

IB 124-2011/3

A Comprehensive Comparison of Various Algorithm for Efficiently Updating Singular Value Decomposition Based Reduced Order Models

R. Zimmermann

Verteiler:

Institut für Aerodynamik und Strömungstechnik, BS		1	Exemplar
Institut für Aerodynamik und Strömungstechnik, GÖ		1	Exemplar
Verfasser	je	1	Exemplar
Prof. Dr.-Ing. habil. C. Rossow		1	Exemplar
Prof. Dr.-Ing. N. Kroll		1	Exemplar
Dr.-Ing. D. Schwamborn		1	Exemplar
Technische Informationsbibliothek Hannover		1	Exemplar
Niedersächsische Landesbibliothek Hannover		1	Exemplar
Deutsche Bibliothek Frankfurt am Main		2	Exemplare
Niedersächsische Landesbibliothek Hannover		1	Exemplar
Technische Informationsbibliothek Hannover		1	Exemplar
Zentralbibliothek		2	Exemplare
Reserve		5	Exemplare
		19	Exemplare