Model Reduction using Incremental Decomposition Techniques to support Unsteady Adjoint-Based Fluid Dynamic Shape Optimization

Master Thesis

by

Hendrik Fischer

Supervision: Prof. Dr. M. Siebenborn
Prof. Dr. T. Rung

University of Hamburg
Department of Mathematics

Hamburg University of Technology
Institute for Fluid Dynamics and Ship Theory

Hamburg, September 2021

Contents

1 Introduction **1**

2 Unsteady Adjoint-Based Shape Optimization **3**

3 Model reduction via Singular Value Decomposition **7**

3.1 Singular Value Decomposition 7

3.2 Additive Modification of the SVD 10

4 Application to reduced Flow Field Model and Optimization **14**

4.1 Incremental Update of reduced Flow Field 14

4.2 Extensions 19

4.2.1 Incremental Bunch Update 19

4.2.2 Adaptivity Techniques 21

4.2.3 Parallelization Techniques 23

5 Numerical Results **26**

5.1 Flow Field Reconstruction 26

5.1.1 Approximation Quality 27

5.1.2 Performance 31

5.1.3 Incremental Bunch Update 32

5.1.4 Adaptivity Techniques 34

5.1.5 Parallelization Techniques 37

5.2 Optimization 40
5.2.1 Laminar Flow around 2D Cylinder 40
5.2.2 Turbulent Two-Phase Flow around 3D Sphere 47

6 Conclusion and Outlook 50

Chapter 1

Introduction

The research field of shape optimization provides a variety of potential applications to engineering topics. With respect to fluid dynamics, industrial scale applications include the shape optimization of ship hulls, curved ducts, as well as centrifugal pumps. Common motivations for shape optimization are the minimization of drag or power loss and the active flow control.

In this context, the adjoint-based shape optimization is often applied, since it offers an efficient method to calculate the shape sensitivity. The main feature of this method is that the computational costs are independent of the number of design variables [1]. The method requires to solve the adjoint equations in addition to the physical (primal) ones. The application of the adjoint-based optimization to unsteady problems causes that the adjoint equations have to be integrated backward in time, which requires the primal solution to be available in the solution process [2]. However, industrial applications often involves small time steps and large computational grids such that the storage of the whole primal solution leads to a memory overhead [3]. Moreover, for large problems the memory requirements can exceed the maximal capacity of the system making it impossible to store the primal solution and thus prevent the optimization from being carried out properly.

A remedy for this issue are the well known and widely used checkpointing techniques [4]. Here, the full storage of the primal solution is omitted and the solution is only stored for a fixed number of time instances, the checkpoints, and the missing information is recomputed on demand. Although this approach reduces the storage requirements, it significantly raises the computational effort simultaneously [3].

In contrast, this work pursues the approach of using a Reduced Order Model (ROM) to obtain an approximation of the primal flow field that has a greatly reduced memory requirement. In fluid dynamics problems, the Proper Orthogonal Decomposition (POD) is widely used for this purpose [3]. The main idea of this technique derives from the notion of the Principal Component Analysis (PCA) with the aim of the extraction of the predominant part of the flow field to obtain a compact representation which is optimal with respect to its memory consumption [2].

However, the typical POD algorithm requires the complete flow field to be available beforehand. This requirement makes the method inappropriate for the unsteady adjoint-based optimization, as it also results in a memory exhaustion. Therefore, the objective of this thesis is to develop an incremental variant of the POD such that the reduced primal field can be computed in a computational efficient way on the fly during the primal computation. Since the POD and the Singular Value Decomposition (SVD) can be used indifferently [2] and for the SVD an incremental variant already exists [5] it is more reasonable to resort to the SVD instead of the POD in the course of this thesis.

Therefore, this thesis is organized as follows: In chapter 2 the mathematical model and the optimization framework for the unsteady adjoint-based fluid dynamic shape optimization are introduced. Subsequently, in chapter 3 the model reduction via the SVD is outlined and further a general approach for the additive modification of a SVD is presented. Based on this, different incremental SVD variants can be derived. In chapter 4, the incremental construction of the reduced primal flow field and its application to the optimization process is discussed. In addition, extensions that aim to increase the overall power of the incremental approach are provided. Afterwards, in chapter 5 the incremental SVD and its extensions are tested numerically and the shape optimization based on the reduced primal field is applied to two test cases of different complexity. Finally, the thesis closes with a conclusion and outlines further research in chapter 6.

Chapter 2

Unsteady Adjoint-Based Shape Optimization

In this chapter, the approach of the unsteady adjoint-based fluid dynamic shape optimization is briefly prescribed. The main focus is on the development of an optimization method for dealing with unsteady laminar flows of incompressible fluids. For this purpose, the chapter is structured as follows. Initially, the mathematical modeling of the physical problem is presented. Based on this, the optimization problem is examined and the adjoint equations are derived. Finally, the calculation of the surface sensitivity is discussed and its application to a gradient based optimization method is pointed out. Within this chapter, Einstein's summation convention is used for repeated lower-case Latin subscripts.

The obtained physical problem is modeled for a domain $\Omega \subset \mathbb{R}^d$ with $d = 2, 3$ and $\Gamma = \partial\Omega$ in a time interval $T = [t_1, t_2]$ by the incompressible Navier-Stokes equations (NSE). Thus in residual form the governing primal equations for the vector-valued velocity u_i and scalar pressure p are given by

$$R_i^u : \rho \frac{\partial u_i}{\partial t} + \rho u_j \frac{\partial u_i}{\partial x_j} + \frac{\partial}{\partial x_j} \left[p\delta_{ij} - 2\mu S_{ij} \right] = 0, \tag{2.1a}$$

$$R^p : - \frac{\partial u_i}{\partial x_i} = 0. \tag{2.1b}$$

Here, $S_{ij} = 0.5(\partial u_i/\partial x_j + \partial u_j/\partial x_i)$, δ_{ij}, ρ and μ with $i, j \leq d$ refer to the components of the strain-rate tensor, the Kronecker delta components, the fluid density and the molecular viscosity, respectively.

Starting from the primal flow described by the NSE, the optimization problem is examined. Since the shape optimization is subject to the behavior of the surrounding flow field it is expressed by an PDE-constrained optimization, viz.

$$(P)\ \begin{cases} \min\limits_{c \in C_{ad}} J(y,c), \\ \text{s.t. } R(y,c) = 0, \end{cases} \tag{2.2}$$

where J denotes the objective or cost functional, y the vector of primal states, which consists of u_i and p, and R the PDE constrains in form of the residuals in Eqs. 2.1. Furthermore, c is the control parameter and C_{ad} the set of all admissible states from which c is selected. In the case of shape optimization c refers to the shape of the structure which should be optimized.
In this work, the focus is on minimizing the drag force acting on the considered structure. According to [1, 6], the cost functional for the time-averaged fluid flow induced force projected in a spatial direction d_i is given by

$$J = \frac{1}{t_2 - t_1} \int\limits_T \int\limits_{\Gamma_O} \left[p\delta_{ij} - 2\mu S_{ij} \right] n_j d_i d\Gamma dt \tag{2.3}$$

where $\Gamma_O \subset \Gamma$ defines the boundary of the structure which shape should be optimized or merely a section of it. After having formulated the optimization in Eq. 2.2 as a constrained problem, the Lagrangian principle can be applied to eliminate the constrains [7]. Here, from the Lagrangian formalism the continuous Lagrange functional

$$\mathcal{L} = J + \int\limits_T \int\limits_\Omega \left[\hat{u}_i R_i^u + \hat{p} R^p \right] d\Omega dt \tag{2.4}$$

is obtained [1, 8]. The Lagrangian multipliers $\hat{u}_i$ and $\hat{p}$ refer to the adjoint velocity and the adjoint pressure, respectively. From the necessary first order optimality conditions it then follows that the derivatives of the objective have to vanish at the optimal point in all directions [1]. Thus for the total variation at the minimum

$$\delta\mathcal{L} = \delta_{\hat{y}}\mathcal{L} \cdot \delta\hat{y} + \delta_y\mathcal{L} \cdot \delta y + \delta_c\mathcal{L} \cdot \delta c \overset{!}{=} 0, \tag{2.5}$$

have to hold and each term must disappear [8]. The single terms denote the variation of the Lagrangian in the direction of adjoint $\delta_{\hat{y}}$ and primal state δy as well as control δc. This condition leads to the adjoint equations [9] and also permits a definition of the sensitivity rule along the design surface.
The requirement of vanishing derivatives yields to the adjoint equations:

$$\hat{R}_i^u : -\rho \frac{\partial \hat{u}_i}{\partial t} - \rho u_j \frac{\partial \hat{u}_i}{\partial x_j} + \rho \hat{u}_j \frac{\partial u_i}{\partial x_j} + \frac{\partial}{\partial x_j} \left[\hat{p}\delta_{ij} - 2\mu \hat{S}_{ij} \right] = 0, \tag{2.6a}$$

$$\hat{R}^p : -\frac{\partial \hat{u}_i}{\partial x_i} = 0. \tag{2.6b}$$

The adjoint equations are similar to the primal equations and represent the adjoint companions to the continuity and momentum equation, respectively. Nevertheless, in the adjoint momentum equation an additional advection term occur and the sign of the time derivative has changed. Further, the primal velocity occurs in the adjoint convection terms. Thus, in order to solve the adjoint problem, the primal solution of the same time instant must be available. However, since the adjoint solution points backwards in time [2, 1], the primal equations are solved first, followed by the adjoint equations. Thereby the data of the primal calculation must be available. The boundary conditions of the adjoint equations result from the boundary integrals occurring due to the partial integration in Eq. 2.5, which must vanish to satisfy the optimality condition. The specific consideration is beyond the scope of this thesis but the interested reader can find more information in [2, 3, 1, 8].
Due to the unsteady nature of the flow problem, in addition to the boundary conditions an initial condition for the adjoint problem needs to be determined. The term containing the time derivative in Eq. 2.5 yields

$$\int_T \int_\Omega \rho \hat{u}_i \frac{\partial \delta u_i}{\partial t} d\Omega dt = \left[\int_\Omega \rho \hat{u}_i \delta u_i d\Omega \right]_{t=t_1}^{t=t_2} - \int_T \int_\Omega \rho \frac{\partial \hat{u}_i}{\partial t} \delta u_i d\Omega dt \tag{2.7}$$

where the integral evaluated at the boundaries of the time interval have to vanish. For $t = t_1$, the initial condition for the primal equations leads to $\delta u_i = 0$ [3]. Since no information exists for u_i at time $t = t_2$, to guarantee that the boundary integral vanishes $\hat{u}_i = 0$ have to be set as the initial condition for the adjoint flow at $t = t_2$.

After having solved the primal (physical) and the adjoint problem, finally a sensitivity rule along the controlled design wall can be specified with the aid of the optimality condition on $\delta_c \mathcal{L} \cdot \delta c$ from Eq. 2.5 according to [8, 1, 3] by

$$S_L = - \int_T \int_{\Gamma_O} \mu \frac{\partial u_i}{\partial x_j} \frac{\partial \hat{u}_i}{\partial x_k} n_j n_k d\Gamma_O dt. \tag{2.8}$$

Next, the gradient can be extracted from the sensitivity S_L. However, it is possible that the computed surface sensitivity is rough. Thus the Laplace-Beltrami metric [10] is employed to extract a smooth gradient G_L out of the sensitivity by

$$G_{L_i} - \lambda \Delta_\Gamma G_{L_i} = S_L n_i \tag{2.9}$$

with the Laplace-Beltrami operator $\Delta_\Gamma = \Delta - \Delta_n$, the user-defined control of smoothing λ and the normal vector n_i at each face of Γ [8].

Based on the resulting gradient field, the displacement field d_i is computed according to

$$\frac{\partial}{\partial x_j}\left[q\frac{\partial d_i}{\partial x_j}\right] = 0 \text{ in } \Omega, \quad d_i = G_{L_i} \text{ in } \Gamma_O, \quad d_i = 0 \text{ in } \Gamma \setminus \Gamma_O \tag{2.10}$$

where $q = 1/(W_D + \epsilon)$, with wall normal distance W_D and $\epsilon = 10^{-20}$ [8]. Subsequently, the shape can be updated according to the displacement field.

The overall optimization framework is summarized in algorithm 1 for a time step based numerical scheme with q time steps t_n which are equidistantly distributed in the interval T. Here, the optimization is initialized with the shape c^0 and the shape is updated with means of the presented method i_{max} times.

Algorithm 1 Optimization Framework

Require: Initial shape c^0

```
i = 1
while i ≤ i_max do
    n ← 1
    while n ≤ q do
        Solve primal problem in Eq. 2.1 at time t_n in c^i
        n ← n + 1
    end while
    n ← q
    while n ≥ 0 do
        Solve adjoint problem in Eq. 2.6 at time t_n in c^i
        n ← n − 1
    end while
    Compute surface sensitivity according to Eq. 2.8
    Update shape to c^{i+1} using the steepest decent method
    i ← i + 1
end while
```

Chapter 3

Model reduction via Singular Value Decomposition

In this chapter, first the theoretical foundations of the Singular Value Decomposition (SVD) and the model reduction using the truncated SVD are presented. Subsequently, the additive modification of a SVD is examined. This is regarded as the basis of the incremental SVD.

3.1 Singular Value Decomposition

This section gives an introduction to the basic theory of the singular value decomposition (SVD) for real matrices. For this purpose, different variants of the SVD are presented and the for this thesis relevant properties are summarized. Since the following statements are standard information, their proofs are assumed to be known.

Theorem 3.1.1 (Singular Value Decomposition (SVD))**.** *Let $A \in \mathbb{R}^{p\times q}$. Then there exist the orthogonal matrices $U = (u_1, \dots, u_p) \in \mathbb{R}^{p\times p}$ and $V = (v_1, \dots, v_q) \in \mathbb{R}^{q\times q}$ as well as a diagonal matrix $\Sigma = (\sigma_i \delta_{ij})_{ij} \in \mathbb{R}^{p\times q}$ such that*

$$A = U\Sigma V^T = \sum_{i=1}^{d} \sigma_i u_i v_i^T \tag{3.1}$$

with singular values $\sigma_1 \geq \cdots \geq \sigma_d \geq 0$, singular vectors u_i, v_j and $d = \min\{p, q\}$.

Theorem 3.1.2. *The singular values of a matrix $A \in \mathbb{R}^{p\times q}$ are unique.*

Corollary 3.1.3. *Let $A \in \mathbb{R}^{p\times q}$. Then the invariant $||A|| = ||\Sigma||$ yields*

1. $||A||_2 = \sigma_1$,
2. $||A||_F = \sqrt{\sum_{i=1}^{p}\sum_{j=1}^{q} |a_{ij}|^2} = \sqrt{\sum_{i=1}^{d} \sigma_i^2}$.

Corollary 3.1.4. *Let $A \in \mathbb{R}^{p\times q}$ with corresponding SVD $A = U\Sigma V^T$, $U = (u_1, \dots, u_p) \in \mathbb{R}^{p\times p}$, $V = (v_1, \dots, v_q) \in \mathbb{R}^{q\times q}$ and $r = \max\{k \in \mathbb{N} : \sigma_k \neq 0\}$ the amount of positive singular values. Then*

1. $rank(A) = r$,
2. $\ker(A) = span(v_{r+1}, \dots v_q)$,
3. $im(A) = span(u_1, \dots, u_r)$,
4. $A = \sum_{i=1}^{r} \sigma_i u_i v_i^T$

hold.

Thus Cor. 3.1.4 yields that only the first r singular vectors are needed to determine the matrix A via the SVD. This results in the following corollary.

Corollary 3.1.5 (Reduced singular value decomposition (rSVD))**.** *Let $A \in \mathbb{R}^{p\times q}$ with $rank(A) = r \leq \min\{p, q\}$. Then there exist the orthogonal matrices $U_r = (u_1, \dots, u_r) \in \mathbb{R}^{p\times r}$ and $V_r = (v_1, \dots, v_r) \in \mathbb{R}^{q\times r}$ as well as $\Sigma_r = (\sigma_i \delta_{ij})_{ij} \in \mathbb{R}^{r\times r}$ such that*

$$A = U_r \Sigma_r V_r^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T \tag{3.2}$$

with singular values $\sigma_1 \geq \dots \geq \sigma_r > 0$ and singular vectors u_i, v_j.

The methodology of neglecting zero-valued singular values inspires the approximation approach of additionally truncating the smallest non-zero singular values. This represents the assumption that small singular values have only a minor impact on the approximation quality. This conjecture will be discussed afterwards.

Definition 3.1.6 (Truncated singular value decomposition (tSVD))**.** *Let $A \in \mathbb{R}^{p\times q}$ with $rank(A) = r$ and corresponding rSVD $A = U_r \Sigma_r V_r^T$. Then the rank-t tSVD of A with $0 < t < r$ is defined as*

$$A_t = U_t \Sigma_t V_t^T = \sum_{i=1}^{t} \sigma_i u_i v_i^T \tag{3.3}$$

with $U_t = (u_1, \dots, u_t) \in \mathbb{R}^{p\times t}$, $V_t = (v_1, \dots, v_t) \in \mathbb{R}^{q\times t}$ *and* $\Sigma_t = diag(\sigma_1, \dots, \sigma_t) \in \mathbb{R}^{t\times t}$. *Thereby* $A_t \in \mathbb{R}^{p\times q}$ *denotes the matrix obtained by neglecting the* $r - t$ *smallest singular values of* A *in its rSVD.*

Next, the impact of the truncation of the singular values on the approximation quality is examined.

Theorem 3.1.7 (Eckart-Young Theorem [11])**.** *Let* $A \in \mathbb{R}^{p\times q}$ *with* $rank(A) = r$ *and* $A_t \in \mathbb{R}^{p\times q}$ *obtained by the rank-t tSVD of* A *with* $0 < t < r$. *Then*

$$\min_{\substack{rank(B)=t \\ B\in\mathbb{R}^{p\times q}}} ||A - B|| = ||A - A_t|| = \begin{cases} \sigma_{t+1} & \text{if } ||\cdot||_2, \\ \sqrt{\sum\limits_{i=t+1}^{r} \sigma_i^2} & \text{if } ||\cdot||_F \end{cases} \tag{3.4}$$

holds. Here σ_{t+1} *denotes the largest singular value of* A *which is no singular value of* A_t.

Definition 3.1.8 (Retained Energy)**.** *A criterion for the quality of the truncation of the tSVD is the retained energy of the approximation. Since the energy of a matrix* A *is definded via its squared Frobenius norm [12] the retained energy is given by*

$$\eta = \frac{||A_t||_F^2}{||A||_F^2} = \frac{\sum_{i=1}^{t} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2} \tag{3.5}$$

where $t < r$ *denotes the rank of the tSVD and* r *the rank of* A.

Remark 3.1.9. *Since the calculation of all singular values* σ_i *of a matrix* A *in Def. 3.1.8 is impractical for large rank* $r \gg t$ *the following bounds are introduced. Let* $r \geq \tilde{t} \geq t$ *be the amount of economically computable singular values. Then an upper bound is given by*

$$\eta_{upper} = \frac{\sum_{i=1}^{t} \sigma_i^2}{\sum_{i=1}^{\tilde{t}} \sigma_i^2} \tag{3.6}$$

and a lower bound by

$$\eta_{lower} = \frac{\sum_{i=1}^{t} \sigma_i^2}{\sum_{i=1}^{\tilde{t}} \sigma_i^2 + (r - \tilde{t})\sigma_{\tilde{t}}^2} \tag{3.7}$$

such that

$$\eta_{lower} \leq \eta \leq \eta_{upper} \tag{3.8}$$

holds.

3.2 Additive Modification of the SVD

The objective of this chapter is to derive an algorithm that allows the modification of an already existing SVD without any knowledge about the underlying system matrix. This methodology can then be used for incrementally update the SVD. For this purpose, at first a general approach of an additive rank-c modification of the SVD is revealed. Subsequently, the for this work relevant column-extending modification of the system matrix is derived from these results.

Theorem 3.2.1 (Additive rank-c modification of a rSVD)**.** *Let $A \in \mathbb{R}^{p\times q}$ with rank-r rSVD $A = U\Sigma V$ and $X \in \mathbb{R}^{p\times c}$, $Y \in \mathbb{R}^{q\times c}$ be arbitrary matrices of rank c. Then the rSVD $\tilde{U}\tilde{\Sigma}\tilde{V}^T$ of $A + XY^T$ is given by*

$$\tilde{U} = \begin{bmatrix} U & Q_X \end{bmatrix} U' \tag{3.9}$$

$$\tilde{V} = \begin{bmatrix} V & Q_Y \end{bmatrix} V' \tag{3.10}$$

$$\tilde{\Sigma} = \Sigma' \tag{3.11}$$

where $Q_X R_X$ and $Q_Y R_Y$ are QR decompositions of $(I - UU^T)X$ and $(I - VV^T)Y$, respectively. Additionally, $U'\Sigma' V'^T$ denotes the rSVD of K with

$$K = \begin{bmatrix} I & U^T X \\ 0 & R_X \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & V^T Y \\ 0 & R_Y \end{bmatrix}^T = \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} U^T X \\ R_X \end{bmatrix} \begin{bmatrix} V^T Y \\ R_Y \end{bmatrix}^T . \tag{3.12}$$

Proof. Let $U\Sigma V^T$ be the rSVD of $A \in \mathbb{R}^{p\times q}$. Then $A + XY^T = U\Sigma V^T + XY^T$ holds. This can be reformulated into the SVD resembling expression

$$A + XY^T = \begin{bmatrix} U & X \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} V & Y \end{bmatrix}^T . \tag{3.13}$$

Since the matrix X is arbitrary $\begin{bmatrix} U & X \end{bmatrix}$ has to be reorthogonalized to satisfy the properties of a SVD . Therefore, the component of X that is orthogonal to U is extracted and orthogonalized as follows

$$\begin{aligned} \begin{bmatrix} U & X \end{bmatrix} &= \begin{bmatrix} U & (I - UU^T)X + UU^T X \end{bmatrix} \\ &= \begin{bmatrix} U & (I - UU^T)X \end{bmatrix} \begin{bmatrix} I & U^T X \\ 0 & I \end{bmatrix} \\ &= \begin{bmatrix} U & Q_X \end{bmatrix} \begin{bmatrix} I & U^T X \\ 0 & R_X \end{bmatrix} \end{aligned}$$

where $Q_X R_X = (I - UU^T)X$ is a QR decomposition.
The analogue approach for $\begin{bmatrix} V & Y \end{bmatrix}$ yields

$$\begin{bmatrix} V & Y \end{bmatrix} = \begin{bmatrix} V & Q_Y \end{bmatrix} \begin{bmatrix} I & V^T Y \\ 0 & R_Y \end{bmatrix}$$

with QR decomposition $Q_Y R_Y = (I - VV^T)Y$. Then the substitution of both expressions in Eq. 3.13 leads to

$$A + XY^T = \begin{bmatrix} U & Q_X \end{bmatrix} \begin{bmatrix} I & U^T X \\ 0 & R_X \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & V^T Y \\ 0 & R_Y \end{bmatrix}^T \begin{bmatrix} V & Q_Y \end{bmatrix}^T$$

with outer orthogonal matrices. Further, let K be defined as the factorization of the inner matrices such that

$$K = \begin{bmatrix} I & U^T X \\ 0 & R_X \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & V^T Y \\ 0 & R_Y \end{bmatrix}^T = \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U^T X \\ R_X \end{bmatrix} \begin{bmatrix} V^T Y \\ R_Y \end{bmatrix}^T .$$

Then Theorem 3.1.1 yields the existence of the singular value decomposition $U'\Sigma'V'^T$ of K. Thus, $\tilde{U} = \begin{bmatrix} U & Q_X \end{bmatrix} U'$, $\tilde{\Sigma} = \Sigma'$ and $\tilde{V} = \begin{bmatrix} V & Q_Y \end{bmatrix} V'$ are obtained and constitute the SVD of $A + XY^T$. □

Since this work mainly deals with rank-1 modifications ($c = 1$) the following corollary covers this case more precisely.

Corollary 3.2.2 (Additive rank-1 modification of a rSVD)**.** *Let the same assumptions be given as in Th. 3.2.1 with $c = 1$ and thus $x = X \in \mathbb{R}^p$, $y = Y \in \mathbb{R}^q$. Then let $Q_X = R_x^{-1} q_x$ and $Q_Y = R_y^{-1} q_y$ with $q_x = (x - UU^T x)$, $R_x = ||x - UU^T x||_2$ and $q_y = (y - VV^T y)$, $R_y = ||y - VV^T y||_2$, respectively. K is then formulated by*

$$K = \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} m \\ ||q_x||_2 \end{bmatrix} \begin{bmatrix} n \\ ||q_y||_2 \end{bmatrix}^T \tag{3.14}$$

with $m = U^T x$ and $n = V^T y$ and is given as a sum of a diagonal matrix and a rank-1 matrix.

The approach outlined above provides a variety of possibilities to modify the system matrix. For instance, in addition to updating and downdating, individual values can be modified or rows and columns can be exchanged [13]. However, in this work the focus is on the update or column extension of the system matrix. Therefore, this special case is derived from the general formulation.

Corollary 3.2.3 (Additive column adding rank-c modification of a rSVD)**.** *Let $A \in \mathbb{R}^{p \times q}$ with rank-r rSVD $A = U\Sigma V$. The column updating by the matrix $B \in \mathbb{R}^{p \times c}$ is achieved by extending A and V by c additional zero columns such that $\begin{bmatrix} A & 0 \end{bmatrix} = U\Sigma \begin{bmatrix} V & 0 \end{bmatrix}^T$ and by setting $X = B$ and $Y = \begin{bmatrix} 0 & I \end{bmatrix}$. This yields to*

$$\begin{bmatrix} A & B \end{bmatrix} = \begin{bmatrix} A & 0 \end{bmatrix} + B \begin{bmatrix} 0 & I \end{bmatrix}^T = \begin{bmatrix} A & 0 \end{bmatrix} + \begin{bmatrix} 0 & B \end{bmatrix} . \tag{3.15}$$

Then the additive modification can again be performed according to Theorem 3.2.1.

The procedure is pointed out for a single column extension in the following remark.

Remark 3.2.4 (Additive column adding rank-1 modification of a rSVD)**.** *Based on the results of Cor. 3.2.2 and 3.2.3 a column-adding rank-1 modification can be performed for the vector $b \in \mathbb{R}^p$ by setting $x = b$ and $y = [0, \dots, 0, 1]^T \in \mathbb{R}^q$. Then the updated rSVD is given by*

$$\begin{bmatrix} A & b \end{bmatrix} = \begin{bmatrix} U & Q_x \end{bmatrix} \underbrace{\begin{bmatrix} \Sigma & m \\ 0 & ||q_x||_2 \end{bmatrix}}_{K} \begin{bmatrix} V & 0 \\ 0 & 1 \end{bmatrix}^T = \underbrace{\tilde{U}U'}_{\tilde{U}} \underbrace{\Sigma'}_{\tilde{\Sigma}} \underbrace{V'^T \tilde{V}^T}_{\tilde{V}^T} \tag{3.16}$$

with $K = U'\Sigma'V'^T$.

All of the additive modifications presented so far result in a rank enhancement and thus in a higher memory consumption. The integrated truncation of a tSVD in the additive modification is depicted below.

Remark 3.2.5 (Additive rank-c modification of a rank-t tSVD)**.** *Since $\tilde{\Sigma} = \Sigma'$ in Eq. 3.11 the rank-t truncation of an additive rank-c modification can be achieved by truncating K of Eq. 3.12 according to Def. 3.1.6. This yields*

$$\tilde{U} = \begin{bmatrix} U & Q_X \end{bmatrix} U'(:, 1:t), \tag{3.17}$$

$$\tilde{V} = \begin{bmatrix} V & Q_Y \end{bmatrix} V'(:, 1:t), \tag{3.18}$$

$$\tilde{\Sigma} = \Sigma'(1:t, 1:t). \tag{3.19}$$

Remark 3.2.6 (Re-orthogonalization of SVD)**.** *Since $\tilde{U}$ and $\tilde{V}$ are tall thin matrices, repeatedly rotating their column spaces in Eq. 3.9, 3.10 makes loss of orthogonality through numerical error an issue. This problem can be prevented by orthogonalization with means of QR decomposition before performing the SVD of K in Eq. 3.12 [5]. This yields to an modification of K of the form*

$$\begin{bmatrix} A & B \end{bmatrix} = Q_{\tilde{U}} \underbrace{R_{\tilde{U}} K_{old} R_{\tilde{V}}^T}_{K_{new}} Q_{\tilde{V}}^T \tag{3.20}$$

with $Q_{\tilde{U}} R_{\tilde{U}} = \begin{bmatrix} U & Q_X \end{bmatrix}$ and $Q_{\tilde{V}} R_{\tilde{V}} = \begin{bmatrix} V & Q_Y \end{bmatrix}$.

In conclusion, the additive column adding rank-1 modification of a tSVD is presented in algorithm 2 and additional remarks are provided about the memory consumption of the approximation by a tSVD and the computational cost of the additive modification.

Algorithm 2 Additive column adding rank-1 modification of a tSVD

Require: $U_i \in \mathbb{R}^{p\times k}$, $V_i \in \mathbb{R}^{q\times k}$, $\Sigma_i \in \mathbb{R}^{k\times k}$, $x \in \mathbb{R}^q$, $r \in \mathbb{N}$

```
if k = 0 then
    Σ_{i+1} ← ||x||_2
    U_{i+1} ← x/Σ_{i+1}
    V_{i+1} ← 1
else
    m ← U_i^T x
    p ← x − U_i m
    P ← p/||p||_2
    Q ← [U_i  P]
    K ← [Σ  m; 0  ||p||_2]
    if Q^T Q ≠ I then
        [Q, R] ← rQR(Q)
        K ← RK
    end if
    [U_temp, Σ_temp, V_temp] ← rSVD(K)
    t ← min(r, k + 1)
    Σ_{i+1} ← Σ_temp[1 : t :, 1 : t]
    U_{i+1} ← Q U_temp[1 : k + 1, 1 : t]
    V_{i+1} ← [V_i  0; 0  1] V_temp[1 : k + 1, 1 : t]
end if
```

Remark 3.2.7 (Memory Consumption of tSVD)**.** *Let the memory consumption be measured by the total number of entries of the matrices. Then the consumption of the rank-t truncated SVD with respect to approximated matrix $A \in \mathbb{R}^{p\times q}$ is given by*

$$MEM_{rank-t} = \frac{t(p+q+1)}{pq}. \tag{3.21}$$

Hereby, instead of storing the diagonal matrix Σ the singular values are stored vector-wise in $(\sigma_i)_i \in \mathbb{R}^t$.

Remark 3.2.8 (Computational Effort of tSVD)**.** *The main computational effort is in the following three operations. Firstly, the QR-similar decompositions to obtain Q_X and Q_Y take $\mathcal{O}((p+q)c^2)$. Secondly, the SVD of K in Eq. 3.12 has a complexity of $\mathcal{O}((t+c)^3)$. Thirdly, the rotations of the subspaces in Eq. 3.9 and Eq. 3.10 need $\mathcal{O}((p+q)(t+c)t)$ operations. Since $p, q \gg t > c$ the bulk of computational effort is needed for the subspace rotations.*

Chapter 4

Application to reduced Flow Field Model and Optimization

The previous chapter introduced the theoretical fundamentals of the approximation of a matrix by the truncation of the corresponding SVD and further an algorithm of its additive modification. In this chapter its application to unsteady fluid dynamics problems will be discussed.

For this purpose, this chapter is organized as follows. First, the general procedure of the construction of an approximation of the primal solution field using the incremental tSVD is outlined. As this work relies on the in-house finite volume solver FreSCo$^+$ for the numerical calculations, the procedure is tailored as well as possible to this circumstance. Afterwards, approaches to improve the performance are presented. These include an attempt for a more efficient update of the SVD, a possibility for adaptive rank determination and a parallelization technique suited for the solver FresCo$^+$.

4.1 Incremental Update of reduced Flow Field

This chapter provides first an insight into the storage of the full primal solution field. Based on this, it is revealed how the itSVD can be integrated into the optimization process to achieve a significant reduction of the memory consumption as outlined in Remark 3.2.7.

The storage of the full primal solution is equivalent to store the solution of all time steps. Based on the requirement of a time invariant grid with a constant number of flow variables, the storage can be realized by the so-called snapshot

matrix [2]. This matrix shall be defined in below for a primal simulation with q time steps.

Definition 4.1.1 (Snapshot Matrix)**.** *Let $a(t_n) \in \mathbb{R}^p$ denote the discrete solution vector of Eq. 2.1 at time $t_n = t_{n-1} + \Delta t_n$ with $1 \leq n \leq q$. Here, p corresponds to the total number of unknowns and q to the total number of time steps. The snapshot matrix $A \in \mathbb{R}^{p \times q}$ is then defined by*

$$A = \begin{bmatrix} | & | & & | \\ a(t_1) & a(t_2) & \dots & a(t_q) \\ | & | & & | \end{bmatrix} \tag{4.1a}$$

$$= \begin{bmatrix} a^1 & a^2 & \dots & a^q \end{bmatrix}. \tag{4.1b}$$

Remark 4.1.2. *Note that $a(t_n)$ contains the spatial solution of all physical unknowns in stacked format, eg.*

$$a(t_n) = \begin{bmatrix} u_1(t_n) \\ u_2(t_n) \\ u_3(t_n) \end{bmatrix} \tag{4.2}$$

with $u_i(t_n) \in \mathbb{R}^m$. Since in this work a cell-centered finite volume framework is used m refers to the total number of cells.

The snapshot matrix defined above is the result of a finished calculation, which already includes the solution of all time steps. However, since the entirety of the data is not available until the end of the computations, the snapshot matrix is formed incrementally by expanding it after each time step by the recently computed results. Let the snapshot matrix up to time step n be denoted as $A^n = \begin{bmatrix} a^1 & a^2 & \dots & a^n \end{bmatrix} \in \mathbb{R}^{p \times n}$. If the solution a^{n+1} of the subsequent time step is available, the update of the snapshot matrix is done by

$$A^{n+1} = \begin{bmatrix} A^n & a^{n+1} \end{bmatrix} \in \mathbb{R}^{p \times (n+1)}. \tag{4.3}$$

Overall, the procedure has a low extra computational effort. However, with respect to industrial applications which often involve small time steps and large computational grids [3] the snapshot matrix has a memory consumption too high to be stored in the main memory. Thus, the slower levels of the memory hierarchy must be resorted to such that the update might be inefficient due to memory bandwidth limitations caused by this circumstance [14]. Further, the memory requirements of the snapshot matrix might also exceed the maximum capacity of the system making it impossible to store the full snapshot matrix.

In the following, two possible approaches to circumvent the storage problem are briefly discussed. The first approach directly reduces the storage overhead by

saving not the whole solution but only at a small subset of c selected time steps, the so-called checkpoints [4], viz

$$A = \begin{bmatrix} a^{n_1} & a^{n_2} & \dots & a^{n_c} \end{bmatrix} \in R^{p\times c}\text{, with } \{n_i : i \leq c\} \subset \{n \leq q\}. \tag{4.4}$$

The neglected time steps are then reconstructed by restarting the primal calculation from the checkpoints [15]. If old checkpoints are no longer needed to be available for the adjoint problem they can be replaced by new checkpoints created in the recalculation process. On the one hand, the reduction of saved time steps leads to a significant reduction in memory requirements, but on the other hand this also increases the computational effort. This penalty can be reduced by minimizing the number of recalculations utilizing the binomial checkpointing algorithm proposed in [16]. In Summary, the checkpointing techniques trades computational effort for memory [14].

In contrast to checkpointing, the second approach aims to bypass the memory problem without the computationally expensive recalculations. Here, the snapshot matrix is not leaned by ignoring time steps in the storage process. Instead, the snapshot matrix is replaced by a low-rank approximation that fulfills the underlying memory requirements. According to the Eckart-Young Theorem 3.1.7 the optimal low-rank approximation of rank $t \leq p, q$ with respect to the 2- and Frobenius-norm is obtained by its rank-t truncated SVD

$$A = U\Sigma V^T \approx U_t\Sigma_t V_t^T = A_t \tag{4.5}$$

with $U_t \in R^{p\times t}, \Sigma_t \in \mathbb{R}^{t\times t}$ and $V \in \mathbb{R}^{q\times t}$. Thus the adjoint problem does not rely on the primal field stored in the snapshot matrix but on a low rank approximation of it. The impact of the approximation quality, controlled by the number of considered singular values, will be investigated in chapter 5.

In order to calculate the tSVD, standard algorithms require access to the associated matrix [3]. This would imply that for the memory reducing approximation the memory intensive snapshot matrix must be available. This contradicts the original idea. The method presented in the following is inherently inspired by the incremental updating of the snapshot matrix, but replaces the matrix from the start by the approximation. Thus, an incremental update step is carried out by

$$A^{n+1} = \begin{bmatrix} U_t^n \Sigma_t^n V_t^{nT} & a^{n+1} \end{bmatrix} \in \mathbb{R}^{p\times(n+1)}. \tag{4.6}$$

Consequently, the approximation is updated at each time step using the new solution of time step $n + 1$. However, the update shown in Eq. 4.6 is no longer a tSVD and does not benefit from its properties described in section 3.1. Here, this issue is obviated by relying on the findings of the additive modification of an SVD of section 3.2. The update of the approximation can be treated as a additive column adding rank-1 modification of a tSVD presented in Remark

3.2.4 and 3.2.5. The resulting update of the so-called incremental truncated SVD (itSVD) can then be computed by

$$A^{n+1} \approx U_t^{n+1}\Sigma_t^{n+1}V_t^{n+1^T} \overset{Alg.2}{\longleftarrow} \left[U_t^n\Sigma_t^nV_t^{n^T} \quad a^{n+1}\right]. \tag{4.7}$$

This incremental updating leads to the availability of the approximation of the snapshot matrix at the end of the calculation of the primal flow, without the necessity to buffer more than one column simultaneously. Under the assumption that this column needs a minor amount of memory the itSVD reduces the number of stored entries from pq to $r(p+q+1)$. This results in a mayor decrease of memory consumption if $r \ll p, q$.

After the reduced flow is stored in the form of the itSVD and thus not directly accessible, it must be recovered for adjoint computation. The complete extraction of the snapshot matrix would again lead to the known problems, hence an incremental evaluation is carried out by

$$a^n = A(:,n) \approx A_t(:,n) = a_t^n = U_t\Sigma_tV_t(n,:)^T \tag{4.8}$$

within a computational complexity of $\mathcal{O}(t(p+1))$ since Σ_t is a diagonal matrix. The overall resulting optimization framework based on the flow field approximation by means of the itSVD is depicted in algorithm 3.

Algorithm 3 Optimization Framework with itSVD

Require: Initial shape c^0
$i = 1$
while $i \leq i_{max}$ **do**
  $n \leftarrow 1$
  while $n \leq q$ **do**
    $a^n \leftarrow$ solution of primal problem in Eq. 2.1 at time t_n in c^i
    $U_t^n\Sigma_t^nV_t^{n^T} \overset{Alg.2}{\longleftarrow} \left[U_t^{n-1}\Sigma_t^{n-1}V_t^{n-1^T} \quad a^n\right]$
    $n \leftarrow n+1$
  end while
  $n \leftarrow q$
  while $n \geq 0$ **do**
    $a_t^n = U_t\Sigma_tV_t(n,:)^T$
    Solve adjoint problem Eq. 2.6 at time t_n in c^i
    $n \leftarrow n-1$
  end while
  Compute surface sensitivity according to Eq. 2.8
  Update shape to c^{i+1} using the steepest decent method
  $i \leftarrow i+1$
end while

In common to checkpointing, the idea of memory reduction through approximation by means of the itSVD also introduces extra computational cost. However, in this case, these costs are mainly incurred in the construction instead of in the recalculation of the primal solution. Therefore, they do not depend on the complexity of the underlying solver and the number of checkpoints as in the case of checkpointing, but purely on the costs associated with the itSVD itself. Remark 3.2.8 shows that these costs are in turn determined by the size of the snapshot matrix and the quality of the approximation expressed by the rank of the itSVD.

4.2 Extensions

This chapter presents possible extensions to the itSVD algorithm to further reduce the memory consumption or enhance the performance. In this context, the focus is on the construction of the reduced flow field since this determines the amount of storage needed and accounts for most of the computational effort.

For this purpose, the incremental updating is first investigated in more detail for its efficiency. Next, an adaptive rank determination is discussed allowing the minimal required memory consumption to be used for a given accuracy. Finally, a method to parallelize the itSVD with respect to the FresCo$^+$ parallelization procedure is outlined.

4.2.1 Incremental Bunch Update

The incremental rank-1 update of the truncated SVD in Eq. 4.7 appears to be at first the most intuitive approach in case of a time step based solution algorithm.

However, utilizing this ansatz the matrix-matrix multiplications in Eq. 3.17 and Eq. 3.18 have to be performed repetitively in each time step. Thus as prescribed in Remark 3.2.6 this repetitive matrix operations cause the method to be sensitive to numerical error unless appropriate countermeasures are taken. Besides this Remark 3.2.8 states that the main computational effort of the iSVD method results out of these matrix operations. Thus the rank-1 update can yield to a computational overhead.

Therefore, it is desirable to avoid or at least reduce these multiplications if possible. A strategy to achieve this is to reduce the total number of itSVD updates during the primal flow calculation. This can be accomplished by substituting b rank-1 updates by one rank-b update. For this purpose, the b successive solutions after the n-th update are buffered in the so-called bunch matrix B, which is defined as

$$B = \begin{bmatrix} a^{n+1} & \cdots & a^{n+b} \end{bmatrix} \in \mathbb{R}^{p \times b}. \tag{4.9}$$

The bunch matrix is then processed in one update according to

$$A^{n+b} = \begin{bmatrix} A^n & B \end{bmatrix} \approx \begin{bmatrix} U_t^n \Sigma_t^n V_t^{nT} & B \end{bmatrix}. \tag{4.10}$$

Similar to the rank-1 update, the result of this matrix expansion would not constitute an SVD and destroys the approximation property. But again, the update can be understood as the additive modification of the underlying matrix given by

$$\begin{bmatrix} A^n & B \end{bmatrix} \approx U_t^{n+b} \Sigma_t^{n+b} V_t^{n+b^T} \overset{Th.3.2.1}{\longleftarrow} \begin{bmatrix} U_t^n \Sigma_t^n V_t^{nT} & 0 \end{bmatrix} + B \begin{bmatrix} 0 & I \end{bmatrix}^T. \tag{4.11}$$

Thus the tSVD is only updated every b-th time step incrementally. The presented algorithm 4 here is executable for an arbitrary number of buffered solutions.

Algorithm 4 Incremental Bunch-b Update of itSVD

Require: $q, b \in \mathbb{N}$
1: $B \leftarrow [\,]$
2: **for** n=1,q **do**
3: $a^n \leftarrow$ solution of primal problem in Eq. 2.1 at time t_n
4: **if** #columns$(B) = b$ **then**
5: Update itSVD according to Eq. 4.11
6: $B \leftarrow [\,]$
7: **else**
8: $B \leftarrow \begin{bmatrix} B & a^n \end{bmatrix}$
9: **end if**
10: **end for**

Yet, for practical applications the choice of the bunch matrix size has a significant impact on the memory consumption and the execution time. Therefore this impact is discussed in the following. The main motive of the bunch update is to reduce the computational effort by decrease the frequency the itSVD algorithm has to be performed. For the investigation let t_1 and t_b denote the time needed for the construction of the whole approximation of the snapshot matrix with a rank -1 or rank-b scheme, respectively. Also let the average computation time to perform onbe incremental update be given by $\overline{t_1^i}$ and $\overline{t_b^i}$. Then the connection of the computational effort of both methods can be expressed by

$$t_b \approx \frac{t_1}{b} + \frac{q}{b}\left[\overline{t_b^i} - \frac{\overline{t_1^i}}{b}\right]. \tag{4.12}$$

Here, the first term of the right hand side (RHS) represents the performance enhancing that results of the decrease of the update frequency. The term decreases linearly with respect to the size of the bunch matrix. The second term indicates the penalty of carrying out a rank-b instead of a rank-1 update. The additional costs arise mainly in the calculation of the inner SVD of the matrix K Eq. 3.12. According to Remark 3.2.8, the complexity of that calculation is cubic with respect to b. As a consequence, the choice of a bunch matrix that is excessively large leads to the second term dominating and affecting the performance. Furthermore, a disproportionately sized bunch matrix also affects the memory requirements. The maximal storage requirement for b buffered solutions is given by

$$MEM_{\text{rank-b}} = MEM_{\text{rank-1}} + MEM(B) = \frac{t(p+q+1)+bp}{pq}. \tag{4.13}$$

In order to keep the influence of the bunch matrix on the memory requirements and the additional computational effort of the update as small as possible, the choice of $b \ll t$ is suggested.

4.2.2 Adaptivity Techniques

In the previous chapters the rank of the approximation was taken as granted, but in the subsequent chapter the choice of the rank will be discussed in more detail. According to Theorem 3.1.7, the number of singular values determines the accuracy of the approximation. In the following it is therefore the goal to select the optimal number of singular values in order to approximate the primal flow sufficiently well. In order to determine the quality of the approximation, error norms and heuristics are used. However, no a priori rules are available to specify an optimal rank before the simulation [17]. A naive approach is to compute an approximation with high accuracy and then reduce it according to the chosen error bounds. But this contradicts the main objective of memory efficiency of the previous chapters. Also a coarse simulation with larger time steps to record the general behavior of the flow field could be computed in order to estimate the rank based on these results. Yet, this requires additional computational effort and does not guarantee the choice of an optimal rank. In the following, an approach is presented that identifies the necessary rank of the SVD on the fly and thus does not require any prior knowledge about the desired flow field. According to this approach, the rank and thus the memory requirement is not known in advance, but this method can be applied universally and guarantees the compliance with previously defined error bounds under the assumption of no numerical error. The problem of the unknown memory demand can be solved by defining a maximal rank.

The approach exploits the fact that the update of the SVD is incremental. This allows to adapt the required number of singular values to the defined error norms in each step. In more detail, for a rank-1 update of a rank-t tSVD, this means that it is examined whether the updated tSVD requires t or $t+1$ singular values to satisfy the error bounds. This can be generalized for a bunch b-update by determine the rank l such that

$$\underset{t \leq l \leq t+b}{\operatorname{argmin}} \; \varepsilon(l) \leq \varepsilon_{bound} \tag{4.14}$$

where $\varepsilon(l)$ denote the error of the updated SVD with rank l and ε_{bound} the upper error bound. As this minimization is performed at each update, the approximation of the snapshot matrix is guaranteed to have the minimal required rank.

Subsequently, the error bounds are presented, that are used in this work for rank determination. It is important to note that although the focus is on comparing

the snapshot matrix to the approximation, this matrix is not directly available. Therefore, measures are chosen that do not require its direct presence. Additionally, it is mandatory that the computation of the error consumes only negligible computational effort. The first approach is to consider the relative error of the approximation given by

$$\varepsilon_1(l) = \frac{||A - A_t(l)||}{||A||} = \begin{cases} \frac{\sigma_{l+1}}{\sigma_1} & \text{, for } ||\cdot||_2 \\ \sqrt{\frac{\sum_{i=l+1}^{r} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2}} & \text{, for } ||\cdot||_F \end{cases} \tag{4.15}$$

where $A_t(l)$ denotes the approximation by the tSVD of rank l and r the rank of the snapshot matrix. Here, the main advantage is that both norms only requires the singular values to be available and not the matrix. However, only the first $t+b$ singular values are known during the update. Thus to use the Frobenius norm directly is rather impractical for the adaptivity method since $t+b \ll r$. Also the singular value σ_{t+b+1} which is needed to evaluate the spectral norm for $l = t+b$ is unknown. This is not an obstacle, since this evaluation is not necessary in practice. The reason is that if the error condition is not fulfilled for $l = t+b-1$, the maximal number of singular values, i.e. $l = t+b$, is automatically used.

A further approach that is often used in practice is based on the retained energy η presented in Def. 3.1.8. The basic idea is to observe the ratio of the energy of the approximation to the energy of the full system [17]. But the control of this ratio contradicts the minimization approach of Eq. 4.14, due to the fact that the retained energy rises with increasing number of singular values. Therefore, it is not the obtained energy that is considered in the following, but its complement the energy lost by the approximation. This is set by

$$\varepsilon_2(l) = 1 - \eta(l) = 1 - \frac{\sum_{i=1}^{l} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2} \tag{4.16}$$

where $\eta(l)$ denotes the energy retained if l singular values are used. Similar to the Frobenius norm, also in this error metric all r singular values are required to evaluate it. However, a significant decrease in the value of the singular values is assumed, so they are of minor importance. Consequently, the lower bound η_{lower} introduced in Remark 3.1.9 is applied to estimate the retained energy. This yields to

$$\varepsilon_2(l) = 1 - \eta(l) \geq 1 - \eta_{lower} = 1 - \frac{\sum_{i=1}^{l} \sigma_i^2}{\sum_{i=1}^{t+b} \sigma_i^2 + (r - (t+b))\sigma_{t+b}^2}. \tag{4.17}$$

This procedure guarantees that not less than the minimum number of required singular values are used, but does not exclude an overestimation. Given the previously made assumption of the singular value progression, the bound can lead to a sufficiently good result even with a small number of singular values.

Algorithm 5 Adaptivity Technique

Require: $U_t \in \mathbb{R}^{q \times t+b}$, $V_t \in \mathbb{R}^{q \times t+b}$, $\Sigma_t \in \mathbb{R}^{t \times t+b}$

```
l ← t
for i = t, t + b − 1 do
    if ε(i) > ε_bound then
        l ← i + 1
    end if
end for
U_t ← U_t(1 : p, 1 : l)
V_t ← V_t(1 : q, 1 : l)
Σ_t ← Σ_t(1 : l, 1 : l)
```

4.2.3 Parallelization Techniques

In this chapter, an approach for the parallelization of the itSVD is investigated. The goal is to align this routine with the parallelization of the associated finite volume solver, such that their interaction can be implemented as efficiently as possible. For this purpose, the parallelization approach of the solver is presented in short.

The FresCo$^+$ algorithm is parallelized using a domain-decomposition technique based on a Single Instruction Multiple Data (SIMD) message-passing model. So every process executes the same program instructions on its subset of the domain. The algorithm also provides a load balance. Consequently, the solver divides the domain Ω into k disjoint partitions Ω_i such that

$$\Omega = \bigcup_{i=1}^{k} \Omega_i \text{ , with } |\Omega_i| \approx |\Omega_j| \text{ and } \Omega_i \cap \Omega_j = \emptyset \; \forall i, j \leq k. \tag{4.18}$$

Here it is assumed that the number of processes coincides with the number of partitions.

A combination of this procedure with the serial implementation of the incremental update by means of the itSVD would require that the necessary data of all processes must be sent to one process, where the tSVD is then updated. Depending on the number of processes and the size of the data, this would lead to a throughput overhead. A parallelization approach that builds on this ansatz and parallelizes by distributing the data back to the processes will further exacerbate this problem. Here the communication between the processes is the limiting factor.

Instead, an approach is presented that, like the solver itself, is based on the idea of the SIMD architecture and eliminates the need for communication between the

processes during the update of the reduced flow. For this, a dedicated snapshot matrix A_i is defined for each partition Ω_i by

$$A_i = \begin{bmatrix} a_i^1 & a_i^2 & \cdots & a_i^q \end{bmatrix} \in \mathbb{R}^{p_i \times q} \tag{4.19}$$

where a_i^n with $1 \leq n \leq q$ and $1 \leq i \leq k$ contains the primal solution for all finite volumes in Ω_i. Further p_i denotes the number of cells in Ω_i with $\sum_{i=1}^{k} p_i = p$ and $p_i \approx p/k \ \forall i \leq k$.

Rather than approximating the snapshot matrix A, the approximation of each matrix A_i can then be constructed independently according to the update routines presented in chapter 4.1 and 4.2.1. This yields to

$$A_i \approx A_{ti} = U_{ti} \Sigma_{ti} V_{ti}^T \tag{4.20}$$

with $U_{ti} \in \mathbb{R}^{p_i \times q}$, $\Sigma_{ti} \in \mathbb{R}^{t \times t}$ and $V_{ti}^T \in \mathbb{R}^{q \times t}$. The index i in the approximation indicates the affiliation to the partition. Overall, this translates to the replacement of the computation of the approximation of the matrix $A \in \mathbb{R}^{p \times q}$ by k computations of the approximations $A_{ti} \in \mathbb{R}^{p_i \times q}$.

This distribution and the circumstance that the approximations A_{ti} can be computed simultaneously on the associated processes have a significant impact on the computational effort required by the method. Hence, the modification of the number of rows of the matrix A_t from p to $p_i \approx p/k$ for A_{ti} leads to that the matrix U_{ti} also has approximately p/k rows. This has the consequence that the effort of updating the matrix U_{ti} from Eq. 3.17 decreases linearly with k. According to Remark 3.2.8 and the assumption that the number of finite volumes significantly exceeds the number of time steps, this operation occupies the main share of the computational effort and consequently a considerable performance increase is to be expected. If it is further expected that due to the reduced size of the matrix to be approximated fewer singular values are needed, this additionally decreases the computational effort.

Besides, this kind of parallelization of itSVD also impacts the memory consumption. Since this is the main objective of the overall approximation approach, a significantly higher memory consumption would be unacceptable and would severely constrain the method. The memory consumption in form of the total number of stored entries is in the parallel scenario given by

$$MEM_{parallel} = \sum_{i=1}^{k} t(p_i + q + 1) \overset{p_i \approx p/k}{\approx} t(p + k(q+1)) \tag{4.21}$$

since the spatial domain can be split into partitions but the time domain cannot. Given the previously established assumption that the number of finite volumes significantly exceeds the number of time steps $p \gg q$, the additional memory

overhead is a reasonable trade-off for the high performance benefits. Further the application of the adaptivity techniques presented in chapter 4.2.2 to each subdomain approximation can yield to an optimal exploitation of memory.

Chapter 5

Numerical Results

In this chapter the results of the application of the incremental truncated SVD to the construction of a reduced flow field are discussed. In this context, the impact of this approach on the optimization process is examined in more detail. For this purpose, the approximation algorithm was integrated into the in-house finite volume based solver FresCo$^+$. The solver is capable of solving both the primal and adjoint equations on structured and unstructured meshes and moreover can deal with multiphase flow problems. In addition, it has built-in routines to calculate the shape gradient and the mesh deformation for shape optimization based on gradient-based methods.

The tSVD construction was implemented into the solver routine with minimal interfaces such that their functionalities are not affected. According to Algorithm 3, the only interaction is the reading and writing of the primal flow. Beyond that, the algorithm acts independently.

5.1 Flow Field Reconstruction

In this section, the itSVD algorithm is tested on the laminar flow around a circular cylinder. This is a well documented scenario and the expected periodic behavior for a sufficiently large Reynolds number is ideal for testing the itSVD approximation approach to the unsteady adjoint-based optimization [3]. Thereby, the time periodic von-Karman street is developed, inducing a periodic force on the cylinder surface.

In this study, the inflow velocity $u_1 = 0.2\,\mathrm{m/s}$, the kinematic viscosity $10^{-5}\,\mathrm{m^2/s}$ and the diameter of the cylinder $D = 0.01\,\mathrm{m}$ were chosen such that the Reynolds number $\mathrm{Re} = 200$ is obtained. For the simulation a 2D computational grid is used, which covers the physical domain $[-100D, 100D] \times [-50D, 50D]$. Here, the

center of the cylinder is $[0\,\mathrm{m}, 0\,\mathrm{m}]$. In order to observe the physical phenomena, the area $[-5D, 30D] \times [-3.5D, 3.5D]$ is resolved finer. In addition, the immediate area around the cylinder receives further refinement, as this is the area of greatest interest with respect to upcoming optimization. This yields that per time step approximately $p \approx 3.3 \cdot 10^5$ data points have to be stored.

In contrast to the spatial domain, the time domain is resolved equidistantly with a time step size of $\Delta t = 10^{-3}$ s. For the simulation of 10^3 time steps this results in the time interval $[0\mathrm{s}, 1\mathrm{s}]$. However, since the periodic behavior of the flow is developed over time, prior to the simulation another $5 \cdot 10^3$ time steps are calculated, allowing the periodic behavior to evolve. Subsequently, the result of the last time step is used as the initial condition for the simulation. Thus, merely the desired periodic behavior is captured by the itSVD.

The following sections deal with the numerical results of the approximation approach proposed in chapter 4.1. For this purpose, the method is first examined in general terms for approximation quality and performance with respect to the memory consumption. Afterwards, the effects of the extensions proposed in the chapter 4.2 are examined, also for the same criteria.

5.1.1 Approximation Quality

This section investigates the numerical results of the approximation of the primal velocity field using the rank-1 itSVD method and benchmarks it against the exact representation. The number of finite volumes was chosen such that the memory requirement of the full solution does not exceed the maximum capacity of the system. Regarding the storage, it should be noted that due to the size of the snapshot matrix, it cannot be stored in the main memory. Thus, accessing the snapshot matrix takes more time than accessing the approximation, which is stored in the main memory. In the following, the effects of the memory consumption on the approximation method are examined. Here it is to be emphasized that the memory consumption is proportional to the number of singular values. In this case, this is reflected by the fact that 1% memory consumption corresponds to approximately 10 singular values.

The first insight into the quality of the approximation is obtained due to the comparison of the primal and the reconstructed reduced primal field. Therefore snapshots of the fields are presented in Fig. 5.1 at $t = 0.5$ s and $t = 0.9$ s. Additionally, the absolute approximation error per finite volume is shown. The used approximation requires approximately 2% of the memory of the exact field.

The results indicate that the approximation can replicate the general velocity profile with high accuracy at both time points. Here, the vortex shedding

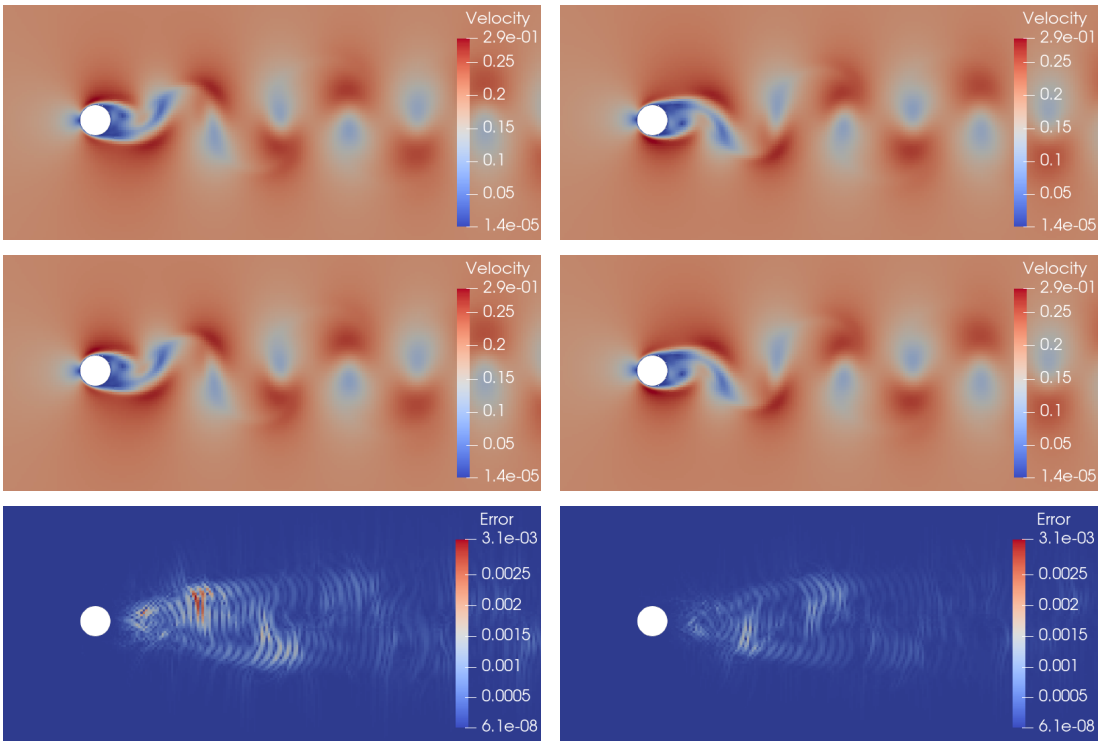

Figure 5.1: The primal flow field (top) in contrast to the reduced primal flow field (mid) and the absolute error made per finite volume (bottom) for $t = 0.5$ s (left) and $t = 0.9$ s (right).

characteristic for this scenario is reproduced in detail by the reduced model. A difference of both flows is first revealed by the absolute error plot. This illustrates that the main part of the inaccuracy of the approximation occurs in the representation of von-Karman vortex street, while the far field has a negligible error most of the time. Indeed, this is due to the fact that the far field is almost constant in time and thus can be represented with high accuracy with few singular values. The vortex street, on the other hand, exhibits an unsteady behavior, requiring more singular values to fully capture this time-varying nature. It is due to the periodicity of the vortex shedding that the behavior can be reproduced successfully with a small amount of singular values.

After having illustrated the general behavior of the low-rank reduced flow field, the approximation property is examined in the following by comparing the snapshot matrix A and its approximation $A_t = U_t \Sigma_t V_t^T$. Here, the development of the relative error when changing the memory requirement of the approximation is relevant. The results of the investigation are depicted for the spectral and Frobenius norm for a memory requirement up to approximately 15% in

Fig. 5.2 (left). The courses of both norms exhibit a similar behavior, whereby the errors decrease sharply at the beginning, but the trend starts to flatten out starting at a memory consumption of approx. 4-5%. This implicates that an approximation with a relative error of the order of magnitude 10^{-4} can be obtained with a low memory consumption and thus can adequately replicate the primal flow. However, the error profile also reveals that if higher accuracy is required, disproportionately more memory is needed to achieve it.

The second measure to evaluate the quality of the approximation is the amount of lost energy depending on the memory consumption. The result is presented in Fig. 5.2 (right). This evaluation confirms the observations of the relative error. Also here the steep decrease at the beginning and the reduction of the slope at a memory consumption of about 5% can be observed. At a memory consumption of 5%, 99.99% of the total energy is obtained. A further increase of the obtained energy leads here also to a highly enhanced memory load.

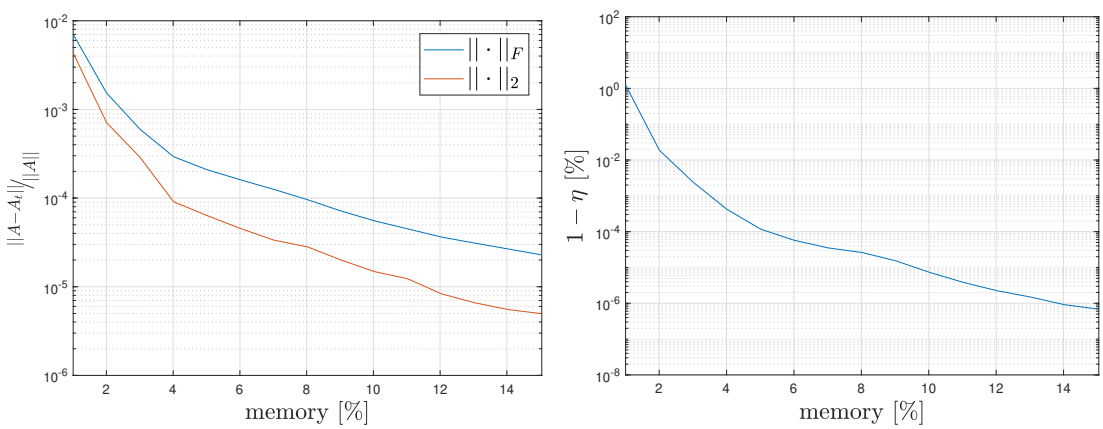

Figure 5.2: The relative error of the approximation in spectral and Frobenius norm and the amount of lost energy by the approximation with respect to memory consumption.

Besides the evaluation of the approximation of the snapshot matrix, the analysis of the tSVD resulting from the incremental approach is also of interest. This is justified by the fact that the incremental updating procedure makes the construction sensitive to numerical errors. If this caused a significant deterioration of the tSVD, the optimal low-rank approximation postulated by the Eckart-Young Theorem 3.1.7 could not be guaranteed. Therefore, the deviation of the incremental tSVD from the tSVD calculated from the snapshot matrix is investigated. For this purpose, Theorem 3.1.2, proving the uniqueness of the singular values of a matrix, is used and the difference between the singular values of both tSVDs is considered. Fig. 5.3 shows on the left side an exemplary representation of the singular values for an approximation with 150 singular values of both tSVD.

The comparison reveals that the singular values are almost identical, with the exception of the smallest singular values. Here, those of the tSVD are smaller than those of the tSVD.

Beyond the examination of the variation in the singular values, the general development of the singular values can provide a foundation for understanding the error curves shown in Fig. 5.2. The course of the singular values displays the already known course of an initial sharp decrease, which flattens out in the further progress. By means of the energy definition this can be interpreted that the first singular values contain this major part of the total energy and thus determined substantially the approximation behavior. The following sharp decrease suggests that the subsequent singular values contain a low amount of energy and therefore provide a moderate contribution to the approximation quality. Thus the inaccuracy in the smallest singular values is relatively insignificant. However, it should be noted that this can affect the evaluation of the error norms and the retained energy.

In addition, the relative error of the singular values with respect to the size of the tSVD is given on the right side in Fig 5.3. It is calculated as the error of the singular value vectors defined by $\sigma_{SVD} = (\sigma_i)_i \in \mathbb{R}^t$ in the Euclidean norm. The evaluation indicates that the error decreases with increasing size of the itSVD. Also in this case, the error decreases steeply initially, followed by a gradual decline. This indicates that the sensitivity to numerical errors of the itSVD is directly related to its size. The results of the singular value progression can justify it. It was pointed out exemplarily that the itSVD and the tSVD distinguish in their smallest singular values. Therefore, the more singular values an approximation considers, the smaller are the differing singular values and thus the less is its effect on the error and vice versa.

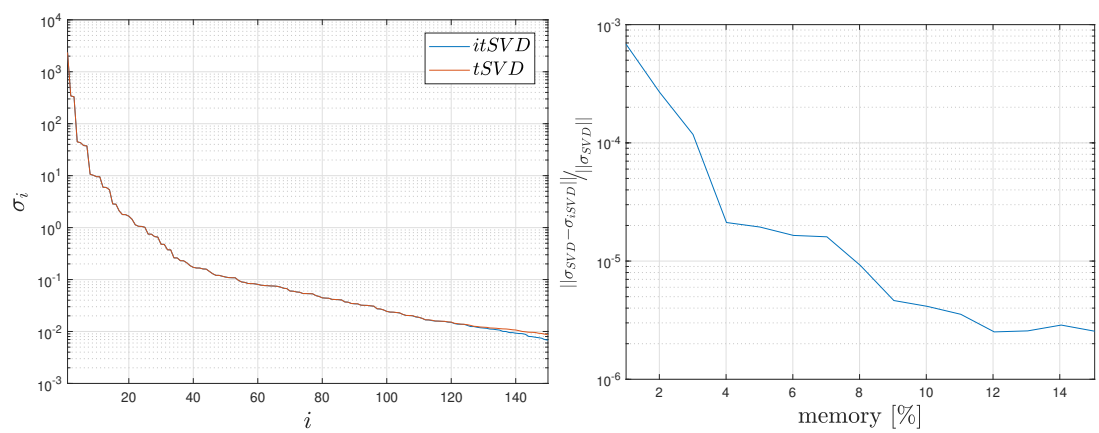

Figure 5.3: Comparison of the singular values of itSVD to those of the ordinary tSVD.

5.1.2 Performance

After having previously investigated the approximation property of the itSVD, this chapter tests the itSVD algorithm for its computational cost. For this purpose, the relative additional computational effort that arises when the itSVD is performed instead of storing the full snapshot matrix is examined. The extra computational effort is determined by the time factor

$$TF = \frac{t_{itSVD}}{t_{full}}, \tag{5.1}$$

where t_{itSVD} and t_{full} denote the execution times of the corresponding cases. Since the snapshot matrix is fixed t_{full} is constant while t_{itSVD} varies with respect to the rank of the approximation. In order to provide comparability of the results, all computations were performed on the same computer. In addition, each test was performed 5 times to reduce the effect of external influences.
The result of this investigation is illustrated in Figure 1. The development of the time factor reveals a polynomial growth with increasing memory requirements of the itSVD. Already a memory requirement of the SVD of 5% causes a duplication of the runtime. At the maximum, the computation time is increased to a factor of more than eight for 15% memory consumption. This demonstrates that the itSVD loses its characteristic of an efficient approximation construction rapidly and is only conditionally suitable for practical applications. In applications that require a higher resolution approximation, relying on alternatives such as checkpointing might be appropriate. Therefore, the extension of the bunch updated itSVD, which promises a more efficient approach, is discussed below.

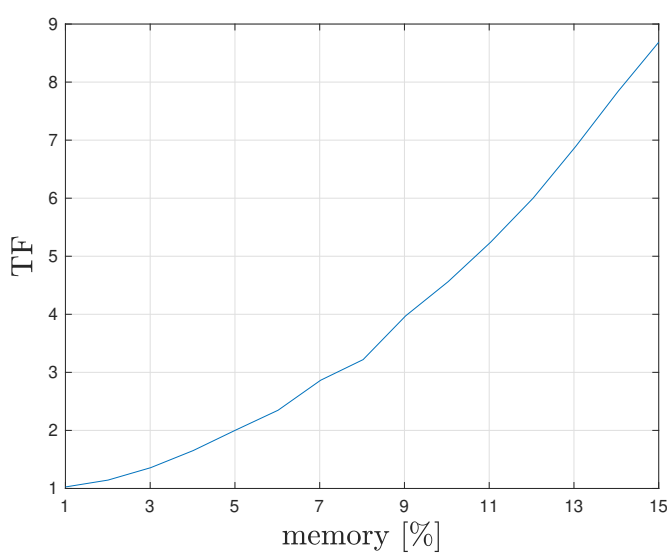

Figure 5.4: Performance of the itSVD with respect to its memory consumption.

5.1.3 Incremental Bunch Update

The incremental bunch update was introduced as an extension of the itSVD algorithm in chapter 4.2.1. It was motivated due to the reduction of the numerical error and the computational effort of the method in comparison to the classical rank-1 update. The objective of this chapter is the verification of these statements.

Therefore, the error development for different bunch sizes are analyzed first. This is followed by a review of the computational effort. The bunch sizes $b \in \{1, 2, 5, 10\}$ have been considered.

The relative error of the approximations with respect to the bunch size is pointed out for different memory requirements in Fig. 5.5. The plot of the Frobenius norm (left) as well as of the spectral norm (right) implicate that the bunch update preserves the general error pattern, but a decrease of the update frequency reduces the relative approximation error. Yet, this reduction is small and the quality is only improved marginally. Considered in isolation, this provides no incentive to apply the bunch update.

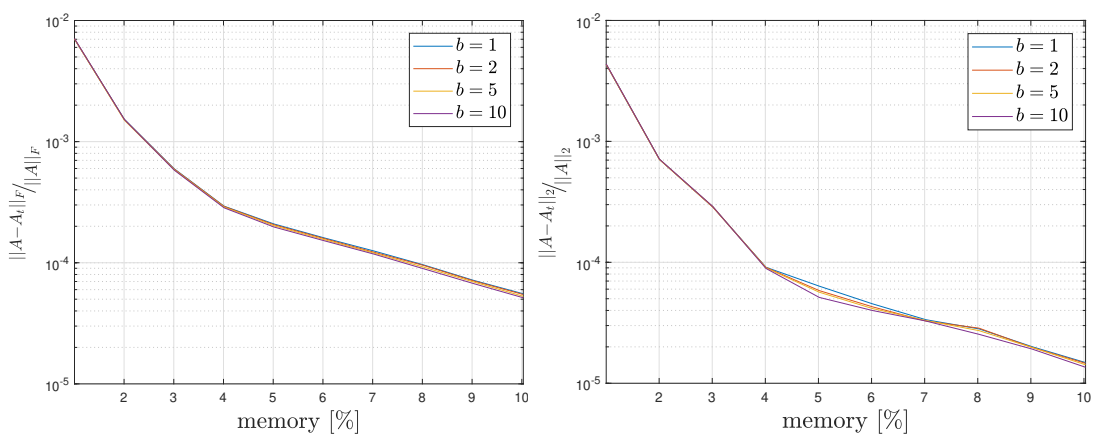

Figure 5.5: Relative Error of approximation with respect to bunch size.

But besides the relative error, the amount of lost energy and the quality of the tSVD, measured by the relative error of the singular values, are also relevant for the evaluation. Thus Fig. 5.6 illustrates these quantities for different bunch sizes as a function of memory. The trajectories of the lost energy indicate that the choice of a update with larger bunches influences the error course only by small variations. In contrast to the relative error, where a larger bunch reduces the error, in this case it causes an increased amount of lost energy. This variation is inherent in the observation that increased bunch sizes mostly lead to a decrease in the error of the singular values. The error plot of the singular values shown

in Fig. 5.6 (right) proves this behavior. As a result, the underestimation of the singular values due to itSVD revealed in Fig. 5.3 diminishes and yields a more accurate estimate of the lost energy. This also suggests that, in the case of an adaptive approach, the bunch update could translate into better maintenance of the energy limit. The error of the singular values in Fig. 5.6 shows that in most cases there is a reduction of the error with an increase of the bunch size, but the error progression for each bunch size is not monotone. Further it is observed that the error in singular values is higher for big bunch updates when a low amount of energy is used.

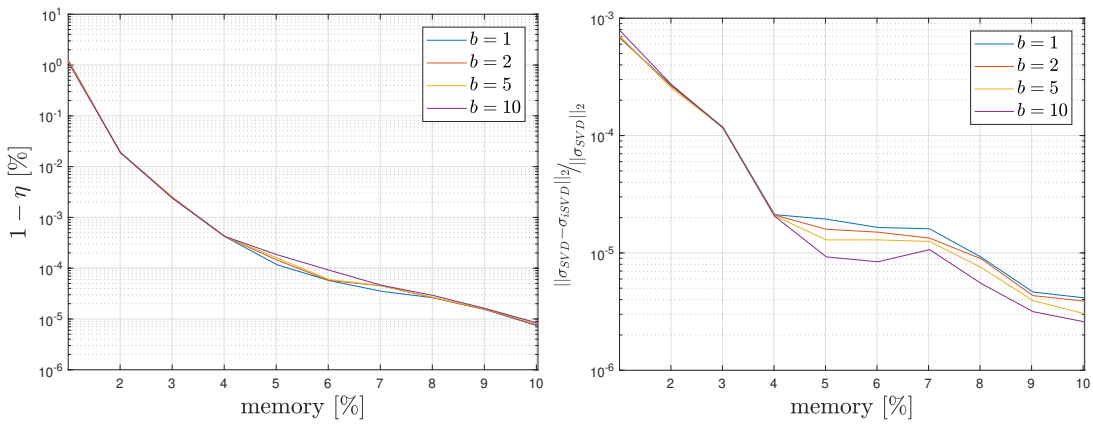

Figure 5.6: Lost energy and relative error of singular values with respect to bunch size.

Overall, it has been shown that a bunch update can marginally reduce the error. However, this alone does not give much incentive to its application. For this reason, the performance depending on the bunch size is investigated in the following.

According to the definition in Eq. 5.1, the performance of the bunch update is determined by the time factor. The test setup is identical to that of the classical itSVD algorithm. Fig. 5.7 presents the results of the bunch updated itSVD in relation to the memory usage. The effort of computing an itSVD with moderate memory requirements can be performed by all tested update sizes with minimal additional effort. In this interval, no bunch size can be identified as preferable to the others. With increasing size of the itSVD, the significantly deviating progressions implicate a distinct dependency of the computation time on the size of the bunch matrix. Furthermore, the performance developments for the tested bunch sizes confirm the relationship postulated in Eq. 4.12 between the computation time of the classical and the bunch update. Thus, the time required for the larger update prevents the computation time from being reduced linearly and decreases the performance gain for larger bunch updates.

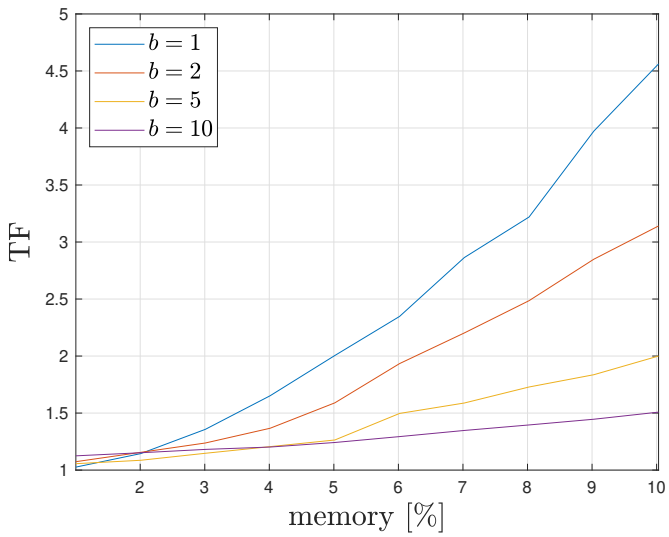

Figure 5.7: Performance of the itSVD with respect to bunch size.

In summary, the bunch updated itSVD provides a most effective method for reducing the computational time to construct the approximation. In contrast to the classical itSVD, the approach yields practically reasonable computation times and is classified in this work as an essential extension of the itSVD that ensures efficient computation. The additional improvement in approximation quality should also be emphasized, but its contribution must be considered minor.

5.1.4 Adaptivity Techniques

In this section, the adaptive extension of itSVD presented in chapter 4.2.2 is applied to the test problem and based on the results, its benefits are discussed.

Here, this investigation is limited to the consideration of a single error bound such that the adaptive behavior of itSVD is uniquely determined by it. Furthermore, setting multiple bounds on a prior unknown problem provides an increased potential that resources will be wasted. It is not known a priori how various bounds are related to each other, thus an intuitive specification is difficult. Therefore, only a bound for the amount of lost energy is set, since it can provide an intuitive interpretation for the approximation of the physical problem. A more relaxed bound for the relative error in spectral norm could also be set to guarantee a minimum level of approximation quality. However, this is not done in this work.

The adaptive approach is characterized by determining the optimal number of singular values to satisfy the error bound in each update step. Therefore, the evolution of the number of singular values with respect to the time steps is in-

vestigated in the following. Fig. 5.8 shows the results for several energy bounds. Here it is still valid that 10 singular values correspond to approximately 1% memory.

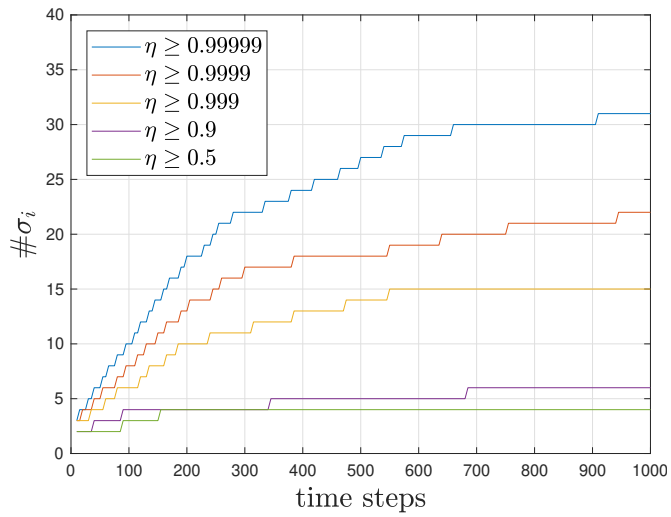

Figure 5.8: Evolution of the size of the itSVD with respect to energy bound.

The results indicate that if a minimal amount of energy is demanded, the size of the itSVD is variable over the entire time course and is subject to frequent adjustments. The progression can be approximately divided into 2 phases. At the beginning of the calculation, setting a high energy bound results in a rapid increase of considered singular values. But this growth decreases with time and after a certain point of time the number of considered singular values remains almost constant in the further course. The more demanding the energy requirement, the later this behavior occurs. This behavior in the second phase is due to the periodicity of the model problem, since after a certain time step the newly calculated solution vectors can be sufficiently well approximated by the already existing itSVD, since similar information has already been processed earlier. In case of a high energy conservation already a small deviation of the new solution vector can violate the prescribed limit, such that an increase of the rank must take place in order to comply with it and thus the consideration of almost constant number of singular values occurs later.
Besides the temporal course of the number of singular values, the results show large differences of the total number of singular values with respect to the energy requirements. It can be concluded that maintaining a 90% bound is slightly more demanding than maintaining a 50% threshold and achieving an increase of the minimum energy by 40% requires 50% more singular values. On the other hand, the effort required to achieve an enhancement in the high energy range increases strongly. An improvement starting from retaining 99.9% of the energy

by 0.099% results here in a doubling of the required singular values. Thus, the previously described fact that the largest part of the total energy is stored in a few singular values and that the following ones have a diminishing influence can be demonstrated.
It was shown that the choice of the energy bound has a significant influence on the number of singular values, especially at high boundaries, and that small changes can have large effects. In the following it is examined how this affects the relative error of the approximation. Therefore, Fig. 5.9 presents the relationships between the energy bound and the relative approximation error.

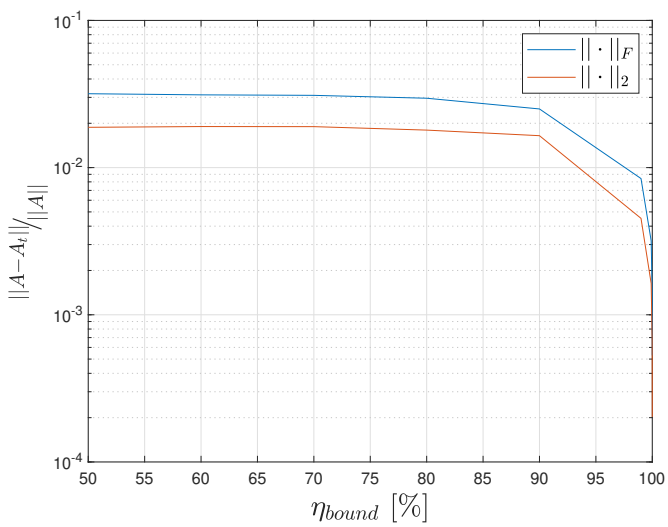

Figure 5.9: Relative error of approximation with respect to energy bound.

The evaluation reveals that the error in both the Frobenius and the spectral norm is practically constant up to an energy boundary of 90%. Thereafter, it commences to decline. The largest reduction takes place above 99% of the obtained energy. Hence, it is recommended from a practical point of view to set energy thresholds that allow at most the loss of a few percent of the energy.

In conclusion, the analysis indicated that the adaptive algorithm provides a method to satisfy an energy constraint in a memory-efficient fashion without prior specification of the rank of the tSVD. It was shown how the approach benefits from the periodicity of the flow field. Also, a relationship between the amount of retained energy and the relative error could be described for this test case. Nevertheless, it can be expected that this relation is problem specific and thus, when applied to further problems, the choice of an appropriate energy bound can pose a challenge.

5.1.5 Parallelization Techniques

The parallelized itSVD algorithm is examined in this section. The focus is on the impact of the number of partitions on the memory requirement, the approximation error and the computation time. Since the primal solution of each partition is approximated by a separate itSVD and the solution behavior varies across partitions, it is expected that if a fixed rank is chosen for all partitions, the approximation quality will also differ. This yields an inefficient exploitation of the available storage capacity. Therefore, the adaptive rank determination is applied. The results shown in section 5.1.4 implicate that an energy bound of a minimum of 99.99% has a sufficiently good approximation quality. Hence, this bound is applied here.

The influence of the number of assigned partitions on the memory consumption and the approximation error are presented in Fig. 5.10. Here on the one hand the mean value and on the other hand the quantities for each partition are displayed. The partition evaluations are marked by diamonds. In the representation an overlapping can occur, such that the visible number of evaluations may differ from the number of partitions.

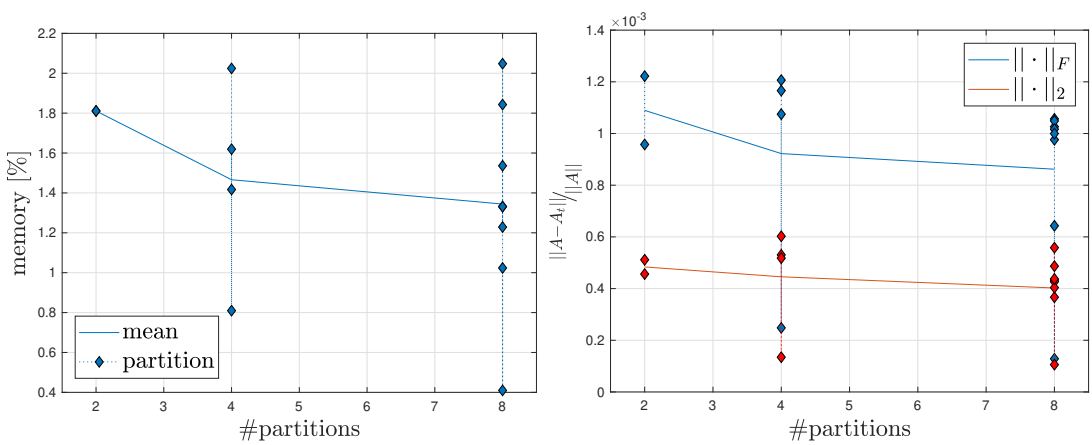

Figure 5.10: Memory consumption and relative approximation error with respect to the number of partitions.

The presented mean memory consumption implicates that splitting the domain into several partitions with the same requirement for the amount of retained energy promotes a memory reduction. The increasing deviation of the individual evaluations from the mean value when multiple partitions are considered provide an indication of a potential source for this phenomenon. As shown in Fig 5.1, the complexity of the primal velocity field in the computational domain varies in different areas. Progressively splitting the domain into partitions benefits these

complexity differences to be accounted for in each itSVD. Thus, a simple domain can be approximated with low memory requirements and a memory intensive approximation can be set up for a more complex domain. As the result demonstrates, this yields an altogether lower memory consumption on average.
A similarly positive result can be observed for the error trend with an ascending number of partitions. Taking into account the results from the memory investigation, it can be concluded that the increase in the number of partitions reduces the relative approximation error, while less memory is required. There are two potential reasons for this behavior, one could be the different relation of the energy bound to the relative error per partition as described in section 5.1.4. The other is that due to the partitioning the orthogonal matrices of the itSVDs have fewer rows such that according to remark 3.2.6 the update procedure is less sensitive to numerical errors.

Besides the advantages of the parallelized itSVD in connection with the adaptive rank determination regarding memory and error reduction, the impact on performance has to be considered as well. In contrast to the previous performance studies in which all configurations were benchmarked against the constant execution time of the solver, it is now variable and scales like itSVD with the number of partitions. Additionally, it needs to be emphasized that for saving the full solution, the dedicated snapshot matrices are saved individually like itSVD and incrementally updated in parallel. The size of the approximations allows them to be stored in main memory, while the exact snapshot matrices must be stored on hard disk. Again, the performance is interpreted according to the time factor defined in Eq. 5.1. The evaluation is shown in Fig. 5.11 with respect to the number of partitions.

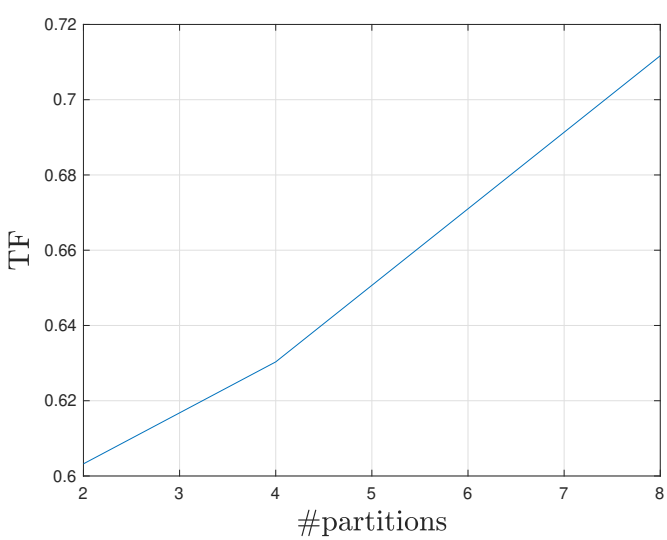

Figure 5.11: Performance of the parallelized itSVD with respect to the number of partitions.

In contrast to the previous performance studies, in this case the time factors are consistently less than 1. This implicates that the incremental construction of the approximation takes less time than storing the full solution. Since the approximation definitely requires more computational effort than storing the full solution, the read and write operations of the snapshot matrices stored on the hard disk can be identified as limiting factors of this approach. Thus, the replacement of memory by arithmetic operations yields to an increase in performance in terms of the time factor.
In addition, the observed rise of the time factor with more partitions is due to the fact that the solver of the primal problem scales better with the number of partitions with respect to the execution time than the parallelized itSVD algorithm.

5.2 Optimization

In this section, the results of the application of the itSVD approach to the unsteady adjoint-based shape optimization are presented and discussed. The optimization is carried out based on the procedure outlined in chapter 2. The objective is to minimize the drag force utilizing the cost functional described in Eq. 2.3. The itSVD is integrated into the solution process of the primal and adjoint equations according to algorithm 3. Thereby, all three extensions of the itSVD described in section 4.2 are employed for the incremental construction of the approximation. The encountered parameters are chosen problem specific.

In this work, the approach is tested for two scenarios of different complexity. First, the shape optimization is applied to the cylinder of the two-dimensional model problem introduced in chapter 5.1. Since this model was chosen to be compact, in this case the optimization based on the reduced primal field obtained by the itSVD can be benchmarked against the optimization using the full primal solution. The second test case is intentionally chosen to be more complex in order to investigate the applicability to more realistic problems and to identify possible limitations of the itSVD approach. Therefore a turbulent two-phase flow around a 3D sphere is considered.

5.2.1 Laminar Flow around 2D Cylinder

Within this section, the shape optimization of a 2D cylinder using the reduced primal velocity field is investigated. As a starting point, the scenario presented in Section 5.1 is selected. This is a very suitable case for a first test of the developed approach, because in addition to the data about the behavior of the reduced flow, general information about the unsteady shape optimization exists as well. Here, it is referred to the findings of [6]. These findings can serve as verification of the unsteady fluid flow shape optimization process used here. This is necessary, since the optimization routine of FresCo$^+$ was originally designed for steady and pseudo unsteady problems [1, 8], such that it had to be modified for this work.

According to algorithm 3, one step of the optimization can be divided into three subsequent subproblems. First the primal problem is solved and its solution or an approximation is stored, then the adjoint solution is calculated using this information, and finally the shape is updated. In the following, the configuration of each individual step is described.
In this case, the Navier-Stokes equations are solved in parallel by means of a domain decompositioning to 8 partitions for 1000 equidistant time steps in the time interval $[0s, 1s]$. The incremental creation of the reduced velocity field by the itSVD included in this process is thus also applied to each partitions individually

according to the parallelization approach of section 4.2.1 by using the bunch size $b = 10$ in order to provide an efficient computation of the approximation also in case of a larger itSVD. Furthermore, the approximation quality is adaptively controlled by the amount of retained energy. The results of the reduced flow field investigation showed that the maintenance of 99.99% of the energy represents a reasonable trade-off between quality and memory consumption. In addition, the optimization performance for poorer quality approximations has been studied to highlight possible weaknesses if the adaptivity parameter is poorly selected.
Once the primal computation is complete and the reduced solution is available, the adjoint problem can be solved. This has to be carried out applying the same parallelization properties and the same grid of the primal computation to ensure an unique mapping of the reduced primal solution. Then, the adjoint problem can be computed backwards in time by evaluating the itSVD at the needed time point. Here, the initial condition of the adjoint problem must be taken into account. In contrast to the primal calculation, where the condition is determined by a prior simulation allowing the periodicity to evolve, the initial adjoint velocity is set to $\hat{u} = 0$ at $t = 1s$. This leads the formation of the periodic behavior to takes place in the considered simulation period. However, the optimization should only consider the cyclic pattern of the flow such that this phase is neglected in the optimization process. Therefore, it is not accounted for the shape update by excluding it from the sensitivity calculation. In the present case, the examination has shown that a periodicity in the adjoint velocity field begins after 200 time steps, hence the sensitivity is calculated in the interval $[0s, 0.8s]$. However, due to the periodic behavior of the primal flow, it can be expected that the drag force will still be minimized over the whole time interval.

Subsequently, the deformation field can be determined and the shape updated with respect to the resulting sensitivity. Here, in order to avoid larger deformations of the mesh and the geometry, the received deformation d_i is scaled to a user defined maximal deformation $\tilde{d}_{max}$. This is done according to [8] by the substitution of d_i by

$$\tilde{d}_i = \frac{d_i}{max(d_i)} \tilde{d}_{max}. \tag{5.2}$$

In this study $\tilde{d}_{max} = 10^{-4}m$ is chosen which corresponds to a maximal deformation of 1% with respect to the diameter of the initial shape.

The results of this optimization approach are presented and analyzed below. For this purpose, the identical optimization is performed for different qualities of the primal flow field. Besides the application of the exact flow field in form of the snapshot matrix, the approximations with a minimum of retained energy of 50%, 75% and 99.99% are used. The exact flow field is described by a perfect conservation of energy with $\eta = 100\%$.

The impact of the approximation quality on the minimization of the cost functional and the memory requirements of each approximation during the optimization process are illustrated in Fig. 5.12. The development of the cost functional is indicated by the decrease in percent compared to the initial shape.

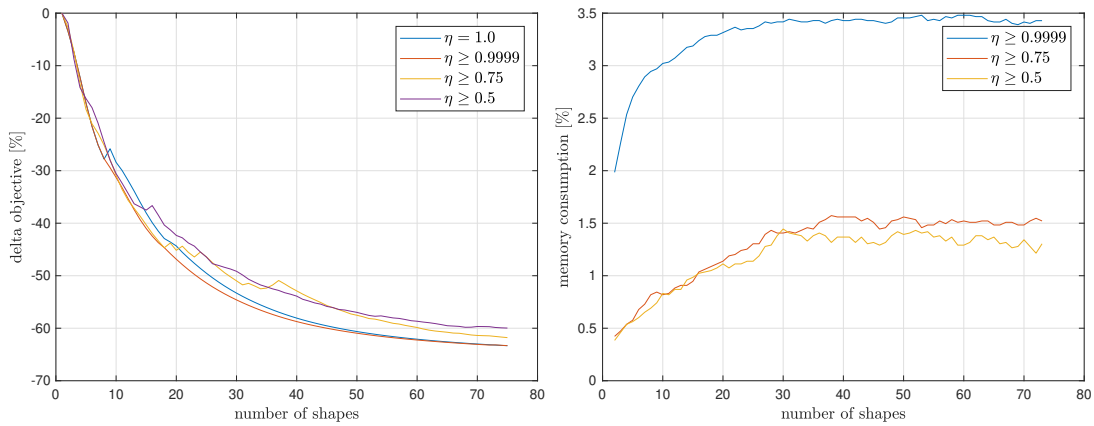

Figure 5.12: Course of the cost functional and the memory consumption during the optimization with respect to retained energy of approximation.

First, the exact optimization is examined. Here, the cost functional exhibits a strong initial reduction, which weakens with increasing shape updates. At the end of the iterations, there is no further relevant decline of the cost functional, such that the optimization is considered to be converged at a reduction of approximately 63%. For the validation of the optimization and to prevent that the decline is not caused by a phenomena that occurs due to the mesh deformation the mesh of the last optimization step is properly rebuild and the last primal simulation is repeated on the new mesh. The new evaluation of the cost functional differs in this case by less than 1%. Therefore, the optimization is assumed to be mesh independent and thus valid .

Nevertheless, in the eighth iteration a rise in the cost function is evident, which otherwise decreases monotonously. Since the exact solution field was employed in this case, the error results immediately out of the shape optimization and can be attributed to a suboptimal choice of the step size length. The choice of a smaller $\tilde{d}_{max}$ could avoid this leap but at the same time results in a higher number of required iterations. A variable step size rule could provide a cure for this. Since this anomaly is independent of the approximation approach, it is expected that similar jumps will occur for the optimization using the itSVD.

However, the evaluation of the cost functional of the optimization based on the approximation with an amount of retained energy of 99.99% does not meet these expectations. As in the exact case, the objective decreases sharply at the start, whereupon the incline decreases and finally converges to approximately the same

limit as the exact optimization. But there is no peak and the monotone decline of the cost function, which is characteristic for a successful optimization, can be observed over the entire shape iteration. Since both optimizations run almost identically up to the eighth iteration, the abandonment of the peak can be attributed to the exploitation of the reduced flow for the adjoint equations and the sensitivity. Yet, it does not follow that the reduced optimization is inherently superior to the exact optimization. In terms of the almost identical achieved reduction of the cost functional, the success of both approaches can be considered equal.
Nevertheless, if the memory requirements are included in the comparison, the reduced approach is definitely preferable. The analysis of the memory consumption in Fig. 5.12 (right) indicates that in this case, from an initial storage requirement of 2% to a maximum of 3.5% of the full storage is required. Moreover, the results from Fig. 5.11 reveal that this approach yields a lower execution time. In summary, if for the approximation a sufficiently high accuracy is chosen, the reduced approach minimizes the cost functional as well as the full optimization approach, but exhibits a significantly decreased memory requirement and is faster at the same time. Anyhow, this conclusion does not hold without restrictions, but depends on the choice of the approximation quality. The impact of a lower approximation quality on the development of the cost functional is also shown in Fig. 5.12 for the energy bounds 50% and 75%. It points out that also in these cases the characteristic course of the objective is kept, but jumps appear more frequently, which affect the optimization success and lead to a lower decrease of the cost functional than in the cases treated before. Thereby, a greater amount of retained energy leads to a better final result. Although the optimization in these two cases leads to a lower reduction of the drag force, the differences are small and taking into account that these cases use a maximum of about 1.5% of the original memory, the optimization can still be regarded as successful.

After evaluating the optimization properties of the reduced flow field by means of the cost functional, the real world implications on the shape of the solid and the physical behaviors of the flow are investigated next. Based on the results of [6], it is expected that the optimization minimizes the frontal area of the body and squeezes it in the transverse direction to suppress vortex shedding. As a consequence, the vortex shedding is substituted by a steady flow around the optimized shape and a constant force on the body results.
For the purpose of examining whether the optimization approach used in this work confirms these findings, Fig. 5.13 depicts the optimized shapes and the optimized drag force compared to their initial states.

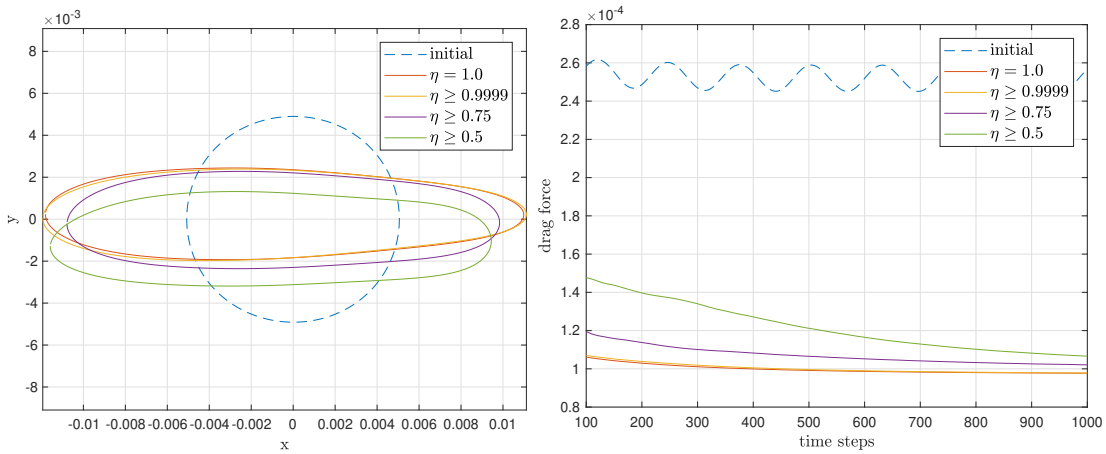

Figure 5.13: Implications of the approximation quality of the reduced flow field on the optimized shape and drag force.

Starting with the results of the optimization based on the full solution, the optimized shape elongated in the flow direction and compressed in orthogonal direction confirms the expected transformation due to the optimization. Moreover, the drag force profile indicates that the initial periodic force pattern becomes almost constant in time. The inconsistent initial behavior is due to the preceding shape update, such that the later time period is of more relevant in the analysis. In addition, the examination of the inital force progression proves that the time interval in which the drag force is optimized was chosen sufficiently large and contains numerous periods. Thus, a larger time interval would lead to almost identical results.

After the results of the exact optimization were validated by means of the results of [6], the optimization results based on the reduced velocity field are compared with the exact results in the following. The comparison of the cost functionals reveals that a higher approximation precision leads to the convergence of the cost functional of the approximation towards the exact objective. Next, it is examined if the same holds for the optimized shapes and the resulting force trajectories. Figure 4 indicates that although the cost functionals of the exact and reduced cases with 99.99% energy are virtually identical, the shapes do differ to some degree. The shape of the reduced case is marginally more elongated. The irregularity in the update of the full optimization can be identified as a potential source of this variation. Since it did not occur in the reduced case, it can be assumed that the shapes begin to differ from each other at this point and tend to converge towards the same shape only approximately. With regard to the force profile, the differences in shape have only a marginal effect and especially in the later course, the graphs prove to be congruent. In contrast to this optimization with an approximation of high quality, there are significant deviations in the op-

timized shape when a lower energy parameter is chosen. Although the overall expected shape is preserved, the stretching in both cases is much less prominent than in the optimization with the exact velocity field. Moreover, both shapes are shifted in the negative y-direction, with the magnitude of the displacement increases with lower approximation quality. Since due to the periodic force and the large time interval a nearly symmetric deformation is expected and indeed appears with an exact velocity field, this shift can be attributed to the failure of the reduced velocity field to retain the time-averaged symmetry of the velocity field. Despite the fact that the overall shape variations are substantial, the force profile exhibits that towards the end of the simulation interval, the evaluation deviates only marginally from each other and every profile becomes approximately constant. Therefore, all optimizations considered here based on reduced velocity fields result in a significantly reduced and steady force acting on the solid.

Finally, Fig. 5.14 presents the effects of the optimization based on the reduced velocity field with 99.99% retained energy on the flow behavior in comparison to the initial situation. Here, the result also matches the findings of [6] and reinforces the confidence in the optimization approach using a approximated flow field. As a result of the deformation of the shape, the vortex shedding is replaced by a steady behavior.

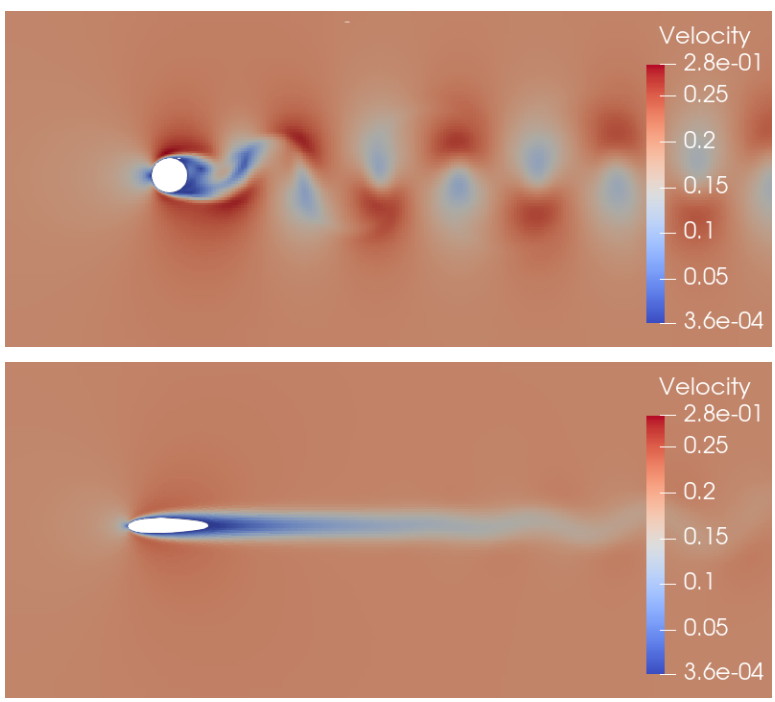

Figure 5.14: The flow field for the initial and optimized shape at $t = 1\,\mathrm{s}$ when performing the optimization based on the itSVD approach ($\eta \geq 99.99\%$).

In summary, in this section the application of the itSVD to the unsteady adjoint-based fluid dynamic shape optimization was tested and validated on a 2D laminar model problem for minimizing the drag force on a cylinder. It was shown that the approach provides a memory efficient alternative to the original approach of storing the full snapshot matrix and can replicate the exact optimization with nearly the same quality using only about 3.5% of the initial memory requirement. The approach also needs less computing time due to the parallelization, such that ultimately the optimization based on the itSVD algorithm both requires less memory capacity and is executed faster.

5.2.2 Turbulent Two-Phase Flow around 3D Sphere

In this section, the itSVD approach is applied to the shape optimization of a 3D sphere. The flow around the sphere is chosen to be two-phase and turbulent in this test case. The two immiscible phases can be interpreted as water and air, with the sphere submerged in the water. The sphere has a diameter of $D = 1\,\mathrm{m}$ and is positioned with its center 1 m below the free surface. The inflow velocity in the water phase is set to $u_1 = 1\,\mathrm{m/s}$ and its kinematic viscosity to $\mu = 10^{-5}\,\mathrm{m^2/s}$. This yields a Reynolds number of $\mathrm{Re} = 10^6$ and thus a turbulent flow. In addition, a periodic wave is induced on the free surface of the two phases. The wave is characterized by a wave height of 0.1 m and a wavelength of 4 m and propagates in flow direction. A snapshot of the resulting simulation scenario is presented in Fig. 5.15 by illustrating the volume fraction field.

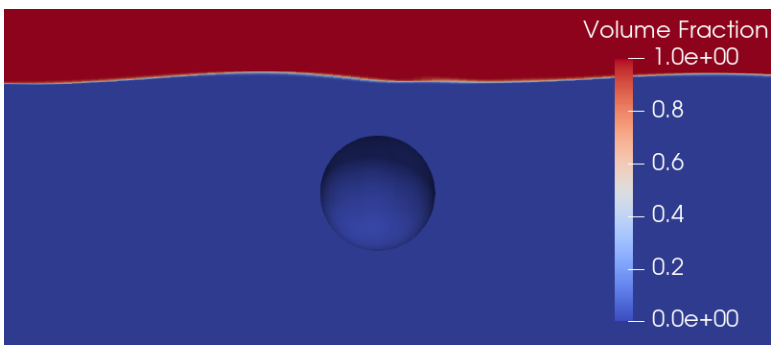

Figure 5.15: Snapshot of the initial state of the optimization.

For the numerical treatment of this problem, it is evident that the optimization framework presented in this study in chapter 2 is inadequate to deal with the novel added phenomena. The presentation of the required extensions is beyond the scope of this work, such that for this purpose, it is referred to the work of [1] to complement the approach by the turbulent and multi-phase elements. However, the basic approach of the optimization remains the same, but the primal and adjoint equations are modified. For the concrete case, this indicates that in addition to the velocity field, the eddy viscosity and the volume fraction of the primal solution have to be stored to be available for the adjoint equations. This is accomplished by means of the itSVD by extending the solution vector $a(t_n)$ of each time step t_n by these quantities. Nevertheless, the approximation of the new quantities introduces a new source of error. While the evaluations of the reduced velocity field tolerate deviations in negative and positive directions, the volume fraction and the eddy viscosity are limited to certain range of values. So the volume fraction is restricted to the values in the interval $[0, 1]$ and the eddy viscosity must be positive.

These restrictions are not taken into account by the itSVD, such that for the evaluation of the reduced field these quantities have to be controlled again and if the conditions are violated the values must be set to the nearest interval boundary.

On the basis of these insights, the shape optimization can be carried out. For this purpose 1000 equidistant time steps with a step size of $\Delta t = 10^{-2}$ s are calculated in each optimization iteration utilizing a parallelization on 22 partitions. As a result of considering a 3D computational domain with mesh refinements in the regions around the sphere and the free surface, it follows that $1.6 \cdot 10^6$ values must be stored in the associated snapshot matrix per time step. Here, the exact optimization is subject to significant limitations due to the maximal storage capacity of the system, such that in this case only an optimization based on a reduced primal flow field is supported.
For this optimization, the maximum deformation is scaled according to Eq. 5.2 via $d_{max} = 10^{-3}$ m to a 0.1% deformation of the initial shape with respect to the diameter of the sphere. Similar to the 2D optimization case, the first 200 steps of the adjoint solution were neglected for the optimization to allow the adjoint flow to establish its periodic behavior.
The itSVD is executed in parallel on each of the 22 partitions and is generated incrementally by means of a bunch update of size $b = 10$. In contrast to the previous test case, the requirement to preserve 99.99% of the energy was insufficient and it caused the adjoint solution process to diverge in the first step. In order to avoid this behavior, an energy bound of $\eta \geq 99.9999\%$ per partition was required. The courses of the resulting cost functional and the memory requirements with respect to the number of shape updates are presented in Fig. 5.16.

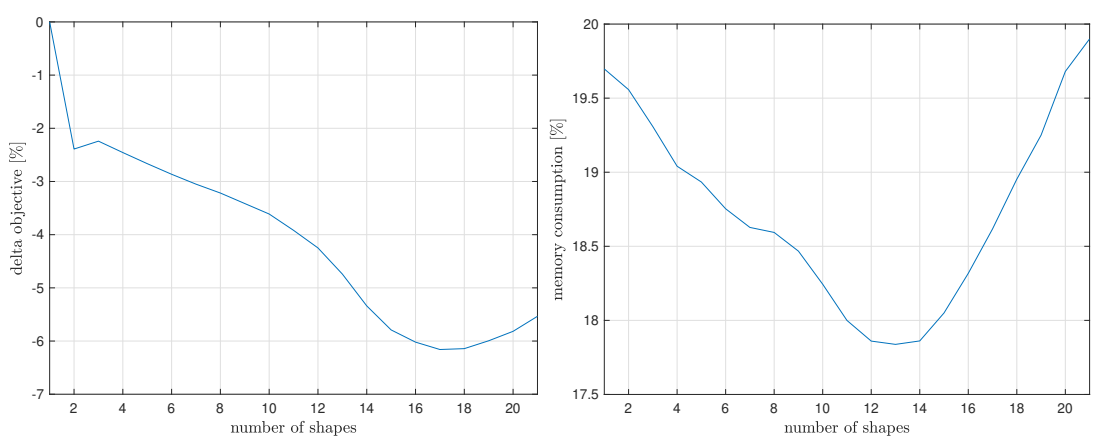

Figure 5.16: Course of the cost functional and the memory consumption with respect to optimization cycles.

The analysis of the cost functional reveals that, except for a jump at the beginning, it declines monotonically until the 17th iteration step. Thereby, the course

begins to flatten towards the end. Although this pattern could indicate a potential convergence, the cost functional subsequently starts to increase again until the 21st form update, contradicting a potential convergence. The execution of a further iteration step was impossible since during the adjoint simulation the solver begins to diverge and thus the shape optimization is aborted. To investigate whether this behavior is caused by a poor approximation quality of the primal flow, the optimization step was repeated again with a manually significant increased accuracy of the reduced flow. But the divergence occurred again in the adjoint problem.
In addition to the cost functional, the memory consumption during optimization also reveals that here the memory requirements can be reduced far less impressively than in the 2D test case. From an initial memory consumption of about 19.7%, it declines to below 18%, but then starts to increase again from the 14th shape update onwards, and reaches a maximal memory consumption of almost 20% before the optimization is aborted.

Lastly, the drag force resulting from the optimization is compared with the initial drag force curve in Fig. 5.17. The comparison reveals that both force trajectories differ only at the start of the simulation period, but align in the later course. This observation suggests that the reduction in the cost functional is a result of that differences and that the optimization eliminates the irregular oscillation that occurs for the initial shape, but otherwise does not have a sustainable impact due to the shape optimization.

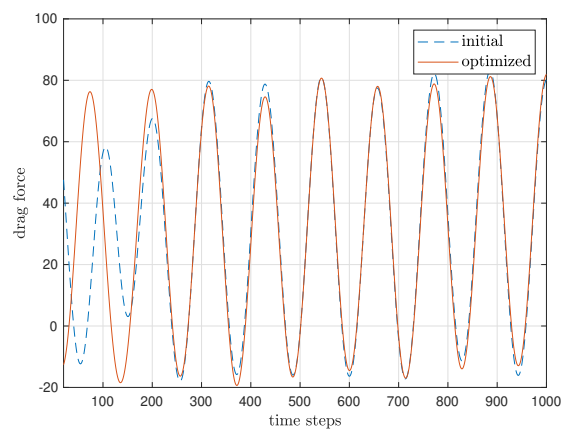

Figure 5.17: Initial drag force in contrast to optimized drag force over time.

In summary, the shape optimization of a 3D sphere in a two-phase turbulent flow presented here does not represent a fully satisfactory result and thus does not confirm the previous success of applying the itSVD approximation to the shape optimization of a 2D cylinder.

Chapter 6

Conclusion and Outlook

Within this thesis, a memory efficient model reduction method for the primal flow field based on the incremental truncated Singular Value Decomposition (itSVD) was developed and integrated into the in-house finite volume based solver FreSCo$^+$. The goal of this effort was to establish a shape optimization framework that ensures the optimization of large problems without being constrained by memory limitations. Concurrently, the approach was intended to have only a marginal effect on the computational time of the optimization, rather than the checkpointing techniques which cause a great computational overhead.

The memory efficiency of the reduced model is ensured in this work by the application of a adaptive rank determination routine, which minimizes the storage requirements with respect to a predefined minimal approximation quality. In practice, this minimal quality is often prescribed by the amount of retained energy of the reduced flow field. Moreover, the computational effort of constructing the reduced primal solution is significantly reduced by the utilization of the bunch update which lowers the frequency of the incremental update of the reduced primal flow field. Furthermore, the itSVD is parallelized according to a domain decomposition approach to match the parallelization of the FreSCo$^+$ routine and to grant an optimal interplay of both parts.

The applicability of the resulting method to the unsteady adjoint-based shape optimization was tested for a periodic laminar flow around a 2D cylinder, where the minimization of drag was demanded. With the requirement that the reduced flow field retains 99.99% of the original energy, it was demonstrated that the optimization based on the reduced flow yields almost the identical result as the optimization based on the full primal solution by using only 3.5% of the original memory requirement. In addition to this high memory reduction, the optimization of the reduced case also required less computation time.

However, the results of the second test case with the aim of the shape optimization of the 3D sphere in a turbulent two-phase flow field do not confirm the unrestricted applicability of the proposed approach. In this case, the approximation in maximum requires almost 20% of the original memory. Furthermore, a divergence behavior occurs in the course of the computation of the adjoint equations, such that the optimization fails. As a possible source of error, the approximations of the additional flow quantities that have to be stored in this case can be identified here. A potential countermeasure to this problem could be to create a dedicated approximation for each field quantity instead of one large itSVD in which all field quantities are stored. The resulting ability to adjust the approximation quality for each field quantity individually could offer interesting opportunities for future works with respect to a problem-specific choice of the approximation accuracy for each field quantity.

Furthermore, combining the incremental approximation approach with the checkpointing techniques could provide a suitable trade-off between computational and memory cost for rather sophisticated memory intensive optimizations and therefore offers promising prospects with respect to the application of the unsteady adjoint-based shape optimization to real-world engineering problems.

Bibliography

[1] Niklas Kühl et al. "Adjoint complement to the volume-of-fluid method for immiscible flows". In: *Journal of Computational Physics* 440 (2021), p. 110411.

[2] C Vezyris et al. "Unsteady continuous adjoint method using POD for jet-based flow control". In: *11th World congress on computational mechanics, ECCOMAS. Barcelona, Spain.* 2014.

[3] Christos Vezyris, Evangelos Papoutsis-Kiachagias, and Kyriakos Giannakoglou. "On the incremental singular value decomposition method to support unsteady adjoint-based optimization". In: *International Journal for Numerical Methods in Fluids* 91.7 (2019), pp. 315–331.

[4] Andreas Griewank and Andrea Walther. "Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation". In: *ACM Transactions on Mathematical Software (TOMS)* 26.1 (2000), pp. 19–45.

[5] Matthew Brand. "Incremental singular value decomposition of uncertain data with missing values". In: *European Conference on Computer Vision.* Springer. 2002, pp. 707–720.

[6] A-SI Margetis, EM Papoutsis-Kiachagias, and KC Giannakoglou. "Lossy compression techniques supporting unsteady adjoint on 2D/3D unstructured grids". In: *Computer Methods in Applied Mechanics and Engineering* 387 (2021), p. 114152.

[7] Orlando Soto. "On the Computation of Flow Sensitivities from Boundary Integrals". In: *AIAA Paper* (Jan. 2004).

[8] Georgios Bletsos, Niklas Kühl, and Thomas Rung. "Adjoint-based Shape Optimization for the Minimization of Flow-induced Hemolysis in Biomedical Applications". In: *arXiv preprint arXiv:2101.10715* (2021).

[9] Jan Philipp Heners et al. "Adjoint shape optimization for fluid–structure interaction of ducted flows". In: *Computational Mechanics* 61.3 (2018), pp. 259–276.

[10] Jörn Kröger and Thomas Rung. "CAD-free hydrodynamic optimisation using consistent kernel-based sensitivity filtering". In: *Ship Technology Research* 62.3 (2015), pp. 111–130.

[11] Nicholas J Higham. *Matrix nearness problems and applications*. Citeseer, 1988.

[12] Gaetan Kerschen et al. "The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems: an overview". In: *Nonlinear dynamics* 41.1 (2005), pp. 147–169.

[13] Matthew Brand. "Fast low-rank modifications of the thin singular value decomposition". In: *Linear algebra and its applications* 415.1 (2006), pp. 20–30.

[14] William W Symes. "Reverse time migration with optimal checkpointing". In: *Geophysics* 72.5 (2007), SM213–SM221.

[15] Qiqi Wang, Parviz Moin, and Gianluca Iaccarino. "Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calculation". In: *SIAM Journal on Scientific Computing* 31.4 (2009), pp. 2549–2567.

[16] Andreas Griewank. "Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation". In: *Optimization Methods and software* 1.1 (1992), pp. 35–54.

[17] Stefan Volkwein. "Model reduction using proper orthogonal decomposition". In: *Lecture Notes, Institute of Mathematics and Scientific Computing, University of Graz. see http://www. uni-graz. at/imawww/volkwein/POD. pdf* 1025 (2011).

Erklärung

Die vorliegende Arbeit habe ich selbständig verfasst und keine anderen als die angegebenen Hilfsmittel - insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen - benutzt. Die Arbeit habe ich vorher nicht in einem anderen Prüfungsverfahren eingereicht. Die eingereichte schriftliche Fassung entspricht genau der auf dem elektronischen Speichermedium.

Unterschrift	Datum