

A Reduced-Order Modeling Approach to Shape Optimization Via Time-Dependent Bases

by

Adam Vieno

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Applied Mathematics

Waterloo, Ontario, Canada, 2026

© Adam Vieno 2026

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Partial differential equation (PDE)-constrained optimization plays a central role in engineering applications such as fluid flow control, thermal analysis, and aerodynamic shape optimization. However, its practical use is limited by the need to repeatedly evaluate high-fidelity full-order models (FOMs), which involves solving large-scale nonlinear PDE systems that can become computationally prohibitive, especially in high-dimensional design spaces.

Reduced-order modeling (ROM) techniques mitigate this cost by constructing low-dimensional surrogates of the FOM. However, traditional static approaches often require extensive pre-processing and can struggle to adapt to dynamic nonlinear changes within the design space.

To address these challenges, we employ the time-dependent basis-CUR (TDB-CUR) framework, which constructs reduced bases dynamically using a time-discrete variational principle. This approach yields a low-rank approximation that remains well-conditioned while naturally adapting to evolving nonlinear solution structures, without relying on a fixed global basis.

Hence, the main contribution of this work is the integration of the TDB-CUR algorithm within a surrogate-inspired shape optimization framework that progressively refines the design space. We propose an iterative strategy that shrinks the design domain toward promising configurations while maintaining a low, static rank during ROM evaluations. This enables improved local approximation quality near the optimum without increasing the reduced dimension. We further introduce heuristics for adaptive rank selection and shrink factor control to balance accuracy and computational cost.

The approach is tested on shape optimization problems governed by the quasi-one-dimensional compressible Euler equations. Numerical experiments with up to ten design variables show that the proposed framework accurately reconstructs the target solutions while substantially reducing computational cost relative to the FOM. Overall, the proposed method represents a promising first step toward more efficient PDE-constrained shape optimization through the integration of dynamic low-rank modeling and iterative design space refinement.

Acknowledgements

First, I would like to express my sincere gratitude to my supervisor, Professor David Del Rey Fernández, for his guidance, patience, and support throughout my graduate studies. I am especially thankful for the patience and understanding he consistently demonstrated whenever I encountered difficulties or uncertainty. His willingness to discuss ideas, answer questions, and provide thoughtful feedback, regardless of the time or circumstance, helped me overcome many challenges and was crucial to the completion of this thesis. I am appreciative of the time and effort he invested in my work, as well as for the many conversations that broadened my understanding and encouraged me to think more carefully about difficult problems. It has been a privilege to work under his supervision, and I sincerely appreciate all that he has done to support me throughout my degree.

I would also like to thank my co-supervisor, Professor Hans De Sterck, for his assistance throughout my graduate studies. I appreciate the advice and perspective he provided on this work.

I would also like to sincerely thank Professor Hessam Babaei at the University of Pittsburgh for his time and input throughout this work. He played a key role in helping me understand the underlying methodology and in shaping my approach to the problem. His ideas and suggestions were central to the direction of this project.

Throughout this project, I benefited greatly from the support of Ramy Rashad from Pratt & Whitney Canada Corp. Our regular meetings over the past years provided valuable feedback and insight, and I am very grateful for the time and effort he consistently invested in discussing and refining this work.

I am also very grateful to Grishma Palkar, who was always generous with her time and consistently willing to help whenever I had questions. She was a constant support for both the coding and the underlying concepts behind the method, and I greatly appreciated her patience and clarity.

I would also like to thank my lab mates Andrew and Zelalem for their support throughout my studies. Andrew was especially helpful in the early stages of this work and played an important role in helping me get started with the project, including coding support throughout. Zelalem provided ongoing guidance over the last few years, with regular meetings and valuable feedback that helped shape the direction of the work.

Finally, I would like to thank my parents Marcel and Eija, my siblings Kayla and Jacob, and my girlfriend Erica, for their patience and support throughout my academic journey. Their belief in me, even when I did not always believe in myself, has been a constant source of motivation.

Dedication

Dedicated to my family and loved ones.

Table of Contents

Author's Declaration	ii
Abstract	iii
Acknowledgements	iv
Dedication	v
List of Figures	ix
List of Tables	xiii
List of Abbreviations	xiv
1 Introduction	1
1.1 Thesis Objectives and Outline	3
2 SBP-SAT Discretization of the Quasi-1D Euler Equations	5
2.1 Motivation	5
2.2 Quasi-1D Euler Equations	6
2.3 Overview of SBP Operators	6
2.3.1 Application to the Linear Advection Equation	8
2.4 Selected Discretization	11

3	TDB-CUR	13
3.1	Full-Order Model Setup	13
3.2	Definitions and Preliminaries	14
3.3	Time-Discrete Variational Principle and Oblique Projection	16
3.3.1	Theoretical Equivalence	20
3.3.2	Key features of TDB-CUR	22
3.4	Additional Details of the TDB-CUR Algorithm	23
3.4.1	Computing $V^k(\mathbf{p}, :)$	23
3.4.2	Error Reduction (Oversampling and Rank Adaptivity)	25
3.4.3	TDB-CUR Algorithm with Rank Adaptivity and Oversampling	28
3.4.4	Application to the Inviscid and Viscous Burgers' Equation	29
4	Optimization Methodology	39
4.1	Problem Layout	39
4.2	Presented Methodology	40
4.2.1	Description of <code>interpolator</code> and <code>getSamples</code>	46
4.2.2	Description of <code>objectiveFunction</code> and <code>terminationCondition</code>	48
4.2.3	Description of <code>optimizer</code>	49
4.2.4	Description of <code>newDomain</code>	50
4.3	Proof-of-Concept Results	52
5	Shrink Factor and Rank Adaptivity	55
5.1	Motivation and Description of Adaptive Shrink Factor	55
5.1.1	Singular Value Tail Ratio	59
5.1.2	Singular Value Slope	60
5.1.3	Relative Change of Singular Values	63
5.1.4	Combining Signals and Aggressiveness	64
5.1.5	Rank Adaptivity	66

5.2	Results	69
5.2.1	Two Design Variables	70
5.2.2	Four Design Variables	73
5.2.3	Six Design Variables	77
5.2.4	Ten Design Variables	82
6	Conclusion and Future Work	84
6.1	Conclusion	84
6.2	Future Work	85
	References	88
	APPENDICES	97
A	Application of TDB-CUR to the Quasi-1D Euler Equations	98
A.1	Nondimensionalization of the Quasi-1D Compressible Euler Equations . . .	99
A.2	Solution and SBP Operator Form for the Quasi-1D Compressible Euler Equations	105
A.3	Indexing in TDB-CUR	109
A.4	Computing the RHS	114
A.4.1	A Note on Artificial Dissipation and TDB-CUR	122
A.4.2	A Note on Different Discretizations	123

List of Figures

3.1	The evolution of the first 50 singular values of the FOM solution matrix over time for the split-form discretization of the inviscid Burgers' equation. . .	31
3.2	The mean solution, as defined in Equation (3.13), for the inviscid Burgers' equation, evaluated at five distinct time instances $t = 0, 0.25, 0.5, 0.75, 1.0$ with $N_s = 400$. The results correspond to the fourth rank-adaptivity configuration, with $\epsilon_u = 10^{-10}$ and $\epsilon_l = 10^{-12}$. No visually discernible difference was observed in the mean solution across the four tested configurations. . .	32
3.3	The maximum absolute error between the FOM and ROM, as defined in (3.14), for the four rank-adaptive cases considered. Case 1 corresponds to $\epsilon_u = 10^{-4}$, $\epsilon_l = 10^{-6}$; Case 2 to $\epsilon_u = 10^{-6}$, $\epsilon_l = 10^{-8}$; Case 3 to $\epsilon_u = 10^{-8}$, $\epsilon_l = 10^{-10}$; and Case 4 to $\epsilon_u = 10^{-10}$, $\epsilon_l = 10^{-12}$	33
3.4	The evolution of the rank r over time for the four rank-adaptive cases. The initial rank is taken as $r = 10$ for all cases. Case 1 corresponds to $\epsilon_u = 10^{-4}$, $\epsilon_l = 10^{-6}$; Case 2 to $\epsilon_u = 10^{-6}$, $\epsilon_l = 10^{-8}$; Case 3 to $\epsilon_u = 10^{-8}$, $\epsilon_l = 10^{-10}$; and Case 4 to $\epsilon_u = 10^{-10}$, $\epsilon_l = 10^{-12}$	34
3.5	The evolution of the first 50 singular values of the FOM solution matrix over time for the split-form discretization of the viscous Burgers' equation. . . .	35
3.6	The mean solution, as defined in (3.13), for the viscous Burgers' equation, evaluated at five distinct time instances $t = 0, 0.25, 0.5, 0.75, 1.0$ with $N_s = 400$. The results correspond to the fourth rank-adaptive configuration, with $\epsilon_u = 10^{-10}$ and $\epsilon_l = 10^{-12}$. No visually discernible difference was observed in the mean solution across the four tested configurations.	36
3.7	The maximum absolute error between the FOM and ROM, as defined in (3.14), for the four rank-adaptive cases considered. Case 1 corresponds to $\epsilon_u = 10^{-4}$, $\epsilon_l = 10^{-6}$; Case 2 to $\epsilon_u = 10^{-6}$, $\epsilon_l = 10^{-8}$; Case 3 to $\epsilon_u = 10^{-8}$, $\epsilon_l = 10^{-10}$; and Case 4 to $\epsilon_u = 10^{-10}$, $\epsilon_l = 10^{-12}$	37

3.8	The evolution of the rank r over time for the four rank-adaptive cases. The initial rank is taken as $r = 10$ for all cases. Case 1 corresponds to $\epsilon_u = 10^{-4}$, $\epsilon_l = 10^{-6}$; Case 2 to $\epsilon_u = 10^{-6}$, $\epsilon_l = 10^{-8}$; Case 3 to $\epsilon_u = 10^{-8}$, $\epsilon_l = 10^{-10}$; and Case 4 to $\epsilon_u = 10^{-10}$, $\epsilon_l = 10^{-12}$	38
4.1	Example of a one-dimensional converging-diverging nozzle parameterized by two design variables over the domain $x \in [x_L, x_R]$. The channel width is given by $a(x, \boldsymbol{\theta})$, where $\boldsymbol{\theta} = [\theta_1, \theta_2]$, with θ_1 and θ_2 controlling the inlet and outlet widths, respectively.	41
4.2	Illustration of the limitations of TDB-CUR when applied to optimization problems, where evaluations are required at parameter values outside the sampled design space.	42
4.3	First 100 singular values of the FOM plotted over nondimensional time for the full design space $\Theta = [0.4, 0.7]$	44
4.4	First 100 singular values of the FOM plotted over nondimensional time for half of the design space $\Theta_\epsilon = [0.475, 0.625]$	45
4.5	The trajectory of the estimated parameter $\boldsymbol{\theta}^*$ within the admissible design space over the course of the optimization algorithm. For this simple test case, the method converges rapidly to the optimal parameter value.	53
4.6	Comparison between the target and reconstructed pressure profiles, with the gray curves representing the sampled profiles from the initial design domain $\Theta^{(0)}$. The maximum absolute error is also reported	54
4.7	Comparison of the target and found nozzle geometries, with the maximum absolute error between the two reported.	54
5.1	An image of the conceptual difference between a standard trust-region method and the proposed methodology. In the trust-region approach (left), the algorithm begins from a local region in the design space and iteratively moves toward the optimum. In contrast, the proposed method (right) starts from the full design space and progressively refines it, converging to the optimum through successive contractions.	57
5.2	Mapping from the singular value tail ratio ϵ_t , to the corresponding shrink factor z_t for the proposed heuristic. Small values of ϵ_t , indicating a low-rank structure, lead to more aggressive shrinkage, whereas values close to one, indicating a flat spectrum, result in more conservative shrinkage.	60

5.3	Mapping between the slope of the singular value spectrum ϵ_m , and the corresponding shrink factor z_m . Large negative values of ϵ_m , corresponding to rapid spectral decay, are mapped to a maximum shrink factor of $z_m = 1.75$. For intermediate decay rates, the mapping is approximately linear, while for a flat spectrum ($\epsilon_m \approx 0$), the shrink factor reduces to $z_m = 1.0$	62
5.4	Mapping between the relative change in singular values across successive iterations ϵ_Δ , and the corresponding shrink factor z_Δ . This assumes that changes in the spectrum between successive iterations are not excessively large, with “large” defined via a tolerance $\tau = 0.05$	65
5.5	Trajectory of the optimal solutions and design domains over optimization iterations for two different rank cases.	71
5.6	Trajectory of the shrink factor over optimization iterations for two different rank cases.	72
5.7	Final pressure profiles found for two different rank cases.	73
5.8	Progression of the max absolute and relative L^2 errors with respect to the target profile over successive algorithm iterations for two different rank cases.	74
5.9	Trajectory of the four design variables over optimization iterations. In gray is the design domain for each variable, with the trajectory of the best sample and optimal θ plotted overtop. The best sample is taken to be the column in the ROM that produces the smallest objective value.	75
5.10	Four design variable optimization results illustrating algorithm performance and convergence.	77
5.11	Adaptive evolution of rank and shrink factor over algorithm iterations.	78
5.12	Effect of different shrink factor aggressiveness (a_g) on algorithm performance.	81
5.13	Ten design variable optimization results illustrating algorithm performance and convergence.	83
A.1	Nondimensional and dimensional pressure profiles for the subsonic nozzle problem from [61], computed to steady state ($t_F = 3.0$). The solution is obtained using the spatial discretization in (2.9) with $k = 50$ elements and polynomial degree $q = 2$, and advanced in time using an RK4 scheme with time step $\Delta t = 5.0 \times 10^{-5}$	103

A.2	Nondimensional and dimensional pressure profiles for the transonic nozzle problem from [61], computed to steady state ($t_F = 5.0$). The solution is obtained using the spatial discretization in (2.9) with $k = 75$ elements and polynomial degree $q = 2$, and advanced in time using an RK4 scheme with time step $\Delta t = 5.0 \times 10^{-5}$	104
A.3	A one-dimensional domain discretized into three elements with four nodes per element. The indices above each node correspond to the global DOF numbering used in the TDB-CUR implementation of the quasi-1D Euler equations. The global ordering is defined by flattening over element index, nodal position within each element, and conserved-variable components. Each index therefore represents a single conserved-variable DOF at a specific node and element. For instance, indices 1 and 25 correspond to the conserved variable $a\rho$ at the first node of the first and third elements, respectively.	125

List of Tables

5.1	Runtime and error comparison across 4D test cases relative to an 11-iteration FOM baseline. Speedup factors are evaluated relative to the matching baseline for each respective sample size N_s . Colour highlights indicate the highest and lowest bounds for runtime, speedup, and $\ p - p^*\ _\infty$ error metrics for the ROM only, as detailed below.	76
5.2	Runtime and error comparison across 6D test cases relative to a single-iteration FOM baseline. Speedup factors are evaluated relative to the matching baseline for each respective sample size N_s . Colour highlights indicate the highest and lowest bounds for runtime, speedup, and $\ p - p^*\ _\infty$ error metrics for the ROM only, as detailed below.	80

List of Abbreviations

BO	Bi-Orthogonal Decomposition
CG	Continuous Galerkin
CUR	CUR Matrix Decomposition
DBO	Dynamically Bi-Orthogonal Decomposition
DE	Differential Evolution
DEIM	Discrete Empirical Interpolation Method
DG	Discontinuous Galerkin
DLRA	Dynamical Low-Rank Approximation
DO	Dynamical Orthogonal Decomposition
DOF	Degree of Freedom
FD	Finite Difference
FOM	Full-Order Model
IBP	Integration-by-Parts
L-BFGS-B	Limited-Memory Broyden-Fletcher-Goldfarb-Shanno Method with Bounds
LGL	Legendre-Gauss-Lobatto
LHS	Latin Hypercube Sampling
PDE	Partial Differential Equation
POD	Proper Orthogonal Decomposition
RBF	Radial Basis Function

RHS	Right-Hand Side
RK4	Fourth-Order Runge-Kutta
ROM	Reduced-Order Model
SAT	Simultaneous Approximation Term
SBP	Summation-by-Parts
SPD	Symmetric Positive Definite
SVD	Singular Value Decomposition
TDB	Time-Dependent Basis
TDB-CUR	Time-Dependent Basis-CUR

Chapter 1

Introduction

Partial differential equation (PDE)-constrained optimization problems arise in a variety of industrial applications, including fluid flow control, thermal analysis, and aerodynamic shape optimization. However, they can be costly to evaluate due to the large number of queries to the underlying PDE solver, or more generally, the full-order model (FOM). This cost becomes particularly apparent in gradient-based optimization methods, such as gradient descent or quasi-Newton schemes, where each iteration requires not only a full solve of the PDE, but also the computation of sensitivities with respect to the design variables. To illustrate this computational burden, consider a central finite-difference (FD) approximation of the gradient $\nabla \mathcal{J}(\boldsymbol{\theta})$:

$$\frac{\partial \mathcal{J}}{\partial \theta_i}(\boldsymbol{\theta}) \approx \frac{\mathcal{J}(\boldsymbol{\theta} + h\mathbf{e}_i) - \mathcal{J}(\boldsymbol{\theta} - h\mathbf{e}_i)}{2h}, \quad (1.1)$$

$$\nabla \mathcal{J}(\boldsymbol{\theta}) \approx \left(\frac{\partial \mathcal{J}}{\partial \theta_1}, \dots, \frac{\partial \mathcal{J}}{\partial \theta_d} \right)^T, \quad (1.2)$$

where $\mathbf{e}_i \in \mathbb{R}^d$ is the i^{th} canonical basis vector, $\boldsymbol{\theta} \in \mathbb{R}^d$ is the vector of d design variables, and $h > 0$ is a small step size. Each evaluation of the objective function $\mathcal{J}(\boldsymbol{\theta})$ requires a solve of the FOM. Therefore, computing a single gradient via (1.1) requires two FOM evaluations per parameter direction. If we assume $\boldsymbol{\theta} \in \mathbb{R}^d$, a full gradient evaluation using a FD approach requires $2d$ FOM evaluations. When embedded within iterative optimization algorithms, this cost is incurred at every iteration, leading to a computational burden that scales linearly with both the number of design variables and the number of optimization steps.

As such, various researchers have developed methods and algorithms aimed at reducing the computational cost associated with PDE-constrained optimization by limiting the

number of FOM evaluations. One prominent class of approaches is based on reduced-order models (ROMs), which seek to approximate the behavior of the FOM within a low-dimensional representation, thereby enabling significantly cheaper evaluations of the governing equations.

For example, ROM trust-region methods [3, 33, 45, 82, 83] solve optimization problems by constructing a sequence of locally accurate surrogate models. At each iteration, a ROM is used to predict the effect of candidate design updates, and this surrogate is restricted to a trust region that controls the validity of the approximation [82]. Based on agreement between the ROM and FOM, the trust-region is dynamically adjusted, ensuring convergence of the optimization process while significantly reducing the number of expensive FOM evaluations [33]. However, the performance of such methods is strongly dependent on the quality and stability of the ROM within the region of interest, and maintaining robustness outside the span of the snapshot data remains a key challenge [45].

In the context of shape optimization problems, surrogate-assisted approaches are widely used. Here, the FOM is evaluated at a selected set of design points, and a lower-cost surrogate model is constructed to guide the optimization process. These surrogate models can take several forms. Classical approaches include interpolation-based and probabilistic models, such as radial basis function (RBF) interpolants [77] and Gaussian process regression (Kriging) [79]. These methods construct a global approximation of the objective or state variables based on sampled evaluations, and are often valued for their simplicity and flexibility. Data-driven machine learning approaches have also been widely adopted, including support vector regression [43] and neural network-based models [84]. These methods aim to learn nonlinear mappings from design variables to quantities of interest, often requiring large datasets to improve predictive capability.

A third class of approaches is based on ROMs, such as proper orthogonal decomposition (POD) [22] and other reduced-basis techniques [22, 68]. In contrast to purely data-driven surrogates, ROMs explicitly incorporate information from the governing equations and are typically constructed via projection onto a low-dimensional subspace. In these settings, they are often used in place of the FOM, and may also be combined with the other aforementioned surrogate modeling approaches to improve efficiency and robustness [32].

However, when working with ROMs, it is important to consider the choice of technique and the associated trade-offs. In snapshot-based approaches such as POD, a global reduced basis is constructed from FOM solutions and subsequently used to project the governing equations onto a low-dimensional subspace. This enables efficient repeated evaluations of the reduced system, however, the construction of the basis still requires access to the FOM.

In contrast, time-dependent basis (TDB) methods, including dynamical low-rank ap-

proximation (DLRA) [46], dynamically orthogonal decomposition (DO) [67], bi-orthogonal decomposition (BO) [18], and dynamically bi-orthogonal decomposition (DBO) [59, 62], allow the reduced basis to evolve in time according to the dynamics of the system or through variational principles. By adapting the approximation space during the evolution, these methods can better represent transient and nonlinear phenomena compared to static bases [27, 67]. However, they may introduce additional challenges, including numerical ill-conditioning and varying degrees of intrusiveness in applying such methods [25].

The time-dependent basis-CUR (TDB-CUR) algorithm [25] addresses several of the challenges associated with TDB-ROMs by combining a time-evolving basis construction with an efficient interpolatory approximation strategy. In particular, the method updates the reduced basis at each time step using a time-discrete variational principle, enabling it to capture the evolving system dynamics throughout the simulation [25]. In addition to this dynamic basis update, the algorithm employs a greedy point-selection strategy to construct a CUR-type low-rank representation [57]. This interpolation-based structure allows the ROM to approximate nonlinear terms efficiently by evaluating them only at a carefully selected subset of degrees of freedom (DOFs), rather than over the full computational domain [25]. As a result, the TDB-CUR framework is able to maintain numerical stability and good conditioning while significantly reducing the computational cost associated with both basis evolution and nonlinear term evaluation [57].

1.1 Thesis Objectives and Outline

In this thesis, we will investigate the applicability of the TDB-CUR algorithm to solve shape optimization problems involving the quasi-1D compressible Euler equations with varying nozzle width. In particular, we aim to formulate an algorithm, similar to that of surrogate-based shape optimization, that focuses on using the TDB-CUR algorithm, and its low-rank capabilities. Specifically, we develop a methodology that progressively reduces the design space at each iteration to locate the optimum, while maintaining a fixed, low rank r . Additionally, we introduce adaptive heuristics to control the shrink factor z , which governs the reduction of the design domain, and rank r . Hence, the specific goals are as follows:

1. Develop an efficient and systematic approach to PDE-constrained shape optimization problems through the use of the TDB-CUR algorithm, and design heuristics to aide in the optimization procedure. This includes investigating the effects of the heuristics on the performance of the presented method.

2. Demonstrate the proposed workflow on the quasi-one-dimensional Euler equations in a converging-diverging geometry, considering multiple design space dimensions.

The remainder of this thesis is structured as follows. In Chapter 2, we introduce the governing equations, namely the quasi-one-dimensional Euler equations, and briefly present the summation by parts (SBP) operators and simultaneous approximation terms (SATs) framework that forms the basis of the FOM discretization. Chapter 3 presents the TDB-CUR algorithm with oversampling and rank adaptivity. The chapter also demonstrates the application of the algorithm to both the inviscid and viscous Burgers' equations, providing insight into the method's behavior across different dynamical regimes. Chapter 4 introduces the main optimization algorithm, outlining its underlying mathematical structure, and providing detailed explanations of its selected components. A proof-of-concept example is also presented for a simple single-design-variable problem. Chapter 5 presents the adaptive heuristics developed to control the shrink factor and rank, and discusses the motivation for the chosen techniques as well as the rationale behind each metric. The main results of the method are then presented for design spaces with $d = 2, 4, 6$, and 10 design variables, highlighting its capabilities. Finally, Chapter 6 summarizes the main contributions of the thesis and outlines directions for future work.

Chapter 2

SBP-SAT Discretization of the Quasi-1D Euler Equations

In this chapter, we introduce the governing equations and briefly describe the numerical framework that defines the FOM used in this thesis. Specifically, we consider the quasi-one-dimensional compressible Euler equations with varying nozzle width, together with their spatial discretization using summation-by-parts (SBP) operators and simultaneous approximation terms (SATs) for the weak imposition of inter-element coupling and boundary conditions. We also motivate the use of the SBP-SAT framework and present the chosen spatial discretization.

2.1 Motivation

Although the spatial discretization is not the central objective of this thesis, its properties are nonetheless important for robust computations. The behaviour of the ROM is inherently based on the underlying FOM, and deficiencies of the FOM, such as numerical instability or nonphysical solutions, may adversely affect the ROM.

For this reason, we aim to construct the FOM using spatial discretization techniques that admit desirable theoretical properties. Specifically, we employ SBP operators, which are matrix difference operators for approximating derivatives at nodes in a mesh [20, 24, 71, 72], in conjunction with simultaneous approximation terms (SATs) which weakly enforce boundary conditions and inter-element coupling [14, 20, 31]. This approach enables the

development of high-order methods that are provably stable by mimicking integration-by-parts (IBP) at the semi-discrete level [20, 72]. Consequently, SBP-SAT discretizations are able to admit discrete energy estimates that mimic those used in continuous stability analysis. This structural connection to continuous stability theory is a key reason for adopting SBP-SAT discretizations in this thesis, as it ensures a robust and therefore sound foundation for the ROM.

2.2 Quasi-1D Euler Equations

In this thesis, the goal is to solve shape optimization problems involving the quasi-one-dimensional compressible Euler equations with varying nozzle width of the form:

$$\frac{\partial}{\partial t} \begin{bmatrix} a\rho \\ a\rho u \\ aE \end{bmatrix} + \frac{\partial}{\partial x} \begin{bmatrix} a\rho u \\ a\rho u^2 \\ au(E+p) \end{bmatrix} + a \frac{\partial}{\partial x} \begin{bmatrix} 0 \\ p \\ 0 \end{bmatrix} = \mathbf{0}. \quad (2.1)$$

Here $\mathbf{0} \in \mathbb{R}^3$, ρ denotes the density, u denotes the velocity in the x direction, p denotes the pressure, and $a = a(x)$ defines the nozzle geometry, which is time independent. Additionally E is the total energy and is given as:

$$E := \frac{1}{2}\rho u^2 + \frac{p}{\gamma - 1},$$

where γ is the ratio of specific heat of a diatomic gas; here we use $\gamma = 1.4$.

2.3 Overview of SBP Operators

To numerically solve (2.1), we now introduce the spatial discretization framework used in this work. To discretize the spatial domain in one dimension, we partition the interval $x \in [x_L, x_R]$ into N_x nodes:

$$\mathbf{x} := [x_1, x_2, \dots, x_{N_x}]^T,$$

where x_i denotes the i^{th} grid point. This allows us to define the grid function $\mathbf{u} := [u_1, u_2, \dots, u_{N_x}]$, where $u_i := \mathcal{U}(x_i, t)$, representing the evaluation of $\mathcal{U}$ at the x_i^{th} mesh node.

As previously discussed, SBP operators are matrix-valued difference operators that, when applied to grid functions, approximate spatial derivatives at the discrete nodes. In particular, an operator $D_x \in \mathbb{R}^{N_x \times N_x}$ approximates the derivative of the form:

$$D_x \mathbf{u} \approx \left[\left(\frac{\partial \mathcal{U}}{\partial x} \right)_1, \left(\frac{\partial \mathcal{U}}{\partial x} \right)_2, \dots, \left(\frac{\partial \mathcal{U}}{\partial x} \right)_{N_x} \right]^T.$$

A key requirement for these operators is that they mimic the IBP property of the first derivative, which is a key step in developing stability estimates using the energy method. In the continuous setting, IBP takes the form:

$$\int_a^b \mathcal{V} \frac{\partial \mathcal{U}}{\partial x} dx = \mathcal{U} \mathcal{V} \Big|_a^b - \int_a^b \mathcal{U} \frac{\partial \mathcal{V}}{\partial x} dx. \quad (2.2)$$

SBP operators are constructed precisely to reproduce this identity at the discrete level [20]. In addition to this structural property, they are required to achieve a prescribed order of accuracy. Specifically, an SBP operator of degree q satisfies the polynomial exactness condition:

$$D_x \mathbf{x}^j = j \mathbf{x}^{j-1}, \quad j \in [0, q].$$

To mirror the continuous setting, we define a discrete inner product and norm as:

$$\langle \mathbf{u}, \mathbf{v} \rangle := \mathbf{u}^T \mathbf{H} \mathbf{v}, \quad \|\mathbf{u}\|_{\mathbf{H}}^2 := \mathbf{u}^T \mathbf{H} \mathbf{u}, \quad (2.3)$$

where $\mathbf{H}$ is symmetric positive definite (SPD). With this notation, the discrete analogue of (2.2) can be expressed as:

$$\mathbf{v}^T \mathbf{H} D_x \mathbf{u} = \mathbf{u}^T \mathbf{E} \mathbf{v} - \mathbf{u}^T \mathbf{H} D_x \mathbf{v}, \quad (2.4)$$

where $\mathbf{E} = \mathbf{e}_N \mathbf{e}_N^T - \mathbf{e}_1 \mathbf{e}_1^T$, and $\mathbf{e}_1 = [1, 0, \dots, 0]^T$, $\mathbf{e}_N = [0, \dots, 0, 1]^T$ are the first and last canonical basis vectors. We can now formalize the concept of an SBP operator (see Definition 1 in [21]):

Definition 1 (Summation-by parts operator) *An operator is an approximation to the first derivative of degree q with the SBP property if:*

1. $D_x \mathbf{x}^j = \mathbf{H}^{-1} \mathbf{Q} \mathbf{x}^j = j \mathbf{x}^{j-1}, j \in [0, q],$
2. $\mathbf{H}$ is a SPD matrix,

$$\mathcal{Q} + \mathcal{Q}^T = \mathbf{E} = \text{diag}(-1, 0, \dots, 0, 1) = \mathbf{e}_N \mathbf{e}_N^T - \mathbf{e}_1 \mathbf{e}_1^T.$$

One can show that Definition 1 leads to a derivative operator $\mathbf{D}_x$ that approximates the first derivative with a truncation error of order q :

$$\mathbf{D}_x \mathbf{f} = \frac{\partial \mathcal{F}}{\partial x}(\mathbf{x}) + \boldsymbol{\tau},$$

where $\max(|\boldsymbol{\tau}|) = \mathcal{O}(h^q)$, with h denoting the mesh spacing of the discretization. Furthermore, $\mathbf{D}_x \mathbf{f}$ represents the vector of derivatives of $\mathcal{F}$ evaluated at the nodal points $\mathbf{x}$.

2.3.1 Application to the Linear Advection Equation

To illustrate the benefits of SBP operators, we consider the one-dimensional linear advection equation:

$$\frac{\partial \mathcal{U}}{\partial t} + \frac{\partial \mathcal{U}}{\partial x} = 0, \quad (2.5)$$

$$\mathcal{U}(x_L, t) = \mathcal{U}_L(t), \quad t \geq 0, \quad (2.6)$$

$$\mathcal{U}(x, 0) = \mathcal{F}(x), \quad x \in [x_L, x_R]. \quad (2.7)$$

To derive an energy estimate, we multiply the governing equation by $\mathcal{U}$ and integrate over the spatial domain:

$$\int_{x_L}^{x_R} \mathcal{U} \frac{\partial \mathcal{U}}{\partial t} dx + \int_{x_L}^{x_R} \mathcal{U} \frac{\partial \mathcal{U}}{\partial x} dx = 0.$$

We first rewrite the temporal term as:

$$\begin{aligned} \int_{x_L}^{x_R} \mathcal{U} \frac{\partial \mathcal{U}}{\partial t} dx &= \int_{x_L}^{x_R} \frac{1}{2} \frac{\partial \mathcal{U}^2}{\partial t} dx \\ &= \frac{1}{2} \frac{d}{dt} \|\mathcal{U}\|_2^2, \quad (\text{Leibniz's rule}), \end{aligned}$$

where $\|\mathcal{U}\|_2^2 := \int_{x_L}^{x_R} \mathcal{U}^2 dx$. For the spatial term, we apply IBP to yield:

$$\int_{x_L}^{x_R} \mathcal{U} \frac{\partial \mathcal{U}}{\partial x} dx = \mathcal{U}^2 \Big|_{x_L}^{x_R} - \int_{x_L}^{x_R} \mathcal{U} \frac{\partial \mathcal{U}}{\partial x} dx.$$

Combining the identical integral terms onto the left-hand side results in:

$$\int_{x_L}^{x_R} \mathcal{U} \frac{\partial \mathcal{U}}{\partial x} dx = \frac{1}{2} \mathcal{U}^2 \Big|_{x_L}^{x_R}.$$

Substituting these expressions into the integrated equation yields:

$$\frac{1}{2} \frac{d}{dt} \|\mathcal{U}\|_2^2 + \frac{1}{2} \mathcal{U}^2 \Big|_{x_L}^{x_R} = 0.$$

Multiplying both sides by 2 gives:

$$\frac{d}{dt} \|\mathcal{U}\|_2^2 = -\mathcal{U}^2(x_R, t) + \mathcal{U}^2(x_L, t).$$

Using the boundary condition $\mathcal{U}(x_L, t) = \mathcal{U}_L(t)$, we obtain:

$$\frac{d}{dt} \|\mathcal{U}\|_2^2 = -\mathcal{U}^2(x_R, t) + \mathcal{U}_L^2(t).$$

Integrating this expression in time over the interval $t \in [0, T]$ yields:

$$\|\mathcal{U}(\cdot, T)\|_2^2 - \|\mathcal{U}(\cdot, 0)\|_2^2 = - \int_0^T \mathcal{U}^2(x_R, t) dt + \int_0^T \mathcal{U}_L^2(t) dt.$$

Since the first term on the RHS is non-positive, we can obtain the upper bound:

$$\|\mathcal{U}(\cdot, T)\|_2^2 - \|\mathcal{U}(\cdot, 0)\|_2^2 \leq \int_0^T \mathcal{U}_L^2(t) dt.$$

By substituting the initial condition and defining the temporal norm of the boundary data as:

$$\|\mathcal{U}_L\|_{L^2(0, T)}^2 := \int_0^T \mathcal{U}_L^2(t) dt,$$

we arrive at the final energy estimate:

$$\|\mathcal{U}(\cdot, T)\|_2^2 \leq \|\mathcal{F}\|_2^2 + \|\mathcal{U}_L\|_{L^2(0, T)}^2.$$

Now consider the same problem discretized in space yielding the semi-discrete form:

$$\frac{d\mathbf{u}}{dt} = -\mathbf{H}^{-1} \mathbf{Q} \mathbf{u} - \tau \mathbf{H}^{-1} \mathbf{e}_1 (u_1 - \mathcal{U}_L), \quad (2.8)$$

where, as shown in [20], τ is a penalty parameter ensuring energy stability. Following the continuous analysis, we multiply (2.8) from the left by $\mathbf{u}^T \mathbf{H}$:

$$\mathbf{u}^T \mathbf{H} \frac{d\mathbf{u}}{dt} = -\mathbf{u}^T \mathbf{Q} \mathbf{u} - \tau u_1 (u_1 - \mathcal{U}_L).$$

Using the discrete energy norm (2.3), we obtain:

$$\frac{d}{dt} \|\mathbf{u}\|_{\mathbf{H}}^2 = -2\mathbf{u}^T \mathbf{Q} \mathbf{u} - 2\tau u_1 (u_1 - \mathcal{U}_L).$$

Using the identity for a scalar quantity $\mathbf{u}^T \mathbf{Q} \mathbf{u} = (\mathbf{u}^T \mathbf{Q} \mathbf{u})^T = \mathbf{u}^T \mathbf{Q}^T \mathbf{u}$, we can rewrite the matrix term as:

$$\mathbf{u}^T \mathbf{Q} \mathbf{u} = \frac{1}{2} \mathbf{u}^T (\mathbf{Q} + \mathbf{Q}^T) \mathbf{u}.$$

Substituting the SBP property $\mathbf{Q} + \mathbf{Q}^T = \mathbf{E} = \text{diag}(-1, 0, \dots, 0, 1)$, the internal matrix term collapses entirely into evaluations at the boundary nodes:

$$\begin{aligned} \mathbf{u}^T \mathbf{Q} \mathbf{u} &= \frac{1}{2} \mathbf{u}^T \mathbf{E} \mathbf{u} \\ &= \frac{1}{2} (-u_1^2 + u_N^2), \end{aligned}$$

which yields:

$$\begin{aligned} \frac{d}{dt} \|\mathbf{u}\|_{\mathbf{H}}^2 &= -(u_N^2 - u_1^2) - 2\tau u_1 (u_1 - \mathcal{U}_L) \\ &= -u_N^2 + (1 - 2\tau)u_1^2 + 2\tau u_1 \mathcal{U}_L. \end{aligned}$$

Setting the penalty parameter to $\tau = 1$ and completing the square for the remaining boundary terms gives:

$$\begin{aligned} \frac{d}{dt} \|\mathbf{u}\|_{\mathbf{H}}^2 &= -u_N^2 - (u_1 - \mathcal{U}_L)^2 + \mathcal{U}_L^2 \\ &\leq \mathcal{U}_L^2, \end{aligned}$$

where the inequality holds because the first two terms are strictly non-positive ($-u_N^2 - (u_1 - \mathcal{U}_L)^2 \leq 0$). Integrating this inequality in time over the interval $t \in [0, T]$ yields:

$$\|\mathbf{u}(T)\|_{\mathbf{H}}^2 - \|\mathbf{u}(0)\|_{\mathbf{H}}^2 \leq \int_0^T \mathcal{U}_L^2(t) dt.$$

Finally, by applying the discrete initial condition $\mathbf{u}(0) = \mathbf{f}$ and employing the temporal norm definition $\|\mathcal{U}_L\|_{L^2(0,T)}^2 := \int_0^T \mathcal{U}_L^2(t) dt$, we obtain the semi-discrete stability estimate:

$$\|\mathbf{u}(T)\|_{\mathbf{H}}^2 \leq \|\mathbf{f}\|_{\mathbf{H}}^2 + \|\mathcal{U}_L\|_{L^2(0,T)}^2,$$

which perfectly mirrors the continuous energy bound as desired.

2.4 Selected Discretization

It is now clear that the SBP property enables the construction of discretizations that mimic continuous energy estimates and provide linear stability guarantees. However, for nonlinear systems such as the Euler equations, these properties alone are not necessarily sufficient to ensure physically meaningful solutions, particularly in the presence of discontinuities. This motivates the consideration of entropy stability [35, 75, 76]. We now give a brief qualitative overview of entropy stability, with the aim of motivating entropy-stable SBP discretizations.

While energy estimates provide a natural framework for establishing stability in linear problems, an analogous approach for nonlinear systems is provided by entropy stability. Beyond its physical interpretation, entropy stability also has important analytical implications. In particular, bounds on the entropy can often be related to bounds on the solution in appropriate norms. Mimicking the continuous analysis then allows the construction of schemes that are stable and therefore robust. In this sense, entropy stability may be viewed as a nonlinear analogue of L^2 -based stability analysis for non-symmetric systems [76].

The construction of entropy-stable numerical methods has therefore been widely studied, leading to different frameworks for nonlinear stability based on entropy analysis [39, 50, 74]. Within the SBP setting, these ideas have been extended to develop high-order discretizations for compressible flow, such as the Euler equations, combining the SBP property with entropy-stable fluxes and boundary treatments [16, 19, 28, 29, 38, 73].

Hence, for this thesis we choose to discretize (2.1) in space using the entropy stable flux differencing scheme from [15], given as:

$$\frac{d\mathbf{u}}{dt} = -\mathbf{H}^{-1}((\mathbf{Q} - \mathbf{Q}^T) \circ \mathbf{F})\mathbf{1} + \mathbf{H}^{-1}\mathbf{E}\mathbf{f}^*(\mathbf{u}, \mathbf{u}^+), \quad \mathbf{F}_{ij} = \mathbf{f}_{EC}(\mathbf{u}_i, \mathbf{u}_j), \quad (2.9)$$

where $\circ$ denotes the Hadamard (entrywise) product and $\mathbf{1}$ is the vector of all ones of appropriate size. Additionally, $\mathbf{u}^+$ denotes the “exterior” solution state at an element interface or domain boundary. It is used to weakly enforce either continuity between elements or appropriate boundary conditions. Furthermore, $\mathbf{f}_{EC}(\cdot, \cdot)$ is the entropy conservative flux defined as [15]:

$$\begin{aligned} f_\rho &= \{\{\rho\}\}_{\log} \{\{au\}\}, \\ f_{\rho u} &= \{\{\rho\}\}_{\log} \{\{au\}\} \{\{u\}\} + a_L \{\{p\}\}, \\ f_E &= \frac{1}{2} \{\{\rho\}\}_{\log} \{\{au\}\} ((u \cdot u)) + \frac{1}{\gamma - 1} \{\{\rho\}\}_{\log} \{\{\rho/p\}\}_{\log}^{-1} \{\{au\}\} + ((p \cdot au)), \end{aligned} \quad (2.10)$$

where all quantities are evaluated at the left and right states of an interface. Further, the symmetric average is defined as:

$$((a \cdot b)) := \frac{1}{2}(a_L b_R + a_R b_L),$$

and the arithmetic and logarithmic means are given by:

$$\{\{a\}\} := \frac{1}{2}(a_L + a_R), \quad \{\{a\}\}_{\log} := \frac{a_L - a_R}{\log(a_L) - \log(a_R)}.$$

Finally, $\mathbf{f}^*(\cdot, \cdot)$ is an interface numerical flux taken as a local Lax-Friedrichs like penalty:

$$\mathbf{f}^*(\mathbf{u}, \mathbf{u}^+) = \mathbf{f}_{EC}(\mathbf{u}, \mathbf{u}^+) - \frac{\lambda}{2}(\mathbf{u}^+ - \mathbf{u})n, \quad \lambda > 0,$$

where $n = \pm 1$ denotes the outward normal and λ is the maximum wave speed computed as:

$$\lambda := \max(|u_L|, |u_R|) + \max(c_L, c_R), \quad c_L := \sqrt{\frac{\gamma p_L}{\rho_L}}, \quad c_R := \sqrt{\frac{\gamma p_R}{\rho_R}}.$$

Here, u_L and u_R denote the velocity components of the left and right states, respectively, while $p_{L/R}$ and $\rho_{L/R}$ are the corresponding pressure and density values. The quantities c_L and c_R denote the corresponding speeds of sound.

With the governing equations and the SBP-SAT discretization framework for the FOM in place, the foundation is set for the construction of the ROM. We now proceed to the next chapter to present the TDB-CUR algorithm, which constitutes the main computational component of the proposed optimization approach.

Chapter 3

TDB-CUR

In this chapter, we review the time-dependent basis ROM algorithm introduced by Donello et al. [25], referred to as TDB-CUR. We begin by outlining its key definitions and core components, followed by a detailed description of the algorithmic procedure and associated error control strategies. Finally, we demonstrate its application to the inviscid and viscous Burgers' equations. This approach forms the cornerstone of the thesis and serves as the primary framework for our investigation, particularly in the context of shape optimization problems discussed in Chapters 4 and 5.

3.1 Full-Order Model Setup

Consider the quasi-1D Euler equations written in a general parametric form:

$$\frac{\partial \mathcal{V}}{\partial t} = f(\mathcal{V}, x, t; \boldsymbol{\theta}),$$

augmented with appropriate boundary and initial conditions. Here, $\mathcal{V} = \mathcal{V}(x, t; \boldsymbol{\theta})$ is the state vector containing the conserved variables $[a\rho, a\rho u, aE]^T$, x denotes the spatial coordinate, t denotes time, $\boldsymbol{\theta} \in \mathbb{R}^d$ denotes the set of design parameters, and $f(\mathcal{V}, x, t; \boldsymbol{\theta})$ encodes the nonlinear flux and source terms from (2.1). Discretizing in space yields the associated matrix differential equation (MDE) which is of the form:

$$\frac{d\mathbf{V}}{dt} = \mathcal{F}(\mathbf{V}, t), \quad t \in [0, t_F]. \quad (3.1)$$

Here, $\mathcal{F}(\mathbf{V}, t)$ is the right-hand side (RHS) of the spatial discretization (2.9), which is assumed to be augmented with the discretization/sampling of $\boldsymbol{\theta}$ as well. Hence, $\mathbf{V} \in \mathbb{R}^{N \times N_s}$ is the solution matrix, where N denotes the total number of DOFs of the semi-discrete system, and N_s represents the number of parameter samples. For scalar PDEs, $N = N_x$, while for the quasi-1D Euler equations (with stacked conserved variables), $N = 3N_x$. To be explicit, the solution matrix will be of the form:

$$\mathbf{V}(t) := \begin{pmatrix} \mathcal{V}(x_1, t; \boldsymbol{\theta}_1) & \mathcal{V}(x_1, t; \boldsymbol{\theta}_2) & \cdots & \mathcal{V}(x_1, t; \boldsymbol{\theta}_{N_s}) \\ \vdots & \vdots & \vdots & \vdots \\ \mathcal{V}(x_i, t; \boldsymbol{\theta}_1) & \mathcal{V}(x_i, t; \boldsymbol{\theta}_2) & \cdots & \mathcal{V}(x_i, t; \boldsymbol{\theta}_{N_s}) \\ \vdots & \vdots & \vdots & \vdots \\ \mathcal{V}(x_{N_x}, t; \boldsymbol{\theta}_1) & \mathcal{V}(x_{N_x}, t; \boldsymbol{\theta}_2) & \cdots & \mathcal{V}(x_{N_x}, t; \boldsymbol{\theta}_{N_s}) \end{pmatrix},$$

where $\boldsymbol{\theta}_j$ denotes the j^{th} sample of the parameter vector $\boldsymbol{\theta}$. In general, Equation (3.1) is referred to as the FOM for the remainder of this thesis, and incurs a computational cost of $\mathcal{O}(NN_s)$ to evaluate at each time step of an explicit time integration scheme.

3.2 Definitions and Preliminaries

In this section we introduce some of the important definitions and notions that are required to understand the TDB-CUR algorithm.

Definition 2 (Low-rank matrix manifold) *The low-rank matrix manifold $\mathcal{M}_r$ is defined as the set:*

$$\mathcal{M}_r := \{\hat{\mathbf{V}} \in \mathbb{R}^{N \times N_s} : \text{rank}(\hat{\mathbf{V}}) = r\}$$

of matrices of fixed rank- r . In other words, $\mathcal{M}_r$ contains all rank- r matrices $\hat{\mathbf{V}}$, where $(\hat{\cdot})$ denotes a rank- r matrix.

Any member of this set $\mathcal{M}_r$ can be represented as a decomposition of three matrices of the form $\hat{\mathbf{V}} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{Y}^T$. Here, $\mathbf{U} \in \mathbb{R}^{N \times r}$ and $\mathbf{Y} \in \mathbb{R}^{N_s \times r}$ are matrices consisting of orthonormal columns, and $\boldsymbol{\Sigma} \in \mathbb{R}^{r \times r}$ is a rank- r matrix.

Definition 3 (Orthogonal projection) *Let $\mathbf{U} \in \mathbb{R}^{N \times r}$ and $\mathbf{Y} \in \mathbb{R}^{N_s \times r}$ be orthonormal matrices. Then orthogonal projectors onto the range of $\mathbf{U}$, $\text{Ran}(\mathbf{U})$, and $\mathbf{Y}$, $\text{Ran}(\mathbf{Y})$, are given by:*

$$\mathbf{P}_{\mathbf{U}}^{\perp} := \mathbf{U}\mathbf{U}^T, \text{ and } \mathbf{P}_{\mathbf{Y}}^{\perp} := \mathbf{Y}\mathbf{Y}^T,$$

where the symbol $\perp$ denotes an orthogonal projection operator.

Let $\mathbf{p} := [p_1, p_2, \dots, p_{r'}] \in \mathbb{N}^{r'}$ and $\mathbf{s} := [s_1, s_2, \dots, s_{r'}] \in \mathbb{N}^{r'}$ denote the vectors containing selected row and column indices, where $r \leq r' \ll \min\{N, N_s\}$. To stay consistent with the notation in [25], we use MATLAB indexing where $\mathbf{V}(\mathbf{p}, :) \in \mathbb{R}^{r' \times N_s}$ selects all columns at the $\mathbf{p}$ rows and $\mathbf{V}(:, \mathbf{s}) \in \mathbb{R}^{N \times r'}$ selects all rows at the $\mathbf{s}$ columns of the matrix $\mathbf{V}$. Let $\mathbf{P} := \mathbf{I}_N(:, \mathbf{p}) \in \mathbb{R}^{N \times r'}$ and $\mathbf{S} := \mathbf{I}_{N_s}(:, \mathbf{s}) \in \mathbb{R}^{N_s \times r'}$ be indexing matrices where $\mathbf{I}_N$ and $\mathbf{I}_{N_s}$ are identity matrices of size $N \times N$ and $N_s \times N_s$, respectively. Then it follows that $\mathbf{P}^T \mathbf{U} = \mathbf{U}(\mathbf{p}, :)$ and $\mathbf{S}^T \mathbf{Y} = \mathbf{Y}(\mathbf{s}, :)$. Finally, let $(\cdot)^\dagger$ denote the Moore-Penrose pseudoinverse of a matrix, i.e., $\mathbf{A}^\dagger = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T$. We can then define oblique projectors as follows [25]:

Definition 4 (Oblique projection) Let $\mathbf{U} \in \mathbb{R}^{N \times r}$ and $\mathbf{Y} \in \mathbb{R}^{N_s \times r}$ be orthonormal matrices such that $\mathbf{U}^T \mathbf{U} = \mathbf{I}$ and $\mathbf{Y}^T \mathbf{Y} = \mathbf{I}$ and let $\mathbf{p} \in \mathbb{N}^{r'}$ and $\mathbf{s} \in \mathbb{N}^{r'}$ be sets of distinct row and column indices, respectively. Oblique projectors onto $\text{Ran}(\mathbf{U})$ and $\text{Ran}(\mathbf{Y})$ are:

$$\mathbf{P}_U^\angle := \mathbf{U}(\mathbf{P}^T \mathbf{U})^\dagger \mathbf{P}^T, \text{ and } \mathbf{P}_Y^\angle := \mathbf{S}(\mathbf{Y}^T \mathbf{S})^\dagger \mathbf{Y}^T,$$

provided $(\mathbf{P}^T \mathbf{U})^T (\mathbf{P}^T \mathbf{U}) \in \mathbb{R}^{r \times r}$ and $(\mathbf{Y}^T \mathbf{S})^T (\mathbf{Y}^T \mathbf{S}) \in \mathbb{R}^{r \times r}$ are invertible.

Here, the symbol $\angle$ is used to denote the oblique projector and differentiate it from orthogonal projectors. We note that the oblique projection of any matrix $\mathbf{A}$ belongs to the manifold of rank- r matrices, i.e., $\mathbf{P}_U^\angle \mathbf{A} \mathbf{P}_Y^\angle \in \mathcal{M}_r$.

We now show that $\mathbf{P}_U^\angle$ and $\mathbf{P}_Y^\angle$ are projection operators, i.e., they are idempotent:

$$(\mathbf{P}_U^\angle)^2 = \mathbf{P}_U^\angle, \quad (\mathbf{P}_Y^\angle)^2 = \mathbf{P}_Y^\angle.$$

To this end, first consider $\mathbf{P}_U^\angle$. Using the definition of the oblique projection and the Moore-Penrose pseudoinverse to obtain:

$$\begin{aligned} (\mathbf{P}_U^\angle)^2 &= \mathbf{U}(\mathbf{P}^T \mathbf{U})^\dagger \mathbf{P}^T \mathbf{U}(\mathbf{P}^T \mathbf{U})^\dagger \mathbf{P}^T \\ &= \mathbf{U}((\mathbf{P}^T \mathbf{U})^T (\mathbf{P}^T \mathbf{U}))^{-1} (\mathbf{P}^T \mathbf{U})^T \mathbf{P}^T \mathbf{U} (\mathbf{P}^T \mathbf{U})^\dagger \mathbf{P}^T \\ &= \mathbf{U}((\mathbf{P}^T \mathbf{U})^T (\mathbf{P}^T \mathbf{U}))^{-1} ((\mathbf{P}^T \mathbf{U})^T (\mathbf{P}^T \mathbf{U})) (\mathbf{P}^T \mathbf{U})^\dagger \mathbf{P}^T \\ &= \mathbf{U}(\mathbf{P}^T \mathbf{U})^\dagger \mathbf{P}^T \\ &= \mathbf{P}_U^\angle. \end{aligned}$$

The result for $\mathbf{P}_Y^\angle$ follows analogously. For the special case when $r' = r$, the oblique projectors reduce to interpolatory projectors, since they reproduce the original matrix at

the selected rows and columns. In this context:

$$\begin{aligned} \mathbf{P}_U^\angle &= \mathbf{U}(\mathbf{P}^T \mathbf{U})^{-1} \mathbf{P}^T, \\ \mathbf{P}_Y^\angle &= \mathbf{S}(\mathbf{Y}^T \mathbf{S})^{-1} \mathbf{Y}^T. \end{aligned}$$

At the opposite extreme, when $r' = N$ (or $r' = N_s$), all rows (or columns) are selected, so that $\mathbf{P} = \mathbf{I}_N$. In this case:

$$\begin{aligned} \mathbf{P}_U^\angle &= \mathbf{U}(\mathbf{P}^T \mathbf{U})^\dagger \mathbf{P}^T \\ &= \mathbf{U}(\mathbf{I}_N^T \mathbf{U})^\dagger \mathbf{I}_N^T \\ &= \mathbf{U}(\mathbf{U})^\dagger \\ &= \mathbf{U}(\mathbf{U}^T \mathbf{U})^{-1} \mathbf{U}^T \\ &= \mathbf{U} \mathbf{U}^T \\ &= \mathbf{P}_U^\perp, \end{aligned}$$

where we used the orthonormality of $\mathbf{U}$. The same result holds for $\mathbf{P}_Y^\angle$ when $r' = N_s$.

Definition 5 (CUR decomposition) *A CUR decomposition of a matrix $\mathbf{V}$ is a rank- r approximation of $\mathbf{V}$ of the form $\mathbf{V} \approx \mathbf{C} \mathbf{U} \mathbf{R}$ where $\mathbf{C} \in \mathbb{R}^{N \times r}$ and $\mathbf{R} \in \mathbb{R}^{r \times N_s}$ contain the columns and rows of $\mathbf{V}$, i.e., $\mathbf{C} = \mathbf{V}(:, \mathbf{s})$ and $\mathbf{R} = \mathbf{V}(\mathbf{p}, :)$. Finally, the matrix $\mathbf{U} \in \mathbb{R}^{r \times r}$ is constructed such that $\mathbf{C} \mathbf{U} \mathbf{R}$ is a “good” approximation of $\mathbf{V}$.*

It can be seen that the matrices $\mathbf{C}$ and $\mathbf{R}$ are formed from selected columns and rows of the original matrix, respectively [54]. Consequently, the CUR decomposition is not unique, as it depends on both the selection of the indices and on the procedure used to compute the matrix $\mathbf{U}$.

3.3 Time-Discrete Variational Principle and Oblique Projection

The main idea behind any TDB-ROM is that its bases evolve to minimize the residual arising from low-rank approximation error. Methods such as DLRA [46], DO [67], BO [18], and DBO [59, 62] employ a time-continuous variational principle, or an analogous construction

obtained by substituting an SVD-like low-rank approximation into the FOM. Specifically, the solution is expressed in terms of the SVD-like decomposition:

$$\hat{\mathbf{V}}(t) := \mathbf{U}(t)\mathbf{\Sigma}(t)\mathbf{Y}(t)^T,$$

where $\mathbf{U}(t) \in \mathbb{R}^{N \times r}$ and $\mathbf{Y}(t) \in \mathbb{R}^{N_s \times r}$ are time-dependent orthonormal bases, and $\mathbf{\Sigma}(t) \in \mathbb{R}^{r \times r}$ is, in general, full rank. Substituting this low-rank approximation into Equation (3.1) yields the residual:

$$\mathbf{R}_{\text{es}}(t) := \frac{d(\mathbf{U}(t)\mathbf{\Sigma}(t)\mathbf{Y}(t)^T)}{dt} - \mathcal{F}(t, \mathbf{U}(t)\mathbf{\Sigma}(t)\mathbf{Y}(t)^T). \quad (3.2)$$

A time-continuous variational formulation is then obtained by minimizing the residual in Equation (3.2) subject to orthonormality constraints on $\mathbf{U}$ and $\mathbf{Y}$. This leads, via first-order optimality conditions, to evolution equations for $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{Y}$. Although this formulation provides a structured time-dependent low-rank representation that can reduce the computational cost relative to the FOM, several challenges remain. In particular, these approaches can lose efficiency in the presence of nonlinearities, may be intrusive in nature, and can exhibit stability issues in the presence of small singular values [5].

To address these issues, TDB-CUR utilizes a time-discrete variational principle for rank-adaptive matrix approximations. Consider a temporal discretization (i.e., Euler or Runge-Kutta) of the FOM in (3.1):

$$\mathbf{V}^k = \hat{\mathbf{V}}^{k-1} + \Delta t \bar{\mathbf{F}}, \quad (3.3)$$

where Δt is the time step and $\bar{\mathbf{F}}$ denotes the time-discrete evaluation of the RHS resulting from the selected temporal discretization scheme (i.e., for explicit Euler, $\bar{\mathbf{F}} = \mathcal{F}(\hat{\mathbf{V}}^{k-1}, t^{k-1})$). In this thesis, we consider only explicit temporal discretizations. For the application of TDB-CUR using implicit discretizations, we refer to [55]. From Equation (3.3), it follows that the resulting matrix is, in general, no longer contained in the rank- r manifold $\mathcal{M}_r$. This is due to the fact that even a single time step of Equation (3.3) typically maps $\mathbf{V}^k$ outside of the desired manifold. As such, a mechanism is required to project the solution back onto $\mathcal{M}_r$ at each time step.

We can achieve this by approximating $\mathbf{V}^k$ with a rank- r matrix:

$$\mathbf{V}^k = \hat{\mathbf{V}}^k + \mathbf{A}^k, \quad (3.4)$$

where $\mathbf{A}^k$ is the low-rank approximation error. Like the continuous case, the time-discrete variational principle can then be stated as finding the best $\hat{\mathbf{V}}^k \in \mathcal{M}_r$ such that the Frobenius norm of the residual is minimized:

$$\mathcal{Z}(\hat{\mathbf{V}}^k) := \|\mathbf{V}^k - \hat{\mathbf{V}}^k\|_F^2. \quad (3.5)$$

By the Eckart-Young theorem [9, 26], the solution to (3.5) is obtained via:

$$\hat{\mathbf{V}}_{best}^k = \text{SVD}(\mathbf{V}^k), \quad (3.6)$$

where $\text{SVD}(\mathbf{V}^k) := \mathbf{U}_{best}^k \Sigma_{best}^k \mathbf{Y}_{best}^{kT}$ is the best rank- r truncated SVD of the matrix $\mathbf{V}^k$. Geometrically, the $\text{SVD}(\mathbf{V}^k)$ can be thought of as the orthogonal projection of $\mathbf{V}^k$ onto the column and row bases $\mathbf{U}_{best}^k$ and $\mathbf{Y}_{best}^k$:

$$\hat{\mathbf{V}}_{best}^k = \text{SVD}(\mathbf{V}^k) = \mathbf{P}_{\mathbf{U}_{best}^k}^\perp \mathbf{V}^k \mathbf{P}_{\mathbf{Y}_{best}^k}^\perp = \mathbf{P}_{\mathbf{U}_{best}^k}^\perp (\hat{\mathbf{V}}^{k-1} + \Delta t \bar{\mathbf{F}}) \mathbf{P}_{\mathbf{Y}_{best}^k}^\perp, \quad (3.7)$$

where $\mathbf{P}_{\mathbf{U}_{best}^k}^\perp = \mathbf{U}_{best}^k \mathbf{U}_{best}^{kT}$ and $\mathbf{P}_{\mathbf{Y}_{best}^k}^\perp = \mathbf{Y}_{best}^k \mathbf{Y}_{best}^{kT}$ are orthogonal projections onto the column and row space of $\hat{\mathbf{V}}_{best}^k$. This time-discrete variational principle has several advantages such as robustness in the presence of small singular values, however, it retains the FOM's computational cost due to the evaluation of the nonlinear map $\bar{\mathbf{F}}$ as well as computing $\text{SVD}(\mathbf{V}^k)$. While approximate SVD algorithms can reduce the latter cost, $\bar{\mathbf{F}}$ remains the bottleneck. TDB-CUR addresses this by using an oblique projection scheme, representing $\hat{\mathbf{V}}^k$ as a CUR decomposition:

$$\hat{\mathbf{V}}^k = \text{CUR}(\mathbf{V}^k) := \mathbf{P}_{\mathbf{U}^k}^\angle \mathbf{V}^k \mathbf{P}_{\mathbf{Y}^k}^\angle = \mathbf{P}_{\mathbf{U}^k}^\angle (\hat{\mathbf{V}}^{k-1} + \Delta t \bar{\mathbf{F}}) \mathbf{P}_{\mathbf{Y}^k}^\angle, \quad (3.8)$$

where $\mathbf{P}_{\mathbf{U}}^\angle = \mathbf{U}(\mathbf{P}^T \mathbf{U})^\dagger \mathbf{P}^T$ and $\mathbf{P}_{\mathbf{Y}}^\angle = \mathbf{S}(\mathbf{Y}^T \mathbf{S})^\dagger \mathbf{Y}^T$ are oblique projections onto rank- r orthonormal column and row bases, $\mathbf{U}^k$ and $\mathbf{Y}^k$, respectively. Here it is important to note that since $\text{SVD}(\mathbf{V}^k)$ is considered the best rank- r approximation of $\mathbf{V}^k$, the oblique projection is not the optimal approximation. However, compared to the orthogonal projection that requires access to the entire $\hat{\mathbf{V}}^{k-1} + \Delta t \bar{\mathbf{F}}$ matrix, the oblique projectors can be designed to require the computation of $\hat{\mathbf{V}}^{k-1} + \Delta t \bar{\mathbf{F}}$ at $\mathcal{O}(r)$ rows $\mathbf{p}$ and columns $\mathbf{s}$, significantly reducing cost.

Although multiple strategies exist for selecting the index sets $\mathbf{p}$ and $\mathbf{s}$, they should be chosen in a principled manner in order to control the approximation error induced by the oblique projection. In particular, the sampling is designed such that the residual is enforced to be small at the selected interpolation points, which is achieved by requiring $\hat{\mathbf{V}}^k$ to interpolate $\mathbf{V}^k$ on the sampled rows and columns. This leads naturally to the use of interpolatory projectors, which form the basis of the TDB-CUR framework. In this work, we consider DEIM [17] and GappyPOD+E [60] as sampling strategies. In particular, DEIM has been shown to provide near-optimal sampling points for CUR-type decompositions [69]. Although TDB-CUR can be interpreted as an oblique projection method, the basis matrices $\mathbf{U}^k$ and $\mathbf{Y}^k$ are not available at the current time step. Furthermore, a direct application of standard CUR decompositions (3.8) would require the full assembly of $\mathbf{V}^k$, thereby negating the advantages of the ROM framework and effectively reverting to a full-order computation.

Additionally, certain CUR variants require inverting the intersection matrix $\mathbf{U}$, which can be highly ill-conditioned when $\mathbf{U}$ is nearly singular, leading to numerical instability and degraded approximation accuracy [58].

To circumvent these limitations, TDB-CUR employs a sequence of algorithmic steps that leverage the low-rank approximation available from the previous time step,

$$\hat{\mathbf{V}}^{k-1} = \mathbf{U}^{k-1} \Sigma^{k-1} (\mathbf{Y}^{k-1})^T.$$

Specifically, the following steps are designed to obtain the orthonormal bases $\mathbf{U}^k$ and $\mathbf{Y}^k$, from which the sparse sampling indices $\mathbf{p}$ and $\mathbf{s}$ are selected, together with the matrices $\mathbf{Q}^k$ and $\mathbf{Z}^k$ defining the factorization:

$$\hat{\mathbf{V}}^k = \mathbf{Q}^k \mathbf{Z}^k = \mathbf{U}^k \Sigma^k (\mathbf{Y}^k)^T,$$

all without explicitly constructing $\hat{\mathbf{V}}^k$. Beginning with the rank- r SVD of the initial condition $\mathbf{V}_0$, the resulting algorithm proceeds as follows:

1. Compute the sampling indices $\mathbf{s} \leftarrow \text{DEIM}(\mathbf{Y}^k)$ for the columns and $\mathbf{p} \leftarrow \text{DEIM}(\mathbf{U}^k)$ for the rows (in parallel).
2. Compute $\mathbf{V}^k(\mathbf{p}, :)$ and $\mathbf{V}^k(:, \mathbf{s})$ by taking one step according to (3.3) at the selected rows and columns.
3. Compute $\mathbf{Q} \in \mathbb{R}^{N \times r}$ as the orthonormal basis for the range of $\mathbf{V}^k(:, \mathbf{s})$ using QR decomposition such that $\mathbf{V}^k(:, \mathbf{s}) = \mathbf{Q}\mathbf{R}$ where $\mathbf{R} \in \mathbb{R}^{r \times r}$.
4. Interpolate every column of $\mathbf{V}^k$ onto the orthonormal basis $\mathbf{Q}$ at sparse indices $\mathbf{p}$:

$$\mathbf{Z} = \mathbf{Q}(\mathbf{p}, :)^{-1} \mathbf{V}^k(\mathbf{p}, :),$$

where $\mathbf{Z} \in \mathbb{R}^{r \times N_s}$ is the matrix of interpolation coefficients such that $\mathbf{Q}\mathbf{Z}$ interpolates $\mathbf{V}^k$ onto the basis $\mathbf{Q}$ at the interpolation points indexed by $\mathbf{p}$.

5. Compute the SVD of $\mathbf{Z}$ such that:

$$\mathbf{Z} = \mathbf{U}_Z \Sigma^k (\mathbf{Y}^k)^T,$$

where $\mathbf{U}_Z \in \mathbb{R}^{r \times r}$, $\Sigma^k \in \mathbb{R}^{r \times r}$, and $\mathbf{Y}^k \in \mathbb{R}^{N_s \times r}$.

6. Compute $\mathbf{U}^k \in \mathbb{R}^{N \times r}$ as the in-subspace rotation:

$$\mathbf{U}^k = \mathbf{Q}\mathbf{U}_Z.$$

3.3.1 Theoretical Equivalence

Despite the practical advantages of these steps, it is not immediately obvious how they provide exact mathematical equivalence to the oblique projection framework in (3.8). To illuminate this connection, recall the row and column selection matrices $\mathbf{P} \in \mathbb{R}^{N \times r}$ and $\mathbf{S} \in \mathbb{R}^{N_s \times r}$, which map to the sampling index vectors $\mathbf{p}$ and $\mathbf{s}$ such that $\mathbf{P}^T \mathbf{V} = \mathbf{V}(\mathbf{p}, :)$ and $\mathbf{V}\mathbf{S} = \mathbf{V}(:, \mathbf{s})$.

If we let $\hat{\mathbf{V}} = \mathbf{Q}\mathbf{Z}$ be the low-rank reconstruction generated by the TDB-CUR algorithm, we can leverage Theorems 1 and 2 and their subsequent proofs from [25] to demonstrate three core properties:

(i) $\hat{\mathbf{V}}$ is algebraically identical to the classical interpolatory CUR decomposition:

$$\text{CUR} = (\mathbf{V}\mathbf{S})(\mathbf{P}^T \mathbf{V}\mathbf{S})^{-1}(\mathbf{P}^T \mathbf{V}).$$

(ii) $\hat{\mathbf{V}}$ satisfies the exact interpolatory conditions at the sampled rows and columns:

$$\hat{\mathbf{V}}(\mathbf{p}, :) = \mathbf{V}(\mathbf{p}, :) \quad \text{and} \quad \hat{\mathbf{V}}(:, \mathbf{s}) = \mathbf{V}(:, \mathbf{s}).$$

(iii) $\hat{\mathbf{V}}$ is identically equivalent to the oblique projection framework in (3.8):

$$\hat{\mathbf{V}} = \mathbf{P}_U^\perp \mathbf{V} \mathbf{P}_Y^\perp.$$

To establish statement (i), consider Step 3 of the algorithm, where $\mathbf{Q}$ is defined as an orthonormal basis for $\text{Ran}(\mathbf{V}(:, \mathbf{s}))$. Through its QR decomposition, we can express the sampled columns as $\mathbf{V}(:, \mathbf{s}) = \mathbf{V}\mathbf{S} = \mathbf{Q}\mathbf{R}$, which directly implies that $\mathbf{P}^T \mathbf{V}\mathbf{S} = \mathbf{P}^T \mathbf{Q}\mathbf{R}$. Substituting these expressions into the classical CUR formulation explicitly yields:

$$\begin{aligned} \text{CUR} &= (\mathbf{V}\mathbf{S})(\mathbf{P}^T \mathbf{V}\mathbf{S})^{-1}(\mathbf{P}^T \mathbf{V}) \\ &= (\mathbf{Q}\mathbf{R})(\mathbf{P}^T \mathbf{Q}\mathbf{R})^{-1} \mathbf{P}^T \mathbf{V} \\ &= \mathbf{Q}\mathbf{R}\mathbf{R}^{-1}(\mathbf{P}^T \mathbf{Q})^{-1} \mathbf{P}^T \mathbf{V} \\ &= \mathbf{Q} \left(\mathbf{Q}(\mathbf{p}, :)^{-1} \mathbf{V}(\mathbf{p}, :) \right). \end{aligned}$$

Recognizing the quantity inside the parentheses as the interpolation coefficient matrix $\mathbf{Z}$ computed in Step 4, we arrive at the expected algorithmic identity:

$$\text{CUR} = \mathbf{Q}\mathbf{Z} = \hat{\mathbf{V}}.$$

Moving to property **(ii)**, we can use this verified identity to show that the reconstruction preserves exact values at our interpolation coordinates. Left-multiplying $\hat{\mathbf{V}}$ by the row-selection matrix $\mathbf{P}^T$ confirms that the selected rows are exact:

$$\begin{aligned}\hat{\mathbf{V}}(\mathbf{p}, :) &= \mathbf{P}^T \hat{\mathbf{V}} = \mathbf{P}^T (\mathbf{V}\mathbf{S})(\mathbf{P}^T \mathbf{V}\mathbf{S})^{-1} (\mathbf{P}^T \mathbf{V}) \\ &= (\mathbf{P}^T \mathbf{V}\mathbf{S})(\mathbf{P}^T \mathbf{V}\mathbf{S})^{-1} (\mathbf{P}^T \mathbf{V}) \\ &= \mathbf{P}^T \mathbf{V} = \mathbf{V}(\mathbf{p}, :).\end{aligned}$$

In an identical fashion, right-multiplying by the column-selection matrix $\mathbf{S}$ yields the column interpolation criteria:

$$\begin{aligned}\hat{\mathbf{V}}(:, \mathbf{s}) &= \hat{\mathbf{V}}\mathbf{S} = (\mathbf{V}\mathbf{S})(\mathbf{P}^T \mathbf{V}\mathbf{S})^{-1} (\mathbf{P}^T \mathbf{V}\mathbf{S}) \\ &= \mathbf{V}\mathbf{S} = \mathbf{V}(:, \mathbf{s}).\end{aligned}$$

Finally, we can connect these insights to verify property **(iii)**, which establishes that the algorithm correctly executes the oblique projection framework. We begin by expressing the left oblique projector $\mathbf{P}_U^\angle$ using the orthogonal basis $\mathbf{Q}$. Substituting the subspace rotation $\mathbf{U} = \mathbf{Q}\mathbf{U}_Z$ from Step 6 into the definition of the projector results in:

$$\begin{aligned}\mathbf{P}_U^\angle &= \mathbf{U}(\mathbf{P}^T \mathbf{U})^{-1} \mathbf{P}^T = \mathbf{Q}\mathbf{U}_Z(\mathbf{P}^T \mathbf{Q}\mathbf{U}_Z)^{-1} \mathbf{P}^T \\ &= \mathbf{Q}\mathbf{U}_Z \mathbf{U}_Z^{-1} (\mathbf{P}^T \mathbf{Q})^{-1} \mathbf{P}^T \\ &= \mathbf{Q}(\mathbf{P}^T \mathbf{Q})^{-1} \mathbf{P}^T,\end{aligned}$$

where we leverage the property that $\mathbf{U}_Z$ is a square, orthonormal matrix, meaning $\mathbf{U}_Z \mathbf{U}_Z^{-1} = \mathbf{I}$.

Next, we rewrite the row-space projector $\mathbf{P}_Y^\angle$ relative to $\mathbf{Z}$. Inverting the Step 5 SVD relations give the row bases as $\mathbf{Y}^T = \Sigma^{-1} \mathbf{U}_Z^{-1} \mathbf{Z}$. Substituting this into the definition of $\mathbf{P}_Y^\angle$ yields:

$$\begin{aligned}\mathbf{P}_Y^\angle &= \mathbf{S}(\mathbf{Y}^T \mathbf{S})^{-1} \mathbf{Y}^T = \mathbf{S}(\Sigma^{-1} \mathbf{U}_Z^{-1} \mathbf{Z}\mathbf{S})^{-1} \Sigma^{-1} \mathbf{U}_Z^{-1} \mathbf{Z} \\ &= \mathbf{S}(\mathbf{Z}\mathbf{S})^{-1} \mathbf{U}_Z \Sigma \Sigma^{-1} \mathbf{U}_Z^{-1} \mathbf{Z} \\ &= \mathbf{S}(\mathbf{Z}\mathbf{S})^{-1} \mathbf{Z}.\end{aligned}$$

Compounding these updated projection representations together, we find that the total oblique projection expands to:

$$\mathbf{P}_U^\angle \mathbf{V} \mathbf{P}_Y^\angle = (\mathbf{Q}(\mathbf{P}^T \mathbf{Q})^{-1} \mathbf{P}^T) \mathbf{V} (\mathbf{S}(\mathbf{Z}\mathbf{S})^{-1} \mathbf{Z}). \quad (3.9)$$

Using the interpolation constraints verified in part **(ii)**, the matrix intersection satisfies $\mathbf{V}(\mathbf{p}, \mathbf{s}) = \hat{\mathbf{V}}(\mathbf{p}, \mathbf{s})$. Evaluated in matrix operator form, this equivalence reads:

$$\mathbf{P}^T \mathbf{V} \mathbf{S} = \mathbf{P}^T \hat{\mathbf{V}} \mathbf{S} = \mathbf{P}^T \mathbf{Q} \mathbf{Z} \mathbf{S}.$$

Subbing this cross-term identity directly into equation (3.9) gives:

$$\begin{aligned} \mathbf{P}_U^\perp \mathbf{V} \mathbf{P}_Y^\perp &= \mathbf{Q} (\mathbf{P}^T \mathbf{Q})^{-1} (\mathbf{P}^T \mathbf{V} \mathbf{S}) (\mathbf{Z} \mathbf{S})^{-1} \mathbf{Z} \\ &= \mathbf{Q} (\mathbf{P}^T \mathbf{Q})^{-1} (\mathbf{P}^T \mathbf{Q} \mathbf{Z} \mathbf{S}) (\mathbf{Z} \mathbf{S})^{-1} \mathbf{Z} \\ &= \mathbf{Q} ((\mathbf{P}^T \mathbf{Q})^{-1} \mathbf{P}^T \mathbf{Q}) \mathbf{Z} (\mathbf{S} (\mathbf{Z} \mathbf{S})^{-1} \mathbf{Z}) \\ &= \mathbf{Q} \mathbf{I} \mathbf{Z} \\ &= \mathbf{Q} \mathbf{Z} = \hat{\mathbf{V}}. \end{aligned}$$

This completes our discussion, confirming that the multi-step execution of the TDB-CUR algorithm perfectly replicates the geometric intent of the oblique projection framework.

3.3.2 Key features of TDB-CUR

In comparison to other aforementioned TDB algorithms, TDB-CUR improves three common issues [25]:

1. **Intrusiveness:** The two primary computations required by TDB-CUR are the evaluation of $\mathbf{V}^k(\mathbf{p}, :)$ and $\mathbf{V}^k(:, \mathbf{s})$. In the case where the samples of $\boldsymbol{\theta}$ are independent, the RHS of Equation (3.1) can be treated as a black box for computing $\mathbf{V}^k(:, \mathbf{s})$. In contrast to methods such as DLRA, which require the derivation and implementation of reduced governing equations, this enables the use of the RHS without prior modification or model reformulation. However, the evaluation of $\mathbf{V}^k(\mathbf{p}, :)$ generally remains dependent on the underlying dynamics. As a result, some knowledge of the governing equations and adjacency structure is required. While this introduces additional complexity, it is still significantly less demanding than deriving reduced-order operators for each term in the governing equations.
2. **Ill-conditioning:** TDB-CUR is robust in the presence of small or even zero singular values. In particular, the algorithm does not require inversion of the singular value matrix $\boldsymbol{\Sigma}$, so instability associated with small singular values is avoided. Instead, the conditioning of the method is governed by the matrices $\mathbf{Q}(\mathbf{p}, :)$ and $\mathbf{Y}(\mathbf{s}, :)$. While

there is no guarantee that these submatrices are perfectly conditioned, greedy selection strategies such as DEIM are designed to keep quantities like $\|\mathbf{Q}(\mathbf{p}, :)^{-1}\|$ as small as possible, thereby improving numerical stability. Similarly, $\|\mathbf{Y}(\mathbf{s}, :)^{-1}\|$ plays an equally important role in ensuring a well-conditioned algorithm.

3. **Computational efficiency:** The output of the TDB-CUR algorithm requires sampling $\mathbf{V}^k$ only at r selected rows and columns, which is a major source of computational savings. Instead of computing the SVD of the full $N \times N_s$ matrix, the algorithm splits the computation into two stages: a QR decomposition of an $N \times r$ matrix followed by an SVD of the resulting $r \times N_s$ matrix $\mathbf{Z}$. This yields a computational cost of $\mathcal{O}(r^2(N + N_s))$. Compared to the full SVD cost of $\mathcal{O}(NN_s \min(N, N_s))$, this represents a substantial reduction in complexity. In addition, the TDB-CUR method avoids full-state operations and instead evaluates the dynamics using sampled entries of $\mathbf{V}^k$. Under this formulation, the dominant computational cost arises from evaluating the system at sampled indices together with reduced evolution steps, again leading to an overall complexity of $\mathcal{O}(r(N + N_s))$. Finally, the computation of $\mathbf{V}^k(:, \mathbf{s})$ can typically be performed in parallel across independent samples, providing additional computational speedup.

3.4 Additional Details of the TDB-CUR Algorithm

3.4.1 Computing $\mathbf{V}^k(\mathbf{p}, :)$

One of the more subtle aspects of the TDB-CUR algorithm is the evaluation of the temporal discretization at the selected row indices $\mathbf{p}$. In particular, updating the solution at these indices requires identifying a corresponding set of adjacent indices $\mathbf{p}_a$, which is determined by the sparsity pattern of the underlying spatial discretization. To illustrate this, consider a simple second-order centered FD operator given by:

$$\mathbf{D}_x = \frac{1}{2h} \begin{bmatrix} -2 & 2 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -2 & 2 \end{bmatrix}.$$

The adjacency structure is determined by the sparsity pattern of $\mathbf{D}_x$. Each row couples only to a small set of neighboring indices corresponding to the nonzero entries of that

row. For example, the first row depends on indices $\{1, 2\}$, while the fourth row depends on indices $\{3, 5\}$. Hence, if $\mathbf{p} = [1, 4]$, then the corresponding adjacency set is given by:

$$\mathbf{p}_a = [1, 2, 3, 5].$$

This addresses how to construct the adjacency set $\mathbf{p}_a$. We now consider how to evaluate the temporal discretization at the selected rows $\mathbf{p}$. To illustrate this, consider the second-order explicit Runge-Kutta scheme, where:

$$\bar{\mathbf{F}} = \mathcal{F} \left(t^{k-1} + \frac{1}{2}\Delta t, \hat{\mathbf{V}}^{k-1} + \frac{1}{2}\Delta t \mathcal{F}(t^{k-1}, \hat{\mathbf{V}}^{k-1}) \right).$$

After determining the index sets $\mathbf{p}$ and $\mathbf{p}_a$, $\mathbf{V}^k(\mathbf{p}, :)$ can be computed as follows:

1. Compute the reduced solution at the selected and adjacent rows:

$$\begin{aligned} \hat{\mathbf{V}}^{k-1}(\mathbf{p}, :) &= \mathbf{U}^{k-1}(\mathbf{p}, :)\Sigma^{k-1}(\mathbf{Y}^{k-1})^T, \\ \hat{\mathbf{V}}^{k-1}(\mathbf{p}_a, :) &= \mathbf{U}^{k-1}(\mathbf{p}_a, :)\Sigma^{k-1}(\mathbf{Y}^{k-1})^T. \end{aligned}$$

2. Compute the first-stage Runge-Kutta update at the selected rows:

$$\mathbf{F}_1(\mathbf{p}, :) = \mathcal{F}_{\mathbf{p}}(t^{k-1}, \hat{\mathbf{V}}^{k-1}(\mathbf{p}, :), \hat{\mathbf{V}}^{k-1}(\mathbf{p}_a, :)),$$

where $\mathcal{F}_{\mathbf{p}}$ denotes a local evaluation of the RHS at the rows indexed by $\mathbf{p}$. In particular, $\mathcal{F}_{\mathbf{p}}$ represents a reformulation of $\mathcal{F}$ that takes as input the entries of $\hat{\mathbf{V}}^{k-1}(\mathbf{p}, :)$ together with the required adjacent entries of $\hat{\mathbf{V}}^{k-1}(\mathbf{p}_a, :)$.

3. For the second-order scheme, evaluate the midpoint stage:

$$\mathbf{F}_2(\mathbf{p}, :) = \mathcal{F}_{\mathbf{p}} \left(t^{k-1} + \frac{1}{2}\Delta t, \hat{\mathbf{V}}^{k-1}(\mathbf{p}, :) + \frac{1}{2}\Delta t \mathbf{F}_1(\mathbf{p}, :), \hat{\mathbf{V}}^{k-1}(\mathbf{p}_a, :) + \frac{1}{2}\Delta t \mathbf{F}_1(\mathbf{p}_a, :) \right).$$

One important consequence of the previous construction is that evaluating the second-stage term requires access to $\mathbf{F}_1(\mathbf{p}_a, :)$. While this could, in principle, be computed using the same procedure as in Step 2, doing so would require introducing an additional adjacency set $\mathbf{p}_{aa}$ for $\mathbf{p}_a$. This recursive expansion quickly becomes impractical, particularly for higher-order time integration schemes, where the number of required adjacency layers grows with each stage. As a result, both the computational cost and implementation complexity could increase significantly.

To address this issue, the authors instead construct a low-rank approximation $\hat{\mathbf{F}}_i \approx \mathbf{F}_i$ at each stage of the time integrator. This approximation is formed using only the available sparse measurements $\mathbf{F}_i(\mathbf{p}, :)$ and $\mathbf{F}_i(:, \mathbf{s})$, which are already computed as part of the TDB-CUR algorithm. The procedure is as follows:

1. Compute an orthonormal basis $\mathbf{U}_{F_i}$ for the range of $F_i(:, \mathbf{s})$ via a QR factorization:

$$\mathbf{U}_{F_i} \mathbf{R}_{F_i} = \mathbf{QR}(F_i(:, \mathbf{s})).$$

2. Form a low-rank approximation of F_i via an oblique projection:

$$\hat{F}_i = \mathbf{U}_{F_i} (\mathbf{U}_{F_i}(\mathbf{p}, :))^{-1} F_i(\mathbf{p}, :),$$

Using this approximation, $F_i(\mathbf{p}_a, :)$ can be approximated without introducing additional adjacency sets:

$$F_i(\mathbf{p}_a, :) \approx \hat{F}_i(\mathbf{p}_a, :) = \mathbf{U}_{F_i}(\mathbf{p}_a, :) (\mathbf{U}_{F_i}(\mathbf{p}, :))^{-1} F_i(\mathbf{p}, :).$$

This construction naturally extends to higher-order time integration schemes and can be shown to be equivalent to a CUR decomposition of the matrix F_i [56].

3.4.2 Error Reduction (Oversampling and Rank Adaptivity)

From Theorem 2.8 in [25], the error bound of TDB-CUR is given by:

$$\|\mathbf{V} - \mathbf{P}_U^\top \mathbf{V} \mathbf{P}_Y^\top\|_2 \leq (1 + \epsilon_f) \hat{\sigma}_{r+1}, \quad (3.10)$$

where $\epsilon_f \geq 0$ is defined as:

$$\epsilon_f := \min\{\eta_p(1 + \eta_s), \eta_s(1 + \eta_p)\} - 1,$$

with $\eta_p := \|(\mathbf{P}^T \mathbf{U})^\dagger\|_2$ and $\eta_s := \|(\mathbf{S}^T \mathbf{Y})^\dagger\|_2$. Moreover,

$$\hat{\sigma}_{r+1} := \max\{\|\mathbf{V} - \mathbf{U} \mathbf{U}^T \mathbf{V}\|_2, \|\mathbf{V} - \mathbf{V} \mathbf{Y} \mathbf{Y}^T\|_2\}.$$

This bound shows that the error of the oblique projection is controlled by the optimal rank- r approximation error, i.e., the orthogonal projection error, scaled by an amplification factor $(1 + \epsilon_f)$. The magnitude of this amplification depends on η_p and η_s , which reflect the conditioning of $\mathbf{P}^T \mathbf{U}$ and $\mathbf{S}^T \mathbf{Y}$, respectively. In particular, better conditioning, i.e., smaller values of η_p and η_s , leads to a tighter error bound.

This observation regarding the role of η_p and η_s in the amplification factor motivates the extension of TDB-CUR to incorporate row oversampling. In particular, in [25] they consider selecting $r' = r + m \ll \min\{N, N_s\}$, where m denotes the number of additional sampling points. This extension is motivated by results from the DEIM framework,

where oversampling has been shown to improve the conditioning of oblique projections [86]. Specifically, choosing $m = \mathcal{O}(r)$ can reduce the value of η_p , and consequently tighten the associated error bound.

The introduction of oversampling leads to a modification of the underlying system, which becomes overdetermined. As a result, the solution is obtained via a least-squares formulation rather than interpolation. The resulting changes to the algorithm are given by:

1. $Z = Q(\mathbf{p}, :)^{-1} V^k(\mathbf{p}, :) \rightarrow Z = Q(\mathbf{p}, :)^{\dagger} V^k(\mathbf{p}, :)$
2. $\hat{F}_i = U_{F_i} U_{F_i}(\mathbf{p}, :)^{-1} F_i(\mathbf{p}, :) \rightarrow \hat{F}_i = U_{F_i} U_{F_i}(\mathbf{p}, :)^{\dagger} F_i(\mathbf{p}, :)$
3. $\text{DEIM}(\mathbf{U}, r) \rightarrow \text{GappyPOD}(\mathbf{U}, r + m)$

The modification in 3 arises from the fact that DEIM selects a number of sampling points equal to the number of basis vectors, and therefore does not naturally accommodate oversampling. Consequently, an alternative sampling procedure is required. One such approach is the GappyPOD+E algorithm introduced in [60].

Another strategy explored in TDB-CUR for controlling the approximation error is the adaptive selection of the rank. Since the true error cannot be evaluated at each time step without reconstructing the FOM, an error proxy is introduced based on the preceding error analysis to estimate the low-rank approximation error. This proxy is defined as:

$$\epsilon(t) := \frac{\hat{\sigma}_r(t)}{\|\hat{\mathbf{V}}(t)\|_F} = \frac{\hat{\sigma}_r(t)}{(\sum_{i=1}^r \hat{\sigma}_i(t)^2)^{1/2}}, \quad (3.11)$$

where $\hat{\sigma}_i(t)$ denotes the singular values obtained at the previous time step. Rather than prescribing a fixed tolerance, the rank is adapted dynamically to maintain the proxy within a specified range $\epsilon_l \leq \epsilon(t) \leq \epsilon_u$, where ϵ_l and ϵ_u are user-defined lower and upper bounds, respectively. If $\epsilon(t) > \epsilon_u$, the rank is increased to $r + 1$, whereas if $\epsilon(t) < \epsilon_l$, the rank is decreased to $r - 1$. The use of a bounded interval, as opposed to a single threshold, mitigates oscillatory behavior in which modes are repeatedly added and removed. It is worth noting that this proxy is not unique, alternative formulations may be constructed and could be more appropriate depending on the specific problem setting (see, e.g., [78]).

Algorithm 1 TDB-CUR with Row Oversampling and Rank Adaptivity

Input: $\check{\mathbf{U}} \in \mathbb{R}^{N \times \check{r}}, \check{\mathbf{\Sigma}} \in \mathbb{R}^{\check{r} \times \check{r}}, \check{\mathbf{Y}} \in \mathbb{R}^{N_s \times \check{r}}, \check{r}, m$ ▷ ($\check{\cdot}$) Indicates the previous time step
Output: $\mathbf{U} \in \mathbb{R}^{N \times r}, \mathbf{\Sigma} \in \mathbb{R}^{r \times r}, \mathbf{Y} \in \mathbb{R}^{N_s \times r}, r$
 $\epsilon = \check{\mathbf{\Sigma}}(\check{r}, \check{r}) / \|\text{diag}(\check{\mathbf{\Sigma}})\|_2$ ▷ Error proxy for rank adaptivity
if $\epsilon > \epsilon_u$ **then** ▷ Increase r if ϵ is greater than ϵ_u
 $r = \check{r} + 1$
else if $\epsilon < \epsilon_l$ **then** ▷ Decrease r if ϵ is less than ϵ_l
 $r = \check{r} - 1$
 $\check{\mathbf{U}} = \check{\mathbf{U}}(:, : r), \check{\mathbf{\Sigma}} = \check{\mathbf{\Sigma}}(1 : r, 1 : r), \check{\mathbf{Y}} = \check{\mathbf{Y}}(:, 1 : r)$ ▷ Truncate the SVD components
else
 $r = \check{r}$
end if
 $\mathbf{s} \leftarrow \text{sparseAlgorithm}(\check{\mathbf{Y}}, r)$ ▷ Compute column indices
 $\mathbf{p} \leftarrow \text{sparseAlgorithm}(\check{\mathbf{U}}, r + m)$ ▷ Compute row indices with m oversampling points
 $\mathbf{p}_a \leftarrow \text{findAdjacent}(\mathbf{p})$ ▷ Some function that computes the adjacent points, given $\mathbf{p}$
 $\hat{\mathbf{V}}(:, \mathbf{s}) = \check{\mathbf{U}}\check{\mathbf{\Sigma}}\check{\mathbf{Y}}(\mathbf{s}, :)^T$ ▷ Compute the low-rank approximation at the $\mathbf{s}$ columns
 $\hat{\mathbf{V}}(\mathbf{p}, :) = \check{\mathbf{U}}(\mathbf{p}, :)\check{\mathbf{\Sigma}}\check{\mathbf{Y}}^T$ ▷ Compute the low-rank approximation at the $\mathbf{p}$ rows
 $\hat{\mathbf{V}}(\mathbf{p}_a, :) = \check{\mathbf{U}}(\mathbf{p}_a, :)\check{\mathbf{\Sigma}}\check{\mathbf{Y}}^T$ ▷ Compute the low-rank approximation at the $\mathbf{p}_a$ rows
 $\mathbf{V}(:, \mathbf{s}) = \hat{\mathbf{V}}(:, \mathbf{s}) + \Delta t \bar{\mathbf{F}}(:, \mathbf{s})$ ▷ Take one step at selected columns
 $\mathbf{V}(\mathbf{p}, :) = \hat{\mathbf{V}}(\mathbf{p}, :) + \Delta t \bar{\mathbf{F}}(\mathbf{p}, :)$ ▷ Take one step at selected rows
 $\mathbf{QR} = \mathbf{QR}(\mathbf{V}(:, \mathbf{s}), \text{'econ'})$ ▷ Compute the economy QR of $\mathbf{V}(:, \mathbf{s})$
 $\mathbf{Z} = \mathbf{Q}(\mathbf{p}, :)^{\dagger} \mathbf{V}(\mathbf{p}, :)$ ▷ Compute $\mathbf{Z}$ as the oblique projection of $\mathbf{V}$ onto $\mathbf{Q}$
 $\mathbf{U}_Z, \mathbf{\Sigma}, \mathbf{Y}^T = \mathbf{SVD}(\mathbf{Z}, \text{'econ'})$ ▷ Compute the economy SVD of $\mathbf{Z}$
 $\mathbf{U} = \mathbf{Q}\mathbf{U}_Z$ ▷ Compute the in subspace rotation of the orthonormal basis $\mathbf{Q}$

3.4.3 TDB-CUR Algorithm with Rank Adaptivity and Oversampling

The complete TDB-CUR algorithm, incorporating both oversampling and rank adaptivity, is presented in Algorithm 1. In this work, the GappyPOD+E method is employed as the `sparseAlgorithm`, however, alternative choices may be used depending on the application. A second aspect that requires careful consideration is the computation of $\bar{\mathbf{F}}_i$, i.e., the evaluation of each stage of the time integrator at the selected rows $\mathbf{p}$ and $\mathbf{p}_a$, and columns $\mathbf{s}$. The update at the selected columns $\mathbf{s}$ is straightforward, as the RHS can be evaluated directly according to the chosen spatial discretization. The primary consideration in this step is the consistent evaluation of the SAT terms to ensure that boundary conditions are properly enforced at the indices corresponding to $\mathbf{s}$. For the indices $\mathbf{p}_a$, the update is obtained by computing the low-rank approximation, as described in Section 3.4.1. The most challenging component is the evaluation of the RHS at the selected rows $\mathbf{p}$. In particular, the computation of the volume terms must be carried out without using the full RHS. To achieve this, a common approach is to utilize the approximations $\hat{\mathbf{V}}_{\mathbf{p}}$ and $\hat{\mathbf{V}}_{\mathbf{p}_a}$ together with a restricted derivative operator of the form $\mathbf{D}_x(\mathbf{p}, :)(:, \mathbf{p}_a)$.

Finally, the treatment of the boundary SAT terms requires additional care. In particular, it must be ensured that the boundary indices are included in the sampling set $\mathbf{p}$. Taking all of these considerations into account, one possible procedure for computing $\bar{\mathbf{F}}_i$ for $\mathbf{s}, \mathbf{p}$, and $\mathbf{p}_a$, is given in Algorithm 2. Additional implementation details of the TDB-CUR algorithm, focusing on its application to the quasi-1D Euler equations, are provided in Appendix A.

Algorithm 2 Computing the i^{th} stage of the time integrator F_i at the selected rows and columns, $\mathbf{s}, \mathbf{p}$, and $\mathbf{p}_a$

Input: $t, D_x, \hat{V}_s, \hat{V}_p, \hat{V}_{p_a}, \mathbf{s}, \mathbf{p}, \mathbf{p}_a, \mathbf{g}_L, \mathbf{g}_R$

Output: $F_s, F_p, \hat{F}_{p_a}$ $\triangleright$ Updated stage of the time integrator at the selected indices

$F_s \leftarrow \text{RHS}(t, D_x, \hat{V}_s)$ $\triangleright$ Compute the volume terms at the $\mathbf{s}$ columns

$F_s(1, :) \leftarrow \text{SAT}(\hat{V}_s, \mathbf{g}_L(\mathbf{s}))$ $\triangleright$ Compute the left boundary SAT

$F_s(\text{end}, :) \leftarrow \text{SAT}(\hat{V}_s, \mathbf{g}_R(\mathbf{s}))$ $\triangleright$ Compute the right boundary SAT

$F_p \leftarrow \text{RHS}_p(t, D_x(\mathbf{p}, :)(:, \mathbf{p}_a), \hat{V}_p, \hat{V}_{p_a})$ $\triangleright$ Compute the volume terms at the $\mathbf{p}$ rows

if 1 in $\mathbf{p}$ **then**

$F_p(1, :) \leftarrow \text{SAT}(\hat{V}_p, \mathbf{g}_L)$ $\triangleright$ Compute the left boundary SAT

end if

if N in $\mathbf{p}$ **then**

$F_p(\text{end}, :) \leftarrow \text{SAT}(\hat{V}_p, \mathbf{g}_R)$ $\triangleright$ Compute the right boundary SAT

end if

$U_{F_s} R_{F_s} = \text{QR}(F_s)$ $\triangleright$ Get the orthonormal basis of F_s

$\hat{F}_{p_a} = U_{F_s}(\mathbf{p}_a, :) U_{F_s}(\mathbf{p}, :)^{\dagger} F_p$ $\triangleright$ Get the low-rank approximation of F_{p_a}

Note that $\mathbf{g}_L$ and $\mathbf{g}_R$ denote the vectors corresponding to the left and right boundary conditions, respectively, each of length N_s . Additionally, RHS_p refers to a reformulation of RHS designed to compute the volume terms for the selected indices $\mathbf{p}$, as briefly discussed in Section 3.4.1.

3.4.4 Application to the Inviscid and Viscous Burgers' Equation

In this section, the TDB-CUR algorithm is applied to both the inviscid and viscous Burgers' equations in order to illustrate its key features. In particular, the focus is on how different dynamical behaviours influence the solution structure and how TDB-CUR adapts to these variations.

To demonstrate the algorithm in practice, we first consider the inviscid Burgers' equation of the form:

$$\begin{aligned} \frac{\partial \mathcal{V}}{\partial t} + \mathcal{V} \frac{\partial \mathcal{V}}{\partial x} &= 0, \\ \mathcal{V}(x, 0; \boldsymbol{\theta}) &= \boldsymbol{\theta} \sin(2\pi x), \quad x \in [0, 1], \quad \boldsymbol{\theta} \in [0.1, 1.0], \\ \mathcal{V}(0, t; \boldsymbol{\theta}) &= \mathcal{V}(1, t; \boldsymbol{\theta}), \quad t \in [0, 1]. \end{aligned}$$

For this problem, we employ a global continuous Galerkin (CG) discretization using polynomial degree $q = 3$ over $k = 150$ elements, resulting in a total of $N = 451$ DOFs. The semi-discrete system is written in split form as:

$$\frac{d\mathbf{V}}{dt} = -\frac{1}{3}\mathbf{D}_x\mathbf{A}\mathbf{V} - \frac{1}{3}\mathbf{A}\mathbf{D}_x\mathbf{V} + \mathbf{SAT}^I,$$

where in general $\mathbf{A} := \text{diag}(\mathbf{V})$. The corresponding inviscid SAT term given by:

$$\begin{aligned} \mathbf{SAT}^I = \mathbf{H}^{-1} & \left(-\mathbf{e}_1 \left(\frac{1}{3}\mathbf{V}_1^2 - \frac{1}{6}\mathbf{V}_1\mathbf{V}_N - \frac{1}{6}\mathbf{V}_N^2 \right) \right. \\ & \left. + \mathbf{e}_N \left(\frac{1}{3}\mathbf{V}_N^2 - \frac{1}{6}\mathbf{V}_N\mathbf{V}_1 - \frac{1}{6}\mathbf{V}_1^2 \right) \right). \end{aligned}$$

Here, $\mathbf{V} \in \mathbb{R}^{N \times N_s}$, where $N_s = 400$ uniform samples of $\boldsymbol{\theta} \in [0.1, 1.0]$ are considered. The boundary values are defined as $\mathbf{V}_1 = \mathbf{V}(1, :)$ and $\mathbf{V}_N = \mathbf{V}(\text{end}, :)$. Since we are first considering the inviscid Burgers' equation, shock formation is expected. To mitigate the associated high-frequency oscillations, a dissipation operator is introduced to smooth the solution [38]. The impact of such dissipation on the implementation of the TDB-CUR algorithm is discussed in Appendix A.4.1. Finally, time integration is performed using a classical fourth-order Runge-Kutta (RK4) scheme with a time step of $\Delta t = 1.0 \times 10^{-4}$.

For this example, we analyze the singular value decay, the maximum absolute error between the FOM and ROM, and the mean solution at five time instances: $t = 0$, $t = 0.25$, $t = 0.5$, $t = 0.75$, and $t = 1.0$. The mean solution is defined as the average of the N_s sampled solutions:

$$\mathbf{u}_{\text{mean}} := \frac{1}{N_s} \sum_{i=1}^{N_s} \mathbf{V}(:, i). \quad (3.12)$$

Using the low-rank representation $\hat{\mathbf{V}} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{Y}^T$, this may be computed as:

$$\mathbf{u}_{\text{mean}} := \frac{1}{N_s} \sum_{i=1}^{N_s} (\mathbf{U}\boldsymbol{\Sigma}\mathbf{Y}^T)(:, i), \quad (3.13)$$

where $(\mathbf{U}\boldsymbol{\Sigma}\mathbf{Y}^T)(:, i)$ denotes the i^{th} column of the reconstructed solution matrix.

One of the key aspects to examine is the evolution of the singular values over time, as this provides insight into the underlying dimensionality of the system and its suitability

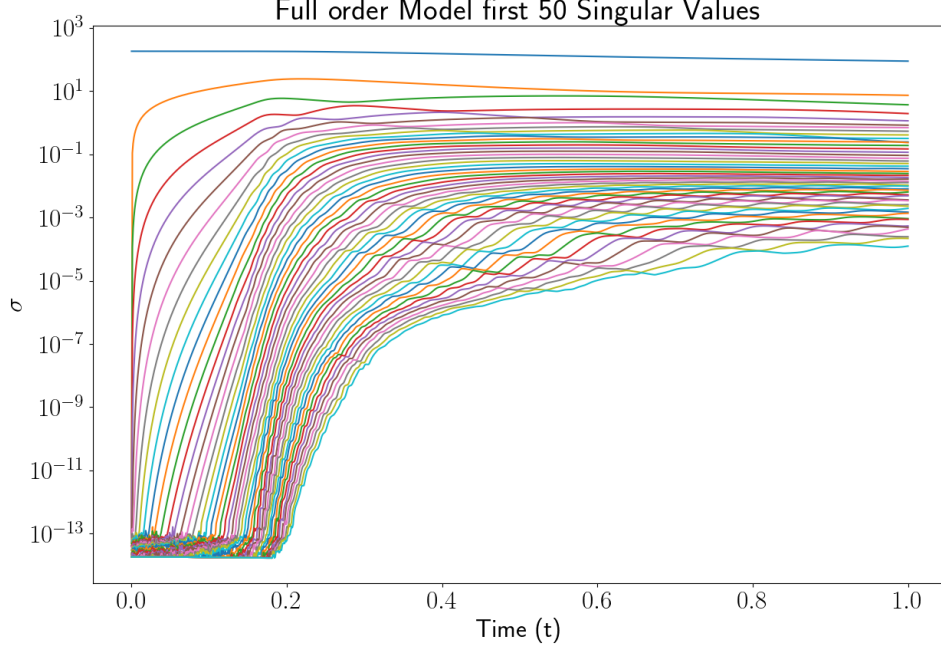


Figure 3.1: The evolution of the first 50 singular values of the FOM solution matrix over time for the split-form discretization of the inviscid Burgers' equation.

for ROMs. From Figure 3.1, it can be observed that the leading singular values are well separated initially and remain dominant throughout the simulation. However, as time evolves, the lower-energy modes become increasingly clustered, indicating a gradual rise in the observed rank. This increase in rank can be attributed primarily to the parametric sensitivity of the shock formation, where the variations in the parameter θ directly dictate both how sharply the wave steepens and exactly when the shock develops. In addition, the presence of spurious numerical oscillations, as seen in Figure 3.2, contributes to this behaviour, as these high-frequency features are interpreted by the ROM as dynamically relevant structures, thereby enriching the observed rank of the system. Given this observed trend, a natural question is how the rank adaptivity strategy will respond to this evolving spectral structure, and whether it is capable of accommodating the sharp increase in rank exhibited by the FOM.

For this problem, we consider an initial rank of $r = 10$ and $m = 5$ oversampling points. The four cases of the rank-adaptive bounds are defined as follows:

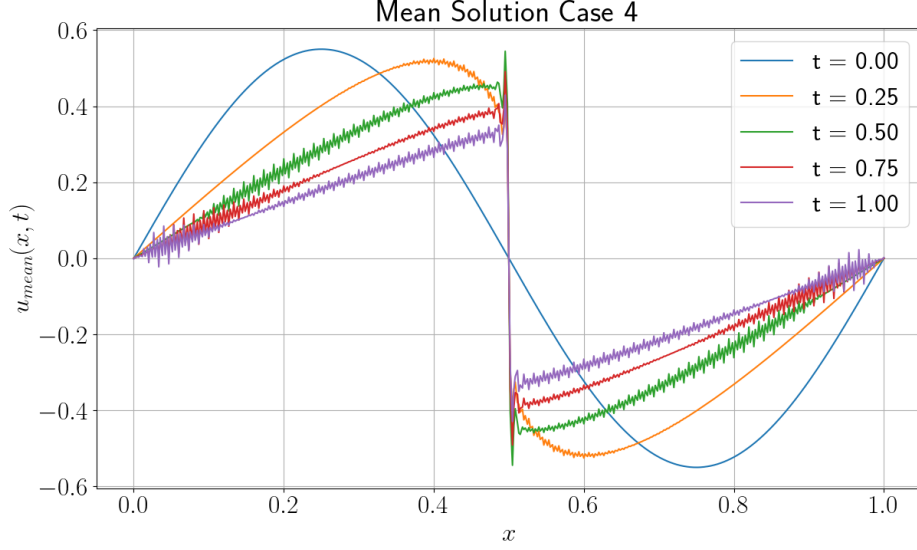


Figure 3.2: The mean solution, as defined in Equation (3.13), for the inviscid Burgers' equation, evaluated at five distinct time instances $t = 0, 0.25, 0.5, 0.75, 1.0$ with $N_s = 400$. The results correspond to the fourth rank-adaptivity configuration, with $\epsilon_u = 10^{-10}$ and $\epsilon_l = 10^{-12}$. No visually discernible difference was observed in the mean solution across the four tested configurations.

1. **Case 1:** $\epsilon_u = 10^{-4}$, $\epsilon_l = 10^{-6}$.
2. **Case 2:** $\epsilon_u = 10^{-6}$, $\epsilon_l = 10^{-8}$.
3. **Case 3:** $\epsilon_u = 10^{-8}$, $\epsilon_l = 10^{-10}$.
4. **Case 4:** $\epsilon_u = 10^{-10}$, $\epsilon_l = 10^{-12}$.

For each case, we compute the maximum absolute error between the FOM and ROM, defined as:

$$e := \max \left| \mathbf{V} - \hat{\mathbf{V}} \right|, \quad (3.14)$$

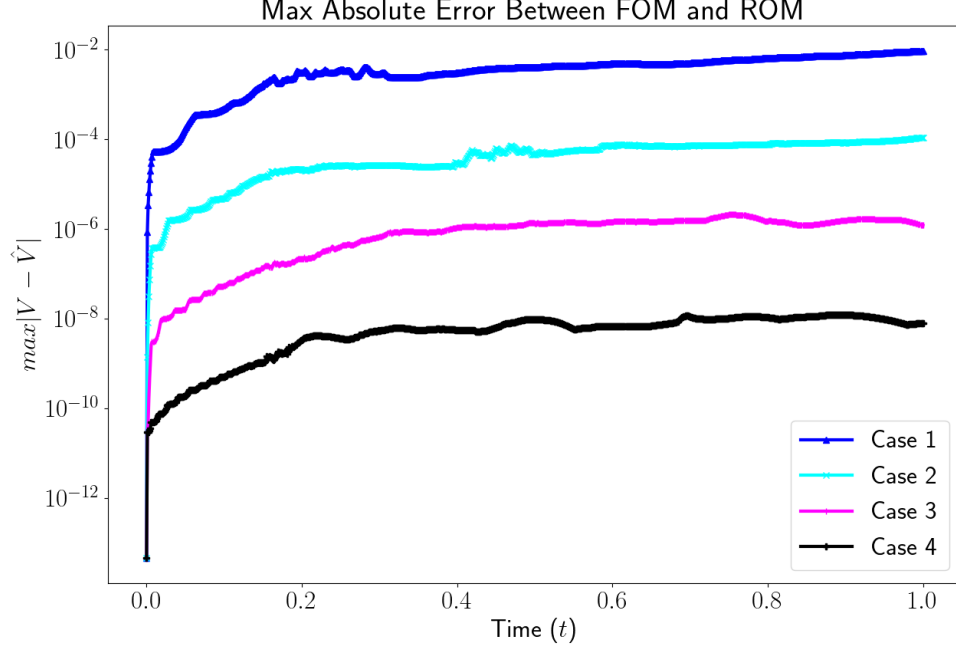


Figure 3.3: The maximum absolute error between the FOM and ROM, as defined in (3.14), for the four rank-adaptive cases considered. Case 1 corresponds to $\epsilon_u = 10^{-4}$, $\epsilon_l = 10^{-6}$; Case 2 to $\epsilon_u = 10^{-6}$, $\epsilon_l = 10^{-8}$; Case 3 to $\epsilon_u = 10^{-8}$, $\epsilon_l = 10^{-10}$; and Case 4 to $\epsilon_u = 10^{-10}$, $\epsilon_l = 10^{-12}$.

where $\mathbf{V}$ denotes the FOM solution, and $\hat{\mathbf{V}}$ the corresponding ROM solution obtained from TDB-CUR. Figure 3.3 shows a clear separation in the error levels across the four cases, illustrating the effectiveness of the proposed error proxy in controlling the ROM accuracy. In particular, imposing tighter bounds on the rank adaptivity leads to a reduction in the error of approximately two orders of magnitude between successive cases.

However, the caveat is that achieving such accurate solutions requires an increased number of modes, as illustrated in Figure 3.4, which shows the evolution of the rank r over time. In particular, Case 1 reaches a final rank of $r = 28$. This increases to $r = 53$ in Case 2, $r = 67$ in Case 3, and $r = 89$ in Case 4. Although this represents a significant increase across the cases, it remains substantially lower than the FOM, which requires the update of the full matrix $\mathbf{V} \in \mathbb{R}^{450 \times 400}$ at every time step. In contrast, the ROM at its largest rank requires updates of two sub-matrices of size $(89 + 5) \times 400$ and 450×89 , respectively.

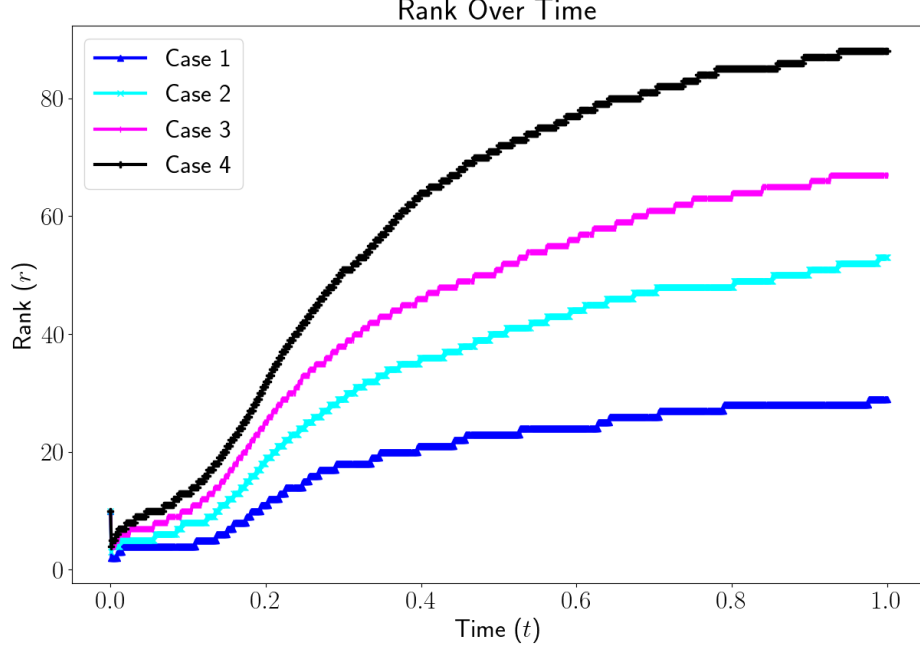


Figure 3.4: The evolution of the rank r over time for the four rank-adaptive cases. The initial rank is taken as $r = 10$ for all cases. Case 1 corresponds to $\epsilon_u = 10^{-4}$, $\epsilon_l = 10^{-6}$; Case 2 to $\epsilon_u = 10^{-6}$, $\epsilon_l = 10^{-8}$; Case 3 to $\epsilon_u = 10^{-8}$, $\epsilon_l = 10^{-10}$; and Case 4 to $\epsilon_u = 10^{-10}$, $\epsilon_l = 10^{-12}$.

Furthermore, if the rank evolution is compared with the singular value decay, it can be observed that increases in rank coincide with regions where additional modes gain relative significance in the energy spectrum.

Now consider a second example in which the TDB-CUR algorithm is applied to the viscous Burgers' equation of the form:

$$\begin{aligned} \frac{\partial \mathcal{V}}{\partial t} + \mathcal{V} \frac{\partial \mathcal{V}}{\partial x} &= \nu \frac{\partial^2 \mathcal{V}}{\partial x^2}, \\ \mathcal{V}(x, 0; \boldsymbol{\theta}) &= \boldsymbol{\theta} \sin(2\pi x), \quad x \in [0, 1], \quad \boldsymbol{\theta} \in [0.1, 1.0], \\ \mathcal{V}(0, t; \boldsymbol{\theta}) &= \mathcal{V}(1, t; \boldsymbol{\theta}), \quad t \in [0, 1]. \end{aligned}$$

where $\nu = 2.5 \times 10^{-2}$. For this problem, we again employ global CG operators, this time using degree $q = 2$ on $k = 150$ elements, resulting in a total of $N = 301$ DOFs. The

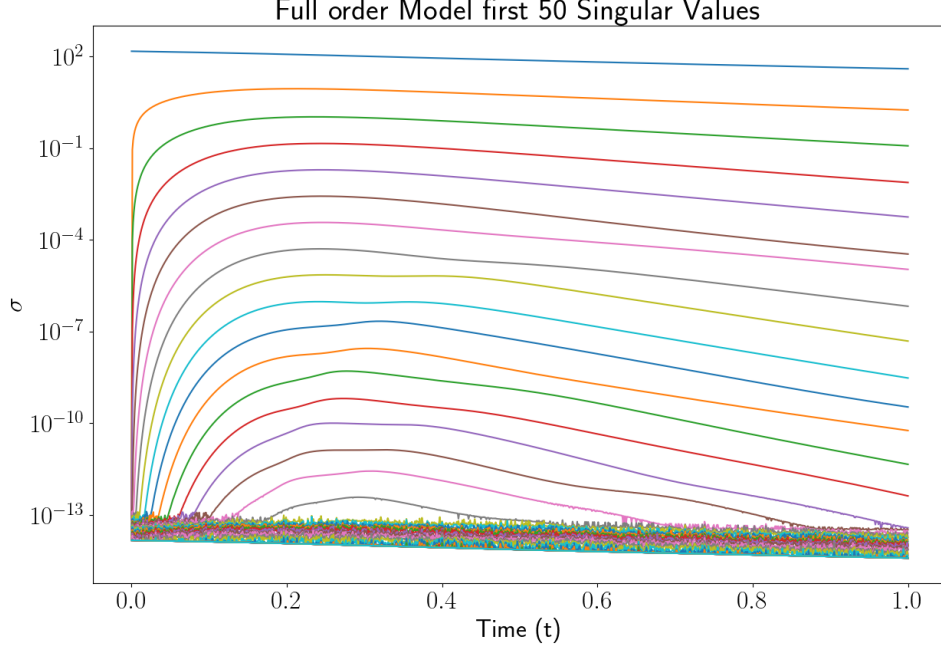


Figure 3.5: The evolution of the first 50 singular values of the FOM solution matrix over time for the split-form discretization of the viscous Burgers' equation.

semi-discrete system is discretized in split form as:

$$\frac{dV}{dt} = -\frac{1}{3}D_xAV - \frac{1}{3}AD_xV + \nu D_{xx}V + SAT^I + SAT^V,$$

where D_{xx} is computed as $D_{xx} := D_x D_x$, and SAT^V denotes the viscous penalty terms as defined in [81].

Since no shock formation is expected for this case, no additional dissipation operator is included. Time integration is performed using a classical RK4 scheme with a time step of $\Delta t = 1.0 \times 10^{-4}$. The same TDB-CUR parameters are used as in the inviscid case, with $N_s = 400$ samples of θ , an initial rank of $r = 10$, and $m = 5$ oversampling points. From the evolution of the singular values shown in Figure 3.5, a clear difference in behaviour is observed compared to the inviscid case. The decay of the singular values is significantly faster, with the spectrum reaching machine precision by approximately the 16th mode. Moreover, no pronounced clustering or increase in higher modes is observed.

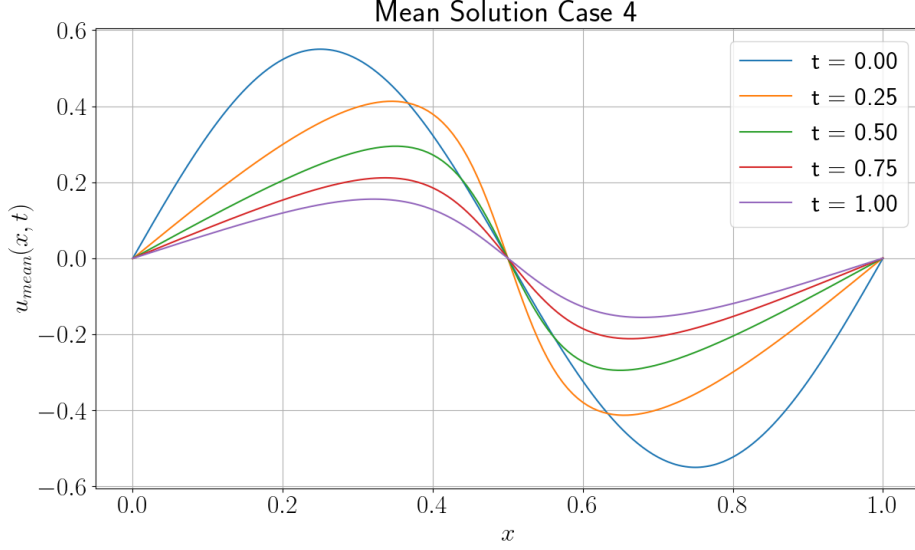


Figure 3.6: The mean solution, as defined in (3.13), for the viscous Burgers' equation, evaluated at five distinct time instances $t = 0, 0.25, 0.5, 0.75, 1.0$ with $N_s = 400$. The results correspond to the fourth rank-adaptive configuration, with $\epsilon_u = 10^{-10}$ and $\epsilon_l = 10^{-12}$. No visually discernible difference was observed in the mean solution across the four tested configurations.

This behaviour is consistent with the smoothness of the solution, as illustrated in Figure 3.6, where no discontinuities or sharp gradients are present.

Hence, if we examine the maximum absolute error and rank over time using the same four cases as in the inviscid problem we can see some of these observations reflected in the plots. In particular, in Figure 3.7, we again observe a separation between the four cases, with approximately two orders of magnitude difference between successive cases.

However, a key difference is observed in the rank evolution. In Figure 3.8, the final ranks are significantly closer across the cases and do not exhibit the pronounced growth seen in the inviscid case. In particular, Case 1 reaches a final rank of $r = 4$, whereas Case 4 attains a maximum rank of $r = 13$. This indicates that only nine additional modes are required to achieve an error that is orders of magnitude smaller than that of the least

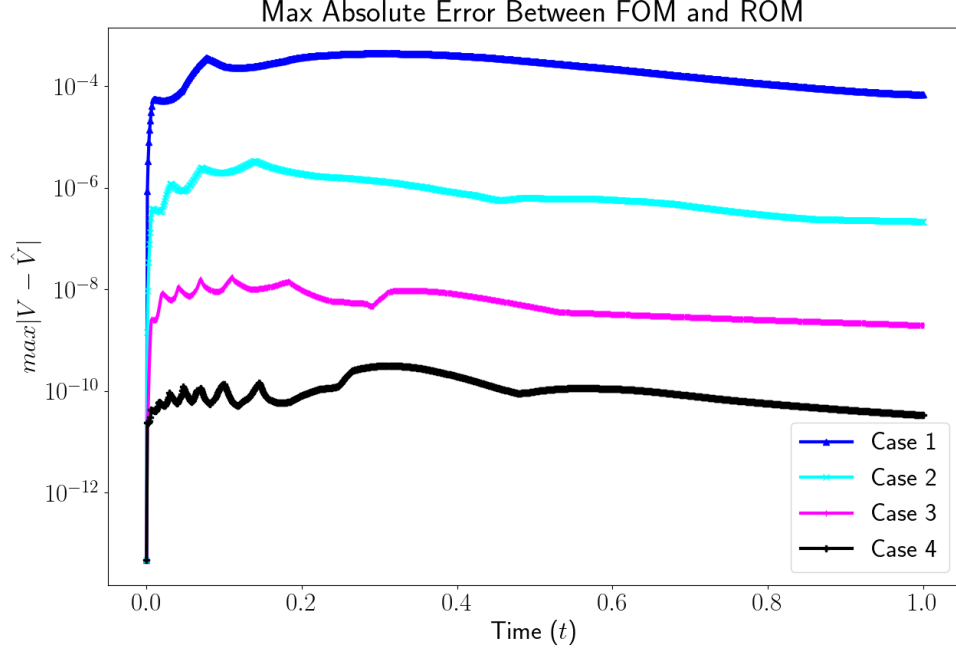


Figure 3.7: The maximum absolute error between the FOM and ROM, as defined in (3.14), for the four rank-adaptive cases considered. Case 1 corresponds to $\epsilon_u = 10^{-4}$, $\epsilon_l = 10^{-6}$; Case 2 to $\epsilon_u = 10^{-6}$, $\epsilon_l = 10^{-8}$; Case 3 to $\epsilon_u = 10^{-8}$, $\epsilon_l = 10^{-10}$; and Case 4 to $\epsilon_u = 10^{-10}$, $\epsilon_l = 10^{-12}$.

accurate case.

In this chapter, we have presented the key components of the TDB-CUR algorithm and established the necessary framework for its application to a range of problems. Two representative examples were also provided, illustrating how the characteristics of the underlying system influence the suitability and performance of the ROM, and highlighting situations in which additional considerations may be required prior to their application.

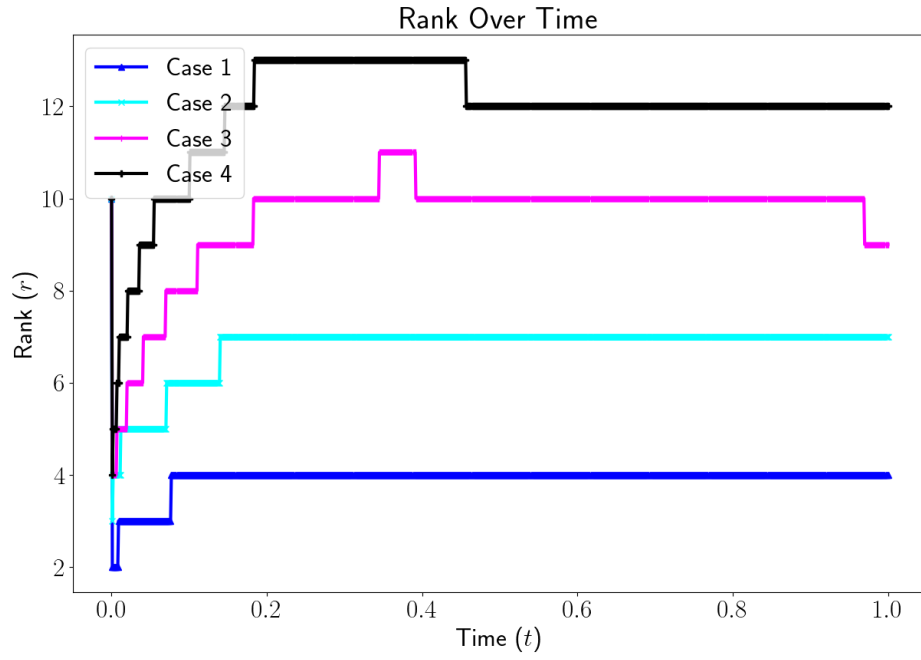


Figure 3.8: The evolution of the rank r over time for the four rank-adaptive cases. The initial rank is taken as $r = 10$ for all cases. Case 1 corresponds to $\epsilon_u = 10^{-4}$, $\epsilon_l = 10^{-6}$; Case 2 to $\epsilon_u = 10^{-6}$, $\epsilon_l = 10^{-8}$; Case 3 to $\epsilon_u = 10^{-8}$, $\epsilon_l = 10^{-10}$; and Case 4 to $\epsilon_u = 10^{-10}$, $\epsilon_l = 10^{-12}$.

Chapter 4

Optimization Methodology

In this chapter, we introduce an optimization framework based on the TDB-CUR algorithm. We begin by formulating the general optimization problem, along with motivation for the proposed approach. We then describe the algorithm in detail, including the role of each of its components. As a proof of concept, we evaluate the method for a single design-variable problem. This framework constitutes the primary contribution of this thesis and serves as the foundation for subsequent extensions presented in Chapter 5, which incorporate adaptive heuristics aimed at improving performance and increasing flexibility of the presented approach.

4.1 Problem Layout

Consider the following problem:

$$\min_{\mathcal{V}, \boldsymbol{\theta}} \mathcal{G}(\mathcal{V}, \boldsymbol{\theta}), \quad \text{s.t.} \quad \mathcal{R}(\mathcal{V}, \boldsymbol{\theta}) = 0,$$

where $\mathcal{V} = \mathcal{V}(x, t; \boldsymbol{\theta})$, $\mathcal{G}(\mathcal{V}, \boldsymbol{\theta})$ is an objective function and $\mathcal{R}(\mathcal{V}, \boldsymbol{\theta})$ is a PDE. By discretizing and solving the PDE (a discretize-then-optimize approach), $\mathcal{V}$ is restricted to the discrete solution manifold defined by the numerical scheme [40, 80]. This yields a reduced, unconstrained minimization problem in $\boldsymbol{\theta}$:

$$\mathcal{J} : \boldsymbol{\theta} \rightarrow \mathcal{G}(\mathcal{V}, \boldsymbol{\theta}),$$

where $\mathbf{V} \in \mathbb{R}^{N \times N_s}$ represents the numerical solution. Our goal is to then solve the following minimization problem:

$$\min_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}).$$

Directly solving for $\boldsymbol{\theta}$ could be costly both in terms of iterations and or queries to the FOM. To alleviate this cost, TDB-CUR can be applied to construct a low-rank surrogate of the state. In this setting, the objective function $\mathcal{J}$ is then defined as:

$$\mathcal{J} : \boldsymbol{\theta} \rightarrow \mathcal{G}(\hat{\mathbf{V}}, \boldsymbol{\theta}),$$

where, as before, $(\hat{\cdot})$ denotes our low-rank solution from the TDB-CUR algorithm. However, a key limitation of TDB-CUR, in this context, is that it does not allow evaluation at arbitrary points in the design space. This becomes problematic when using optimization algorithms that require evaluations outside the set of points sampled to construct the ROM.

To illustrate this limitation, consider a converging-diverging nozzle parameterized by two design variables, θ_1 and θ_2 , as shown in Figure 4.1. In the TDB-CUR framework, the ROM is constructed by sampling the design space at discrete parameter values. If N_{s_1} samples are taken along θ_1 and N_{s_2} along θ_2 , the resulting low-rank approximation is $\hat{\mathbf{V}} \in \mathbb{R}^{N \times N_s}$, where $N_s = N_{s_1} N_{s_2}$. Figure 4.2a illustrates the corresponding sampling grid, where grid lines represent samples along each parameter direction and intersections (blue $\times$ symbols) denote the sampled (θ_1, θ_2) combinations. For example, taking five samples in each direction yields $N_s = 25$. However, optimization methods such as gradient descent typically require evaluations at parameter values that do not lie on this grid. As illustrated in Figure 4.2b, this raises the question of how to evaluate the ROM at new parameter points (dark blue squares) outside the sampled set.

4.2 Presented Methodology

One possible solution is to approach the problem as is done in surrogate assisted optimization, where an expensive objective function is replaced by a lower-cost surrogate to predict objective or constraint values [23, 36]. The general procedure is to sample the design space Θ at N_s samples, evaluate the FOM, or in this case the ROM at all of these samples, form a cheap surrogate of the objective function based on these samples, evaluate, refine, and repeat until a termination condition is met [36]. Broadly, this is the approach we will take. However, we must also control the accuracy of the resulting ROM, since the number of

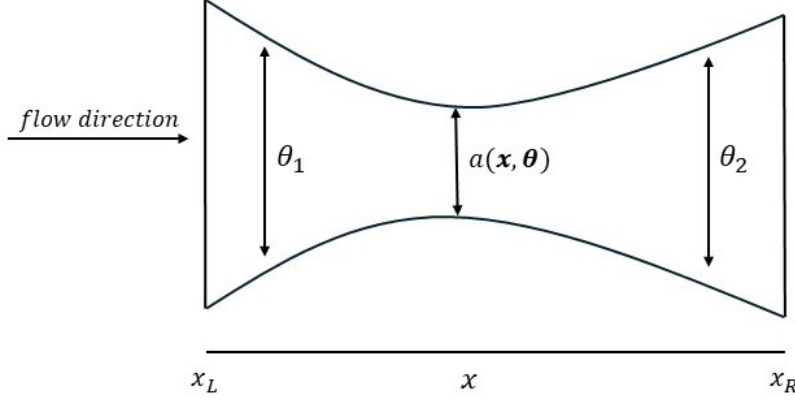
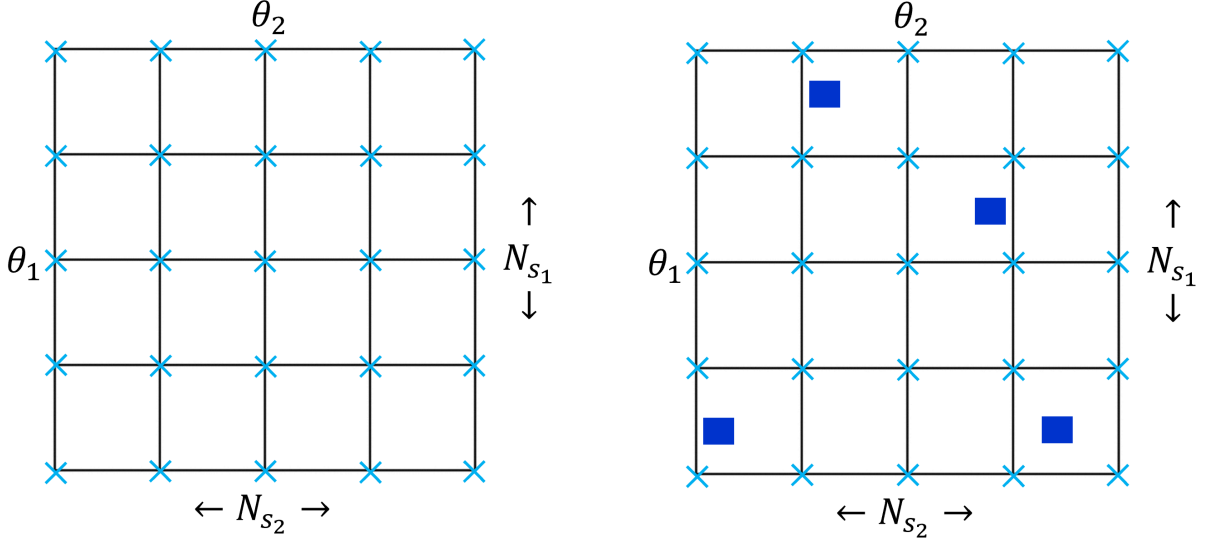


Figure 4.1: Example of a one-dimensional converging-diverging nozzle parameterized by two design variables over the domain $x \in [x_L, x_R]$. The channel width is given by $a(x, \theta)$, where $\theta = [\theta_1, \theta_2]$, with θ_1 and θ_2 controlling the inlet and outlet widths, respectively.

modes required to achieve a given accuracy is not known a priori. If we do not, insufficient modes can lead to errors in the ROM, and subsequently the surrogate, which could steer the optimizer astray. In light of this, we adopt this surrogate assisted optimization perspective, and propose an alternative approach that leverages the relationship between ROM accuracy and the progressive shrinkage of the design space.

Hence, the proposed approach is motivated by the observation that, as the design space shrinks iteratively towards an optimal region, the associated solution manifold becomes more localized and structured, making it more amenable to accurate low-rank approximation. The central idea is to maintain a fixed, low rank throughout the optimization procedure while progressively restricting the design space in order to guide the search toward the optimum. Keeping the rank relatively low at each iteration is a deliberate design choice, particularly in the context of flow problems where the underlying solution dynamics may exhibit high intrinsic complexity. In such cases, the solution manifold can become highly non-smooth, and may contain structures such as shocks or discontinuities.

As observed in Section 3.4.4, the presence of shock-like features can lead to a rapid increase in the number of modes required to achieve a prescribed level of accuracy. In these regimes, the underlying rank of the solution effectively grows, meaning that a straightforward ROM would require many more modes to remain accurate. This can significantly diminish the computational advantage of the ROM, to the point where a surrogate-based optimization strategy may become comparable in cost to the FOM.



(a) Illustration of the sampling grid underlying the TDB-CUR algorithm. Here, $N_{s_1} = 5$ samples are taken along θ_1 and $N_{s_2} = 5$ samples along θ_2 . Each blue $\times$ denotes a sampled parameter combination and corresponds to a column of the resulting ROM.

(b) Illustration of the challenge arising in optimization. Methods such as gradient descent may require evaluating the solution at parameter values that do not lie on the sampling grid (blue square), raising the question of how to approximate the solution at such points.

Figure 4.2: Illustration of the limitations of TDB-CUR when applied to optimization problems, where evaluations are required at parameter values outside the sampled design space.

The key idea in this work is therefore to deliberately avoid tracking this potentially increasing rank. Instead, the rank of the reduced representation is kept fixed and relatively small throughout the optimization process. Rather than increasing the model complexity to capture all fine-scale features, accuracy is instead indirectly controlled through the progressive restriction of the design space.

In this way, as the optimization proceeds and the admissible design space shrinks toward an optimal region, the solution manifold becomes increasingly localized and structured. This improved localization compensates for the fixed low-rank approximation, allowing the ROM to remain both computationally efficient and sufficiently accurate within the evolving region of interest. Specifically, in this setting, restricting the design space at each iteration leads to increased correlation among snapshot vectors, which is typically associated with faster decay of the singular value spectrum and improved low-rank approximation accuracy

for a fixed number of modes.

Although not a formal proof, we provide brief mathematical intuition for this behavior. In particular, consider the FOM solution matrix from a snapshot perspective:

$$\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{N_s}],$$

where each $\mathbf{v}_i \in \mathbb{R}^N$ corresponds to a parameter sample $\boldsymbol{\theta}_i \in \Theta$, i.e., $\mathbf{v}_i = \mathcal{V}(\mathbf{x}, t_F; \boldsymbol{\theta}_i)$. Assuming smooth dependence on the parameters, we impose a Lipschitz condition:

$$\|\mathbf{v}_j - \mathbf{v}_k\| \leq L \|\boldsymbol{\theta}_j - \boldsymbol{\theta}_k\|, \quad \forall \boldsymbol{\theta}_j, \boldsymbol{\theta}_k \in \Theta,$$

for some constant $L > 0$. This condition implies that the variation in the snapshot set is controlled by the diameter of the parameter domain.

Now consider a restricted parameter domain $\Theta_\epsilon \subset \Theta$. As the diameter of the domain decreases, the Lipschitz bound implies that all pairwise differences $\|\mathbf{v}_j - \mathbf{v}_k\|$ are uniformly reduced. In other words, the snapshot vectors become more tightly clustered in $\mathbb{R}^N$. This concentration of the snapshot set suggests that the columns of $\mathbf{V}$ are increasingly well-approximated by a low-dimensional subspace. Equivalently, the snapshots exhibit stronger correlation, indicating a lower effective dimensionality of the associated solution manifold.

From the perspective of low-rank approximation, this behavior is reflected in the singular value spectrum of $\mathbf{V}$. The error of the best rank- r approximation is given by [26]:

$$\|\mathbf{V} - \hat{\mathbf{V}}\|_F = \left(\sum_{i=r+1}^{\min(N, N_s)} \sigma_i^2 \right)^{1/2},$$

where σ_i are the singular values of $\mathbf{V}$. When the snapshot set exhibits stronger correlation, the singular values tend to decay more rapidly, allowing a larger fraction of the total energy to be captured by the leading r modes. Consequently, for fixed rank r and number of samples N_s , restricting the parameter domain is expected to improve the accuracy of the low-rank approximation.

Finally, it is important to mention that the sequence of parameter domains considered in this work is not necessarily nested. Although each domain lies within the original bounds and decreases in size, successive domains are not required to be contained within one another. As a result, monotonic improvement in the approximation error is not guaranteed. Nevertheless, the preceding arguments suggest that restricting the parameter domain promotes increased correlation in the snapshot set and more rapid singular value decay. This

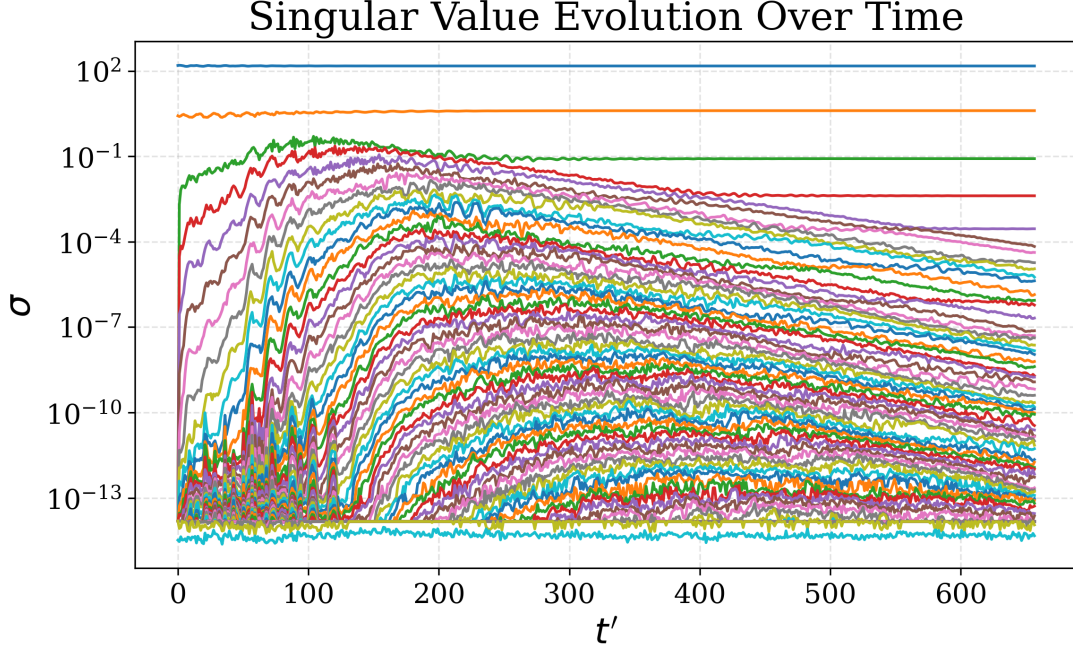


Figure 4.3: First 100 singular values of the FOM plotted over nondimensional time for the full design space $\Theta = [0.4, 0.7]$.

provides a theoretical rationale for the proposed iterative restriction strategy and explains the desired, and consistently observed-numerical behavior of improved ROM accuracy under fixed sampling and rank.

To illustrate this effect, consider a problem for the quasi-1D Euler equations in a parametrized nozzle geometry, where a single design variable $\theta \in \Theta$, with $\Theta = [0.4, 0.7]$, is used to parameterize the geometry. The FOM uses $N_s = 100$ samples on a mesh of $N_x = 101$ nodes. Figure 4.3, shows two dominant modes with the rest tightly clustered. Restricting the design space to $\Theta_\epsilon = [0.475, 0.625]$ (same mesh and samples), increases the spread in singular values, seen in Figure 4.4. This behavior is consistent with the earlier observation that reduced solution variability can lead to more rapidly decaying spectra and lower rank structure.

With this background, we now introduce the main algorithm. For simplicity, we assume a fixed rank and shrink factor across all optimization iterations. Accordingly, we use the TDB-CUR algorithm in Algorithm 1, omitting the rank adaptivity component and keeping the rank constant throughout the ROM evaluation. Let j denote the algorithm iteration

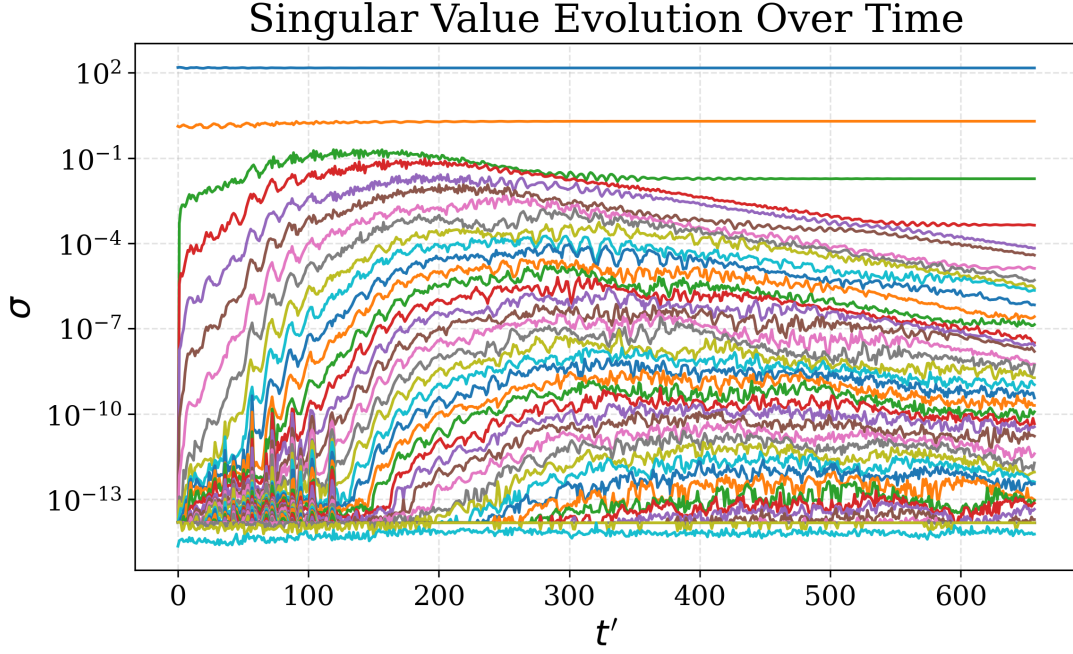


Figure 4.4: First 100 singular values of the FOM plotted over nondimensional time for half of the design space $\Theta_\epsilon = [0.475, 0.625]$.

and $\Theta^{(j)}$ denote the design space at optimization iteration j . In Algorithm 3, we summarize the general steps of the optimization algorithm.

Algorithm 3 Presented Algorithm

Input: $r, m, N_s, \Theta^{(0)}, \text{maxIters}, \text{minIters}, z$ **Output:** $\theta^{(j+1)}$ $\triangleright$ Estimate of the optimal point for the design space $\Theta^{(j)} \leftarrow \Theta^{(0)}$ $\triangleright$ Initialize design domain**for** $\text{iter} \leftarrow \text{maxIters}$ **do** $\triangleright$ Loop over max optimization iterations $\mathbf{T}^{(j)} \leftarrow \text{getSamples}(\Theta^{(j)}, N_s)$ $\triangleright$ Obtain design samples for TDB-CUR $\hat{\mathbf{V}} \leftarrow \text{TDB-CUR}(\mathbf{T}^{(j)}, r, m)$ $\triangleright$ Get the low-rank approximation from TDB-CUR $\mathbf{J} \leftarrow \text{objectiveFunction}(\hat{\mathbf{V}})$ $\triangleright$ Evaluate objective function with $\hat{\mathbf{V}}$ $\mathcal{J}_{\text{interp}} \leftarrow \text{interpolator}(\mathbf{J})$ $\triangleright$ Form surrogate $\theta^{(j+1)} \leftarrow \text{optimizer}(\mathcal{J}_{\text{interp}})$ $\triangleright$ Pass surrogate to optimizer $\text{flag} \leftarrow \text{terminationCondition}(\theta^{(j+1)}, \mathcal{J}_{\text{interp}}, \Theta^{(j)})$ $\triangleright$ Check desired condition**if** $\text{flag} == \text{True}$ **then** EXIT $\triangleright$ Exit loop if termination condition is satisfied**end if** $\Theta^{(j+1)} \leftarrow \text{newDomain}(\theta^{(j+1)}, \Theta^{(j)}, z)$ $\triangleright$ Shrink domain**end for**

Note, here r is the static rank, m is the number of oversampling points, and N_s is the number of samples taken, which is also assumed to be static throughout the duration of the optimization. Additionally, $\mathbf{T} \in \mathbb{R}^{N_s \times d}$ is the matrix of design samples, $\mathbf{J}$ is a vector of length N_s storing each of the objective evaluations of all samples taken.

4.2.1 Description of interpolator and getSamples

The first step in the optimization procedure is the sampling of the design domain. This must be done in a manner consistent with the chosen interpolation strategy. In this work, we consider two approaches for sampling and interpolation:

1. Tensor-product Lagrange interpolation with Legendre-Gauss-Lobatto (LGL) or Chebyshev nodes.
2. Radial basis function (RBF) interpolation with sampling based on Latin hypercube sampling (LHS).

We first consider tensor-product Lagrange interpolation. Given an objective function $\mathcal{J}(\theta)$ defined on a d -dimensional hypercube, we construct a tensor-product grid over the

design space:

$$\boldsymbol{\theta}_{\mathbf{i}} := (\theta_{i_1}, \dots, \theta_{i_d}), \quad i_k = 0, \dots, N_{s_k} - 1, \quad k = 1, \dots, d,$$

where $\mathbf{i} = (i_1, \dots, i_d)$ denotes a multi-index. On this grid, the tensor-product Lagrange interpolant $\mathcal{J}_{\text{interp}}(\boldsymbol{\theta})$ is defined as:

$$\mathcal{J}_{\text{interp}}(\boldsymbol{\theta}) := \sum_{i_1=0}^{N_{s_1}-1} \cdots \sum_{i_d=0}^{N_{s_d}-1} \mathcal{J}(\boldsymbol{\theta}_{\mathbf{i}}) \prod_{j=1}^d \ell_{i_j}(\theta_j), \quad (4.1)$$

where $\ell_{i_j}(\theta_j)$ are the one-dimensional Lagrange basis polynomials in the j^{th} coordinate direction, and $N_s = \prod_{j=1}^d N_{s_j}$ is the total number of grid points. In one dimension, these basis functions are given by:

$$\ell_i(\theta) := \prod_{\substack{j=0 \\ j \neq i}}^{N_s-1} \frac{\theta - \theta_j}{\theta_i - \theta_j},$$

satisfying the interpolation property $\ell_i(\theta_j) = \delta_{ij}$. The second aspect to consider is which nodes to sample with respect to. In general, two natural choices arise, LGL and Chebyshev nodes. Both are able to mitigate the effect of Runge's phenomena, while also providing near optimal Lebesgue constants [13].

Although they have favourable error properties, the main drawback with tensor product interpolation is the curse of dimensionality. Since a structured grid is required, the total number of evaluations scales as $N_s = n^d$, where n is the number of points per dimension. For example, with $d = 20$ and $n = 2$, this already yields $N_s = 2^{20} = 1,048,576$ samples. This exponential growth renders the approach computationally infeasible in high dimensions.

We now consider RBF interpolation combined with LHS. Following [10], RBF interpolation takes the form:

$$\mathcal{J}_{\text{interp}}(\boldsymbol{\theta}) := \sum_{\boldsymbol{\theta}_i \in S} \lambda_i \phi(\|\boldsymbol{\theta} - \boldsymbol{\theta}_i\|), \quad \boldsymbol{\theta} \in \mathbb{R}^d, \quad (4.2)$$

where ϕ is a radially symmetric kernel, S is a set of scattered sample points generated via LHS, and λ_i are coefficients determined by enforcing interpolation conditions.

Unlike tensor-product methods, RBF interpolation does not require a structured grid and can naturally handle scattered sampling, making it well suited for high-dimensional

shape optimization problems [11]. In industrial settings, particularly in aerospace applications, the combination of RBF interpolation and LHS forms the basis of many surrogate-based shape optimization procedures [37, 41, 65, 77]. This approach is popular due to its ability to efficiently approximate high-dimensional design spaces using scattered data.

However, its performance strongly depends on the choice of kernel and associated shape parameters, which can significantly influence accuracy and stability [65]. In addition, optimal sampling and interpolation strategies remain open research problems in ROM frameworks [65].

Given these considerations, we adopt tensor-product interpolation in this work. Although computationally more expensive in high dimensions due to the exponential growth in the number of grid points, it provides a structured setting that is particularly useful for assessing how the TDB-CUR framework handles increased computational cost.

4.2.2 Description of objectiveFunction and terminationCondition

In general, the objective function $\mathcal{J}(\boldsymbol{\theta})$ is evaluated point-wise over the design space, yielding a vector $\mathbf{J}$ of length N_s . This discrete representation is then passed to the tensor-product interpolant in Equation (4.1), where it serves as the set of interpolation coefficients.

Regarding termination conditions, several criteria may be considered. In this work, we focus on a combination of three:

1. **Domain width:** the algorithm terminates once the width of the design domain falls below a prescribed minimum threshold.
2. **Iteration limits:** upper and lower bounds are imposed on the number of optimization iterations to ensure controlled execution.
3. **Objective change:** convergence is assessed via the change in the interpolated objective function between successive iterations:

$$\Delta J = \left| \mathcal{J}_{\text{interp}}(\boldsymbol{\theta}^{(j)}) - \mathcal{J}_{\text{interp}}(\boldsymbol{\theta}^{(j-1)}) \right| < \beta,$$

where β is a prescribed tolerance. In this work, we set $\beta = 1.0 \times 10^{-6}$.

4.2.3 Description of optimizer

The choice of optimizer must be carefully considered. A naïve application of a standard gradient-based method may degrade performance of the overall framework. For instance, at first glance gradient-based methods such as gradient descent appear suitable, since the surrogate model provides access to derivative information. However, this overlooks the fact that the ROM is initially a rough approximation of the true objective landscape.

Hence, in the early stages of the optimization procedure, the ROM may be noisy or inaccurate, implying that gradient information derived from it may be unreliable. Directly following such gradients may therefore lead the search into incorrect regions of the design space. For this reason, a gradient-free global search strategy is preferable in the initial iterations.

A natural choice in this context is an evolutionary algorithm [12], and in particular Differential Evolution (DE). DE is a population-based optimization method widely used in engineering applications [4]. It generates new candidate solutions through mutation, crossover, and selection, allowing for robust exploration of the design space [4, 64]. However, DE can be computationally expensive due to the large number of objective evaluations required [53]. While this cost is mitigated in the present setting since the objective is evaluated through a surrogate model rather than the FOM, the cost may still become significant in high-dimensional design spaces.

To balance exploration and efficiency, we adopt a hybrid optimization strategy combining DE with a gradient-based method. In particular, we employ the Limited-memory Broyden-Fletcher-Goldfarb-Shanno method with bounds (L-BFGS-B) [85]. This is a quasi-Newton method that approximates the Hessian matrix using gradient information, requiring only first-order derivatives while maintaining low memory and computational cost. Within the proposed framework, L-BFGS-B is used both as a refinement step following DE in early iterations, and as a standalone optimizer once the ROM has sufficiently stabilized and gradient information becomes more reliable.

The transition between these two regimes is controlled via a strategy-switching criterion. This decision depends on the iteration number and the current shrink factor of the design domain. In general, the switching function `strategySwitch`, found in Algorithm 4, enforces minimum thresholds on both the iteration count and the shrink factor before transitioning from the global DE-based strategy to the local L-BFGS-B-based strategy. The description of the hybrid optimization approach is summarized in Algorithm 5.

Algorithm 4 strategySwitch

Input: iter, z , minIters, $z_{\min}$ **Output:** strategy ▷ global: DE + L-BFGS-B, local: L-BFGS-B only**if** iter < minIters **then** strategy $\leftarrow$ global ▷ Ensure global exploration during early iterations**else if** $z < z_{\min}$ **then** strategy $\leftarrow$ global ▷ Maintain global search while the design domain remains large**else** strategy $\leftarrow$ local ▷ Switch to local gradient-based refinement (L-BFGS-B only)**end if**

Algorithm 5 optimizer

Input: $\mathcal{J}_{\text{interp}}$, $\nabla \mathcal{J}_{\text{interp}}$, $\mathbf{J}$, bounds, iter, minIters, z , $z_{\min}$ **Output:** $\boldsymbol{\theta}^*$ ▷ Found minimum of surrogate strategy $\leftarrow$ strategySwitch(iter, minIters, z , $z_{\min}$) ▷ Choose method**if** strategy == global **then** ▷ If in early iterations, use combination of methods $\boldsymbol{\theta}^* \leftarrow \text{DE}(\mathcal{J}_{\text{interp}}, \text{bounds})$ ▷ Get rough estimate of optimal point from DE $\boldsymbol{\theta}^* \leftarrow \text{LBFGSB}(\boldsymbol{\theta}^*, \mathcal{J}_{\text{interp}}, \nabla \mathcal{J}_{\text{interp}}, \text{bounds})$ ▷ Pass $\boldsymbol{\theta}^*$ from DE into L-BFGS-B**else** index $\leftarrow \arg \min(\mathbf{J})$ ▷ Get the index of the best sample in $\mathbf{J}$ $\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}(\text{index})$ ▷ Use the best sample as the initial guess $\boldsymbol{\theta}^* \leftarrow \text{LBFGSB}(\boldsymbol{\theta}^*, \mathcal{J}_{\text{interp}}, \nabla \mathcal{J}_{\text{interp}}, \text{bounds})$ ▷ Find min using only L-BFGS-B**end if**

4.2.4 Description of newDomain

The final step in the algorithm is to shrink the design space. Specifically, the idea is for each iteration of the optimization procedure we obtain our optimum for that iteration $\boldsymbol{\theta}^{(j+1)}$, and will build the next domain centered at this point, shrinking each direction equally as defined via the shrink factor z . If we think of the initial design space as a multidimensional box, we can define the domain as follows:

$$\Theta^{(0)} = \prod_{i=1}^d [l_i^{(0)}, r_i^{(0)}],$$

where $l_i^{(0)}$ and $r_i^{(0)}$ are the global left and right endpoints of each design direction in $\mathbb{R}^d$. At iteration j :

$$\Theta^{(j)} = \prod_{i=1}^d [l_i^{(j)}, r_i^{(j)}],$$

where $l_i^{(j)}$, $r_i^{(j)}$ are the left and right bounds of the design space, in each direction d . For each dimension i our width for any iteration j can then be given by

$$w_i^{(j)} := r_i^{(j)} - l_i^{(j)}.$$

This implies, given the shrink factor z , the width of the domain for the next iteration is:

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{z}, \text{ where } z > 1.$$

Ideally we want to then center the interval around the found minimum, i.e.,

$$\tilde{l}_i = \theta_i^{(j+1)} - \frac{w_i^{(j+1)}}{2}, \quad \tilde{r}_i = \theta_i^{(j+1)} + \frac{w_i^{(j+1)}}{2}.$$

Near the boundaries, the new interval may extend outside the global bounds $[a_i, b_i]$. One option is to truncate the interval back within the bounds, but this can unevenly shrink different directions. Instead, we preserve the internal width and shift it to remain within bounds while staying as close as possible to the new optimum. Thus, the updated endpoints are:

$$l_i^{(j+1)} = \max(l_i^{(0)}, \min(\tilde{l}_i, r_i^{(0)} - w_i^{(j+1)})), \quad r_i^{(j+1)} = l_i^{(j+1)} + w_i^{(j+1)}.$$

Hence, the updated design domain is then:

$$\Theta^{(j+1)} = \prod_{i=1}^d [l_i^{(j+1)}, r_i^{(j+1)}].$$

Note that this approach applies a single shrink factor z to every direction d in the design space equally. Ideally, it would be beneficial to have a targeted shrink factor that could shrink each direction individually based on its own needs, however, we did not pursue this idea here.

4.3 Proof-of-Concept Results

In the following, we present proof-of-concept results for a single design variable. Specifically, we consider an inverse problem that seeks to minimize an energy-like functional introduced in [42]:

$$\mathcal{J}(\theta) = \frac{1}{2} \|p - p^*\|_2^2,$$

where θ denotes the design parameter controlling the nozzle geometry. Here $\|\cdot\|_2$ is the L^2 norm over the spatial domain, $p = p(x, t; \theta)$ is the pressure profile predicted by the algorithm at the current design point, and $p^* = p^*(x, t_F; \theta^*)$ is the target profile for the desired design point θ^* .

The flow problem corresponds to a subsonic inflow-outflow configuration, which is advanced in time to a steady state at $t_F = 3.0$. We consider a piecewise-defined nozzle geometry of the form:

$$a(x, \theta) = \begin{cases} 1 + (1 + \theta) \left(1 - \frac{x}{5}\right)^2 & 0 \leq x \leq 5, \\ 1 + \theta \left(1 - \frac{x}{5}\right)^2 & 5 \leq x \leq 10. \end{cases}$$

For the spatial discretization, we employ CG operators of degree $q = 2$ on $k = 50$ elements, resulting in a total of $N = 3N_x = 303$ degrees of freedom. The parameter space is defined as $\Theta^{(0)} = [0.4, 0.7]$, over which $N_s = 100$ samples are taken using LGL nodes. This leads to a solution matrix of size $\mathbf{V} \in \mathbb{R}^{303 \times 100}$.

Time integration is performed using a classical RK4 scheme within the TDB-CUR framework, with a time step of $\Delta t = 5.0 \times 10^{-5}$.

For the optimization setup, we prescribe a rank of $r = 3$ with $m = 2$ oversampling points. The maximum number of optimization iterations is set to $\text{maxIters} = 15$, while a minimum of 10 iterations is enforced for illustrative purposes. Finally, a shrink factor of $z = 2$ is used, corresponding to a halving of the design domain at each iteration.

In Figure 4.5, we show the evolution of the parameter θ over the course of the optimization procedure relative to the admissible design space. We observe that already in the first iteration the estimated value of θ is close to the true optimum, indicating that a rank of $r = 3$ with $m = 2$ oversampling points provides a sufficiently accurate approximation of the underlying dynamics even in the initial stages of the algorithm.

Figures 4.6 and 4.7 show the resulting pressure profiles and nozzle geometries, respectively, compared to the target solution. In Figure 4.6, the target and predicted pressure

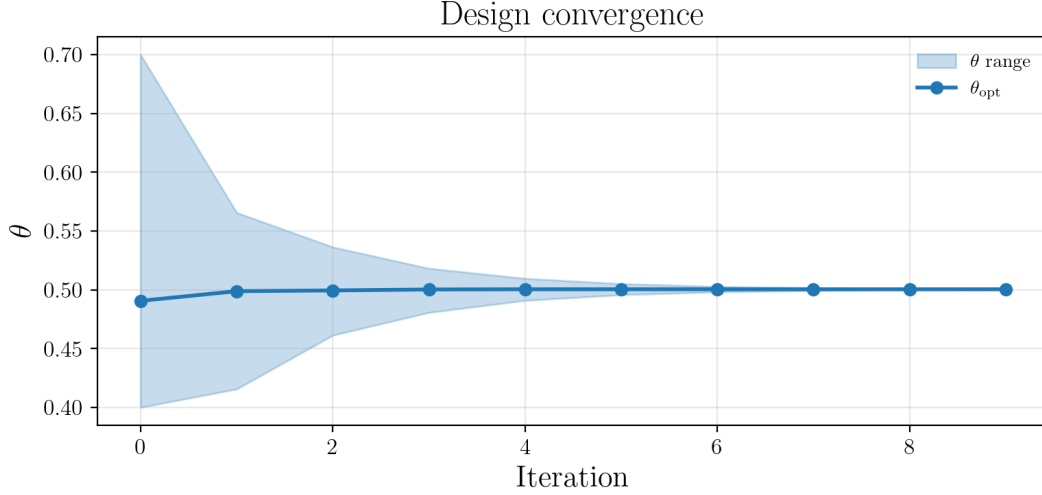


Figure 4.5: The trajectory of the estimated parameter θ^* within the admissible design space over the course of the optimization algorithm. For this simple test case, the method converges rapidly to the optimal parameter value.

profiles are plotted against the range of admissible profiles defined by the initial design space. The algorithm is able to recover the target solution with good accuracy, achieving a final maximum absolute error of $\|p - p^*\|_\infty \approx 4.37 \times 10^{-4}$. A similar trend is observed for the nozzle geometry, where the final maximum absolute error is given by $\|a - a^*\|_\infty \approx 2.837 \times 10^{-3}$.

In this chapter, we have presented the main contribution of this thesis, namely an optimization framework based on the TDB-CUR algorithm applied to shape optimization problems. We have also described the key components of the method and detailed the specific algorithmic choices that define its implementation. In addition, a single-design-variable problem was introduced as a proof-of-concept to demonstrate the performance of the proposed approach.

A natural question that arises is how the domain shrink rate should be selected. In the single-design variable problem, we chose $z = 2.0$ arbitrarily, however, this raises the question of whether such a choice leads to overly aggressive or overly conservative domain reduction. This issue is addressed in the following chapter.

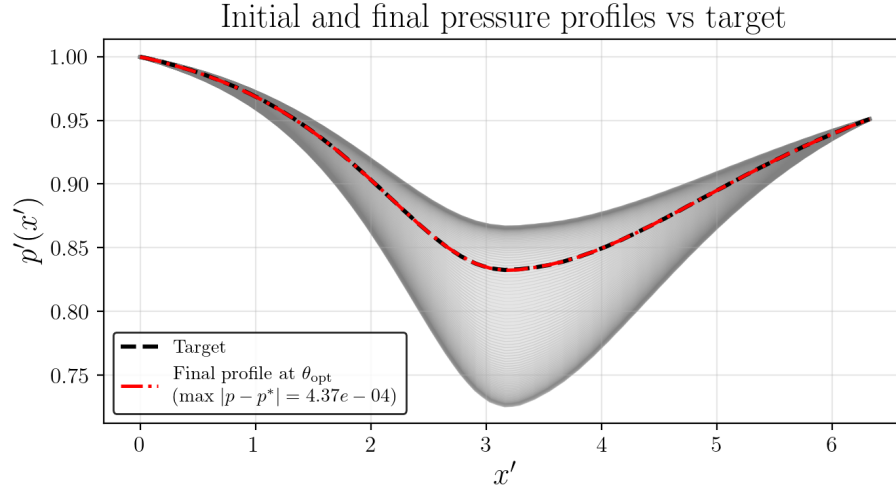


Figure 4.6: Comparison between the target and reconstructed pressure profiles, with the gray curves representing the sampled profiles from the initial design domain $\Theta^{(0)}$. The maximum absolute error is also reported

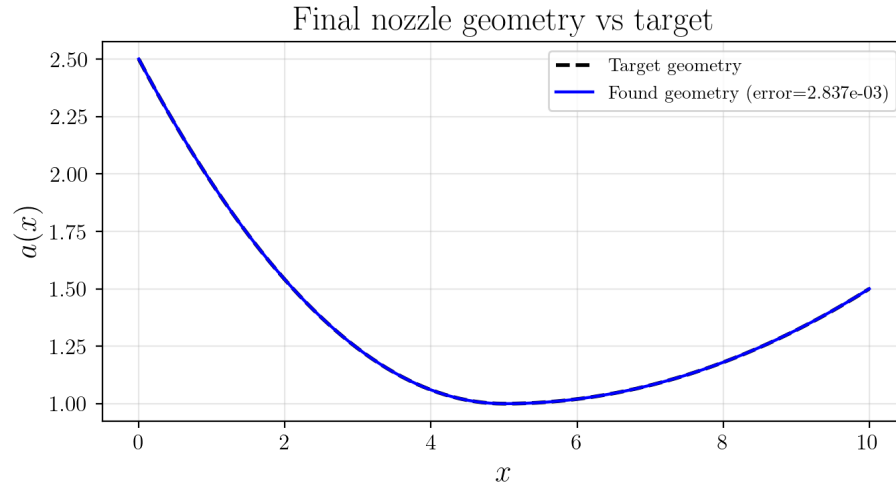


Figure 4.7: Comparison of the target and found nozzle geometries, with the maximum absolute error between the two reported.

Chapter 5

Shrink Factor and Rank Adaptivity

In the previous chapter, we introduced the general methodology for applying the TDB-CUR algorithm to shape optimization problems. A central component of this method is the shrink factor z , which controls the update dynamics of the algorithm. In the initial experiments, this parameter was kept fixed at an arbitrarily chosen constant value throughout all optimization runs. This raises the question of how this value should be selected and whether it leads to appropriate contraction behaviour, avoiding excessively aggressive or overly conservative shrinkage. Unfortunately, there is no simple or exact answer to this question, as the behaviour of the shrink factor is not easily characterized in a rigorous way. For this reason, the present chapter aims to explore this issue in more detail. We discuss the heuristics used to adjust and evaluate the shrink factor, as well as the rank. In addition, we provide motivation for the main design choices made in developing these approaches. Finally, we will present the main results of the method, exploring different problems for $d = 2, 4, 6$, and 10 design variable problems.

5.1 Motivation and Description of Adaptive Shrink Factor

As previously discussed in Section 3.4.2, the concept of rank adaptivity in ROMs is important as it provides a mechanism to control the approximation error of the ROM, while avoiding unnecessary computations. In particular, it allows the model complexity to adjust in response to the evolving system dynamics. This idea motivates the development

of an adaptive strategy for the shrink factor in the TDB-CUR framework. The goal is to regulate the reduction of the design space in a controlled manner, ensuring that:

1. the shrink factor does not become too large, which would cause the algorithm to prematurely concentrate on a narrow region of the design space, and
2. it does not decrease too slowly, which would result in excessive exploration and unnecessary computational cost.

This perspective is closely aligned with the role of adaptivity in rank selection, where the model is refined only when necessary. However, unlike standard ROM settings, the challenge here lies in the absence of FOM information during the optimization procedure. As a result, it is nontrivial to define reliable indicators that capture how the ROM is evolving, or to design heuristics that can guide the adjustment of the shrink factor in a robust and systematic way.

To begin developing this idea, we examine related ROM strategies, with a particular focus on methods that adaptively explore and refine the design space. To the best of our knowledge, the closest class of methods to the present framework are trust-region methods [3, 33, 45, 82, 83]. These methods, similar to the proposed approach, use information obtained during the optimization process to guide model updates [3].

In the ROM setting, trust-region methods iteratively solve reduced optimization subproblems defined over a local region of the design space in which the ROM is assumed to be reliable [45]. Each subproblem yields a candidate update for the trust-region centre. This candidate is then evaluated using acceptance criteria that compare predicted and observed performance, and depending on this assessment, the update is either accepted or rejected. The trust-region size is subsequently adjusted to reflect the perceived reliability of the ROM in the current region [83].

To highlight the distinction between the present method and trust-region approaches, consider Figure 5.1. In the trust-region framework (left), the algorithm starts from an initial point in the design space and solves a sequence of local subproblems to approach an optimum. At each iteration, the trust region defines a neighbourhood in which the ROM is assumed to be reliable. In contrast, the proposed methodology (right) begins with the entire design space as the initial region of interest. The algorithm then refines this domain by shrinking it around points of interest identified during the optimization process, leading to convergence through an adaptive global-to-local shrinkage strategy.

Despite these differences in how the design space is explored, both approaches share the idea of dynamically adapting the region in which the ROM is applied. This motivates

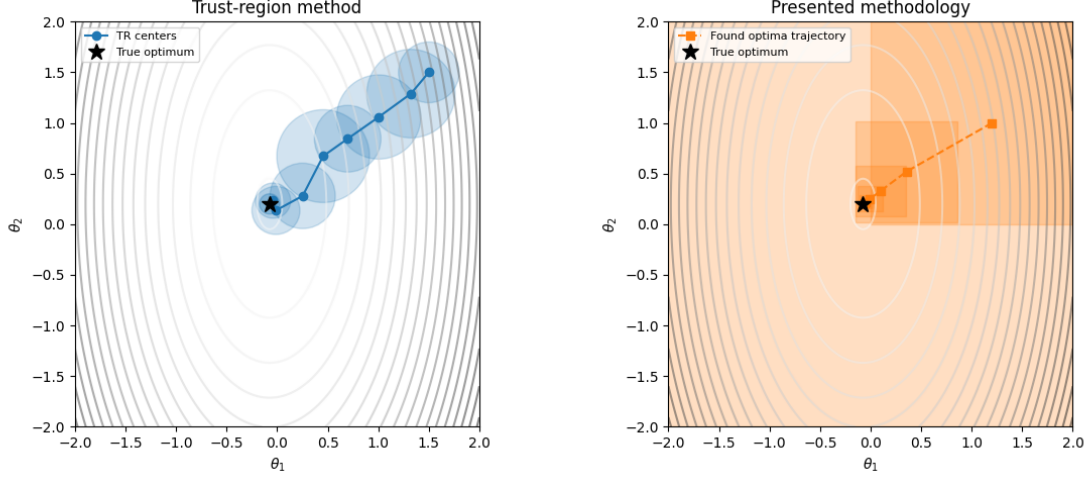


Figure 5.1: An image of the conceptual difference between a standard trust-region method and the proposed methodology. In the trust-region approach (left), the algorithm begins from a local region in the design space and iteratively moves toward the optimum. In contrast, the proposed method (right) starts from the full design space and progressively refines it, converging to the optimum through successive contractions.

the question of whether mechanisms used in trust-region methods for controlling region size can inform the choice of the shrink factor in the present framework.

However, classical trust-region methods control the radius using a ratio comparing the actual and predicted reduction in the objective function:

$$\rho := \frac{v_{\text{actual}}}{v_{\text{pred}}},$$

where v_{actual} denotes the reduction in the objective function evaluated using the FOM, and v_{pred} denotes the reduction predicted by the ROM [7]. The quantity ρ measures the agreement between the ROM prediction and the observed reduction. When ρ is close to one, the ROM is considered reliable in the current region, otherwise the step is rejected and the trust-region size is adjusted accordingly [7].

A key limitation of this approach is that it requires repeated evaluations of the FOM, which is generally infeasible due to its computational cost [7, 83]. As a result, many trust-region methods rely on approximations of this criterion based on surrogate or inexact objective information under suitable assumptions on model accuracy. For example, [83] employs an asymptotic condition derived from [47], where the acceptance criterion is formulated using an inexact model provided by the ROM.

In the present setting, however, these acceptance criteria cannot be directly employed. Their implementation relies on the ability to evaluate or reliably approximate objective function variations across the parameter space. In practice, this would require FOM evaluations over a large number of samples, which is computationally infeasible in the present framework. Although ROM based approximations of the classical trust-region ratio have been proposed in the literature, they typically rely on the availability of sufficiently accurate and consistent information about objective variations across the parameter space.

In our setting, the ROM is constructed under a low-rank assumption and is intended to provide only a rough approximation in the early stages of the optimization. As a result, ROM based estimates of objective variation may be highly inaccurate and may not reflect the true behaviour of the FOM. This lack of reliable global consistency makes ROM based acceptance tests potentially misleading, since they may incorrectly indicate agreement or disagreement between predicted and observed behaviour. Consequently, the standard trust-region acceptance mechanism is not robust in this context. Instead, we retain only the general idea of using model consistency information to guide domain adaptation, which is incorporated in a heuristic manner to update the shrink factor.

Hence, we focus on leveraging a commonly used quantity as a proxy for model quality, the singular values. As discussed in the context of the TDB-CUR algorithm, singular values can guide rank adaptivity by providing information about approximation error. In addition, they reflect the energy distribution of the system, with their decay indicating how well the system can be represented in a low-dimensional subspace.

In this sense, the decay of singular values provides information about the observed dimensionality of the ROM solution at a given stage of the optimization procedure. Since the ROM is constructed as a low-rank approximation, faster decay suggests that the solution is well-captured by a smaller number of dominant modes, whereas slower decay indicates that a higher-dimensional representation is required.

This motivates the use of singular value based quantities for adjusting the shrink factor. Specifically, we use the singular values associated with the ROM obtained at the final integration time t_F . This choice is based on the assumption that the system behaviour at the end of the simulation is representative, allowing the final reduced state to capture its main features. While this does not fully capture the temporal evolution, it serves as a practical proxy for the structure of the ROM solution at that iteration of the optimization procedure. Based on this perspective, we consider the following metrics to guide the update strategy.

5.1.1 Singular Value Tail Ratio

The first metric used to guide design space shrinkage is the singular value tail ratio, which quantifies the relative magnitude of the lower end of the singular value spectrum with respect to its dominant mode. It is given as:

$$\epsilon_t := \frac{\hat{\sigma}_{i_{\text{tail}}}}{\hat{\sigma}_1}, \quad (5.1)$$

where $\hat{\sigma}_1$ denotes the largest singular value and $\hat{\sigma}_{i_{\text{tail}}}$ corresponds to a representative value in the tail of the available spectrum.

Rather than selecting the smallest singular value directly, the index i_{tail} is defined as the singular value corresponding approximately to the 90th percentile of the spectrum. This choice improves robustness, since the smallest singular values are often sensitive to numerical noise and may vary significantly as the ROM is refined. Using a percentile-based index also provides a more consistent relative measure of spectral decay across iterations, allowing for an improved assessment of how the ROM evolves over successive iterations.

This metric can be interpreted as a proxy for spectral decay of the solution matrix, analogous to roughness measures based on singular value decay proposed in prior work [66]. In this sense, it provides a normalized indicator of how rapidly the singular values decrease relative to the dominant mode. Small values of ϵ_t indicate rapid decay and therefore a strongly low-rank structure, corresponding to a more reliable ROM for this iteration. In this regime, the solution is well approximated by a small number of dominant modes.

Conversely, values of ϵ_t close to one indicate a flatter spectrum, suggesting weaker low-rank structure and higher observed dimensionality. In this case, the ROM may require a richer representation to capture the relevant dynamics, and overly aggressive shrinkage could be detrimental.

To convert this signal into a shrink factor, we use a smoothly decreasing mapping that transitions between conservative and aggressive regimes. This leads to the following empirical mapping:

$$z_t := 1.0 + 0.6e^{-6\epsilon_t}. \quad (5.2)$$

This function assigns larger shrink factors to rapidly decaying systems and smoothly transitions toward a baseline value as the spectrum becomes flatter. As shown in Figure 5.2, the mapping yields values in the range $z_t \in [1.0, 1.6]$.

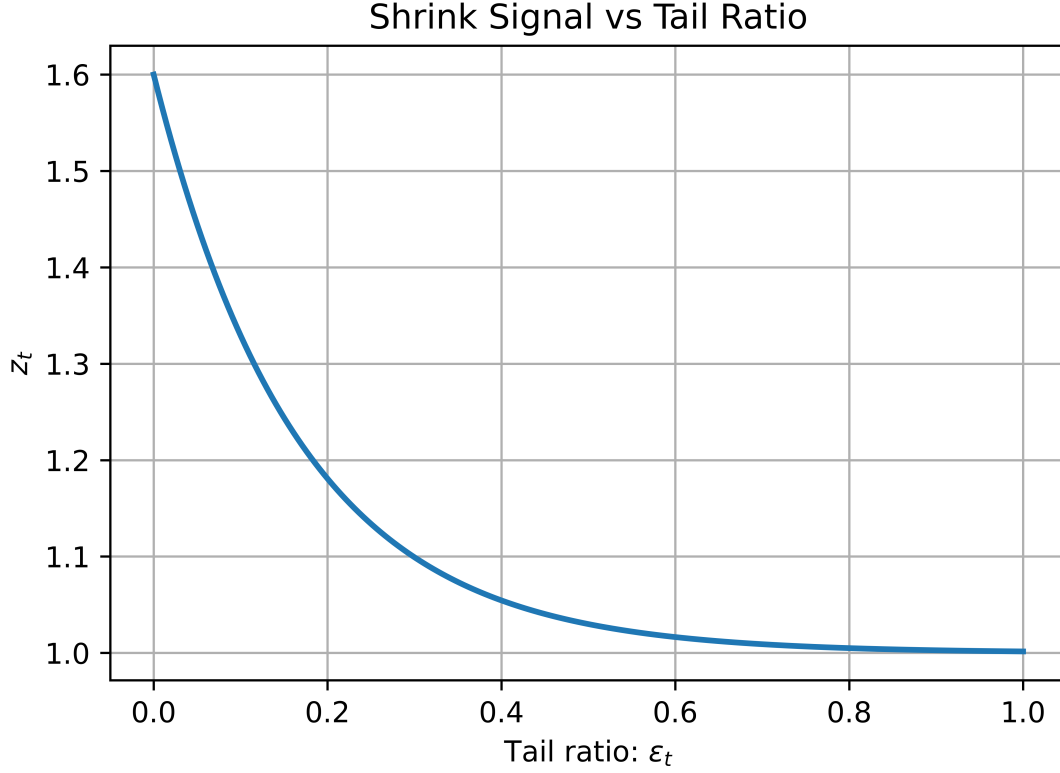


Figure 5.2: Mapping from the singular value tail ratio ϵ_t , to the corresponding shrink factor z_t for the proposed heuristic. Small values of ϵ_t , indicating a low-rank structure, lead to more aggressive shrinkage, whereas values close to one, indicating a flat spectrum, result in more conservative shrinkage.

5.1.2 Singular Value Slope

The second metric considered is the slope of the singular value spectrum, evaluated in the tail region. This metric estimates the effective exponential decay rate of the singular values by computing the slope of the logarithm of the singular values with respect to their index:

$$\epsilon_m \approx \frac{d}{di} \log(\hat{\sigma}_i), \quad i \in \text{tail indices}.$$

The slope of the logarithmic singular value spectrum provides a practical indicator of spectral decay and low-rank structure [34, 63, 66]. In practice, the smallest singular values are often sensitive to numerical noise and rounding errors, which may lead to unstable

estimates if used directly. To improve robustness, we estimate the slope using a least-squares fit over the last $w = \min(5, r)$ singular values, with $w \geq 3$. This choice balances robustness against noise with capturing local tail behaviour, since smaller windows are sensitive to fluctuations while larger windows may smooth out the overall decay trend. Let the selected singular values be $\{\hat{\sigma}_{r-w+1}, \dots, \hat{\sigma}_r\}$. We define:

$$y_k := \log(\hat{\sigma}_{r-w+k}), \quad k = 1, \dots, w,$$

and normalize the indices to the interval $[0, 1]$ via:

$$x_k := \frac{k-1}{w-1}, \quad k = 1, \dots, w.$$

The slope is then computed using standard least-squares regression:

$$\epsilon_m = \frac{\sum_{k=1}^w (x_k - \bar{x})(y_k - \bar{y})}{\sum_{k=1}^w (x_k - \bar{x})^2}, \quad \bar{x} = \frac{1}{w} \sum_{k=1}^w x_k, \quad \bar{y} = \frac{1}{w} \sum_{k=1}^w y_k.$$

Since the regression is performed on $\log(\hat{\sigma}_i)$, we obtain an effective local model:

$$\log(\hat{\sigma}_i) \approx \epsilon_m i + b,$$

which implies:

$$\hat{\sigma}_i \approx e^b e^{\epsilon_m i}.$$

Thus, ϵ_m provides an estimate of the effective exponential decay rate of the singular value spectrum in the tail region. Large negative values of ϵ_m indicate rapid decay and suggest strong low-rank structure, whereas values of ϵ_m close to zero indicate a flatter spectrum and weaker low-rank behavior. We interpret three regimes:

1. $\epsilon_m \ll 0$: rapid spectral decay, indicating strong low-rank structure. The ROM is reliable and supports aggressive shrinkage.
2. $\epsilon_m < 0$ with moderate magnitude: intermediate decay, indicating partial low-rank structure and moderate confidence in the ROM.
3. $\epsilon_m \approx 0$: flat spectrum, indicating weak low-rank structure and reduced reliability of the ROM.

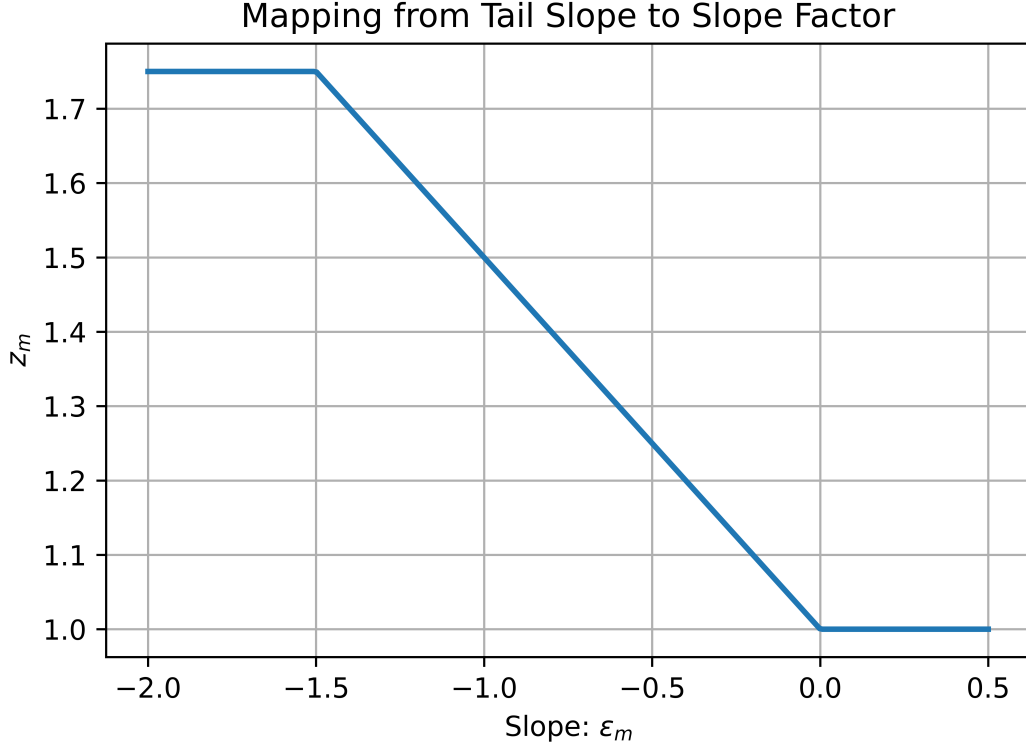


Figure 5.3: Mapping between the slope of the singular value spectrum ϵ_m , and the corresponding shrink factor z_m . Large negative values of ϵ_m , corresponding to rapid spectral decay, are mapped to a maximum shrink factor of $z_m = 1.75$. For intermediate decay rates, the mapping is approximately linear, while for a flat spectrum ($\epsilon_m \approx 0$), the shrink factor reduces to $z_m = 1.0$.

The corresponding shrink factor maps the decay rate into a bounded nonlinear response:

$$z_m := 1 + 0.5\text{clip}(-\epsilon_m, 0, 1.5). \quad (5.3)$$

This mapping increases shrinkage proportionally to the strength of spectral decay while remaining bounded to ensure numerical stability. In the case where $\epsilon_m > 0$, which may occur due to numerical noise, the shrink factor is set to $z_m = 1$. Figure 5.3 illustrates the resulting mapping between the slope and the shrink factor z_m .

5.1.3 Relative Change of Singular Values

The final metric considers the relative change in the singular value spectrum between successive iterations, with the goal of quantifying how the underlying system evolves as the design space is progressively reduced. We define the relative spectral change as:

$$\epsilon_{\Delta} := \frac{\text{change in spectrum}}{\text{magnitude of previous spectrum}},$$

which is expressed in terms of the singular value vectors $\mathbf{S}^{(j)}$ and $\mathbf{S}^{(j-1)}$ as:

$$\epsilon_{\Delta} := \frac{\|\mathbf{S}^{(j)} - \mathbf{S}^{(j-1)}\|_2}{\|\mathbf{S}^{(j-1)}\|_2}.$$

However, since the reduced rank may vary between iterations, the comparison is restricted to the shared indices:

$$\epsilon_{\Delta} = \frac{\|\mathbf{S}^{(j)}(:, r_{\min}) - \mathbf{S}^{(j-1)}(:, r_{\min})\|_2}{\|\mathbf{S}^{(j-1)}(:, r_{\min})\|_2}, \quad r_{\min} = \min(r^{(j)}, r^{(j-1)}).$$

While this comparison does not enforce a one-to-one correspondence between individual singular values across iterations, it provides a measure of the relative change in the truncated singular value spectrum over a common index range. In this sense, ϵ_{Δ} captures changes in the dominant spectrum rather than tracking individual singular value trajectories.

From perturbation theory for singular values, changes in the singular value spectrum are controlled by perturbations of the underlying matrix. In particular, Mirsky's theorem [49, 70] implies that:

$$\|\mathbf{S}^{(j)} - \mathbf{S}^{(j-1)}\|_2 \leq \|\hat{\mathbf{V}}^{(j)} - \hat{\mathbf{V}}^{(j-1)}\|_F,$$

where $\hat{\mathbf{V}}^{(j)}$ denotes the solution matrix at iteration j . This establishes that changes in the singular value spectrum are controlled by perturbations of the underlying solution matrix, providing partial theoretical support for using spectral variation as an indicator of stability.

In particular, large values of ϵ_{Δ} are consistent with significant changes in the underlying solution structure. Based on this interpretation, two regimes are distinguished:

1. $\epsilon_{\Delta} \approx 0$: the spectrum remains nearly unchanged between iterations, suggesting that the system may be stabilizing and allowing for more aggressive shrinkage.

2. $\epsilon_\Delta \gtrsim \tau$: significant spectral change is observed, indicating that the system is still evolving and that a more conservative shrinkage strategy may be appropriate. In this work, $\tau = 0.05$.

To map this signal into a shrink factor, we use a smooth, bounded nonlinear mapping that ensures stable and confined shrink factors:

$$z_\Delta := 1.0 - 0.7 \tanh(3(\epsilon_\Delta - \tau)).$$

This mapping reduces the shrink factor when large spectral changes are detected, and increases aggressiveness as the system stabilizes. Unlike the previous two metrics, this formulation permits $z_\Delta < 1$, which introduces an explicit penalization of instability between successive iterations. In this regime, the shrink factor is reduced below its baseline value when ϵ_Δ exceeds the tolerance threshold, reflecting reduced confidence in the current ROM.

The choice of tolerance τ plays a critical role in controlling the sensitivity of this mechanism. Since the design space is reduced gradually over successive iterations, even moderate changes in the spectrum may indicate meaningful shifts in the dominant subspace. For this reason, a small value of $\tau = 0.05$ is used in this work to ensure early detection of instability while avoiding excessive sensitivity to numerical noise. This metric also acts as a corrective mechanism relative to the previous two heuristics. While the tail ratio and slope quantify instantaneous spectral structure, this metric captures inter-iteration variation in the spectrum. Consequently, a large value of ϵ_Δ not only indicates a change in the system dynamics, but may also signal a shift in the dominant modes, justifying a reduction in shrink aggressiveness. The parameters in this mapping, including τ and the scaling constants, can be adjusted depending on the desired sensitivity to spectral variation. Figure 5.4 illustrates the resulting relationship between ϵ_Δ and z_Δ .

5.1.4 Combining Signals and Aggressiveness

The proposed shrinkage strategy is not based on a single spectral indicator, but instead combines multiple heuristics that use the singular value spectrum in an attempt to approximate the underlying ROM structure at each iteration. Each metric captures a different aspect of spectral behaviour, and their combination is used to reduce the sensitivity of the overall strategy to the limitations of any individual measure. Hence, to combine the three signals, we employ the geometric mean:

$$z_b = (z_t \cdot z_m \cdot z_\Delta)^{\frac{1}{3}}.$$

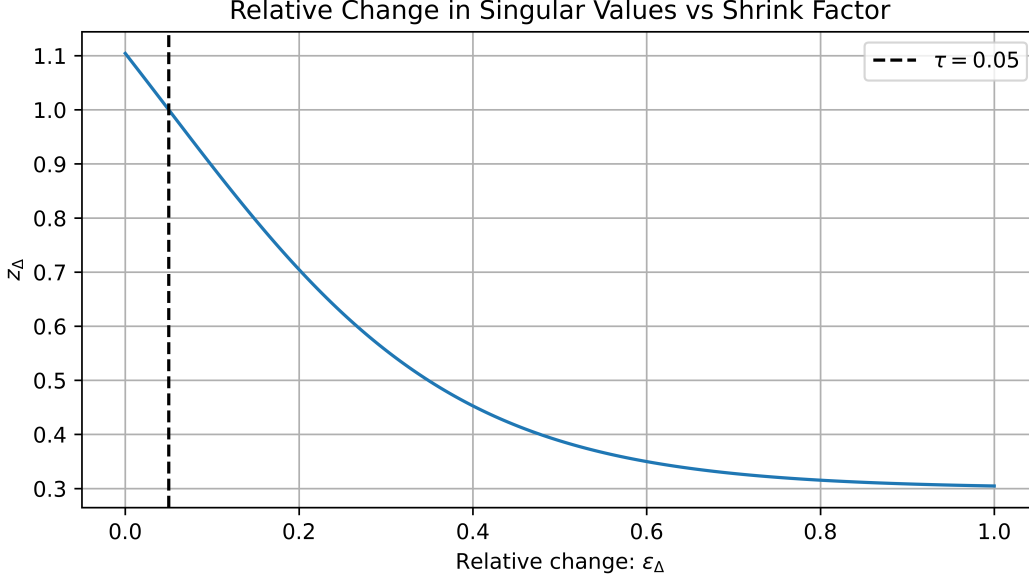


Figure 5.4: Mapping between the relative change in singular values across successive iterations ϵ_Δ , and the corresponding shrink factor z_Δ . This assumes that changes in the spectrum between successive iterations are not excessively large, with “large” defined via a tolerance $\tau = 0.05$.

This choice reflects the multiplicative nature of the proposed criteria, as the three metrics attempt to capture complementary aspects of the singular value spectrum. In particular, z_t measures relative spectral decay, z_m estimates the effective decay rate in the tail, and z_Δ quantifies inter-iteration changes in the spectrum. A large shrink factor is therefore only obtained when all three indicators simultaneously suggest strong low-rank structure. Compared to an arithmetic mean, the geometric mean reduces the influence of any single metric and penalizes disagreement between signals, leading to more conservative behaviour when the indicators are not aligned. In preliminary experiments, arithmetic averaging was found to be more sensitive to individual metrics and resulted in less stable shrinkage behaviour. For these reasons, the geometric mean is used throughout this work.

Additionally, not all problems exhibit the same singular value decay behaviour as the design space evolves. To account for this variability, we introduce an aggressiveness parameter $a_g \in [0, 1]$ that modulates the overall shrinkage rate. A baseline value $b_s = 0.35$, corresponding to the setting used in all reported experiments prior to introducing a_g , is

used for normalization. The final shrink factor is given as:

$$z := \max\left(z_{\min}, z_b^{\frac{a_g}{b_s}}\right).$$

This exponentiation provides a smooth mechanism for adjusting the effective strength of shrinkage. Values $a_g > b_s$ increase the effective shrink factor relative to the baseline setting, leading to more aggressive contraction, while $a_g < b_s$ yields more conservative updates. In the limiting case $a_g \rightarrow 0$, the exponent tends to zero and the shrink factor approaches one. However, to ensure that the algorithm always enforces at least a minimal level of contraction, a lower bound $z_{\min}$ is imposed. This guarantees that the design space is reduced at every iteration, even in cases where the spectral indicators suggest weak or ambiguous structure. As such, a_g provides a simple and continuous means of adapting the method to problems with differing spectral characteristics while preserving guaranteed shrinkage behaviour.

5.1.5 Rank Adaptivity

The final component of the adaptivity heuristics concerns the selection of the rank and its influence on the quality of the approximation. In the initial formulation of the algorithm, the rank was held fixed throughout all iterations. However, as the design space is progressively reduced, the dimensionality of the solution may also decrease. Consequently, the rank required to achieve a prescribed approximation accuracy can vary from one iteration to the next. This observation motivates the development of a controlled strategy for adapting the rank. A natural starting point is the approach used in the TDB-CUR algorithm, discussed in Section 3.4.2, where rank adaptivity is performed during time integration. In that setting, the adaptation is guided by the singular values through the metric:

$$\epsilon(t) = \frac{\hat{\sigma}_r}{(\sum_{i=1}^r \hat{\sigma}_i(t)^2)^{1/2}}, \quad (5.4)$$

where $\hat{\sigma}_i$ denotes the singular values of the low-rank approximation from the previous time step. Upper and lower thresholds, denoted by ϵ_u and ϵ_l , are used to determine whether the rank should be increased or decreased.

In the present work, we do not perform adaptation in time, but rather across successive iterations of the algorithm. Nevertheless, the same underlying idea can be leveraged. In particular, we adopt the metric in (5.4) as an indicator of the relative contribution of the

smallest retained singular value. To avoid counteracting the benefits of domain shrinkage, we restrict the adaptation mechanism so that the rank may only decrease. Accordingly, only an upper threshold ϵ_u is introduced. A subtle complication arises from the fact that the singular values are not guaranteed to correspond exactly between iterations. As a result, direct comparisons based solely on individual singular values may be unreliable. To mitigate this issue, we once again employ a percentile based variant of the criterion. Specifically, we evaluate the error proxy in (5.4) at singular value indices corresponding to the 85% and 95% percentiles of the spectrum, rather than relying solely on the smallest retained singular value. A rank reduction is permitted only if both resulting indicators fall below the threshold ϵ_u . This provides a more robust assessment that the tail of the spectrum is sufficiently small.

In addition, to prevent overly aggressive or premature reductions in rank, a patience mechanism is incorporated. That is, the reduction criterion must be satisfied for two consecutive iterations before the rank is decreased. This safeguard helps ensure that the ROM remains stable and that important modes are not discarded too early in the process.

It is important to note that the strategies proposed here are inherently heuristic. Although they are informed by well-established properties of singular value spectra, including decay behaviour, conditioning, and perturbation sensitivity, they are not derived from a formal optimality principle. As a result, alternative formulations could be considered. The specific choices of mappings, thresholds, and combination rules are empirical in nature and motivated by considerations of interpretability, robustness, and ease of implementation. In the numerical experiments considered in this work, they were observed to perform consistently across the test cases studied. Nevertheless, several limitations should be acknowledged. The effectiveness of the approach may depend on problem-specific spectral characteristics, such as the rate and regularity of singular value decay, as well as the presence of numerical noise. In addition, the use of fixed tolerances and percentile-based measures may reduce sensitivity in cases where the spectrum does not exhibit a clear separation between dominant and trailing modes. Furthermore, the inter-iteration comparison assumes a degree of consistency in the underlying subspaces, which may not hold if the system undergoes significant structural changes. Despite these considerations, the proposed framework provides a simple and flexible basis for adaptive shrinkage and rank selection, and can be readily adjusted or extended for different problem settings. Algorithms 6 and 7 summarize the implementation of the shrink factor and rank adaptivity procedures, respectively.

Algorithm 6 General Shrink Factor Adaptivity

Input: $r_{\text{current}}, \hat{\mathbf{S}}^{(j)}, \hat{\mathbf{S}}^{(j-1)}, \tau, a_g, z_{\min}$

Output: z

```

 $i_{\text{tail}} \leftarrow \max(1, \min(r_{\text{current}} - 1, \text{int}(0.9r_{\text{current}})))$        $\triangleright$  Find index for the 90th percentile
 $w \leftarrow \min(5, r_{\text{current}})$        $\triangleright$  Find the number of indices for the slope computation
 $r_{\min} \leftarrow \min(r_{\text{current}}, \text{len}(\hat{\mathbf{S}}^{(j-1)}))$        $\triangleright$  Find minimum overlapping modes
if  $r_{\text{current}} == 1$  then
     $\epsilon_t = 0$        $\triangleright$  If only one mode, set tail ratio to zero
     $\epsilon_m = 0$        $\triangleright$  If only one mode, set slope to zero
else
     $\epsilon_t \leftarrow \hat{\mathbf{S}}^{(j)}(i_{\text{tail}}) / \hat{\mathbf{S}}^{(j)}(1)$        $\triangleright$  Compute tail ratio
    if  $w \geq 3$  then
         $\mathbf{y} \leftarrow \log(\mathbf{S}^{(j)}(r - w + 1 : r))$        $\triangleright$  Get the logarithm of the singular values
         $\mathbf{x} \leftarrow \text{linspace}(0, 1, w)$        $\triangleright$  Map indices for fitting
         $\epsilon_m \leftarrow \text{leastSquaresFit}(\mathbf{x}, \mathbf{y})$        $\triangleright$  Compute least squares to obtain slope
    else
         $\epsilon_m = 0$        $\triangleright$  If less than three points, set slope to zero
    end if
end if
 $\epsilon_{\Delta} \leftarrow \|\hat{\mathbf{S}}^{(j)}(:, r_{\min}) - \hat{\mathbf{S}}^{(j-1)}(:, r_{\min})\|_2 / \|\hat{\mathbf{S}}^{(j-1)}(:, r_{\min})\|_2$        $\triangleright$  Get relative change signal
 $z_t \leftarrow 1.0 + 0.6e^{-6\epsilon_t}$        $\triangleright$  Map tail ratio to a shrink factor value
 $z_m \leftarrow 1.0 + 0.5\text{clip}(-\epsilon_m, 0, 1.5)$        $\triangleright$  Map slope to a shrink factor value
 $z_{\Delta} \leftarrow 1.0 - 0.7 \tanh(3(\epsilon_{\Delta} - \tau))$        $\triangleright$  Map relative change to a shrink factor value
 $z_b \leftarrow (z_t \cdot z_m \cdot z_{\Delta})^{1/3}$        $\triangleright$  Get the base shrink factor
 $z \leftarrow \max(z_{\min}, z_b^{\frac{a_g}{b_s}})$        $\triangleright$  Scale shrink factor via aggressiveness value  $a_g$ 

```

Algorithm 7 General Rank Adaptivity

Input: $r_{\text{current}}, \hat{\mathbf{S}}^{(j)}, \epsilon_u, \text{patienceBound}$

Output: r

$i_{85} \leftarrow \max(1, \min(r_{\text{current}} - 1, \text{int}(0.85r_{\text{current}})))$ $\triangleright$ Find index for the 85th percentile

$i_{95} \leftarrow \max(1, \min(r_{\text{current}} - 1, \text{int}(0.95r_{\text{current}})))$ $\triangleright$ Find index for the 95th percentile

$\epsilon_{\text{total}} \leftarrow (\sum_{i=1}^{r_{\text{current}}} (\hat{\mathbf{S}}^{(j)}(i))^2)^{1/2}$ $\triangleright$ Compute total energy in the current singular values

$\epsilon_{85} \leftarrow \hat{\mathbf{S}}^{(j)}(i_{85})/\epsilon_{\text{total}}$ $\triangleright$ Get 85% energy estimate

$\epsilon_{95} \leftarrow \hat{\mathbf{S}}^{(j)}(i_{95})/\epsilon_{\text{total}}$ $\triangleright$ Get 95% energy estimate

if $\epsilon_{85} < \epsilon_u$ and $\epsilon_{95} < \epsilon_u$ **then**

$\text{iterCounter} \leftarrow \text{iterCounter} + 1$ $\triangleright$ If both are below ϵ_u , increase counter

else

$\text{iterCounter} \leftarrow 0$ $\triangleright$ Update counter back to zero

end if

if $\text{iterCounter} \geq \text{patienceBound}$ **then**

$r_{\text{new}} \leftarrow \max(r_{\text{min}}, r_{\text{current}} - 1)$ $\triangleright$ Decrease rank by one if patience bound is satisfied

$\text{iterCounter} \leftarrow 0$ $\triangleright$ Update counter back to zero

else

$r_{\text{new}} \leftarrow r_{\text{current}}$ $\triangleright$ If patience bound has not been satisfied do not change rank

end if

$r \leftarrow r_{\text{new}}$ $\triangleright$ Update rank

5.2 Results

In the following, we extend the results from the single-parameter case in Section 4.3 to shape optimization problems involving multiple design variables for a converging-diverging nozzle. The formulation closely follows that introduced previously, and we again consider the inverse problem of minimizing the energy-like functional from [42]:

$$\mathcal{J}(\boldsymbol{\theta}) = \frac{1}{2} \|p - p^*\|_2^2,$$

where $\boldsymbol{\theta}$ now denotes the vector of design parameters controlling the nozzle geometry. The remaining notation follows directly from Section 4.3, with $p = p(x, t; \boldsymbol{\theta})$ denoting the predicted pressure field and $p^* = p^*(x, t_F; \boldsymbol{\theta}^*)$ the corresponding target solution. The L^2 -norm is taken over the spatial domain.

The flow configuration remains a subsonic inflow-outflow problem, which is evolved to a steady state at $t_F = 3.0$, as in the previous chapter. The geometry is parameterized using cubic B-spline basis functions defined over the axial domain. The cross-sectional area is expressed as:

$$a(x, \boldsymbol{\theta}) = \sum_{i=1}^d \theta_i \psi_i(x),$$

where $\psi_i(x)$ denotes the i^{th} basis function and d is the dimension of the design space, with each coefficient controlling a localized geometric region.

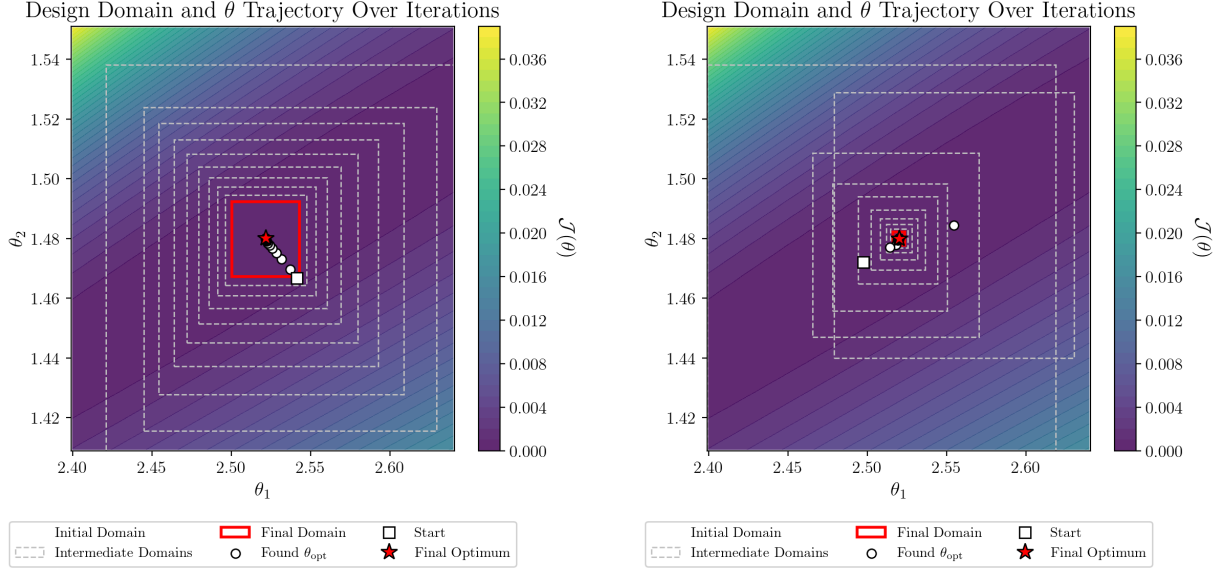
All simulations are performed on a fixed mesh using a global CG discretization of degree $q = 2$ with $k = 50$ elements, resulting in a solution matrix $\mathbf{V} \in \mathbb{R}^{303 \times N_s}$. Time integration is carried out using an explicit RK4 scheme with a time step of $\Delta t = 5.0 \times 10^{-5}$.

5.2.1 Two Design Variables

We first aim to solve a simple two design variable problem, with only $N_s = 400$ samples, corresponding to twenty samples per direction in the tensor-product grid. For this first test we focused on comparing how two slightly different rank values may have an effect on the error and performance of the algorithm. Specifically, we compare the cases $r = 2$ and $r = 4$. In both settings, we use $m = 2$ oversampling points and an adaptive shrink factor with aggressiveness $a_g = 0.35$. The minimum number of iterations before termination is set to $\text{minIters} = 10$.

Figure 5.5 shows the trajectory of the optimal solutions together with the evolution of the design domain over iterations, overlaid on a contour map of the objective function. In particular, Figure 5.5a corresponds to the lower-rank case. In this setting, the design domains appear to be nested across iterations, with the solution trajectory remaining largely linear and progressing toward the optimum in relatively small steps. In contrast, the higher-rank case in Figure 5.5b exhibits a more exploratory behaviour in the early iterations, before gradually converging and shrinking toward the identified optimum.

Figure 5.6 shows the trajectories of the shrink factors over the algorithm iterations for both rank cases. A clear difference is observed between the two settings. In the lower-rank case, Figure 5.6a, the shrink factor increases gradually, reaching values slightly above $z \approx 1.20$. In contrast, the higher-rank case, Figure 5.6b, attains larger shrink factor values, around $z \approx 1.45$.



(a) Trajectory of the optimal solutions over iterations for the $r = 2$ case, overlaid on a contour plot of the objective function. The evolving design domains are shown in white.

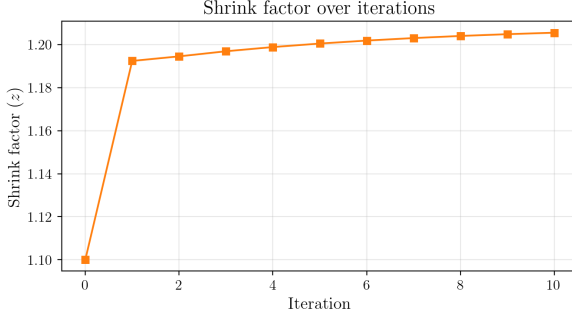
(b) Trajectory of the optimal solutions over iterations for the $r = 4$ case, overlaid on a contour plot of the objective function. The evolving design domains are shown in white.

Figure 5.5: Trajectory of the optimal solutions and design domains over optimization iterations for two different rank cases.

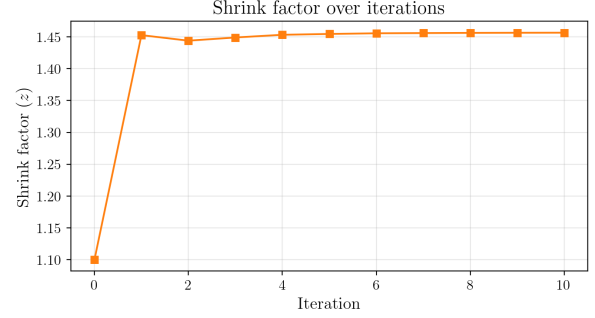
This behaviour suggests that the adaptive shrink strategy responds to differences in model fidelity. In the lower-rank case, the ROM is more strongly constrained and captures only the most dominant modes of the system. As a result, the estimated spectral decay is flatter, the relative variation across iterations is higher, and the tail behaviour is less informative. Collectively, these effects reduce the reliability of the ROM representation and lead to a more conservative shrink update.

In contrast, for the $r = 4$ case, the spectral structure is better resolved, resulting in a steeper decay, improved stability, and reduced tail energy. This in turn supports more aggressive shrinkage during the optimization process. This trend is also reflected in Figure 5.5, where the $r = 2$ case terminates with a larger final design domain compared to the $r = 4$ case, consistent with the differences observed in the shrink factor evolution.

In Figure 5.7, the final pressure profile is compared with the target profile p^* and the initial domain bounds, with both the maximum absolute and relative L^2 errors reported. Both cases successfully converged to the target pressure profiles, at slightly dif-



(a) Trajectory of the shrink factor over algorithm iterations for the $r = 2$ case, hovering to approximately $z \approx 1.20$



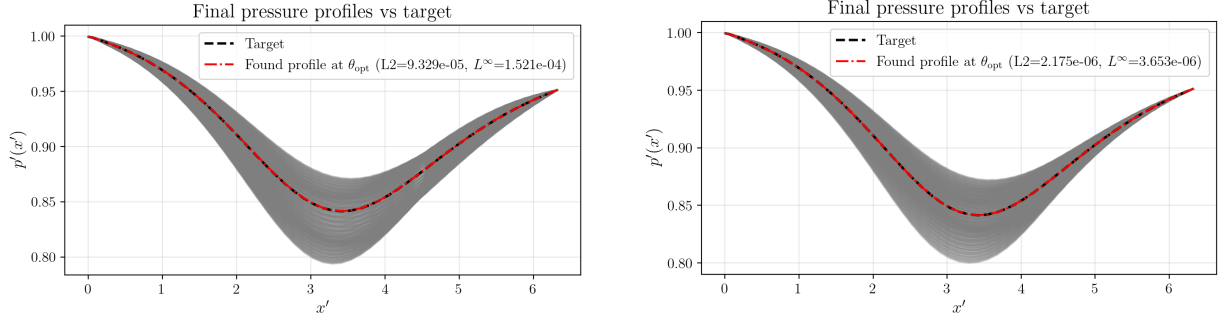
(b) Trajectory of the shrink factor over algorithm iterations for the $r = 4$ case, hovering to approximately $z \approx 1.45$

Figure 5.6: Trajectory of the shrink factor over optimization iterations for two different rank cases.

ferent accuracy levels. In Figure 5.7a, the $r = 2$ case achieves a maximum absolute error of $\|p - p^*\|_\infty \approx 1.52 \times 10^{-4}$. In contrast, in Figure 5.7b, adding just two additional modes significantly improves accuracy, reducing the maximum absolute error to $\|p - p^*\|_\infty \approx 3.65 \times 10^{-6}$. One other piece to note is the difference between the two initial domains the two cases are plotted over. Although both sets of profiles are evaluated for the exact same problem, with identical initial design spaces, the resulting distributions differ noticeably with rank. In the $r = 2$ case, the reduced basis under-resolves the underlying dynamics, leading to a broader spread of admissible profiles, particularly near the outflow. Essentially, with such few modes, in the initial domain, this case is not able to enforce the tighter relationships that would normally constrain those degrees of freedom. In contrast, the $r = 4$ approximation captures more of the system's underlying structure, resulting in profiles that are more tightly clustered and consistent across the domain.

Finally, in Figure 5.8, the error convergence is illustrated, showing a consistent downward trend in both the maximum absolute and relative L^2 errors, for both cases, with a faster convergence found in the $r = 4$ case. This steady decay rate in Figure 5.8a also suggests that if the $r = 2$ case were run for longer, it may continue to reduce its error.

Overall, these results highlight the strong coupling between rank selection, model fidelity, and the adaptive shrinkage mechanism. While both $r = 2$ and $r = 4$ successfully converge toward the target solution, the higher-rank approximation consistently yields improved accuracy, tighter profile distributions, and faster error reduction. This is directly reflected in the shrink factor behavior, the $r = 4$ case admits more aggressive domain reduc-



(a) Comparison between the target and recovered pressure profiles across the design domain, reporting the max absolute and relative L^2 errors for the $r = 2$ case.

(b) Comparison between the target and recovered pressure profiles across the design domain, reporting the max absolute and relative L^2 errors for the $r = 4$ case.

Figure 5.7: Final pressure profiles found for two different rank cases.

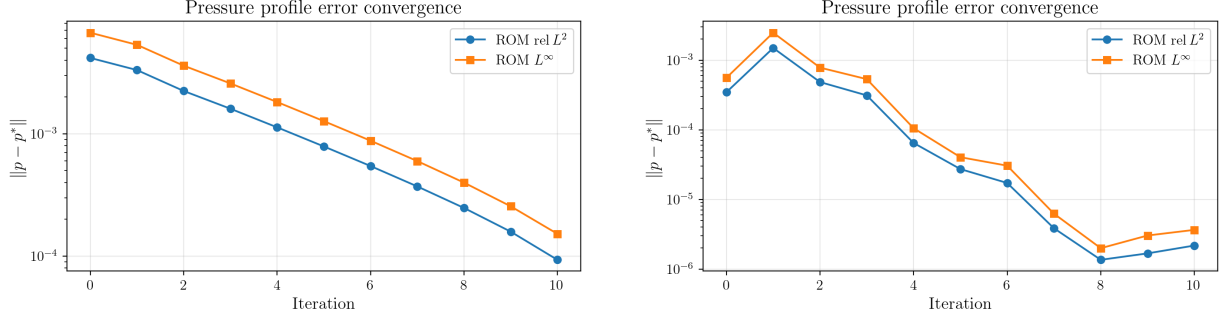
tion due to a more reliable representation of the underlying dynamics, whereas the $r = 2$ case remains more conservative due to its limited expressiveness and increased variability.

Importantly, the wider range of profiles observed in the $r = 2$ case does not reflect more complex dynamics, but rather a reduced ability of the model to accurately represent the system. As a result, the effective design space is less well constrained, which slows convergence and limits the achievable accuracy. Taken together, these findings demonstrate that even modest increases in rank can significantly enhance both the reliability of the ROM and the effectiveness of the adaptive shrinkage strategy, leading to more efficient and accurate optimization.

5.2.2 Four Design Variables

We consider next a four design variable case with $N_s = 4,096$ samples, corresponding to eight samples per direction in the tensor-product grid. The algorithm is initialized with rank $r = 15$, $m = 2$ oversampling points, and an adaptive shrink factor with aggressiveness $a_g = 0.35$. The minimum number of iterations is set to $\text{minIters} = 15$.

Figure 5.9 shows the trajectories of the four design variables over iterations. The gray regions indicate the shrinking design space, while the best ROM sample converges toward the optimum at each iteration. In Figure 5.10a, the final pressure profile is compared with the target profile p^* and the initial domain bounds, yielding a maximum absolute error of



(a) Progression of the max absolute and relative L^2 errors with respect to the target profile over successive algorithm iterations for the $r = 2$ case.

(b) Progression of the max absolute and relative L^2 errors with respect to the target profile over successive algorithm iterations for the $r = 4$ case.

Figure 5.8: Progression of the max absolute and relative L^2 errors with respect to the target profile over successive algorithm iterations for two different rank cases.

$\|p - p^*\|_\infty \approx 9.53 \times 10^{-9}$, demonstrating successful convergence. The error convergence is further illustrated in Figure 5.10b, which shows a consistent downward trend in both the maximum absolute and relative L^2 errors.

Figure 5.11 depicts the effects of adaptive rank and shrink factor. In Figure 5.11a, the rank begins at $r = 15$ and decreases stepwise to $r = 11$, reflecting the desired behaviour of the rank adaptivity. In Figure 5.11b, the shrink factor starts near $z = 1.10$, enforcing early shrinkage, then increases and stabilizes around $z = 1.45$, indicating controlled adaptation as desired.

The second test case for the four design variable problem focuses on comparing the computational runtime of the FOM and ROM approaches. Specifically, we assess the proposed optimization methodology against a FOM-based variant in which the FOM replaces the TDB-CUR component in Algorithm 3, with the goal of quantifying differences in accuracy and computational cost. This test is carried out across three sample sizes, $N_s = 256$, 1,296, and 4,096, for both model types. For the ROM, we consider two rank configurations, $r = 5$ and $r = 15$, with a fixed oversampling parameter $m = 2$. We also test two iteration strategies, one enforcing a minimum of eleven iterations, and one using the termination conditions outlined in Section 4.2.2. For the FOM cases, all runs are fixed to eleven iterations to provide a consistent reference for both error and computational cost under a fixed iteration budget.

In Table 5.1, we present the detailed performance metrics, where the rank r , sample

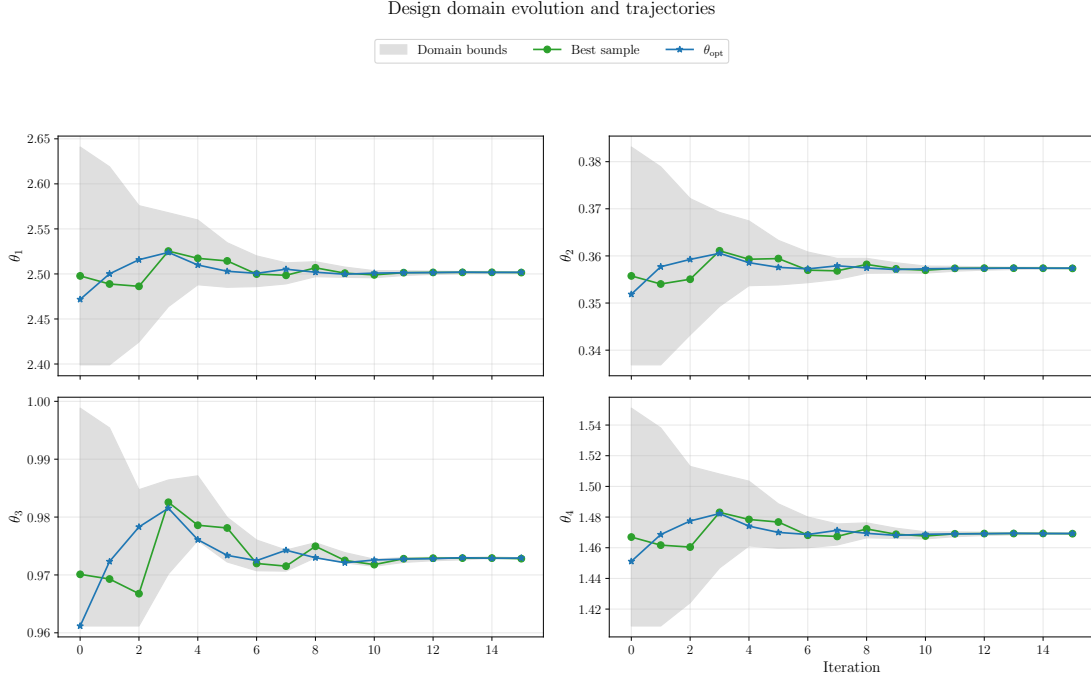


Figure 5.9: Trajectory of the four design variables over optimization iterations. In gray is the design domain for each variable, with the trajectory of the best sample and optimal θ plotted overtop. The best sample is taken to be the column in the ROM that produces the smallest objective value.

size N_s , and iteration counts are tabulated by model configuration. The columns provide a direct comparison of total and per-iteration runtimes against the matching FOM baselines, alongside the corresponding speedup factors and maximum absolute errors for each case.

Across all configurations, the ROM consistently achieves a substantially lower total runtime than the corresponding FOM evaluations, while maintaining comparable accuracy. In particular, the ROM errors remain on the order of $\|p - p^*\|_\infty \approx 1.0 \times 10^{-6}$ to $\|p - p^*\|_\infty \approx 1.0 \times 10^{-8}$, closely matching the FOM error levels across all tested sample sizes. Among the configurations considered, the case $r = 15$ with $N_s = 256$ yields the lowest maximum absolute error of $\|p - p^*\|_\infty \approx 4.55 \times 10^{-8}$, followed closely by $r = 15$ with $N_s = 1,296$, yielding $\|p - p^*\|_\infty \approx 7.92 \times 10^{-8}$, both at a substantially lower computational cost than the FOM runs.

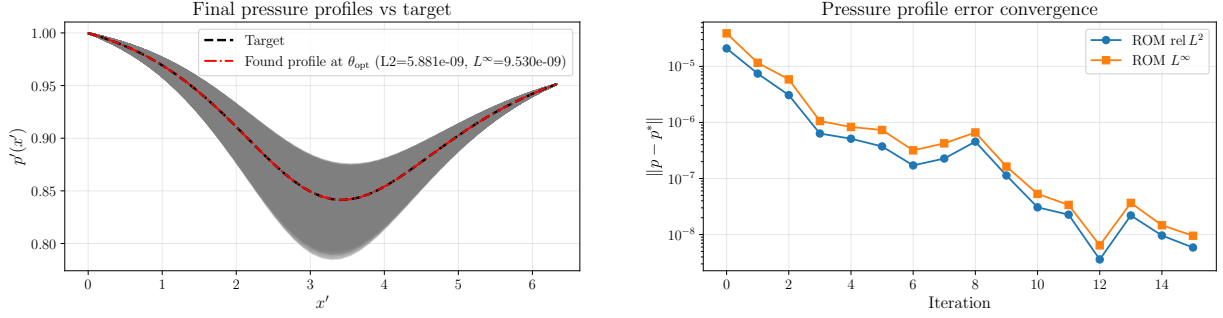
The FOM errors exhibit only mild variation across sample sizes, ranging from $\|p - p^*\|_\infty \approx 7.55 \times 10^{-8}$ to $\|p - p^*\|_\infty \approx 4.0 \times 10^{-8}$, indicating that the optimization problem

Table 5.1: Runtime and error comparison across 4D test cases relative to an 11-iteration FOM baseline. Speedup factors are evaluated relative to the matching baseline for each respective sample size N_s . Colour highlights indicate the highest and lowest bounds for runtime, speedup, and $\|p - p^*\|_\infty$ error metrics for the ROM only, as detailed below.

Runtime Comparison Across Test Cases (4D)

Configuration	N_s	Runtime (s)	Runtime (min)	Speedup	Runtime/Iter (s)	Speedup/Iter	L_∞ Error
FOM (iters=11)	256	10470.82	174.51	1.00x	951.8930	1.00x	7.5465e-08
FOM (iters=11)	1296	45913.99	765.23	1.00x	4173.9993	1.00x	7.6694e-08
FOM (iters=11)	4096	157223.58	2620.39	1.00x	14293.0524	1.00x	3.9982e-08
ROM (r=5, iters=6)	256	2081.65	34.69	5.03x	346.9418	2.74x	1.2584e-04
ROM (r=5, iters=11)	256	3686.39	61.44	2.84x	335.1260	2.84x	2.6079e-06
ROM (r=15, iters=7)	256	4793.92	79.90	2.18x	684.8454	1.39x	2.3513e-06
ROM (r=5, iters=6)	1296	5686.26	94.77	8.07x	947.7098	4.40x	4.4176e-05
ROM (r=15, iters=11)	256	6963.46	116.06	1.50x	633.0418	1.50x	4.5505e-08
ROM (r=5, iters=11)	1296	7667.61	127.79	5.99x	697.0556	5.99x	2.5857e-06
ROM (r=15, iters=6)	1296	10155.63	169.26	4.52x	1692.6053	2.47x	4.3535e-07
ROM (r=5, iters=6)	4096	11380.16	189.67	13.82x	1896.6940	7.54x	6.0548e-05
ROM (r=15, iters=11)	1296	18414.88	306.91	2.49x	1674.0798	2.49x	7.9153e-08
ROM (r=5, iters=11)	4096	20511.78	341.86	7.67x	1864.7072	7.67x	2.5664e-06
ROM (r=15, iters=7)	4096	26040.95	434.02	6.04x	3720.1355	3.84x	2.0400e-07
ROM (r=15, iters=11)	4096	42045.89	700.76	3.74x	3822.3538	3.74x	1.3134e-07

Fastest ROM (Total & Per-Iteration)	Best ROM Speedup (Total & Per-Iteration)	Best ROM Error (L_∞)
Slowest ROM (Total & Per-Iteration)	Worst ROM Speedup (Total & Per-Iteration)	Worst ROM Error (L_∞)



(a) Comparison between the target and recovered pressure profiles across the design domain, reporting the max absolute and relative L^2 errors.

(b) Progression of the max absolute and relative L^2 errors with respect to the target profile over successive algorithm iterations.

Figure 5.10: Four design variable optimization results illustrating algorithm performance and convergence.

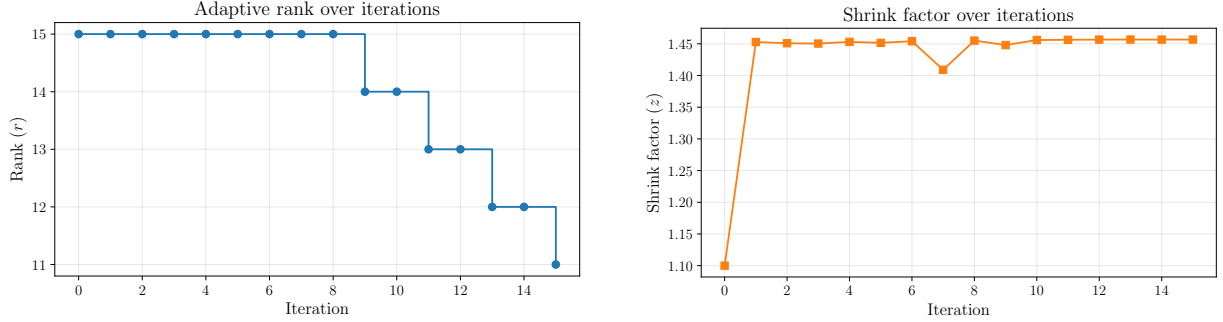
is already well-resolved even at lower sampling densities. Consequently, increasing N_s or the number of iterations primarily increases the total runtime rather than substantially improving accuracy, while the ROM retains its efficiency advantage across all settings.

These results demonstrate the effectiveness of the proposed algorithm. The hybrid optimization strategy, combined with rank and shrink factor adaptivity, enables efficient reduction of the design space while preserving sufficient modes to accurately recover the target profile. This balance is reflected in consistent low errors and observed computational savings compared to the FOM. Overall, the results highlight the method's ability to trade off accuracy and computational cost in a controlled manner.

5.2.3 Six Design Variables

The second case of consideration is that of six design variables. For the first test in this section we focused on comparing the effects of different aggressiveness values on the error and performance of the algorithm. Specifically, we take $r = 10$, $m = 2$ oversampling points, and $N_s = 4,096$ samples equating to four samples per direction in the tensor product grid. All three cases are forced to perform the same number of iterations, which we set to 16, and the three a_g values of consideration are:

1. case 1: $a_g = 0.1$,



(a) Trajectory of the algorithmic rank over iterations, showing how it evolves throughout the process.

(b) Trajectory of the shrink factor over iterations, converging to approximately $z \approx 1.45$ as the algorithm stabilizes.

Figure 5.11: Adaptive evolution of rank and shrink factor over algorithm iterations.

2. case 2: $a_g = 0.5$,
3. case 3: $a_g = 1.0$.

In Figure 5.12a, the value of the shrink factors over algorithm iterations is plotted. Specifically, the least aggressive case $a_g = 0.1$, produces a shrink factor of approximately $z = 1.11$ per iteration, whereas $a_g = 0.5$ produces a shrink factor of approximately $z = 1.7$, and finally the most aggressive case $a_g = 1.0$ produces a shrink factor that sits around $z = 2.89$. In all three cases, we see a similar stable behaviour where it reaches the target shrink factor for that value of a_g and stays around that value for all iterations.

Further, in Figure 5.12b, the target profile is plotted against the results from each of the three different shrink factor cases. All three cases are able to find the target profiles quite easily, however, as we see in Figure 5.12c, their convergence behaviour differs significantly. Case 3 starts out quick, shooting down to a maximum absolute error of around $\|p - p^*\|_\infty \approx 5.6 \times 10^{-5}$ in just six iterations, however, it remains stuck at this level for the remainder of the experiment. This is indicative of convergence to a local minimum, where subsequent iterations fail to identify a descent direction sufficient to escape. The least aggressive case, has the opposite effect. the error progressively decreases, ending at around $\|p - p^*\|_\infty \approx 3.8 \times 10^{-4}$. However, it is clearly trending downwards, indicating that it could generate a solution closer to the target profile if the number of iterations was greatly increased. Finally, the case that produced the best overall performance, Case 2, moved slower and had a higher error than that of Case 3 in early iterations. However, it is able to out perform it by the end of the iterations obtaining a maximum absolute error of about

$\|p - p^*\|_\infty \approx 6.7 \times 10^{-6}$. These results highlight the critical role that the shrink factor, and the aggressiveness a_g play in balancing convergence speed and solution quality. An overly aggressive choice may lead to premature convergence to suboptimal solutions, whereas a moderate setting can enable better refinement.

Similar to Section 5.2.2, the second test for the six design variable case assesses the proposed framework against a FOM baseline. However, due to the computational expense of the FOM, the baseline is limited to the first iteration of the proposed framework, with the FOM replacing the TDB-CUR component. Although more limited than the four design variable study, this comparison still provides insight into the accuracy and computational cost of the proposed methodology.

This comparison is performed across three sample sizes, $N_s = 4,096$, $15,625$, and $46,656$. For the ROM, two rank configurations are considered, $r = 5$ and $r = 15$, both with $m = 2$ oversampling points. Two iteration strategies are also tested, one enforcing a minimum of eleven iterations, and one based on the termination criteria defined in Section 4.2.2.

In Table 5.2, we present the detailed performance metrics, where the rank r , sample size N_s , and iteration counts are tabulated by model configuration. The columns provide a direct comparison of total and per-iteration runtimes against the matching FOM baselines, alongside the corresponding speedup factors and maximum absolute errors for each case.

In general, the ROM achieves maximum absolute errors comparable to the FOM, though with variable total runtime. Specifically, while all $r = 15$ configurations exceed the runtime of their respective FOM baselines, they consistently yield slightly improved error results. This ability to outperform the FOM in accuracy is likely attributable to the algorithm’s capacity for iterative self-refinement during execution.

Furthermore, these results suggest that for the current 6D problem configuration on a coarse mesh, the $r = 15$ ROM cases exhibit a higher total absolute runtime than the FOM baseline. However, this total runtime comparison is inherently biased by the structural setup of the test cases: while the FOM baseline is executed for only a single iteration, the ROM variants undergo a full optimization trajectory. Evaluating performance on a per-optimization-iteration basis corrects for this discrepancy and reveals that the ROM is performing exactly as intended, achieving a minimum of a three-fold speedup per iteration. Notably, these per-iteration savings exceed the performance gains observed in the lower-dimensional 4D cases, demonstrating that the computational efficiency of the reduction strategy scales favorably with increasing problem dimensionality.

One additional observation is that the ROM tests where the algorithm was forced to run longer tend to have better error results compared to the cases that were allowed to finish solely based on termination conditions. This is an indicator that the termination conditions

Table 5.2: Runtime and error comparison across 6D test cases relative to a single-iteration FOM baseline. Speedup factors are evaluated relative to the matching baseline for each respective sample size N_s . Colour highlights indicate the highest and lowest bounds for runtime, speedup, and $\|p - p^*\|_\infty$ error metrics for the ROM only, as detailed below.

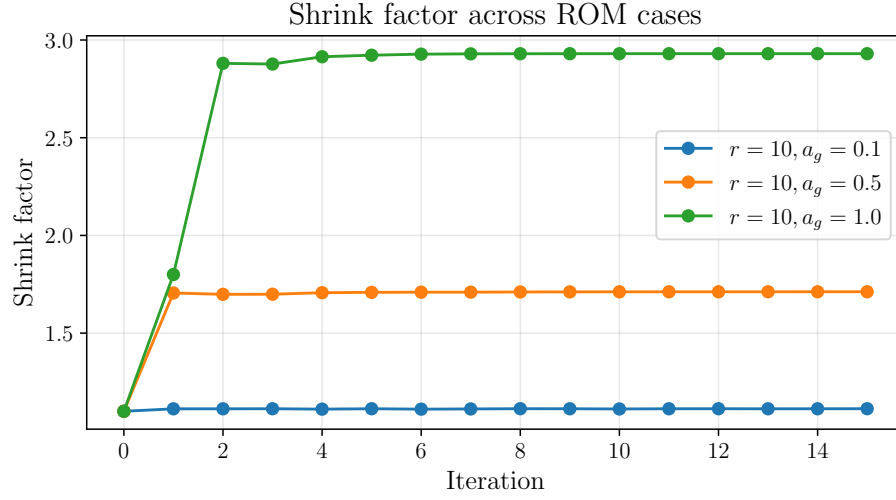
Runtime Comparison Across Test Cases (6D)

Configuration	N_s	Runtime (s)	Runtime (min)	Speedup	Runtime/Iter (s)	Speedup/Iter	L_∞ Error
FOM (iters=1)	4096	19145.49	319.09	1.00x	19145.4890	1.00x	2.6457e-04
FOM (iters=1)	15625	66840.98	1114.02	1.00x	66840.9830	1.00x	7.2060e-05
FOM (iters=1)	46656	240664.98	4011.08	1.00x	240664.9800	1.00x	9.4700e-05
ROM (r=5, iters=6)	4096	11729.63	195.49	1.63x	1954.9385	9.79x	1.2764e-03
ROM (r=5, iters=11)	4096	18814.42	313.57	1.02x	1710.4020	11.19x	3.5849e-04
ROM (r=15, iters=7)	4096	38638.57	643.98	0.50x	5519.7956	3.47x	1.2262e-04
ROM (r=15, iters=11)	4096	45051.45	750.86	0.42x	4095.5864	4.67x	6.8950e-05
ROM (r=5, iters=6)	15625	56534.41	942.24	1.18x	9422.4011	7.09x	1.6625e-03
ROM (r=5, iters=11)	15625	68072.68	1134.54	0.98x	6188.4258	10.80x	3.8523e-04
ROM (r=15, iters=6)	15625	123948.61	2065.81	0.54x	20658.1017	3.24x	4.1056e-05
ROM (r=5, iters=6)	46656	134649.05	2244.15	1.79x	22441.5078	10.72x	1.1168e-03
ROM (r=15, iters=11)	15625	151038.15	2517.30	0.44x	13730.7406	4.87x	1.1844e-05
ROM (r=5, iters=11)	46656	236094.81	3934.91	1.02x	21463.1646	11.21x	3.8258e-04
ROM (r=15, iters=7)	46656	388319.85	6472.00	0.62x	55474.2642	4.34x	3.1097e-05
ROM (r=15, iters=11)	46656	567787.82	9463.13	0.42x	51617.0745	4.66x	3.7542e-06

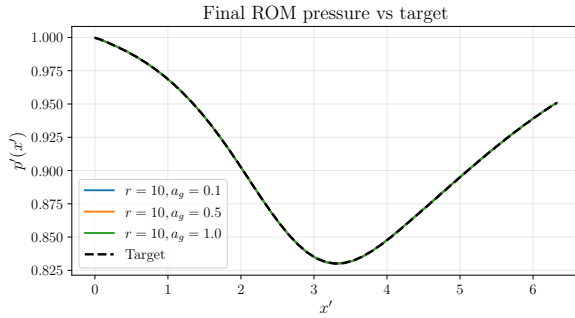
Fastest ROM (Total & Per-Iteration)	Best ROM Speedup (Total & Per-Iteration)	Best ROM Error (L_∞)
Slowest ROM (Total & Per-Iteration)	Worst ROM Speedup (Total & Per-Iteration)	Worst ROM Error (L_∞)

could be too loose and could be tightened depending on the needs of the problem. We also see a similar pattern in the rank cases: typically, the higher the rank, the better the error, yet this comes at a larger computational cost. This is indicative of the tradeoff nature of ROMs in general; higher accuracy often requires higher computational cost.

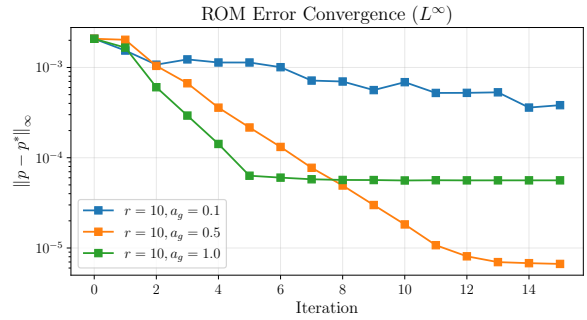
Hence, these results highlight the potential power of the TDB-CUR algorithm to reduce costs, while, when combined with the proposed approach, achieving error metrics that are comparable to, or even better than, those of the corresponding FOM.



(a) Plot of the resulting shrink factor over algorithm iterations. Using $a_g = 0.1$ the shrink factor stabilizes around $z = 1.11$, $a_g = 0.5$ stabilizes around $z = 1.70$, and $a_g = 1.0$ stabilizes around $z = 2.89$.



(b) Final pressure profiles plotted against the target profile for the three shrink factor cases.



(c) Max absolute error convergence over algorithm iterations for the three shrink factor cases.

Figure 5.12: Effect of different shrink factor aggressiveness (a_g) on algorithm performance.

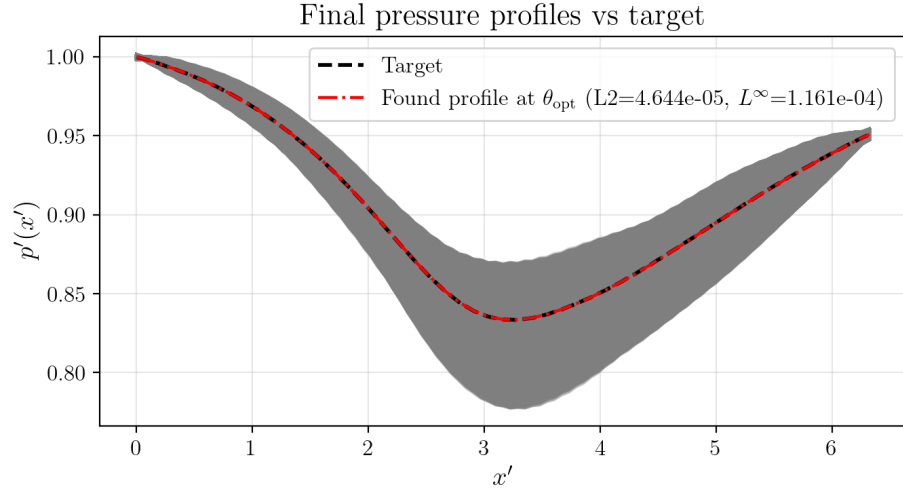
5.2.4 Ten Design Variables

The final test we present is that of the ten design variable case. In particular, we compute the solution on the same mesh as in the previous two cases, and take $N_s = 59,049$ samples, corresponding to only three samples per design direction in the tensor product grid. This represents an intentionally sparse sampling regime relative to the dimensionality of the problem. The rank is kept fixed at $r = 15$, with $m = 2$ oversampling points and an aggressiveness parameter of $a_g = 0.35$. Unlike the previous cases, no minimum number of iterations is enforced, nevertheless, the algorithm satisfies the termination criteria after eight iterations, indicating rapid convergence despite the increased problem complexity.

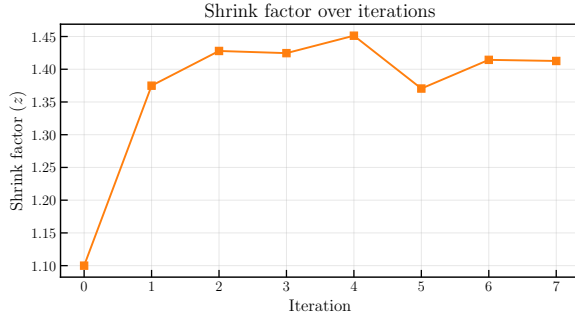
In Figure 5.13a, the final and target pressure profiles are plotted over the initial design domain. The algorithm successfully recovers the target profile, achieving a maximum absolute error of $\|p - p^*\|_\infty \approx 1.61 \times 10^{-4}$. This level of accuracy is notable given the limited sampling density in each design direction. Figure 5.13b shows the evolution of the shrink factor over the optimization iterations. In contrast to the lower-dimensional cases, the shrink factor exhibits mild oscillations around $z = 1.40$, suggesting a slightly less stable but still well-controlled update behavior in higher dimensions.

Finally, Figure 5.13c illustrates the convergence behavior of the algorithm. Both the maximum absolute error and the relative L^2 error decrease monotonically over successive iterations, demonstrating consistent progress toward the target solution. This trend confirms that the algorithm maintains robustness and convergence even in the presence of increased dimensionality and limited sampling.

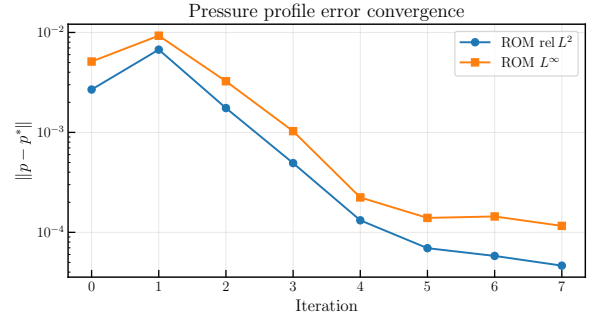
Overall, these results highlight the scalability of the proposed approach. Even in a ten-dimensional setting with sparse sampling, the method achieves accurate reconstruction and stable convergence, underscoring its effectiveness for even higher-dimensional design problems.



(a) Plot of the target and found pressure profiles over the design domain, and the max absolute and relative L^2 error reported.



(b) Trajectory of the shrink factor over algorithm iterations, slightly oscillating around $z \approx 1.40$.



(c) Progression of the max absolute and relative L^2 errors with respect to the target profile over successive algorithm iterations.

Figure 5.13: Ten design variable optimization results illustrating algorithm performance and convergence.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis presented the development of a shape optimization framework based on the TDB-CUR algorithm for PDE-constrained problems governed by the quasi-one-dimensional Euler equations. The proposed approach combines adaptive design space reduction with a low-rank ROM, enabling the optimization procedure to progressively focus on regions near the optimum while maintaining a computationally efficient representation.

To support this development, the core numerical and algorithmic components were first established. In Chapter 2, an entropy-stable FOM based on an SBP discretization of the quasi-one-dimensional Euler equations was introduced, providing a robust and stable foundation for the subsequent formation of the ROM. This was followed in Chapter 3 by a detailed presentation of the TDB-CUR algorithm, including its non-intrusive formulation, robustness properties, and error-adaptive features such as oversampling and rank adaptivity. Applications to both inviscid and viscous Burgers' equations were also considered, offering insight into how the underlying system behavior influences the performance of the method. In particular, smooth solutions were observed to exhibit stronger low-rank structure, whereas more complex behavior involving discontinuities, such as shock formation, even in the one-dimensional settings, led to a higher reported rank for a given level of accuracy.

Building on these elements, Chapter 4 introduced the proposed optimization algorithm, which integrates TDB-CUR within an iterative design space reduction framework. The resulting approach follows a surrogate-inspired strategy, employing a tensor-product

interpolant with favorable approximation properties, coupled with a hybrid optimization scheme. Specifically, a gradient-free method (DE) is used in the early stages when the ROM is less reliable, followed by gradient-based refinement using L-BFGS-B as the model accuracy improves, enabling a transition from global exploration to local refinement.

Finally, Chapter 5 presented adaptive heuristics for controlling both the shrink factor and the rank. These heuristics were based on metrics designed to estimate the decay of the singular value spectrum across iterations, providing a practical mechanism for guiding the reduction of the design space. The effectiveness of these strategies was demonstrated through numerical results spanning multiple design space dimensions.

The primary contribution of this work lies in the formulation of an optimization strategy that leverages both the low-rank structure of the solution and adaptive control of the design domain. The proposed heuristics, based on the behavior of the singular value spectrum, provide a practical and effective mechanism for guiding the optimization process while maintaining flexibility and stability.

The results demonstrate that the method is capable of accurately solving shape optimization problems for design spaces of varying dimensionality ($d = 1, 2, 4, 6, 10$). In lower-dimensional settings, the algorithm reliably recovers the target solution, while in higher-dimensional cases it maintains stable convergence behavior despite increased complexity. The adaptive shrink factor is shown to play a key role in balancing convergence speed and robustness, and rank adaptivity further enhances performance in problems with more complex underlying dynamics.

From a computational perspective, the proposed approach achieves accuracy comparable to, and in some cases exceeding, that of the FOM, while generally requiring less computational time throughout the optimization process. This highlights the efficiency of combining ROMs with adaptive optimization strategies.

Overall, this work demonstrates that integrating TDB-CUR with adaptive design space reduction is a viable and effective approach to PDE-constrained shape optimization. At the same time, the results emphasize the importance of carefully selecting algorithmic parameters, particularly the shrink factor and rank, as these choices directly influence the balance between efficiency, accuracy, and robustness.

6.2 Future Work

While this thesis demonstrates the effectiveness of the proposed shape optimization algorithm for the quasi-one-dimensional Euler equations, several important directions for

future research remain.

A natural extension is the application of the method to higher-dimensional problems, both in terms of physical space and parameter space. This includes more complex geometries, such as airfoil configurations like the NACA 0012, as well as more sophisticated objective functions relevant to practical engineering design. Such extensions would likely involve significantly larger design spaces than the $d = 10$ cases considered in this work, providing a more stringent test of the scalability and efficiency of the proposed approach.

Another key direction is the extension of the framework to more challenging flow regimes, particularly transonic and supersonic flows involving shock formation, allowing the algorithm to explore even larger design spaces. While the low-rank structure exploited by the TDB-CUR algorithm remains effective in subsonic regimes, the presence of shocks introduces significant challenges. In particular, variations in shock location and formation time across parameter samples can severely weaken low-rank structure, leading to a substantial increase in the required rank for accurate representation. A promising avenue for future work would be the implementation of techniques to align or track shock features across samples, thereby improving coherence in the reduced representation and enhancing computational efficiency.

Additionally, the development of mathematical guarantees represents an important theoretical direction. This includes both the optimization and ROM aspects of the framework. On the optimization side, it would be valuable to establish convergence results for the proposed iterative design space reduction strategy, particularly with respect to convergence to critical points and the dependence on ROM accuracy. From the ROM perspective, a central question is whether structural properties of the FOM are preserved under projection. In particular, the FOM used in this work is based on an entropy-stable SBP discretization. It is therefore of interest to determine whether such stability properties are inherited by the ROM constructed via the TDB-CUR projection framework. Establishing such results would provide a stronger theoretical justification for the use of structure-preserving discretizations within ROMs, and would suggest that desirable stability properties can be retained at significantly reduced computational cost.

Finally, the adaptive strategies used to control the shrink factor and rank warrant further investigation. In this work, these heuristics were based on the singular value spectrum extracted from the steady-state solution. However, this may not be optimal, as the steady-state spectrum is often relatively smooth and less informative, exhibiting reduced clustering compared to transient dynamics.

In contrast, during the time evolution of the system, there may exist intermediate time instances where the singular value spectrum is more strongly clustered or structured. Ex-

tracting adaptive information from such time instances could therefore yield different, and potentially more informative estimates of the underlying rank, leading to improved shrink factor behaviour. Exploring alternative temporal sampling strategies for constructing the singular value based indicators may therefore enhance the responsiveness and robustness of the adaptive procedure.

In addition, the mapping functions used to translate spectral metrics into shrink factor updates could be further investigated and refined. While this work introduced heuristic mappings based on tail magnitude, spectral slope, and relative spectral variation, more systematic parameter studies or alternative functional forms may lead to improved stability and performance. This includes both sensitivity analysis of the existing parameters and exploration of alternative nonlinear mappings.

Overall, these extensions would broaden the applicability of the method and further demonstrate the flexibility of the TDB-CUR algorithm within PDE-constrained optimization problems.

References

- [1] R. Abgrall, J. Nordström, P. Öffner, and S. Tokareva. Analysis of the SBP-SAT stabilization for finite element methods part I: Linear problems. *Journal of Scientific Computing*, 85(1):43, November 2020.
- [2] R. Abgrall, J. Nordström, P. Öffner, and S. Tokareva. Analysis of the SBP-SAT stabilization for Finite Element Methods part II: Entropy stability. *Commun. Appl. Math. Comput.*, 5(2):573–595, 2023.
- [3] Anshul Agarwal and Lorenz T. Biegler. A trust-region framework for constrained optimization using reduced order modeling. *Optimization and Engineering*, 14(1):3–35, 03 2013.
- [4] Mohamad Faiz Ahmad, Nor Ashidi Mat Isa, Wei Hong Lim, and Koon Meng Ang. Differential evolution: A recent review based on state-of-the-art works. *Alexandria Engineering Journal*, 61(5):3831–3872, 2022.
- [5] A. Amiri-Margavi and H. Babaei. Time-dependent low-rank input–output operator for forced linearized dynamics with unsteady base flows. *Journal of Fluid Mechanics*, 1001:A22, 2024.
- [6] John D. Anderson. Numerical solutions of quasi-one-dimensional nozzle flows. In *Computational fluid dynamics : the basics with applications*, McGraw-Hill series in mechanical engineering, page 283–373. McGraw-Hill, New York, 1995.
- [7] Peter Benner, Eric Sachs, and Stefan Volkwein. Model order reduction for PDE constrained optimization. In Günter Leugering, Peter Benner, Sebastian Engell, Arno Griewank, Helmut Harbrecht, Michael Hinze, and et al., editors, *Trends in PDE Constrained Optimization*, pages 303–326. Birkhäuser, Basel, 2014.

- [8] S. L. Brunton and J. N. Kutz. Reduced order models (ROMs). In *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*, pages 375–402. Cambridge University Press, 2019.
- [9] S. L. Brunton and J. N. Kutz. Singular value decomposition (SVD). In *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*, pages 3–46. Cambridge University Press, 2019.
- [10] Martin D. Buhmann. *Introduction*, pages 1–10. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge, 2003.
- [11] Martin D. Buhmann. *Radial basis functions on scattered data*, page 99–146. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2003.
- [12] Jonathan Byrne, Philip Cardiff, Anthony Brabazon, and Michael O’Neill. Evolving parametric aircraft models for design exploration and optimisation. *Neurocomputing*, 142:39–47, 2014.
- [13] Claudio Canuto, M. Yousuff Hussaini, Alfio Quarteroni, and Thomas A. Zang. Polynomial approximation. In *Spectral Methods*, Scientific Computation, pages 45–123. Springer, Berlin, Heidelberg, 2006.
- [14] Mark H. Carpenter, David Gottlieb, and Saul Abarbanel. Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: Methodology and application to high-order compact schemes. *Journal of Computational Physics*, 111(2):220–236, 1994.
- [15] J. Chan, K. Shukla, X. Wu, R. Liu, and P. Nalluri. High order entropy stable schemes for the quasi-one-dimensional shallow water and compressible Euler equations. *Journal of Computational Physics*, 504:112876, 2024.
- [16] Jesse Chan. On discretely entropy conservative and entropy stable discontinuous Galerkin methods. *Journal of Computational Physics*, 362:346–374, 2018.
- [17] S. Chaturantabut and D. C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010.
- [18] Mulin Cheng, Thomas Y. Hou, and Zhiwen Zhang. A dynamically bi-orthogonal method for time-dependent stochastic partial differential equations I: Derivation and algorithms. *Journal of Computational Physics*, 242:843–868, 2013.

- [19] Jared Crean, Jason E. Hicken, David C. Del Rey Fernández, David W. Zingg, and Mark H. Carpenter. Entropy-stable summation-by-parts discretization of the Euler equations on general curved elements. *Journal of Computational Physics*, 356:410–438, 2018.
- [20] D. C. Del Rey Fernández, J. E. Hicken, and D. W. Zingg. Review of summation-by-parts operators with simultaneous approximation terms for the numerical solution of partial differential equations. *Computers & Fluids*, 95:171–196, 2014.
- [21] David C. Del Rey Fernández, Pieter D. Boom, and David W. Zingg. A generalized framework for nodal first derivative summation-by-parts operators. *Journal of Computational Physics*, 266:214–239, 2014.
- [22] Nicola Demo, Giovanni Ortali, Giacomo Gustin, et al. An efficient computational framework for naval shape design and optimization problems by means of data-driven reduced order modeling techniques. *Bollettino dell’Unione Matematica Italiana*, 14:211–230, 2021.
- [23] A. Díaz-Manríquez, G. Toscano, J. H. Barron-Zambrano, and E. Tello-Leal. A review of surrogate assisted multiobjective evolutionary algorithms. *Computational Intelligence and Neuroscience*, 2016:9420460, 2016.
- [24] Peter Diener, Ernst N. Dorband, Erik Schnetter, and Manuel Tiglio. Optimized high-order derivative and dissipation operators satisfying summation by parts, and applications in three-dimensional multi-block evolutions. *Journal of Scientific Computing*, 32(1):109–145, 2007.
- [25] M. Donello, G. Palkar, M. H. Naderi, D. C. Del Rey Fernández, and H. Babaei. Oblique projection for scalable rank-adaptive reduced-order modelling of nonlinear stochastic partial differential equations with time-dependent bases. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 479(2278), 2023.
- [26] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.
- [27] Florian Feppon and Pierre F. J. Lermusiaux. A geometric approach to dynamical model order reduction. *SIAM Journal on Matrix Analysis and Applications*, 39(1):510–538, 2018.

- [28] Travis C. Fisher and Mark H. Carpenter. High-order entropy stable finite difference schemes for nonlinear conservation laws: Finite domains. *Journal of Computational Physics*, 252:518–557, 2013.
- [29] Travis Calob Fisher. *High-order L_2 stable multi-domain finite difference method for compressible flows*. PhD thesis, Purdue University, 2012.
- [30] Miguel Fosas de Pando, Peter J. Schmid, and Denis Sipp. Nonlinear model-order reduction for compressible flow solvers using the discrete empirical interpolation method. *Journal of Computational Physics*, 324:194–209, 2016.
- [31] D. Funaro and D. Gottlieb. A new method of imposing boundary conditions in pseudospectral approximations of hyperbolic equations. *Mathematics of Computation*, 51(184):599–613, 1988.
- [32] Matteo Giacomini and Pedro Díez. Surrogates for physics-based and data-driven modelling of parametric systems: Review and new perspectives. *Archives of Computational Methods in Engineering*, 2026.
- [33] Berry Thomas Gibson and Masayuki Yano. Accelerated nonlinear PDE-constrained optimization by reduced order modelling. *8th European Congress on Computational Methods in Applied Sciences and Engineering*, 2022.
- [34] S. N. Grama and S. J. Subramanian. Computation of full-field strains using principal component analysis. *Experimental Mechanics*, 54:913–933, 2014.
- [35] Amiram Harten. On the symmetric form of systems of conservation laws with entropy. *Journal of Computational Physics*, 49(1):151–164, 1983.
- [36] Chunlin He, Yong Zhang, Dunwei Gong, and Xinfang Ji. A review of surrogate-assisted evolutionary algorithms for expensive optimization problems. *Expert Systems with Applications*, 217:119495, 2023.
- [37] Z. He, X. Xiong, B. Yang, et al. Aerodynamic optimisation of a high-speed train head shape using an advanced hybrid surrogate-based nonlinear model representation method. *Optimization and Engineering*, 23:59–84, 2022.
- [38] Jason E. Hicken. Entropy-stable, high-order summation-by-parts discretizations without interface penalties. *Journal of Scientific Computing*, 82(2):50–72, February 2020.

- [39] Andreas Hildebrand and Siddhartha Mishra. Entropy stable shock capturing space–time discontinuous Galerkin schemes for systems of conservation laws. *Numerische Mathematik*, 126(1):103–151, 2014.
- [40] Michael Hinze. Discrete concepts in PDE constrained optimization. In *Optimization with PDE Constraints*, volume 23 of *Mathematical Modelling: Theory and Applications*, pages 229–270. Springer, Dordrecht, 2009.
- [41] Jiangtao Huang, Zhenghong Gao, Zhu Zhou, and Ke Zhao. An improved adaptive sampling and experiment design method for aerodynamic optimization. *Chinese Journal of Aeronautics*, 28(5):1391–1399, 2015.
- [42] Angelo Iollo, Manuel D. Salas, and Shlomo Taasan. Shape optimization governed by the Euler equations using an adjoint method. Technical report, NASA, 1993.
- [43] Rongyuan Jia, Huaijie Xia, Song Zhang, Weiguang Su, and Shuhui Xu. Optimal design of savonius wind turbine blade based on support vector regression surrogate model and modified flower pollination algorithm. *Energy Conversion and Management*, 270:116247, 2022.
- [44] Ki Sung Jung, Cristian E. Lacey, Hessam Babaei, and Jacqueline H. Chen. Accelerating high-fidelity simulations of chemically reacting flows using reduced-order modeling with time-dependent bases. *Computer methods in applied mechanics and engineering*, 437, 2025.
- [45] Michael Kartmann, Benedikt Klein, Mario Ohlberger, Thomas Schuster, and Stefan Volkwein. Adaptive reduced basis trust region methods for parabolic inverse problems. *Inverse Problems*, 41(12):125006, 2025.
- [46] O. Koch and C. Lubich. Dynamical low-rank approximation. *SIAM Journal on Matrix Analysis and Applications*, 29(2):434–454, 2007.
- [47] D. P. Kouri, M. Heinkenschloss, D. Ridzal, and B. G. van Bloemen Waanders. Inexact objective function evaluations in a trust-region algorithm for PDE-constrained optimization under uncertainty. *SIAM Journal on Scientific Computing*, 36(6):A3011–A3029, 2014.
- [48] Dmitri Kuzmin and Joshua Vedral. Dissipation-based weno stabilization of high-order finite element methods for scalar conservation laws. *J. Comput. Phys.*, 487:112153, 2023.

- [49] Ren-Cang Li. Relative perturbation theory: I. eigenvalue and singular value variations. *SIAM Journal on Matrix Analysis and Applications*, 19(4):956–982, 1998.
- [50] Yimin Lin, Jesse Chan, and Ignacio Tomas. A positivity preserving strategy for entropy stable discontinuous Galerkin discretizations of the compressible Euler and Navier-Stokes equations. *Journal of Computational Physics*, 475:111850, 2023.
- [51] Yinmin Liu, Hessam Babaee, Peyman Givi, Daniel Livescu, and Arash Nouri. Skeletal reaction models for gasoline surrogate combustion. 2025.
- [52] Sibusiso Mabuza, John N. Shadid, and Dmitri Kuzmin. Local bounds preserving stabilization for continuous Galerkin discretization of hyperbolic systems. *J. Comput. Phys.*, 361:82–110, 2018.
- [53] Nateri Madavan. On improving efficiency of differential evolution for aerodynamic shape optimization applications. In *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2004.
- [54] M. W. Mahoney and P. Drineas. CUR matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences of the United States of America*, 106(3):697–702, 2009.
- [55] Mohammad Hossein Naderi, Sara Akhavan, and Hessam Babaee. CUR for implicit time integration of random PDEs on low-rank matrix manifolds, 2024.
- [56] Mohammad Hossein Naderi and Hessam Babaee. Adaptive sparse interpolation for accelerating nonlinear stochastic reduced-order modeling with time-dependent bases. *Computer Methods in Applied Mechanics and Engineering*, 405:115813, 2023.
- [57] Grishma Palkar and Hessam Babaee. An adaptive CUR algorithm and its application to reduced-order modeling of random PDEs, 2025.
- [58] Taejun Park and Yuji Nakatsukasa. Accuracy and stability of CUR decompositions with oversampling. *SIAM Journal on Matrix Analysis and Applications*, 46(1):780–810, 2025.
- [59] P. Patil and H. Babaee. Real-time reduced-order modeling of stochastic partial differential equations via time-dependent subspaces. *Journal of Computational Physics*, 415:109511, 2020.

- [60] B. Peherstorfer, Z. Drmač, and S. Gugercin. Stability of discrete empirical interpolation and gappy proper orthogonal decomposition with randomized and deterministic sampling points. *SIAM Journal on Scientific Computing*, 42(5):A2837–A2864, 2020.
- [61] Thomas H Pulliam and David W Zingg. Governing equations. In *Fundamental algorithms in computational fluid dynamics*, volume 940, page 59–74. Springer, 2014.
- [62] Donya Ramezani, Arash G. Nouri, and Hessam Babaei. On-the-fly reduced order modeling of passive and reactive species via time-dependent manifolds. *Computer Methods in Applied Mechanics and Engineering*, 382:113882, 2021.
- [63] G. Reitberger and T. Sauer. Background subtraction using adaptive singular value decomposition. *Journal of Mathematical Imaging and Vision*, 62:1159–1172, 2020.
- [64] Chongle Ren and Zhenyu Meng. A survey on expensive optimization problems using differential evolution. *Applied Soft Computing*, 170:112727, 2025.
- [65] M. B. Russo, P. Franciosa, A. Greco, et al. Reduced-order modelling for real-time physics-based variation simulation enhanced with adaptive sampling and optimized interpolation. *The International Journal of Advanced Manufacturing Technology*, 132:3709–3734, 2024.
- [66] Rowayda A. Sadek. SVD based image processing applications: State of the art, contributions and research challenges. *CoRR*, abs/1211.7102, 2012.
- [67] Themistoklis P. Sapsis and Pierre F. J. Lermusiaux. Dynamically orthogonal field equations for continuous stochastic dynamical systems. *Physica D: Nonlinear Phenomena*, 238(23):2347–2360, 2009.
- [68] Wei Shyy, Young-Choon Cho, Weiqiang Du, et al. Surrogate-based modeling and dimension reduction techniques for multi-scale mechanics problems. *Acta Mechanica Sinica*, 27:845–865, 2011.
- [69] D. C. Sorensen and M. Embree. A DEIM induced CUR factorization. *SIAM Journal on Scientific Computing*, 38(3):A1454–A1482, 2016.
- [70] G. W. Stewart. Perturbation theory for the singular value decomposition. Technical Report UMIACS-TR-90-124, CS-TR-2539, Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, 1990.
- [71] Bo Strand. Summation by parts for finite difference approximations for d/dx . *Journal of Computational Physics*, 110(1):47–67, 1994.

- [72] Magnus Svärd and Jan Nordström. Review of summation-by-parts schemes for initial-boundary-value problems. *Journal of Computational Physics*, 268:17–38, 2014.
- [73] Magnus Svärd. Entropy-stable in- and outflow boundary conditions for the compressible euler equations. *Journal of Computational Physics*, 521:113543, 2025.
- [74] E. Tadmor. Chapter 18 - entropy stable schemes. In Rémi Abgrall and Chi-Wang Shu, editors, *Handbook of Numerical Methods for Hyperbolic Problems*, volume 17 of *Handbook of Numerical Analysis*, pages 467–493. Elsevier, 2016.
- [75] Eitan Tadmor. The numerical viscosity of entropy stable schemes for systems of conservation laws. I. *Mathematics of Computation*, 49:91–103, 1987.
- [76] Eitan Tadmor. Entropy stability theory for difference approximations of nonlinear conservation laws and related time-dependent problems. *Acta numerica*, 12:451–512, 2003.
- [77] Magnus Urquhart, Emil Ljungskog, and Simone Sebben. Surrogate-based optimisation using adaptively scaled radial basis functions. *Applied Soft Computing*, 88:106050, 2020.
- [78] René Vidal, Yi Ma, and S. Shankar Sastry. Generalized principal component analysis (GPCA). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1945–1959, 2005.
- [79] Handing Wang, John Doherty, and Yaochu Jin. Hierarchical surrogate-assisted evolutionary multi-scenario airfoil shape optimization. In *Proceedings of the 2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2018.
- [80] Tianshu Wen. *Adaptive Model Hyperreduction to Accelerate Optimization Problems Governed by Partial Differential Equations*. PhD thesis, University of Notre Dame, 2024.
- [81] Zelalem Arega Worku and David W. Zingg. Simultaneous approximation terms and functional accuracy for diffusion problems discretized with multidimensional summation-by-parts operators. *Journal of computational physics*, 445:110634–, 2021.
- [82] Masayuki Yano, Tianci Huang, and Matthew J. Zahr. A globally convergent method to accelerate topology optimization using on-the-fly model reduction. *Computer Methods in Applied Mechanics and Engineering*, 375:113635, March 2021.

- [83] Matthew J. Zahr. *Adaptive Model Reduction to Accelerate Optimization Problems Governed by Partial Differential Equations*. PhD thesis, Stanford University Institute for Computational and Mathematical Engineering, 2016.
- [84] Xinshuai Zhang, Fangfang Xie, Tingwei Ji, Zaoxu Zhu, and Yao Zheng. Multi-fidelity deep neural network surrogate model for aerodynamic shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 373:113485, 2021.
- [85] Ciyu Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.
- [86] R. Zimmermann and K. Willcox. An accelerated greedy missing point estimation procedure. *SIAM Journal on Scientific Computing*, 38(5):A2827–A2850, 2016.

APPENDICES

Appendix A

Application of TDB-CUR to the Quasi-1D Euler Equations

The TDB-CUR algorithm, along with its core components and several methodological nuances, was examined in detail in Chapter 3. As previously noted, the framework is highly robust and can be adapted to meet the requirements of specific problems or applications. For example, it has been extended to incorporate implicit time integrators, including applications in sensitivity analysis of skeletal reaction models for gas combustion [51, 55]. It has also been applied to uncertainty quantification in chemically reacting flows, where the algorithm is augmented to approximate nonlinear transport and chemical source terms [44]. These examples suggest that, when applying the TDB-CUR algorithm to the quasi-1D compressible Euler equations, slight modifications to both the FOM and the TDB-CUR framework are to be expected.

In this appendix, we detail the modifications required to couple the TDB-CUR algorithm with a SBP discretization of the quasi-1D compressible Euler equations. Specifically, we describe the nondimensionalization of the governing equations, the structure of the solution vector and associated differentiable operators, and key implementation details of the SBP discretization, including indexing conventions and the computation of the corresponding RHS used in this thesis.

A.1 Nondimensionalization of the Quasi-1D Compressible Euler Equations

ROMs at their core, are based on the observation that complex systems often evolve on low-dimensional structures [8]. Many ROM techniques, including TDB-CUR, rely on the SVD, or approximations thereof, to extract these structures. As a result, it is standard practice to normalize variables prior to applying SVD based methods to ensure that all components contribute comparably and to prevent variables with larger magnitudes from dominating the resulting modes.

This consideration is particularly important for the quasi-1D compressible Euler equations, where the primitive variables (ρ, u, p) may differ significantly in scale. Without appropriate normalization, this disparity can degrade the quality of the extracted low-dimensional subspaces and may lead to nonphysical behavior, such as negative density or pressure.

To address this issue, we nondimensionalize the governing equations following [6, 30]. Let:

$$c_0 := \sqrt{\frac{\gamma p_0}{\rho_0}}, \quad a_0 := a(x_0),$$

denote a reference speed of sound and a reference area, respectively. We define nondimensional variables as:

$$x' := \frac{x}{\sqrt{a_0}}, \quad t' := \frac{tc_0}{\sqrt{a_0}},$$

and

$$\rho' := \frac{\rho}{\rho_0}, \quad u' := \frac{u}{c_0}, \quad p' := \frac{p}{p_0}, \quad a' := \frac{a}{a_0}.$$

In what follows, primes are omitted in the final equations for clarity.

Mass conservation

Starting from the dimensional equation:

$$\frac{\partial(a\rho)}{\partial t} + \frac{\partial(a\rho u)}{\partial x} = 0,$$

we substitute the nondimensional variables:

$$\begin{aligned} a\rho &= a_0 a' \cdot \rho_0 \rho' = a_0 \rho_0 (a' \rho'), \\ a\rho u &= a_0 a' \cdot \rho_0 \rho' \cdot c_0 u' = a_0 \rho_0 c_0 (a' \rho' u'). \end{aligned}$$

Applying the chain rule:

$$\frac{\partial}{\partial t} = \frac{c_0}{\sqrt{a_0}} \frac{\partial}{\partial t'}, \quad \frac{\partial}{\partial x} = \frac{1}{\sqrt{a_0}} \frac{\partial}{\partial x'},$$

we obtain:

$$\frac{a_0 \rho_0 c_0}{\sqrt{a_0}} \frac{\partial(a' \rho')}{\partial t'} + \frac{a_0 \rho_0 c_0}{\sqrt{a_0}} \frac{\partial(a' \rho' u')}{\partial x'} = 0.$$

Dividing through by the common factor $\frac{a_0 \rho_0 c_0}{\sqrt{a_0}}$:

$$\frac{\partial(a\rho)}{\partial t} + \frac{\partial(a\rho u)}{\partial x} = 0.$$

Momentum conservation

Starting from:

$$\frac{\partial(a\rho u)}{\partial t} + \frac{\partial(a\rho u^2)}{\partial x} + a \frac{\partial p}{\partial x} = 0,$$

we substitute:

$$\begin{aligned} a\rho u &= a_0 \rho_0 c_0 (a' \rho' u'), \\ a\rho u^2 &= a_0 \rho_0 c_0^2 (a' \rho' (u')^2), \\ p &= p_0 p', \quad a = a_0 a'. \end{aligned}$$

Applying the chain rule:

$$\frac{a_0 \rho_0 c_0^2}{\sqrt{a_0}} \frac{\partial(a' \rho' u')}{\partial t'} + \frac{a_0 \rho_0 c_0^2}{\sqrt{a_0}} \frac{\partial(a' \rho' (u')^2)}{\partial x'} + \frac{a_0 p_0}{\sqrt{a_0}} a' \frac{\partial p'}{\partial x'} = 0.$$

Using $c_0^2 = \frac{\gamma p_0}{\rho_0}$, we rewrite the equation so that all terms share a common factor:

$$\frac{a_0 \rho_0 c_0^2}{\sqrt{a_0}} \left[\frac{\partial(a' \rho' u')}{\partial t'} + \frac{\partial(a' \rho' (u')^2)}{\partial x'} + \frac{1}{\gamma} a' \frac{\partial p'}{\partial x'} \right] = 0.$$

Dividing through by the common factor $\frac{a_0 \rho_0 c_0^2}{\sqrt{a_0}}$:

$$\frac{\partial(a' \rho' u')}{\partial t'} + \frac{\partial(a' \rho' (u')^2)}{\partial x'} + \frac{1}{\gamma} a' \frac{\partial p'}{\partial x'} = 0.$$

Applying the product rule to the pressure term:

$$a' \frac{\partial p'}{\partial x'} = \frac{\partial(a' p')}{\partial x'} - p' \frac{\partial a'}{\partial x'},$$

the momentum equation becomes:

$$\frac{\partial(a' \rho' u')}{\partial t'} + \frac{\partial(a' \rho' (u')^2)}{\partial x'} + \frac{1}{\gamma} \frac{\partial(a' p')}{\partial x'} - \frac{1}{\gamma} p' \frac{\partial a'}{\partial x'} = 0.$$

Combining the conservative flux terms yields:

$$\frac{\partial(a \rho u)}{\partial t} + \frac{\partial}{\partial x} \left(a \left(\rho u^2 + \frac{1}{\gamma} p \right) \right) - \frac{1}{\gamma} p \frac{\partial a}{\partial x} = 0.$$

Energy conservation

Starting from:

$$\frac{\partial(aE)}{\partial t} + \frac{\partial}{\partial x} (au(E + p)) = 0,$$

we define the total energy per unit volume as:

$$E := \frac{p}{\gamma - 1} + \frac{1}{2} \rho u^2.$$

Substituting nondimensional variables:

$$\begin{aligned} E &= \frac{p_0 p'}{\gamma - 1} + \frac{1}{2} \rho_0 \rho' (c_0 u')^2 \\ &= p_0 \left[\frac{p'}{\gamma - 1} + \frac{\gamma}{2} \rho' (u')^2 \right], \end{aligned}$$

where we used:

$$c_0^2 = \frac{\gamma p_0}{\rho_0}.$$

Thus, $E_0 = p_0$, and $E' = E/p_0$. Substituting into the governing equation:

$$\frac{a_0 p_0 c_0}{\sqrt{a_0}} \frac{\partial(a' E')}{\partial t'} + \frac{a_0 p_0 c_0}{\sqrt{a_0}} \frac{\partial}{\partial x'} (a' u' (E' + p')) = 0.$$

Dividing through by the common factor $\frac{a_0 p_0 c_0}{\sqrt{a_0}}$:

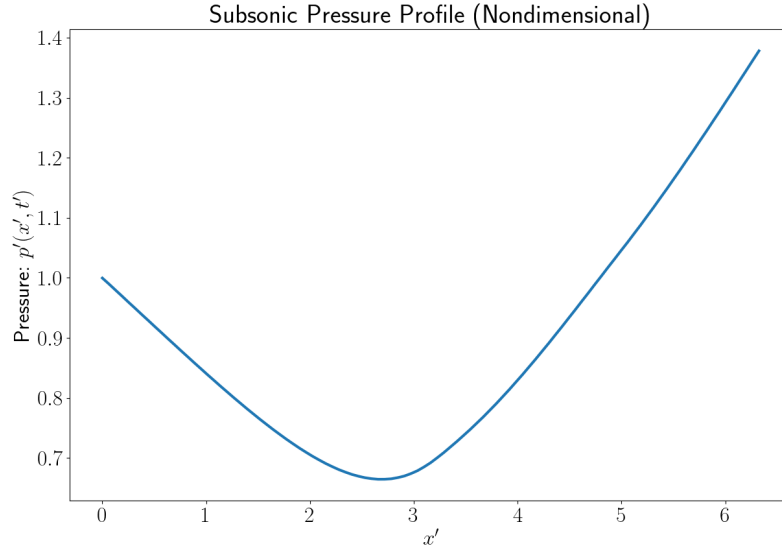
$$\frac{\partial(aE)}{\partial t} + \frac{\partial}{\partial x} (au(E + p)) = 0.$$

Collecting all equations, the nondimensional quasi-1D compressible Euler system becomes:

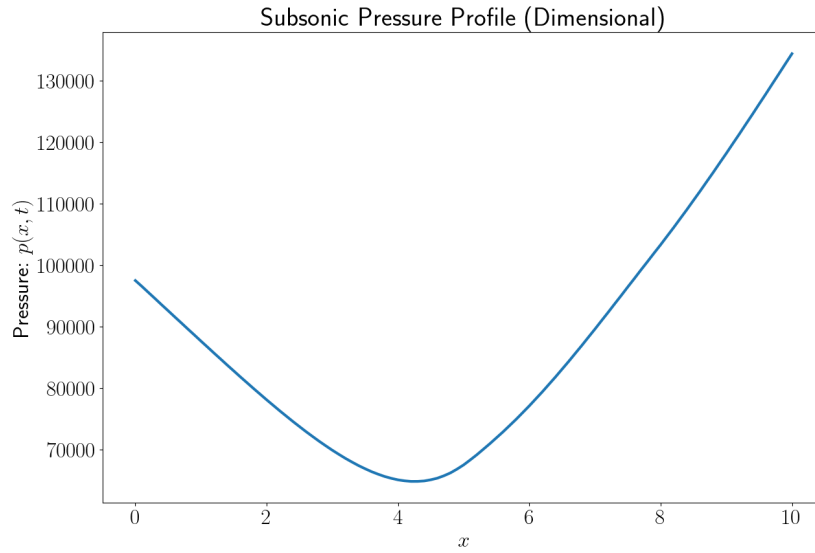
$$\frac{\partial}{\partial t} \begin{bmatrix} a\rho \\ a\rho u \\ aE \end{bmatrix} + \frac{\partial}{\partial x} \begin{bmatrix} a\rho u \\ a \left(\rho u^2 + \frac{1}{\gamma} p \right) \\ au(E + p) \end{bmatrix} - \frac{1}{\gamma} \frac{\partial a}{\partial x} \begin{bmatrix} 0 \\ p \\ 0 \end{bmatrix} = 0.$$

This system retains the same structure as the dimensional formulation, differing only by nondimensional scaling and the appearance of the factor $\frac{1}{\gamma}$ in the geometric source term. The transformation preserves the governing dynamics while improving numerical conditioning by bringing all variables to comparable magnitudes.

For example, in Figures [A.1](#) and [A.2](#), the pressure profiles obtained from the dimensional and nondimensional systems are identical in shape, even in the presence of shocks. This confirms that the transformation preserves the dominant dynamics of the system, which is essential for constructing accurate ROMs using TDB-CUR.

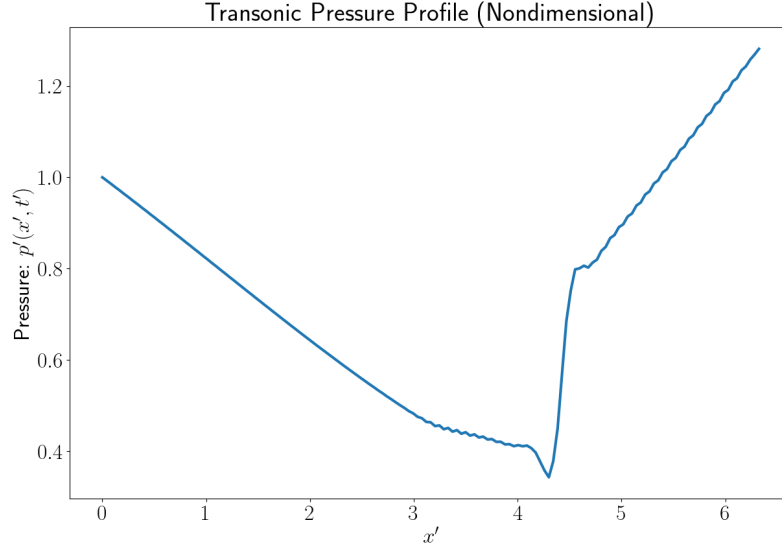


(a) Nondimensional pressure profile.

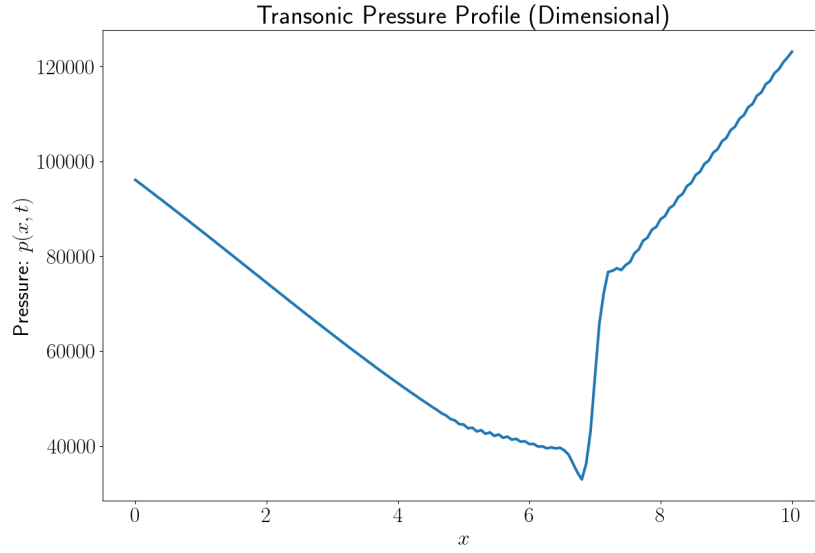


(b) Dimensional pressure profile.

Figure A.1: Nondimensional and dimensional pressure profiles for the subsonic nozzle problem from [61], computed to steady state ($t_F = 3.0$). The solution is obtained using the spatial discretization in (2.9) with $k = 50$ elements and polynomial degree $q = 2$, and advanced in time using an RK4 scheme with time step $\Delta t = 5.0 \times 10^{-5}$.



(a) Nondimensional pressure profile.



(b) Dimensional pressure profile.

Figure A.2: Nondimensional and dimensional pressure profiles for the transonic nozzle problem from [61], computed to steady state ($t_F = 5.0$). The solution is obtained using the spatial discretization in (2.9) with $k = 75$ elements and polynomial degree $q = 2$, and advanced in time using an RK4 scheme with time step $\Delta t = 5.0 \times 10^{-5}$.

A.2 Solution and SBP Operator Form for the Quasi-1D Compressible Euler Equations

An important consideration in applying the TDB-CUR algorithm is the representation of the solution matrix. The chosen data layout directly impacts the indexing of the state variables, as well as the structure of the discrete differential operators. For a one-dimensional discretization of the quasi-1D Euler equations, a common representation of the solution is:

$$\mathbf{V} = \begin{bmatrix} (a\rho)_1 & (a\rho u)_1 & (aE)_1 \\ (a\rho)_2 & (a\rho u)_2 & (aE)_2 \\ \vdots & \vdots & \vdots \\ (a\rho)_{N_x} & (a\rho u)_{N_x} & (aE)_{N_x} \end{bmatrix}, \quad (\text{A.1})$$

so that $\mathbf{V} \in \mathbb{R}^{N_x \times 3}$ for a single sample, i.e., $N_s = 1$.

In some implementations, it is convenient to augment the solution matrix with the geometric term $a(x)$ to simplify the extraction of primitive variables. This leads to the alternative representation:

$$\mathbf{V} = \begin{bmatrix} (a\rho)_1 & (a\rho u)_1 & (aE)_1 & (a)_1 \\ (a\rho)_2 & (a\rho u)_2 & (aE)_2 & (a)_2 \\ \vdots & \vdots & \vdots & \vdots \\ (a\rho)_{N_x} & (a\rho u)_{N_x} & (aE)_{N_x} & (a)_{N_x} \end{bmatrix},$$

with $\mathbf{V} \in \mathbb{R}^{N_x \times 4}$.

However, this augmented representation is not well-suited for use within the TDB-CUR framework. The inclusion of the geometric term $a(x)$, which is not a dynamic state variable, can adversely affect the construction of low-rank bases and the associated point selection procedures. In particular, the resulting decomposition may end up representing the geometry more strongly than the evolving flow dynamics it is meant to capture. To ensure that the ROM captures only the relevant system dynamics, we therefore adopt the representation in (A.1), excluding $a(x)$ from the solution matrix.

Now consider the TDB-CUR setting in which we collect N_s samples over the parameter space. In this case, the solution is represented as a third-order tensor:

$$\mathbf{V} \in \mathbb{R}^{N_x \times 3 \times N_s},$$

where the first mode corresponds to the spatial index, the second mode corresponds to the conserved variable, and the third mode corresponds to the parameter sample. Using MATLAB-style slicing notation, the conserved-variable slices are given by:

$$\begin{aligned} \mathbf{V}(:, 1, :) &= \begin{bmatrix} (a\rho)_{1,1} & \cdots & (a\rho)_{1,N_s} \\ \vdots & \ddots & \vdots \\ (a\rho)_{N_x,1} & \cdots & (a\rho)_{N_x,N_s} \end{bmatrix}, \\ \mathbf{V}(:, 2, :) &= \begin{bmatrix} (a\rho u)_{1,1} & \cdots & (a\rho u)_{1,N_s} \\ \vdots & \ddots & \vdots \\ (a\rho u)_{N_x,1} & \cdots & (a\rho u)_{N_x,N_s} \end{bmatrix}, \\ \mathbf{V}(:, 3, :) &= \begin{bmatrix} (aE)_{1,1} & \cdots & (aE)_{1,N_s} \\ \vdots & \ddots & \vdots \\ (aE)_{N_x,1} & \cdots & (aE)_{N_x,N_s} \end{bmatrix}. \end{aligned}$$

Here, in $(\cdot)_{i,j}$, the first index denotes the spatial node $i \in \{1, \dots, N_x\}$, while the second index denotes the parameter sample $j \in \{1, \dots, N_s\}$. The conserved variable is determined implicitly by the slice $\mathbf{V}(:, k, :)$, where $k = 1, 2, 3$ corresponds to $a\rho$, $a\rho u$, and aE , respectively.

However, this format is not directly compatible with the standard matrix-based formulation used in the TDB-CUR algorithm. In particular, TDB-CUR requires a two-dimensional solution matrix of size $\mathbb{R}^{3N_x \times N_s}$. To obtain this representation, the tensor must be reshaped. There are multiple equivalent ways to construct this matrix, depending on whether one groups by spatial location or by conserved variable.

Option 1: Stacking by spatial location

We first group all conserved variables at each spatial node, yielding:

$$\mathbf{V} := \begin{bmatrix} \begin{pmatrix} a\rho \\ a\rho u \\ aE \end{pmatrix}_1^1 & \begin{pmatrix} a\rho \\ a\rho u \\ aE \end{pmatrix}_1^2 & \cdots & \begin{pmatrix} a\rho \\ a\rho u \\ aE \end{pmatrix}_1^{N_s} \\ \vdots & \vdots & \ddots & \vdots \\ \begin{pmatrix} a\rho \\ a\rho u \\ aE \end{pmatrix}_i^1 & \begin{pmatrix} a\rho \\ a\rho u \\ aE \end{pmatrix}_i^2 & \cdots & \begin{pmatrix} a\rho \\ a\rho u \\ aE \end{pmatrix}_i^{N_s} \\ \vdots & \vdots & \ddots & \vdots \\ \begin{pmatrix} a\rho \\ a\rho u \\ aE \end{pmatrix}_{N_x}^1 & \begin{pmatrix} a\rho \\ a\rho u \\ aE \end{pmatrix}_{N_x}^2 & \cdots & \begin{pmatrix} a\rho \\ a\rho u \\ aE \end{pmatrix}_{N_x}^{N_s} \end{bmatrix}, \quad (\text{A.2})$$

where $(\cdot)_i^j$ denotes evaluation at spatial node $i \in \{1, \dots, N_x\}$ and parameter sample $j \in \{1, \dots, N_s\}$.

Option 2: Stacking by conserved variable

Alternatively, we may group by conserved variable, stacking full spatial fields for each sample:

$$\mathbf{V} := \begin{bmatrix} \begin{pmatrix} (a\rho)_1^1 \\ \vdots \\ (a\rho)_{N_x}^1 \end{pmatrix} & \begin{pmatrix} (a\rho)_1^2 \\ \vdots \\ (a\rho)_{N_x}^2 \end{pmatrix} & \cdots & \begin{pmatrix} (a\rho)_1^{N_s} \\ \vdots \\ (a\rho)_{N_x}^{N_s} \end{pmatrix} \\ \begin{pmatrix} (a\rho u)_1^1 \\ \vdots \\ (a\rho u)_{N_x}^1 \end{pmatrix} & \begin{pmatrix} (a\rho u)_1^2 \\ \vdots \\ (a\rho u)_{N_x}^2 \end{pmatrix} & \cdots & \begin{pmatrix} (a\rho u)_1^{N_s} \\ \vdots \\ (a\rho u)_{N_x}^{N_s} \end{pmatrix} \\ \begin{pmatrix} (aE)_1^1 \\ \vdots \\ (aE)_{N_x}^1 \end{pmatrix} & \begin{pmatrix} (aE)_1^2 \\ \vdots \\ (aE)_{N_x}^2 \end{pmatrix} & \cdots & \begin{pmatrix} (aE)_1^{N_s} \\ \vdots \\ (aE)_{N_x}^{N_s} \end{pmatrix} \end{bmatrix}, \quad (\text{A.3})$$

where $(\cdot)_i^j$ denotes evaluation at spatial node $i \in \{1, \dots, N_x\}$ and parameter sample $j \in \{1, \dots, N_s\}$.

Both representations are mathematically equivalent, however, they induce different memory layouts and affect how indexing and operator application are implemented in practice. However, representing the solution in either of the forms introduced above requires a corresponding reformulation of the SBP differentiation operators. In particular, since the solution is now stored as a matrix in $\mathbb{R}^{3N_x \times N_s}$ rather than $\mathbb{R}^{N_x \times N_s}$, the spatial derivative operators must act consistently across the coupled system of conserved variables.

To construct these operators, we use the Kronecker product. For matrices $\mathbf{A} \in \mathbb{R}^{c \times d}$ and $\mathbf{B} \in \mathbb{R}^{y \times z}$, the Kronecker product $\mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{cy \times dz}$ is defined as:

$$\mathbf{A} \otimes \mathbf{B} := \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1d}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2d}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{c1}\mathbf{B} & a_{c2}\mathbf{B} & \cdots & a_{cd}\mathbf{B} \end{bmatrix},$$

where a_{ij} denotes the (i, j) entry of $\mathbf{A}$. Using this definition, we construct the global SBP differentiation operator in two equivalent forms, depending on how the solution is organized.

Global operator for node-wise stacking (Equation A.2)

For the ordering in which all conserved variables at each spatial node are grouped together, the derivative operator is:

$$\mathbf{D}_{x_g} := \mathbf{D}_x \otimes \mathbf{I}_3 = \begin{bmatrix} d_{11}\mathbf{I}_3 & d_{12}\mathbf{I}_3 & \cdots & d_{1N_x}\mathbf{I}_3 \\ d_{21}\mathbf{I}_3 & d_{22}\mathbf{I}_3 & \cdots & d_{2N_x}\mathbf{I}_3 \\ \vdots & \vdots & \ddots & \vdots \\ d_{N_x1}\mathbf{I}_3 & d_{N_x2}\mathbf{I}_3 & \cdots & d_{N_xN_x}\mathbf{I}_3 \end{bmatrix}, \quad (\text{A.4})$$

where $\mathbf{D}_{x_g} \in \mathbb{R}^{3N_x \times 3N_x}$, $\mathbf{D}_x \in \mathbb{R}^{N_x \times N_x}$ is the SBP differentiation matrix, $\mathbf{I}_3 \in \mathbb{R}^{3 \times 3}$ is the identity matrix corresponding to the three conserved variables, and d_{ij} denotes the (i, j) entry of $\mathbf{D}_x$.

Global operator for variable-wise stacking (Equation A.3)

For the alternative ordering in which each conserved variable is stored as a full spatial field, the operator takes the form:

$$\mathbf{D}_{x_g} := \mathbf{I}_3 \otimes \mathbf{D}_x = \begin{bmatrix} \mathbf{D}_x & 0 & 0 \\ 0 & \mathbf{D}_x & 0 \\ 0 & 0 & \mathbf{D}_x \end{bmatrix}. \quad (\text{A.5})$$

Both formulations are mathematically equivalent and correspond to different permutations of the same underlying coupled system. However, for the implementation used throughout this thesis, we adopt the node-wise ordering in Equation A.2, as it aligns naturally with the structure of the TDB-CUR matrices and simplifies the construction of the ROM.

A.3 Indexing in TDB-CUR

Having established the structure of the solution and SBP operators in the TDB-CUR framework, we now address the practical implementation of indexing and sampling. A central component of the algorithm is the computation of sampling indices $\mathbf{p}$ and $\mathbf{s}$ using sparse sampling techniques such as DEIM [17] or GappyPOD+E [60]. These indices determine which rows and columns of the full solution are retained for constructing the ROM.

The selection of column indices $\mathbf{s}$ is relatively straightforward, as it primarily requires consistent alignment with the parameter samples $\boldsymbol{\theta}$, particularly when evaluating volume or boundary contributions. Specifically, in this case we may revert to the solution representation $\mathbf{V} \in \mathbb{R}^{N_x \times 3 \times |\mathbf{s}|}$, where $|\mathbf{s}|$ denotes the length of the array, which allows the original derivative operators to be used directly without constructing the global derivative operators $\mathbf{D}_{x_g}$. In contrast, the selection and interpretation of row indices $\mathbf{p}$ requires more care, as it directly affects the evaluation of spatial derivatives and fluxes.

To illustrate this, consider first the simpler case of the inviscid Burgers' equation discretized using CG operators. In that setting, adjacent indices can be identified via:

$$\mathbf{p}_a = \text{nonzero}(\mathbf{D}_x(\mathbf{p}, :)), \quad (\text{A.6})$$

where $\mathbf{p}_a$ denotes the set of adjacent grid points required for spatial differencing. Since Burgers' equation involves a single conserved variable, this construction is sufficient for evaluating the RHS.

However, for the quasi-1D Euler equations, the situation is more involved due to the coupled system of conserved variables $[a\rho, a\rho u, aE]^T$. In this case, the structure of the solution vector and SBP operators becomes critical, since the two-point flux evaluations require access to all conserved variables at a given spatial node. Under the node-wise ordering introduced in Equation (A.2), each physical grid point corresponds to a block of three consecutive entries in the global solution vector.

A naïve extension of (A.6) to the global operator:

$$\mathbf{p}_a = \text{nonzero}(\mathbf{D}_{x_g}(\mathbf{p}, :)),$$

correctly identifies nonzero coupling structure at the level of individual variables but does not guarantee that all conserved variables associated with a physical node are included. As a result, the required data for flux evaluation may be incomplete. To make this issue concrete, consider the centered difference operator, with $h = 1$:

$$\mathbf{D}_x = \frac{1}{2} \begin{bmatrix} -1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & -1 & 1 \end{bmatrix},$$

which induces the global operator:

$$\mathbf{D}_{x_g} = \mathbf{D}_x \otimes \mathbf{I}_3 = \begin{bmatrix} -\mathbf{I}_3 & \mathbf{I}_3 & 0 \\ \mathbf{I}_3 & 0 & \mathbf{I}_3 \\ 0 & -\mathbf{I}_3 & \mathbf{I}_3 \end{bmatrix}.$$

Now suppose that the selected row indices are $\mathbf{p} = [1, 4]$. Applying the naïve adjacency rule gives:

$$\mathbf{D}_{x_g}(\mathbf{p}, :) = \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix},$$

from which:

$$\mathbf{p}_a = \text{nonzero}(\mathbf{D}_{x_g}(\mathbf{p}, :)) = [1, 4, 7],$$

where $\text{nonzero}(\cdot)$ returns the indices of the nonzero columns.

While this correctly identifies coupling at the variable level, it does not reflect the structure required for flux evaluations. Specifically, consider the computation of the entropy

conservative two-point flux from (2.10) at the first and fourth rows of the solution matrix $\mathbf{V}$:

$$\mathbf{F}_{14} = \mathbf{f}_{EC} \left(\hat{\mathbf{V}}([1, 2, 3], :), \hat{\mathbf{V}}([4, 5, 6], :) \right).$$

This requires access to all conserved variables associated with each node. However, the reduced index set $\mathbf{p} = [1, 4]$ together with $\mathbf{p}_a = [1, 4, 7]$ does not contain the full block structure needed to evaluate this flux consistently. In other words, they omit the additional indices $\{2, 3, 5, 6\}$ required to assemble the complete nodal states needed to evaluate the flux.

The root of the issue is that entropy conservative fluxes depend on the full vector of conserved variables at each node, and therefore cannot be evaluated on partially sampled variable subsets. Consequently, a more structured indexing strategy is required to ensure that, whenever a spatial node is selected, all associated conserved variables are included in the evaluation of the RHS.

One straightforward way to resolve this issue is to precompute an adjacency structure that stores all indices required for evaluating spatial derivatives and fluxes for each DOF in the global solution matrix $\hat{\mathbf{V}}$. We therefore define an adjacency matrix $\mathbf{A} \in \mathbb{R}^{3N_x \times b}$ of the form:

$$\mathbf{A} := \begin{bmatrix} \cdots & \mathbf{p}_{a_1} & \cdots \\ \cdots & \mathbf{p}_{a_1} & \cdots \\ \cdots & \mathbf{p}_{a_1} & \cdots \\ \cdots & \mathbf{p}_{a_2} & \cdots \\ \cdots & \mathbf{p}_{a_2} & \cdots \\ \cdots & \mathbf{p}_{a_2} & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & \mathbf{p}_{a_{3N_x}} & \cdots \\ \cdots & \mathbf{p}_{a_{3N_x}} & \cdots \\ \cdots & \mathbf{p}_{a_{3N_x}} & \cdots \end{bmatrix}$$

Here, b is chosen sufficiently large to accommodate the maximum adjacency length across all rows. The structure reflects the fact that each physical node contributes three conserved variables, and therefore each node's adjacency pattern is replicated across the corresponding block of three rows in $\hat{\mathbf{V}}$.

Each row of $\mathbf{A}$ stores the indices required to evaluate the RHS at the corresponding DOF. In particular, the adjacency sets $\mathbf{p}_{a_i}$ include both neighboring spatial nodes and all

associated conserved variables required for flux evaluation. For example:

$$\begin{aligned} \mathbf{p}_{a_1} &= [1, 2, 3, \dots], \\ \mathbf{p}_{a_2} &= [4, 5, 6, \dots], \\ &\vdots \\ \mathbf{p}_{a_{3N_x}} &= [3N_x - 2, 3N_x - 1, 3N_x, \dots]. \end{aligned}$$

Once the adjacency matrix is constructed, it can be used efficiently within the time-integration loop to retrieve the required neighboring indices. Given a set of selected rows $\mathbf{p}$, the corresponding adjacency set is obtained via:

$$\mathbf{p}_a = \mathbf{A}(\mathbf{p}, :). \quad (\text{A.7})$$

We now summarize the construction and usage of this structure in Algorithm 8 and Algorithm 9, respectively.

Algorithm 8 Construction of the Global Adjacency Matrix, $\mathbf{A}$

Input: $\mathbf{D}_x$	▷ Local derivative operator
Output: $\mathbf{A}$	▷ Global adjacency structure
$\mathbf{A} \leftarrow -1$	▷ Initialize with placeholder value
for $i = 1, \dots, N_x$ do	
$\mathbf{p}_{a_i} \leftarrow \text{nonzero}(\mathbf{D}_x(i, :))$	▷ Node-level adjacency
$\mathbf{p}_{a_g} \leftarrow \emptyset$	▷ Initialize global adjacency list
for each $j \in \mathbf{p}_{a_i}$ do	
$\text{append}(\mathbf{p}_{a_g}, [3j - 2, 3j - 1, 3j])$	▷ Lift node j to DOF block
end for	
$\text{append}(\mathbf{p}_{a_g}, [3i - 2, 3i - 1, 3i])$	▷ Include self block for node i
$\mathbf{A}(3i - 2 : 3i, :) \leftarrow \mathbf{p}_{a_g}$	▷ Assign adjacency block for node i
end for	

Algorithm 9 Retrieval of Adjacency Indices $\mathbf{p}_a$

Input: $\mathbf{A}, \mathbf{p}$

Output: $\mathbf{p}_a$

```

 $\mathbf{p}_a \leftarrow \emptyset$  ▷ Initialize adjacency list
for  $k = 1, \dots, \text{length}(\mathbf{p})$  do
     $\text{append}(\mathbf{p}_a, \mathbf{A}(\mathbf{p}(k), :))$  ▷ Collect adjacency entries from selected DOFs
end for
 $\mathbf{p}_a \leftarrow \text{unique}(\mathbf{p}_a)$  ▷ Remove duplicates
Remove all entries equal to  $-1$  ▷ Discard placeholder values

```

The above construction assumes a system with three conserved variables and the node-wise ordering defined in Equation (A.2). In practice, the adjacency matrix $\mathbf{A}$ is initialized with a placeholder value, i.e., -1 , and allocated with dimension $\mathbf{A} \in \mathbb{R}^{3N_x \times 3N_x}$, which guarantees sufficient capacity for all adjacency entries. The function `unique` removes duplicate indices and enforces ordering, while invalid entries introduced during initialization are removed after the fact.

It is important to note that this procedure is one of several possible ways to construct adjacency information and is not necessarily the most computationally efficient. However, it is conceptually straightforward, making it suitable for integration within the TDB-CUR framework.

An alternative approach for constructing the adjacency structure is based on a sensitivity analysis of the RHS operator. Since adjacency is fundamentally induced by the coupling structure of the spatial discretization, it can be inferred directly from the dependence of the RHS on the solution variables.

Given a solution $\mathbf{V}$ at a fixed time t , we introduce a small perturbation of magnitude h and compute a FD approximation of the Jacobian of the RHS:

$$\mathbf{J} := \frac{\text{RHS}(t, \mathbf{V} + \mathbf{P}) - \text{RHS}(t, \mathbf{V})}{h}, \quad (\text{A.8})$$

where $\mathbf{P}$ denotes a perturbation matrix with a single nonzero entry. Nonzero entries of $\mathbf{J}$ indicate dependence between DOFs, and thus define adjacency relationships. Specifically, entry (i, j) is nonzero if the i^{th} component of the RHS depends on the j^{th} DOF of the solution. This procedure is summarized in Algorithm 10, where in the following $\mathbb{I}(\cdot)$ denotes the indicator function, which equals 1 if its argument is satisfied and 0 otherwise.

Algorithm 10 Adjacency Construction via FD Sensitivity

Input: V, h **Output:** A

```
J ← 0 ▷ Initialize sensitivity matrix
F0 ← RHS( $t, V$ ) ▷ Unperturbed residual evaluation
for  $i = 1, \dots, N_x$  do
  for  $j = 1, 2, 3$  do
    P ← 0 (same size as V) ▷ Initialize perturbation matrix
    P( $i, j$ ) ←  $h$  ▷ Perturb DOF ( $i, j$ )
    F1 ← RHS( $t, V + P$ ) ▷ Perturbed residual evaluation
    J(:, 3( $i - 1$ ) +  $j$ ) ← (F1 - F0)/ $h$  ▷ FD sensitivity column
  end for
end for
A ←  $\mathbb{I}(|J| > 0)$  ▷ Convert sensitivity to adjacency (nonzero structure)
```

Compared to the stencil-based construction in Algorithm 8, this approach is a black-box method that requires only evaluations of the RHS operator, without explicit knowledge of the underlying discretization or derivative stencils. However, this comes at a significantly increased computational cost due to the repeated RHS evaluations.

As in the previous case, the adjacency matrix can be precomputed and reused throughout the TDB-CUR time integration. The adjacent indices corresponding to a selected set of rows $\mathbf{p}$ are then obtained via:

$$\mathbf{p}_a = \text{nonzero}(A(\mathbf{p}, :)).$$

A.4 Computing the RHS

Now that we have established how the solution is stored and how adjacency is computed for this representation, we turn to how these ideas are incorporated into the algorithm. In particular, one of the more intricate components of the TDB-CUR framework is the evaluation of the temporal discretization, and specifically the RHS contributions at the selected rows $\mathbf{p}$ and columns $\mathbf{s}$.

We begin by considering only the volume contributions, temporarily neglecting boundary terms. Thus, we focus on the volume term from the strong form of the discretization

in (2.9):

$$\frac{d\mathbf{u}}{dt} = -2(\mathbf{D}_x \circ \mathbf{F})\mathbf{1}. \quad (\text{A.9})$$

For a FOM implementation, the RHS could be computed directly by forming the two-point flux matrix and multiplying by the vector of ones, as expressed in (A.9). However, this approach is not ideal for the TDB-CUR framework, where efficient pointwise access to RHS evaluations is required. Instead, we exploit the structure of the vector of ones to reformulate the expression in a pointwise manner. Specifically, the RHS can be written component-wise as:

$$\left(\frac{d\mathbf{u}}{dt}\right)_i = -2 \sum_{j=1}^{N_x} (\mathbf{D}_x)_{ij} \mathbf{F}_{ij}, \quad 1 \leq i \leq N_x \quad (\text{A.10})$$

$$= -2 \sum_{j=1}^{N_x} (\mathbf{D}_x)_{ij} \mathbf{f}_{EC}(\mathbf{u}(i,:), \mathbf{u}(j,:)), \quad 1 \leq i \leq N_x. \quad (\text{A.11})$$

This formulation is significantly more suitable for the TDB-CUR setting, as it allows the RHS to be evaluated in a pointwise manner. Based on this representation, we can implement a function to compute the RHS, as outlined in Algorithm 11.

Algorithm 11 Volume RHS evaluation for quasi-1D Euler equations

Input: $\mathbf{u}, \mathbf{D}_x$

Output: $d\mathbf{u}$

```

( $\mathbf{r}, \mathbf{c}$ )  $\leftarrow$  sparse( $\mathbf{D}_x$ ) ▷ Nonzero index pairs of  $\mathbf{D}_x$ 
 $d\mathbf{u} \leftarrow \mathbf{0}$  ▷ Initialize RHS
for  $(i, j) \in (\mathbf{r}, \mathbf{c})$  do ▷ Loop over sparse indices
     $d\mathbf{u}(i, :) += \mathbf{D}_x(i, j) \mathbf{f}_{EC}(\mathbf{u}(i,:), \mathbf{u}(j,:))$  ▷ Update volume contributions
end for
 $d\mathbf{u} \leftarrow -2 d\mathbf{u}$ 

```

In Algorithm 11, redundant computations can be avoided by exploiting the sparsity structure of the derivative matrix $\mathbf{D}_x$. Having established a general formulation for the volume contribution to the RHS, we now consider how this can be incorporated into the TDB-CUR framework.

For the evaluation of the RHS at the selected columns $\mathbf{s}$, the extension is straightforward. We simply replace $\mathbf{u}$ in Algorithm 11 with the reduced representation $\hat{\mathbf{V}}(:, \mathbf{s}) := \hat{\mathbf{V}}_{\mathbf{s}}$. This leads to the modified operations:

$$\begin{aligned} \mathbf{du} &\leftarrow 0 \text{ of shape } \hat{\mathbf{V}}_{\mathbf{s}}, \\ \mathbf{du}(i, :) &\leftarrow \mathbf{du}(i, :) + \mathbf{D}_x(i, j) \mathbf{f}_{EC}(\hat{\mathbf{V}}_{\mathbf{s}}(i, :), \hat{\mathbf{V}}_{\mathbf{s}}(j, :)), \end{aligned}$$

With the computation at the selected columns clarified, we now turn to the more involved case of evaluating the RHS at the selected rows $\mathbf{p}$, using the associated adjacency set $\mathbf{p}_a$. This case is more subtle and requires careful consideration, particularly in light of the chosen solution representation. Assuming that Algorithms 8 and 9 are used to construct the adjacency structure, several useful properties follow. First, the reduced matrix:

$$\hat{\mathbf{V}}_{\mathbf{p}_a} := \hat{\mathbf{V}}(\mathbf{p}_a, :)$$

has dimensions

$$\hat{\mathbf{V}}_{\mathbf{p}_a} \in \mathbb{R}^{3a \times N_s},$$

where a denotes the number of adjacent points associated with the local stencil of the derivative operator. In particular, since adjacency is constructed across all three conserved variables, this structure provides direct control over the shape of the extracted state matrix, which can be advantageous in the implementation of the RHS evaluation.

A second important property of the adjacency construction is that the indices in $\mathbf{p}_a$ are ordered monotonically. This ordering is particularly useful, as it allows the boundary contributions to be explicitly identified within the adjacency structure when required.

Now, in terms of modifications to Algorithm 11, two additional functions are required to ensure the correct evaluation of the RHS at the selected rows $\mathbf{p}$. As briefly mentioned in Section 3.4.3, we make use of the identity:

$$\mathbf{D}_x \hat{\mathbf{V}}(\mathbf{p}, :) = \mathbf{D}_x(\mathbf{p}, :)(:, \mathbf{p}_a) \hat{\mathbf{V}}_{\mathbf{p}_a},$$

which motivates the structure of the RHS evaluation at the selected rows. In particular, if the FOM is evaluated via a function call of the form:

$$\text{RHS}(t, \mathbf{D}_x, \mathbf{V}),$$

then the corresponding evaluation restricted to the selected rows takes the form:

$$\text{RHS}(t, \mathbf{D}_{x_g}(\mathbf{p}, :)(:, \mathbf{p}_a), \hat{\mathbf{V}}_{\mathbf{p}}, \hat{\mathbf{V}}_{\mathbf{p}_a}).$$

We define:

$$\mathbf{D}_{\mathbf{p}_a} := \mathbf{D}_{x_g}(\mathbf{p}, :)(:, \mathbf{p}_a),$$

as the restricted derivative operator associated with the adjacency structure. The general procedure for computing the RHS at the $\mathbf{p}$ rows using $\mathbf{p}_a$ is summarized in Algorithm 12.

Algorithm 12 Volume RHS evaluation on selected rows $\mathbf{p}$

Input: $\hat{\mathbf{V}}_{\mathbf{p}}, \hat{\mathbf{V}}_{\mathbf{p}_a}, \mathbf{D}_{\mathbf{p}_a}$

Output: $\mathbf{du}$

```

( $\mathbf{r}, \mathbf{c}$ )  $\leftarrow$  sparse( $\mathbf{D}_{\mathbf{p}_a}$ ) ▷ Extract sparsity pattern of  $\mathbf{D}_{\mathbf{p}_a}$ 
 $\mathbf{du} \leftarrow \mathbf{0}$  of shape  $\hat{\mathbf{V}}_{\mathbf{p}}$  ▷ Initialize RHS array
for  $(i, j) \in (\mathbf{r}, \mathbf{c})$  do ▷ Loop over sparse stencil
     $r \leftarrow \text{findRow}(\mathbf{p}(i), \mathbf{p}_a)$  ▷ Map global index in  $\mathbf{p}$  to local adjacency index
     $c \leftarrow \text{findCol}(j, 3)$  ▷ Map conserved-variable index
     $\mathbf{vol} \leftarrow \mathbf{D}_{\mathbf{p}_a}(i, j) \mathbf{f}_{EC}(\hat{\mathbf{V}}_{\mathbf{p}_a}(r, :, :), \hat{\mathbf{V}}_{\mathbf{p}_a}(c, :, :))$  ▷ Two-point flux contribution
     $\mathbf{du}(i, :) \leftarrow \mathbf{du}(i, :) + \mathbf{vol}(\mathbf{p}(r) \bmod 3, :)$  ▷ Compute update at selected DOF
end for
 $\mathbf{du} \leftarrow -2 \mathbf{du}$ 

```

There are several important differences between Algorithm 11 and Algorithm 12 that merit discussion. The first concerns the structure of the solution representation and the associated operators. In practice, the numerical flux function $\mathbf{f}_{EC}(\cdot, \cdot)$ is designed to accept inputs of the form $\mathbf{B} \in \mathbb{R}^{3 \times a}$, where a is arbitrary.

In the case of Algorithm 11 and the evaluation of the RHS for $\hat{\mathbf{V}}_{\mathbf{s}}$, this structure is straightforward to handle. In particular, $\hat{\mathbf{V}}_{\mathbf{s}} \in \mathbb{R}^{3N_x \times |\mathbf{s}|}$ can be reshaped into a tensor of size $\mathbb{R}^{N_x \times 3 \times |\mathbf{s}|}$. This allows Algorithm 11 to be applied directly, where the derivative operator $\mathbf{D}_x$ is the full (local) operator since all spatial nodes are retained in $\hat{\mathbf{V}}_{\mathbf{s}}$, i.e., $\mathbf{D}_x \in \mathbb{R}^{N_x \times N_x}$.

However, the situation in Algorithm 12 is more involved, since neither $\hat{\mathbf{V}}_{\mathbf{p}}$ nor $\hat{\mathbf{V}}_{\mathbf{p}_a}$ contains all spatial nodes in general, i.e.,

$$\hat{\mathbf{V}}_{\mathbf{p}}, \hat{\mathbf{V}}_{\mathbf{p}_a} \notin \mathbb{R}^{N_x \times 3 \times N_s}.$$

This is where the structure of the adjacency set becomes important. In particular, the fact that $|\mathbf{p}_a|$ is a multiple of three is advantageous. It allows $\hat{\mathbf{V}}_{\mathbf{p}_a}$ to be reshaped as

$$\hat{\mathbf{V}}_{\mathbf{p}_a} \in \mathbb{R}^{n \times 3 \times N_s}, \quad n := \frac{|\mathbf{p}_a|}{3},$$

which is compatible with the input requirements of $\mathbf{f}_{EC}(\cdot, \cdot)$. Moreover, since $\mathbf{p}_a$ is ordered and contains all conserved variables at each node, this reshaping preserves the correct grouping of variables.

Furthermore, Algorithm 12 uses the restricted derivative operator $\mathbf{D}_{\mathbf{p}_a}$, meaning that each index $i \in \mathbf{r}$ corresponds directly to a row in $\hat{\mathbf{V}}_{\mathbf{p}}$. A further distinction from Algorithm 11 is the need for explicit index mapping between the global and local representations when accessing $\hat{\mathbf{V}}_{\mathbf{p}_a}$.

The mapping for the column index is straightforward, since $\mathbf{c}$ already corresponds to the ordering used in $\hat{\mathbf{V}}_{\mathbf{p}_a}$. However, since $\hat{\mathbf{V}}_{\mathbf{p}_a}$ is reshaped into a tensor of size $n \times 3 \times N_s$, the global index j must be converted into its local variable index. This is achieved via the mapping

$$c \leftarrow \text{findCol}(j, 3) = j \bmod 3.$$

For example, if $j = 5$ in the global indexing convention, then:

$$c = 5 \bmod 3 = 2,$$

which corresponds to the second conserved variable in the local representation. The remaining mapping that must be addressed concerns the identification of the corresponding row index in the local representation, given the global sparse indexing associated with the selected rows $\mathbf{p}$. Since each entry in $\mathbf{r}$ corresponds directly to an entry in $\hat{\mathbf{V}}_{\mathbf{p}}$, which is constructed in the global indexing space, a mapping is required to recover the appropriate row in $\hat{\mathbf{V}}_{\mathbf{p}_a}$ corresponding to a given value of $\mathbf{p}(i)$.

For instance, the first entry in $\mathbf{r}$ satisfies $\mathbf{r}(1) = \mathbf{p}(1)$, and therefore corresponds to the first row of $\hat{\mathbf{V}}_{\mathbf{p}}$. To construct a general mapping, a straightforward, though naïve, approach is to introduce an auxiliary indexing matrix of size $|\mathbf{p}_a| \times N_s$, where each column is given by $\mathbf{p}_a^T$. This matrix can then be reshaped into a tensor of size:

$$n \times 3 \times N_s,$$

which is consistent with the structure of $\hat{\mathbf{V}}_{\mathbf{p}_a}$.

The corresponding index for $\mathbf{p}(i)$ is then obtained by locating its position within this indexing structure. This procedure is valid due to the construction of $\mathbf{p}_a$, which guarantees that all indices in $\mathbf{p}$ are contained within $\mathbf{p}_a$. By embedding the global indices into the same structural form as the solution representation and subsequently reshaping into the local format, we obtain a consistent mapping that enables direct access to $\hat{\mathbf{V}}_{\mathbf{p}_a}$ during the RHS evaluation.

This construction is well-defined because, by design, all conserved-variable entries associated with any index in $\mathbf{p}$ are present within $\mathbf{p}_a$. Consequently, the resulting mapping can be reliably used in the two-point flux evaluation. This procedure is summarized in Algorithm 13.

Algorithm 13 findRow

Input: index = $\mathbf{p}(i)$, $\mathbf{p}_a$

Output: r

$\mathbf{A} \leftarrow \mathbf{0}$ of shape $(|\mathbf{p}_a| \times N_s)$

▷ Local row index in $\hat{\mathbf{V}}_{\mathbf{p}_a}$

for $i = 1, \dots, N_s$ **do**

▷ Initialize global indexing matrix

$\mathbf{A}(:, i) \leftarrow \mathbf{p}_a^T$

▷ Replicate adjacency indices across all samples

end for

$\mathbf{A} \leftarrow \text{reshape}(\mathbf{A}, n \times 3 \times N_s)$

▷ Reshape to match $\hat{\mathbf{V}}_{\mathbf{p}_a}$, where $n = |\mathbf{p}_a|/3$

$r \leftarrow \text{find}(\mathbf{A}(:, :, 1) = \text{index})$

▷ Identify local row corresponding to global index

The final line in Algorithm 12 is given by:

$$\mathbf{du}(i, :) \leftarrow \mathbf{du}(i, :) + \mathbf{vol}(\mathbf{p}(r) \bmod 3, :).$$

Here, $\mathbf{vol}$ has dimensions $3 \times N_s$. However, only a single conserved variable component should be updated for each contribution. Therefore, it is necessary to determine which of the three conserved variables the global index $\mathbf{p}(r)$ corresponds to.

This is achieved using the operation $\mathbf{p}(r) \bmod 3$, which identifies the appropriate conserved-variable component associated with $\mathbf{p}(r)$. Consequently, the RHS update selects the corresponding row of $\mathbf{vol}$, ensuring that the correct variable component is accumulated in $\mathbf{du}$.

The next component to consider in the evaluation of the RHS is the incorporation of SATs and the imposition of boundary conditions. Since this section focuses on global FD and CG discretizations, the primary concern is the enforcement of boundary conditions. In particular, we consider SAT contributions of the form:

$$\mathbf{SAT} = \mathbf{H}^{-1} \mathbf{E} \mathbf{f}^*(\mathbf{u}, \mathbf{u}^+) \quad (\text{A.12})$$

$$\mathbf{SAT} = \mathbf{H}^{-1} \mathbf{E} (\mathbf{f}_{EC}(\mathbf{u}, \mathbf{u}^+) - \mathbf{f}_{EC}(\mathbf{u}, \mathbf{u})), \quad (\text{A.13})$$

where $\mathbf{u}^+$ denotes the boundary state prescribed at the inflow and outflow boundaries of the computational domain. For simplicity, we restrict attention to the formulation in (A.12).

The key question is how these SAT contributions are incorporated within the TDB-CUR framework. The central idea is to evaluate the SAT terms pointwise at the boundary nodes corresponding to the inflow and outflow of the domain.

For the case in which the RHS is evaluated using $\hat{\mathbf{V}}_s$, this incorporation is straightforward, and the boundary contributions can be directly added to the first and last nodes:

$$\begin{aligned} \mathbf{du}_s(1, :, :) &\leftarrow \mathbf{du}_s(1, :, :) - l \left(\mathbf{f}_{EC}(\hat{\mathbf{V}}_s(1, :, :), \mathbf{g}_L) - \frac{\lambda}{2}(\mathbf{g}_L - \hat{\mathbf{V}}_s(1, :, :)) \right), \\ \mathbf{du}_s(\text{end}, :, :) &\leftarrow \mathbf{du}_s(\text{end}, :, :) - r \left(\mathbf{f}_{EC}(\hat{\mathbf{V}}_s(\text{end}, :, :), \mathbf{g}_R) - \frac{\lambda}{2}(\mathbf{g}_R - \hat{\mathbf{V}}_s(\text{end}, :, :)) \right), \end{aligned}$$

where $l = \frac{n}{\mathbf{H}(1,1)}$ and $r = \frac{n}{\mathbf{H}(N_x, N_x)}$ are constants associated with the unit normals and the diagonal entries of the mass matrix $\mathbf{H}$ at the left and right boundary nodes, respectively. Here, $\mathbf{du}_s = \text{RHS}(t, \hat{\mathbf{V}}_s)$, and $\mathbf{g}_L, \mathbf{g}_R$ denote the prescribed boundary states at the left and right boundaries of appropriate shape.

Moving to the evaluation of the boundary contributions for $\hat{\mathbf{V}}_{\mathbf{p}}$, additional care is required. In particular, it must be ensured that all points required for the evaluation of the SAT terms are contained within $\hat{\mathbf{V}}_{\mathbf{p}}$, and, by extension, within $\hat{\mathbf{V}}_{\mathbf{p}_a}$.

Following the implementation strategy highlighted in Algorithm 2, the SAT evaluation is carried out using the procedure outlined in Algorithm 14. In essence, the algorithm first determines whether any boundary indices are present in the set $\mathbf{p}$ and, if so, computes the corresponding SAT contributions for those indices.

Subsequently, it identifies the locations of these indices within $\mathbf{p}$ and updates $\mathbf{du}_{\mathbf{p}}$ at the corresponding entries with the computed SAT values. This procedure also exploits the structure of $\hat{\mathbf{V}}_{\mathbf{p}_a}$. By construction, if a boundary index is included in $\mathbf{p}_a$, then all three conserved-variable components associated with that node are also present. This guarantees that the evaluation of $\mathbf{f}_{EC}(\cdot, \cdot)$ is well-defined and can be performed consistently within the SAT computation.

Algorithm 14 Computation of SAT contributions at the selected rows $\mathbf{p}$

Input: $\text{du}_p, \hat{\mathbf{V}}_{\mathbf{p}_a}, \mathbf{p}, \mathbf{g}_R, \mathbf{g}_L, r, \lambda$

Output: du_p

```

    if  $\{1, 2, 3\} \cap \mathbf{p} \neq \emptyset$  then ▷ Check if left boundary indices are present
         $\text{SAT}_L \leftarrow l \left( \mathbf{f}_{EC}(\hat{\mathbf{V}}_{\mathbf{p}_a}(1, :, :), \mathbf{g}_L) - \frac{\lambda}{2}(\mathbf{g}_L - \hat{\mathbf{V}}_{\mathbf{p}_a}(1, :, :)) \right)$  ▷ Left boundary SAT
        if  $1 \in \mathbf{p}$  then
             $i_1 \leftarrow \text{where}(\mathbf{p} == 1)$ 
             $\text{du}_p(i_1, :) \leftarrow \text{du}_p(i_1, :) - \text{SAT}_L(1, :)$  ▷ Update mass component
        end if
        if  $2 \in \mathbf{p}$  then
             $i_2 \leftarrow \text{where}(\mathbf{p} == 2)$ 
             $\text{du}_p(i_2, :) \leftarrow \text{du}_p(i_2, :) - \text{SAT}_L(2, :)$  ▷ Update momentum component
        end if
        if  $3 \in \mathbf{p}$  then
             $i_3 \leftarrow \text{where}(\mathbf{p} == 3)$ 
             $\text{du}_p(i_3, :) \leftarrow \text{du}_p(i_3, :) - \text{SAT}_L(3, :)$  ▷ Update energy component
        end if
    end if
    if  $\{3N_x, 3N_x - 1, 3N_x - 2\} \cap \mathbf{p} \neq \emptyset$  then ▷ Check if right boundary indices are present
         $\text{SAT}_R \leftarrow r \left( \mathbf{f}_{EC}(\hat{\mathbf{V}}_{\mathbf{p}_a}(\text{end}, :, :), \mathbf{g}_R) - \frac{\lambda}{2}(\mathbf{g}_R - \hat{\mathbf{V}}_{\mathbf{p}_a}(\text{end}, :, :)) \right)$  ▷ Right boundary SAT
        if  $3N_x - 2 \in \mathbf{p}$  then
             $i_4 \leftarrow \text{where}(\mathbf{p} == 3N_x - 2)$ 
             $\text{du}_p(i_4, :) \leftarrow \text{du}_p(i_4, :) - \text{SAT}_R(1, :)$  ▷ Update mass component
        end if
        if  $3N_x - 1 \in \mathbf{p}$  then
             $i_5 \leftarrow \text{where}(\mathbf{p} == 3N_x - 1)$ 
             $\text{du}_p(i_5, :) \leftarrow \text{du}_p(i_5, :) - \text{SAT}_R(2, :)$  ▷ Update momentum component
        end if
        if  $3N_x \in \mathbf{p}$  then
             $i_6 \leftarrow \text{where}(\mathbf{p} == 3N_x)$ 
             $\text{du}_p(i_6, :) \leftarrow \text{du}_p(i_6, :) - \text{SAT}_R(3, :)$  ▷ Update energy component
        end if
    end if
end if

```

A.4.1 A Note on Artificial Dissipation and TDB-CUR

Although CG discretizations are widely used and integrate naturally within the TDB-CUR framework, they may exhibit difficulties when applied to hyperbolic partial differential equations with solutions exhibiting sharp gradients or discontinuities, as oscillatory behaviour may arise [48, 52]. A variety of works have addressed these difficulties in the context of CG-type discretizations by proposing stabilization and dissipation mechanisms [1, 2, 48, 52]. One common approach is to introduce an explicit dissipation operator into the numerical scheme. This can improve robustness, suppress non-physical oscillations, and enhance the suitability of the method for problems involving strong gradients or discontinuities [48]. However, incorporating such a term requires a slight modification of the TDB-CUR formulation.

In this work, we adopt the dissipation operator proposed in [38] for the one-dimensional setting, defined as:

$$\mathbf{D}_{\text{diss}} := (\mathbf{D}^s)^T \mathbf{H} \mathbf{A} \mathbf{D}^s, \quad (\text{A.14})$$

where $\mathbf{D}_{\text{diss}}$ denotes the dissipation operator, and

$$\mathbf{D}^s := \underbrace{\mathbf{D}_x \cdot \mathbf{D}_x \cdots \mathbf{D}_x}_{s \text{ times}}, \quad s = q - 1,$$

with q denoting the polynomial degree associated with the derivative operator $\mathbf{D}_x$. The matrix $\mathbf{A}$ is diagonal, in the present work, it is taken to be the identity. With this addition, the semi-discrete formulation in (2.9) becomes:

$$\frac{d\mathbf{u}}{dt} = -\mathbf{H}^{-1}((\mathbf{Q} - \mathbf{Q}^T) \circ \mathbf{F})\mathbf{1} + \mathbf{H}^{-1} \mathbf{E} \mathbf{f}^*(\mathbf{u}, \mathbf{u}^+) - \underbrace{\mathbf{H}^{-1} \mathbf{D}_{\text{diss}} \mathbf{u}}_{\mathbf{w}}.$$

In terms of its impact on the TDB-CUR framework, the modification is minimal and primarily affects the construction of adjacency information. Since $\mathbf{D}^s$ is formed through repeated application of the derivative operator $\mathbf{D}_x$, the resulting stencil associated with $\mathbf{D}_{\text{diss}}$ is generally broader than that of $\mathbf{D}_x$. Consequently, the corresponding set of adjacent points must be updated accordingly to ensure all required interactions are captured.

For constructing the adjacent points associated with the dissipation operator, a similar procedure to that used in Section 3.4.1 can be used. In particular, the dissipation adjacency set is defined as:

$$\mathbf{p}_{a_{\text{diss}}} := \text{nonzero}(\mathbf{D}_{\text{diss}}(\mathbf{p}, :)).$$

With this additional adjacency structure, we now require a consistent approach for evaluating both $V(:, \mathbf{s})$ and $V(\mathbf{p}, :)$, as well as for updating the extended state $\hat{V}(\mathbf{p}_{a_{\text{diss}}}, :) := \hat{V}_{\mathbf{p}_{a_{\text{diss}}}}$. For the column-wise evaluation, the dissipation term:

$$W(:, \mathbf{s}) = H^{-1} D_{\text{diss}} \hat{V}(:, \mathbf{s}),$$

can be computed directly in a manner analogous to the volume contribution. For the row-wise evaluation, the dissipation contribution can be written as:

$$W(\mathbf{p}, :) = \left((H^{-1} D_{\text{diss}})(\mathbf{p}, :)(\cdot, \mathbf{p}_{a_{\text{diss}}}) \right) \hat{V}_{\mathbf{p}_{a_{\text{diss}}}}.$$

The final component to address is the update of $\hat{V}(\mathbf{p}_{a_{\text{diss}}}, :)$ at each stage of the time integration scheme. This can be achieved by following the procedure outlined in Section 3.4.1. Since the orthonormal bases U_{F_i} are already available for approximating $F_i(\mathbf{p}_a, :)$, we can similarly approximate $F_i(\mathbf{p}_{a_{\text{diss}}}, :)$ via:

$$F_i(\mathbf{p}_{a_{\text{diss}}}, :) \approx \hat{F}_i(\mathbf{p}_{a_{\text{diss}}}, :) = U_{F_i}(\mathbf{p}_{a_{\text{diss}}}, :) U_{F_i}(\mathbf{p}, :)^{\dagger} F_i(\mathbf{p}, :).$$

A.4.2 A Note on Different Discretizations

Although the procedures and algorithms presented in this section are formulated in the context of CG and FD discretizations, the TDB-CUR framework is not restricted to these choices, and alternative discretization schemes may also be employed. However, different discretizations introduce corresponding modifications and potential challenges in the implementation of the algorithm.

As an example, consider an arbitrary DG discretization. While the majority of the TDB-CUR procedure remains unchanged, a key difference arises in the construction of adjacency information. In particular, since numerical fluxes are used to couple adjacent elements, information from both sides of each element interface must be available. This introduces additional requirements on the indexing structure of the solution in order to ensure that all necessary DOFs are correctly included in the adjacency sets.

For illustration, consider the simple DG configuration shown in Figure A.3, consisting of three elements with four nodes per element, where nodes are assumed to exist on element boundaries. As in the CG case, each node corresponds to three conserved variables. Now consider a selected set of indices:

$$\mathbf{p} = [1, 11, 20, 26].$$

In this setting, the indices 1 and 20 may be treated analogously to the CG case, in the sense that all conserved-variable components associated with the corresponding nodes are included, i.e., [1, 2, 3] and [19, 20, 21], respectively. However, the indices 11 and 26 require additional attention, as they correspond to nodes located at element interfaces, specifically the right interface of the first element and the left interface of the third element.

Consequently, when constructing the adjacency sets for the evaluation of the RHS at the selected indices $\mathbf{p}$, additional DOFs must be included. In particular, $\mathbf{p}_a$ must contain not only the conserved-variable components associated with the selected node itself, but also the DOFs from the neighboring element required to evaluate the interface fluxes. In this example, this leads to adjacency sets of the form:

$$\begin{aligned}\mathbf{p}_a(11) &: [10, 11, 12, 13, 14, 15], \\ \mathbf{p}_a(26) &: [22, 23, 24, 25, 26, 27].\end{aligned}$$

This example highlights that, depending on the chosen discretization, careful construction of adjacency information is required to ensure the correct evaluation of the TDB-CUR algorithm. The specific implementation details are not universal and may vary depending on the problem setup and discretization strategy. Nevertheless, this flexibility is one of the key strengths of the TDB-CUR framework, as different formulations may lead to different but equally valid implementations.

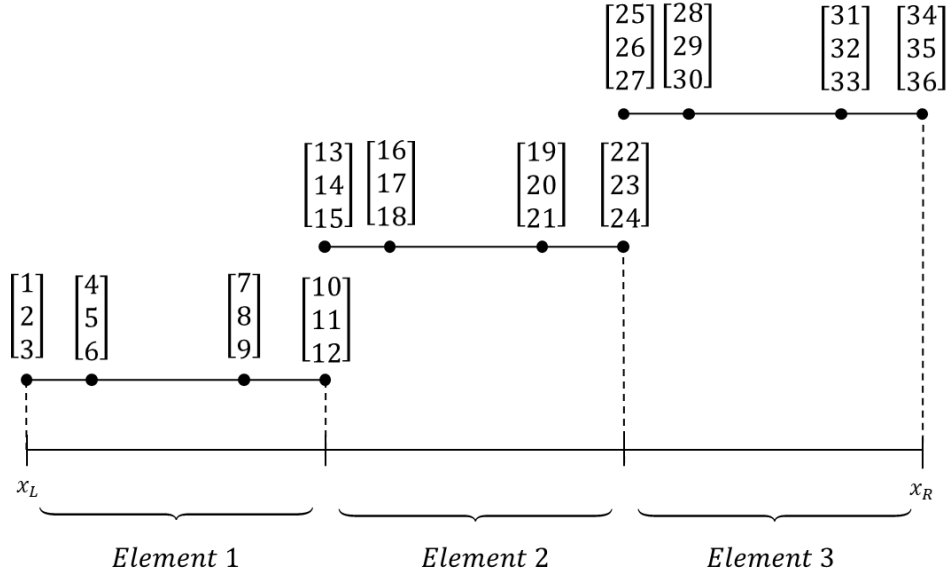


Figure A.3: A one-dimensional domain discretized into three elements with four nodes per element. The indices above each node correspond to the global DOF numbering used in the TDB-CUR implementation of the quasi-1D Euler equations. The global ordering is defined by flattening over element index, nodal position within each element, and conserved-variable components. Each index therefore represents a single conserved-variable DOF at a specific node and element. For instance, indices 1 and 25 correspond to the conserved variable $a\rho$ at the first node of the first and third elements, respectively.