# Accelerated PDE-Constrained Optimization by Adaptive Reduced Order Modelling and Goal-Oriented Hyperreduction

by

Ben Gibson

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science

University of Toronto Institute for Aerospace Studies
University of Toronto

Accelerated PDE-Constrained Optimization by Adaptive Reduced Order Modelling and Goal-Oriented Hyperreduction

Ben Gibson
Master of Applied Science
University of Toronto Institute for Aerospace Studies
University of Toronto
2022

# Abstract

We present a framework to accelerate the solution of optimization problems constrained by nonlinear partial differential equations (PDEs). To reduce the cost of objective function evaluations by several orders of magnitude, we replace the high-fidelity model (HFM) with a reduced order model (ROM). The nonlinearity motivates the use of hyperreduction, for which we use a goal-oriented empirical quadrature procedure. The hyperreduced ROM is trained specifically to preserve zero- and first-order consistency with the HFM, which provides the optimization framework with a convergence guarantee. Several optimization problems are solved using the framework, including a thermal fin problem governed by a nonlinear heat equation, an inverse aerodynamic design problem governed by the Euler equations, and an inverse aerodynamic design problem governed by the Reynolds-averaged Navier-Stokes (RANS) equations. Some speedup is observed for the first two cases, but more research on hyperreduction is necessary before a cost advantage will be seen with RANS.

# Acknowledgements

I would like to thank my advisor Professor Yano. Our weekly meetings were a constant source of renewal, and correspondence between meetings was also extremely helpful. Your passion and enthusiasm for research is highly contagious, and your feedback in preparation of this final manuscript was invaluable. I could not have asked for a better advisor for this project.

Thanks as well to Professors Nair and Zingg for their feedback at RAC meetings. I also greatly enjoyed taking your courses. The broader perspective you provided was helpful in placing this project into context. Thanks also to Professor Nair for providing feedback on this thesis.

Thanks to the other students in the ACEL group, who provided me with technical tips and tricks as well as a fun and relaxed environment at UTIAS, for the few weeks we actually worked in person! Thanks to David, my roommate and CFD wizard. Thanks to all the students at UTWind for enriching my student experience.

And thanks so much to my family – my parents, siblings, and grandparents, for keeping in touch with me and supporting me. These relationships provided my life with a deeper meaning that underpinned my work.

This work was financially supported by the Natural Science and Engineering Research Council of Canada through an Alexander Graham Bell Canada Graduate Scholarship and Discovery Grant. Computations were performed on the Niagara supercomputer at the SciNet HPC Consortium. SciNet is funded by the Canada Foundation for Innovation, the Government of Ontario, Ontario Research Fund – Research Excellence, and the University of Toronto.

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

Numerical approximations to the solutions of partial differential equations (PDEs) are playing an increasingly prominent role in the development of new products at every level of the engineering industry, from aerospace giants down to startups with innovative new products. Many engineering quantities of interest depend on problems governed by PDEs, with examples in structural mechanics, thermodynamics, aerodynamics, electromagnetics, and chemistry. Numerical approximation for PDEs remains an active and broad area of research; however, while simulations become capable of solving increasingly more complex problems at high fidelity, computational resource requirements are becoming increasingly large. This ever-increasing computational cost can put accurate simulations out of reach of a small company without access to large-scale computing facilities. Even with these computational resources, the long turnaround time associated with these simulations extends the development timeline. This motivates the current research, which attempts to preserve the accuracy of high-fidelity simulations while reducing computational cost.

We are primarily interested in reducing computational cost in so-called "many-query scenarios", where similar simulations are performed for a collection of parameter values. In this work, we focus on optimization, in which the parameters are iteratively modified to minimize some cost function. But the method we present would also be useful in many other contexts, including uncertainty quantification, simulations requiring real-time responses (such as control or graphics), or characterizing behaviour under a variety of operating conditions. While the algorithm we develop here is integrally tied to gradient-based optimization, it also demonstrates the adaptability of the method. We employ

a *goal-oriented* surrogate model, meaning the method can be adapted to fit specific requirements of the surrogate model, pivoting to a different application with minimal modification. Data from high-fidelity solutions are used to train the surrogate model on-the-fly, avoiding the need to separate offline and online stages.

## 1.2 Background

Here we provide a broad overview of the literature relevant to the tools we use in this thesis. We discuss the finite element method (FEM), and specifically the subset of methods we use in this thesis to model aerodynamic flows. We then introduce reduced order modelling (ROM), and discuss *a posteriori* error estimation in that context. We also discuss hyperreduction, which is used to apply ROM to nonlinear PDEs. The literature review is most thorough for hyperreduction, since this is the area to which this thesis hopefully contributes the most.

It would be impossible to fully explore the background of the FEM and the extent of its contributions to computational engineering. In this work, we take many principles of the FEM for granted, and treat the solutions it generates as "truth", in keeping with the convention in the ROM community [48]. We use a high-order discontinuous Galerkin (DG) FEM [7, 20] with output-based adaptive mesh refinement [11]. Both methods have a proven successful in computational aerodynamics [9, 10, 34, 31]. The formulation provides stability required for conservation laws, provides higher-order accurate approximations, and facilitates adaptive mesh refinement using unstructured meshes. The use of these state-of-the-art techniques ensures that the further cost reduction obtained by the methods we present here provides a fair comparison, not skewed by the choice of an inefficient baseline.

Despite these advanced techniques, the cost associated with these high-fidelity simulations may be prohibitive. Projection-based ROM is a method used to reduce this cost in many-query contexts, and is described in general in several review papers [53, 12, 47] and books [48, 35, 52]. Application to aerodynamics has successfully been demonstrated by [64, 72], among many others. ROM works by utilizing several "truth" solutions computed using an expensive high-fidelity solver to construct empirical, global reduced basis (RB) functions, which are used to find solutions for other parameter values at a much lower computational cost. Because they rely on precomputed high-fidelity solutions, ROM is only relevant for many-query scenarios, such as optimization, uncertainty quantification, and operational envelope characterization; and real-time scenarios, such as model predictive control and interactive design. In this work, we focus exclusively on optimization.

*A posteriori* error estimation is an important value-adding feature of a model order reduction technique, since we would like some way to estimate how much accuracy we are sacrificing to reduce the computational cost. Equipped with an error estimator, we can claim the method is *reliable* as well as *efficient*. This subject occupies a central place in Patera and Rozza's early monograph on the RB method [48]; however, some of the methods described there are problem-specific or restricted to linear problems. The dual-weighted residual (DWR) error estimation technique, also used for the aforementioned adaptive mesh refinement, has been successfully applied to ROM [43, 65]. We shall describe both the FEM and ROM in the variational context, where the reason for the similarity will become clear – the DWR method was developed for methods based on Galerkin projection without any assumptions about the form of the basis functions [11]. The particular appeal of the DWR method is that it focuses on obtaining an error estimate on the engineering quantity of interest (or "output") directly, rather than the solution to the PDE (in some norm).

Reduction of the number of basis functions alone is often not sufficient for achieving a computational cost independent of the size of the underlying FEM discretization (the technical reason will be shown in Section 2.5). This is only possible in special contexts: linear (or polynomially nonlinear) problems with affine parameter decomposition. In all other contexts, true reduction requires reduction of the cost of residual and Jacobian evaluations. Any solution to this problem is known as "hyperreduction." The first hyperreduction technique was gappy proper orthogonal decomposition (POD), first developed for image analysis [27], but later applied to aerodynamics [14]. In this technique, a randomly selected portion of the original data is used – making it useful for when the data is already randomly marred or "gappy." However, this is not the case when the data comes from the solution of a high-fidelity model, suggesting the use of a more principled approach.

Since the development of gappy POD, a slew of hyperreduction methods have been developed based on the same principle, but using some intelligent means of selecting the points used for online computation, typically referred to as "sensor locations." The empirical interpolation method (EIM) [8], developed specifically for model reduction of PDEs that do not permit affine decomposition, uses a second set of reduced basis to model the nonlinear term itself, and then uses a greedy procedure to select interpolation points that accurately reproduces the nonlinear terms. Many variants of the EIM have been developed [19, 21, 23, 41, 45, 60], but they have struggled to perform well in aerodynamics applications. An alternative hyperreduction method modelled on gappy POD is the Gauss-Newton Approximation Tensor (GNAT) method [15, 16], which uses a Petrov-Galerkin RB projection and gappy POD assisted by a least-squares tensor approximation technique.

The gappy POD-based techniques use an "interpolate-then-integrate" philosophy, where we first

approximate the nonlinear term inside an integral and then integrate exactly. However, another class of hyperreduction techniques approximates the integrals directly. Here, rather than selecting sample locations for measuring the nonlinear function, we select a sparse set of quadrature points and weights used to calculate the integral directly. Methods of this type include optimized cubature [5], engery-conserving sampling and weighting (ESCW) [29, 30, 18] and the empirical quadrature procedure (EQP) [49, 67, 66]. The EQP has been applied to both steady and unsteady elliptic and hyperbolic equations, including aerodynamics problems [24, 56, 57, 25]. One advantage of the EQP is that it permits goal-oriented selection of quadrature points, allowing for error control on the output, or any other quantity of interest, rather than simple residual- or solution-based error control [65].

There are many examples in the literature of applying ROM to various optimization problems. These come from the fields of structural dynamics [69], carbon capture and storage [1], structural topology optimization [32], thermodynamics [51], combustion [4], aortic bypass graft design [42], and aerodynamics [39, 40, 70, 68]. In many of these works, the ROM is built as the optimizer progresses using a trust region (TR) approach, and are therefore referred to as "adaptive" or constructed "on-the-fly". Some of these works make use of hyperreduction [42], but none combine hyperreduction and on-the-fly construction, and we are aware of no examples from the field of aerodynamics of applying hyperreduction to ROM-accelerated optimization. Aerodynamic problems are of particular interest, since they require considerably more cost to solve than some of the other problems mentioned here.

The gap we are trying to fill in this thesis is to combine advantages in ROM-accelerated optimization with state-of-the-art techniques in hyperreduction. We hope this further advances the applicability of ROM to larger-scale optimization problems at low cost.

## 1.3 Thesis Objectives

The overarching goal of this thesis is to demonstrate an integrated optimization framework for general nonlinear PDE-constrained problems, with a guarantee of convergence at least to a local minimum (convergence to a global minimum is virtually impossible to guarantee). We can break this down into several components:

- Develop a TR-based framework for adaptive hyperreduced ROM (HROM)-accelerated nonlinear PDE-constrained optimization.
- Identify a procedure to generate the reduced basis and a set of EQP constraints that guarantees first-order consistency of the HROM surrogate.

- Use *a posteriori* error estimation to build an intelligent TR.
- Demonstrate this framework in a series of increasingly complex problems, including two-dimensional Reynolds-averaged Navier-Stokes (RANS) flow over an airfoil, and report computational cost savings obtained using this framework.

## 1.4 Thesis Outline

In order to achieve the objectives specified above, the thesis first lays out the formulation of the various ingredients used in Chapter 2. In this chapter, we first state the general form of the PDEs under consideration, and then review the basics of the DG FEM, for the purpose of establishing notational familiarity. We then describe the projection-based ROM technique used, the exposition of which closely mirrors the exposition of the FEM. We then describe the EQP used in hyperreduction, which further builds upon the ROM formulation. Shifting the focus from PDE approximation, we also discuss geometry parametrization and optimization. The purpose of this chapter is to establish all the mathematical tools used. None of these mathematical tools are being introduced for the first time, although the EQP has only recently been introduced to the field. Most of the subjects are well-established numerical techniques; we introduce them only for completeness, and to set the stage for the next section.

In Chapter 3, we build on the ideas in Chapter 2, drawing insights from multiple fields simultaneously to build an integrated framework for ROM-accelerated optimization. Specifically, we discuss a suitable choice of basis functions and EQP training parameters for constructing an HROM on-the-fly for use in optimization. We then discuss various trust region definitions, made possible by the *a posteriori* error estimation capabilities of ROM. Both the optimization framework and the trust region definitions constitute novel developments in the field of adaptive ROM optimization. We then demonstrate a final algorithm that summarizes how everything fits together.

Chapter 4 then presents results obtained using this algorithm for a series of progressively more difficult optimization problems. We firstly solve a nonlinear thermal fin problem similar to the case studied by [51]. We then move to an aerodynamic inverse design problem starting with a baseline of NACA0012 in subsonic flow conditions. We study roughly the same problem governed by both the Euler equations and the Reynolds-averaged Navier Stokes (RANS) equations with the Spalart-Allmaras (SA) turbulence model [58, 3]. For each case, the focus of the results is on the computational cost using HROM compared to a baseline finite element discretization.

Chapter 5 covers our vision of what more might be accomplished by applying HROM to nonlinear PDE-constrained optimization problems. We comment on the significance of the findings in this thesis, as well as provide recommendations for future studies in this field.

# Chapter 2

# Mathematical Ingredients

## 2.1 General Strong Form of Nonlinear PDE

We begin by stating the class of problems considered in this thesis. We are interested in problems in which an unknown state variable $u(\mu)$ is known to follow a conservation law described by a (potentially nonlinear) system of PDEs which is parametrized in $\mu$ as follows:

$$\nabla \cdot \big(F(u(\mu);\mu) + G(u(\mu), \nabla u(\mu);\mu)\big) = S(u(\mu), \nabla u(\mu);\mu) \quad \text{in } \Omega(\mu). \tag{2.1}$$

The parameter $\mu$ is selected from a parameter domain $\mathcal{D} \subset \mathbb{R}^{n_p}$, where $n_p$ is the number of scalar parameters, or the dimension of the parameter space. We refer to $F : \mathbb{R}^{n_k} \times \mathcal{D} \to \mathbb{R}^{d \times n_k}$ in (2.1) as the advection flux operator, $G : \mathbb{R}^{n_k} \times \mathbb{R}^{d \times n_k} \times \mathcal{D} \to \mathbb{R}^{d \times n_k}$ as the diffusion flux operator, and $S : \mathbb{R}^{n_k} \times \mathbb{R}^{d \times n_k} \times \mathcal{D} \to \mathbb{R}^{n_k}$ as the source operator, where $n_k$ is the number of coupled scalar equations. The parameter can affect both the flux and source terms and the domain $\Omega(\mu) \subset \mathbb{R}^d$, where $d$ is the spatial dimension of the problem. We restrict the parameter domain $\mathcal{D}$ such that $\Omega(\mu)$ is always a Lipschitz domain. The conservation law may describe several physical quantities, the equations for which are coupled and must therefore be solved concurrently (for example, the Euler equations require the coupling of conservation laws for mass, momentum, and energy). The state maps as $u : \Omega(\mu) \times \mathcal{D} \to \mathbb{R}^{n_k}$.

The state variable is then used to compute a functional output, a scalar quantity of interest $q(u(\mu);\mu)$, consisting of volume and boundary integrals of the form

$$q(u(\mu);\mu) \equiv \int_{\Omega(\mu)} q_V(u(\mu);\mu)\ dV + \int_{\partial\Omega(\mu)} q_B(u(\mu);\mu)\ dS, \tag{2.2}$$

where $q_V$ and $q_B$ are the integrands for the volume and boundary contributions to the output, respectively.

## 2.2 Variational Form

We restate the problem above in the variational (weak) form. We assume that for any $\mu \in \mathcal{D}$, $u(\mu)$ belongs to a suitable function space $\mathcal{V}$ (which is restricted to functions that satisfy Dirichlet boundary conditions). The problem can then be restated using the weighted residual method and a Galerkin scheme. We introduce a residual $r : \mathcal{V} \times \mathcal{V} \times \mathcal{D} \to \mathbb{R}$ of the form

$$r(w, v; \mu) \equiv \int_{\Omega(\mu)} v\Big(\nabla \cdot \big(F(w;\mu) + G(w, \nabla w; \mu)\big) - S(w, \nabla w; \mu)\Big)\ dV. \tag{2.3}$$

Integration by parts produces boundary integral terms, which allow for natural treatment of some boundary conditions; other boundary conditions can be enforced directly. Integrating by parts,

$$\begin{aligned} r(w, v; \mu) = &\int_{\partial\Omega(\mu)} v\big(F(w;\mu) + G(w, \nabla w; \mu)\big) \cdot \hat{n}\ dS \\ &+ \int_{\Omega(\mu)} -\big(\nabla v \cdot \big(F(w;\mu) + G(w, \nabla w; \mu)\big) - vS(w, \nabla w; \mu)\big)\ dV. \end{aligned} \tag{2.4}$$

Enforcing boundary terms usually involves either specifying the value of the solution itself on the boundary through restrictions on $\mathcal{V}$ (Dirichlet) or specifying one of the flux terms on the boundary, which can be directly substituted into (2.4) (Neumann and Robin). We have abused notation by assimilating Neumann and Robin boundary conditions into the definitions of $F$ and $G$ on $\partial\Omega(\mu)$. We then seek $u(\mu) \in \mathcal{V}$ such that

$$r(u(\mu), v; \mu) = 0 \quad \forall v \in \mathcal{V}. \tag{2.5}$$

Because we are interested in performing optimization, we require not only the solution to the PDE, which we call the "primal solution," and the output. We are also interested in computing the gradient of the output functional with respect to the parameter values. We compute the gradient using an adjoint method, where the adjoint $z(\mu)$ is also referred to as the "dual solution". The

adjoint method requires the solution of an adjoint problem: find $z(\mu) \in \mathcal{V}$ such that

$$r'[u(\mu)](v, z(\mu); \mu) = q'[u(\mu)](v; \mu) \quad \forall v \in \mathcal{V}, \tag{2.6}$$

where $r'[u(\mu)](v, w; \mu)$ is the Gâteaux derivative of $r(\cdot, w; \mu)$ about $u(\mu)$ in the direction $v$, and $q'[u(\mu)]$ is the Gâteaux derivative of $q(\cdot; \mu)$ about $u(\mu)$ in the direction $v$. Since $r(w, v; \mu)$ is linear in $v$, the adjoint problem is always linear.

We also define a residual sensitivity operator $\partial_\mu r : \mathcal{V} \times \mathcal{V} \times \mathcal{D} \to R^{n_p}$ where $\partial_\mu r(w, v; \mu) \equiv \frac{\partial r(w,v;\mu)}{\partial \mu}$. A similar output sensitivity operator $\partial_\mu q : \mathcal{V} \times \mathcal{D} \to R^{n_p}$ is defined $\partial_\mu q(v; \mu) \equiv \frac{\partial q(v;\mu)}{\partial \mu}$. The adjoint method then computes the total derivative of the output with respect to the parameter as follows:

$$\frac{D}{D\mu}\big(q(u(\mu); \mu)\big) = -\partial_\mu r(u(\mu), z(\mu); \mu) + \partial_\mu q(u(\mu); \mu). \tag{2.7}$$

Dependence on $\mu$ through $u$ is handled by the derivative of the dual-weighted residual term, and dependence directly through $q$ is handled by the second term.

While this problem is infinite-dimensional, both the FEM and our proposed ROM allow us to use finite-dimensional approximations of $\mathcal{V}$ to approximate the solution to (2.5). Throughout this thesis, we assume that both (2.5) and its finite-dimensional approximations are well-posed.

## 2.3 Finite Element Method

### 2.3.1 Construction of Approximation Space $\mathcal{V}_h$

We use a standard FEM to construct a high-fidelity model (HFM) to approximate the solution. In the FEM, the domain $\Omega(\mu)$ is discretized in space into a mesh, a collection of $n_e$ elements $\mathcal{M}_h \equiv \{E_i\}_{i=1}^{n_e}$. Within each element, the solution is approximated using polynomials of degree $p$. Thus we have

$$\mathcal{V}_h \equiv \big\{ v \in \mathcal{V} \;\big|\; v|_E \in \mathbb{P}^p(E)^{n_k} \quad \forall E \in \mathcal{M}_h \big\}. \tag{2.8}$$

We use a set of Lagrange polynomial basis functions to represent $\mathbb{P}^p(E)$, and define each basis function as being non-zero for only one of the components. This means there are $n_k$ duplicates of each basis function, each one corresponding to a different scalar. Altogether, these basis functions

are collected in the set $\{\phi_i\}_{i=1}^{\mathcal{N}}$, where $\mathcal{N}$ is the number of degrees of freedom (DOF) of the HFM.

This global basis set describes a high-dimensional function space $\mathcal{V}_h \equiv \text{span}\{\phi_i\}_{i=1}^{\mathcal{N}}$. Any function $v_h \in \mathcal{V}_h$ is constructed of piecewise polynomials and coefficients $\hat{v}_h \in \mathbb{R}^{\mathcal{N}}$ as follows:

$$v_h = \sum_{i=1}^{\mathcal{N}} \hat{v}_{h,i}\phi_i. \tag{2.9}$$

By increasing the polynomial degree $p$, we can increase the resolution of the model without changing the number of elements. This is referred to as $p$-refinement, whereas changing the number of elements is referred to as $h$-refinement. Both of these refinement methods increase the resolution by increasing the total number of DOF of the model, meaning they are both associated with increases in computational cost.

In the continuous Galerkin method, $\mathcal{V}_h$ is restricted to functions with $\mathbb{C}^0$ continuity, and so these continuity requirements are enforced during the accumulation of basis functions to form $\{\phi_i\}_{i=1}^{\mathcal{N}}$. In the DG method, however, there are no continuity requirements. Instead, continuity is implicitly encouraged by the addition of an artificial numerical flux term [7, 20]. This term penalizes "jump" in the solution between two elements – it depends on $(u^+ - u^-)$ across the interface. This results in a surface integral over all "interior facets" in addition to the volume and boundary integrals, which is added to (2.4). We therefore define a modified residual term $r_h(w_h, v_h; \mu)$ which follows (2.4), but contains the modifications described here. In this case, $\mathcal{N} = n_e \cdot n_z \cdot n_k$, minus the number of nodes associated with Dirichlet boundary conditions, where $n_z$ is the number of nodes per element.

The HFM approximation is then computed by finding $u_h(\mu) \in \mathcal{V}_h$ such that

$$r_h(u_h(\mu), v_h; \mu) = 0 \quad \forall v_h \in \mathcal{V}_h. \tag{2.10}$$

This results in an $\mathcal{N}$-dimensional system of nonlinear algebraic equations which can be solved using a Newton or Newton-like (e.g. pseudo-transient continuation) solver [36, 17]. The ultimate goal is to compute the solution coefficients $\hat{u}_h(\mu)$ required to represent the solution in the form shown in (2.9). This requires evaluation of the residual Jacobian, $r_h'[u_h(\mu)](\phi_j, \phi_i; \mu)$.

The FEM is primal-dual consistent, meaning the dual problem associated with the discrete approximation of the primal problem can be considered a discrete approximation of the dual problem associated with the true solution. We find $z_h(\mu) \in \mathcal{V}_h$ such that

$$r_h'[u_h(\mu)](v_h, z_h(\mu); \mu) = q'[u_h(\mu)](v_h; \mu) \quad \forall v_h \in \mathcal{V}_h. \tag{2.11}$$

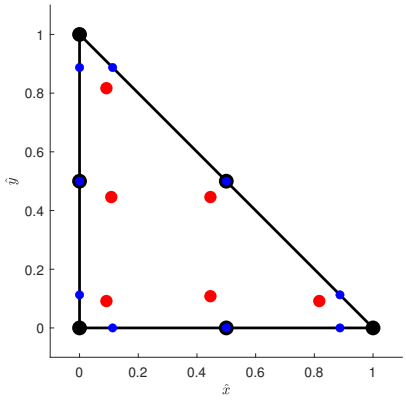



Figure 2.1: Second order ($p = 2$) triangular reference element with quadrature points ($p_q = 4$, $n_q^b = n_q^i = 3$, $n_q^v = 6$). The Lagrange basis nodes are shown in black, the volume quadrature points in red, and the facet quadrature points in blue.

The dual is represented using the same finite element basis functions, in the same form as the solution in (2.9). The residual Jacobian associated with this problem is therefore the same as the one associated with the primal problem, whereas the left hand side in (2.11) is now the dependency of the output on the primal solution.

### 2.3.2 Numerical Integration

Evaluation of both the residual (2.3) and its Jacobian involves integration over the domain volume, boundary facets, and interior facets. We can perform this integration element-wise and facet-wise. A quadrature rule of degree $p_q$ can exactly integrate polynomials of degree $p_q$. For higher order or non-polynomial terms, quadrature integration introduces some error, which decreases as $p_q$ increases. A quadrature rule consists of a set of $n_q$ points $\{x_q\}$ at which to evaluate the function, and weights $\{\rho_q\}$, so that the integration over a domain $\Omega$ of a function $f(x)$ may be approximated as

$$\int_\Omega f(x)\ dx \approx \sum_{q=1}^{n_q} \rho_q f(x_q). \tag{2.12}$$

Provided that a sufficiently dense quadrature rule is used, the error associated with numerical integration is typically much lower than the error associated with the FEM discretization (approximation of $\mathcal{V}_h$) and is not considered further in this thesis. A reference triangular element with $p = 2$ ($n_z = 6$) and quadrature rule with $p_q = 4$ is shown in Figure 2.1.

To integrate over the entire domain, we consider a collection of points distributed throughout the domain $\{x_q^v\}_{q=1}^{N_q^v}$, another for boundary integration $\{x_q^b\}_{q=1}^{N_q^b}$, and another for interior facet integration

$\{x_q^i\}_{q=1}^{N_q^i}$. These points are the collection of all $d$-dimensional, degree $p_q$ quadrature rules on each element, and $(d-1)$-dimensional, degree $p_q$ quadrature rules on each facet. Associated with these points are the quadrature weights $\{\rho_q^v\}_{q=1}^{N_q^v}$, $\{\rho_q^b\}_{q=1}^{N_q^b}$, and $\{\rho_q^i\}_{q=1}^{N_q^i}$. The residual form (2.3) can then be approximated by applying these quadrature rules as

$$\begin{aligned} r_h(w,v;\mu) &= \int_{\Omega(\mu)} r_v(w,v;\mu)\ dV + \int_{\partial\Omega(\mu)} r_b(w,v;\mu)\ dS + \int_{\partial E\setminus\partial\Omega(\mu)} r_i(w,v;\mu)\ dS \\ &\approx \sum_{q=1}^{N_q^v} \rho_q^v r_v(w(x_q^v), v(x_q^v);\mu) + \sum_{q=1}^{N_q^b} \rho_q^b r_b(w(x_q^b), v(x_q^b);\mu) + \sum_{q=1}^{N_q^i} \rho_q^i r_i(w(x_q^i), v(x_q^i);\mu), \end{aligned} \tag{2.13}$$

where $r_v(w,v;\mu)$, $r_b(w,v;\mu)$, and $r_i(w,v;\mu)$ are the contributions to the total residual coming from volume, boundary, and interior facet (denoted $\partial E\setminus\partial\Omega(\mu)$) integrals respectively. They are derived from $F$, $G$, and $S$ in (2.4), including boundary conditions and the numerical fluxes associated with the DG method. We refer to these as the volume, boundary, and interior facet residual operators. In (2.13), the expression $w(x)$ contains more than simply the value of $w$ at the point $x$; it may also imply computing $\nabla w$ about $x$.

### 2.3.3 Error Estimation and Adaptive Mesh Refinement

Because the accuracy of the solution depends on the resolution of the mesh, there is motivation to make the mesh very fine. Yet this comes with increased computational cost. The accuracy of the output is much more sensitive to certain parts of the domain than others; however, it usually involves a high degree of experience to know where to refine *a priori*. For inexperienced users, users looking for an automated process, or even experts looking for a principled approach, an output-based adaptive mesh refinement (AMR) strategy is preferred.

The strategy we use is the DWR procedure described by [11] and used by [34, 31] and others. The dual problem is solved on the same mesh but using polynomials of degree $p+1$, one higher than those used for the primal problem. In other words, we solve the primal problem on the initial mesh to yield $u_h^{(1)}(\mu) \in \mathcal{V}_h^{(1)}$. The same mesh with polynomials of degree $p+1$ generates a space $\mathcal{V}_h^{(1')}$. We then solve the following dual problem: find $z_h^{(1')} \in \mathcal{V}_h^{(1')}$ such that

$$r_h'[u_h^{(1)}](v^{(1')}, z_h^{(1')}(\mu);\mu) = q'[u_h^{(1)}(\mu)](v^{(1')};\mu) \quad \forall v \in \mathcal{V}_h^{(1')}. \tag{2.14}$$

We then compute an output error estimate as follows:

$$|q(u(\mu);\mu) - q(u_h(\mu);\mu)| \approx \eta_h(\mu) \equiv \left|r_h\left(u_h^{(1)}(\mu), z_h^{(1')}(\mu);\mu\right)\right|. \tag{2.15}$$

We can also define a localized error estimate, in which the test function $z_h^{(1')}(\mu)$ in (2.15) is restricted to a single element:

$$\eta_{h,E}(\mu) \equiv \left|r_h\left(u_h^{(1)}(\mu), z_h^{(1')}(\mu)|_E;\mu\right)\right|. \tag{2.16}$$

This gives an element-wise error indicator, or an indicator of how much each element is contributing to the total output error. The elements are then ranked by this element-wise error indicator, and those with the highest error indicator are marked for refinement, while the rest are left as they were in $\mathcal{V}_h^{(1)}$. For example, in this thesis we marked the top 15% elements for refinement. Those marked elements are then subdivided. This yields a new solution space $\mathcal{V}_h^{(2)}$. This process is repeated several times until the error estimate $\eta_h(\mu)$ is below a certain tolerance.

### 2.3.4 Computational Cost

Here we tabulate the many sources of high computational cost associated with approximating the solution using the FEM. For simplicity, we shall define the number of quadrature points $N_q \equiv N_q^v + N_q^b + N_q^i$, and note that $N_q$ is roughly $\mathcal{O}(\mathcal{N})$ (typically higher by some multiple – the reference element in Figure 2.1 had 6 DOF and 15 quadrature points).

1. We require evaluation of the residual $\mathcal{N}$ times, once with respect to each of the FEM basis functions. Each evaluation requires integration, but this does not scale with $N_q$, as each basis function is non-zero only at certain quadrature points (compact support). However, ultimately each quadrature point must be visited, so residual evaluation can be thought of as scaling with either $\mathcal{O}(\mathcal{N})$ or $\mathcal{O}(N_q)$. We also require evaluation of the residual Jacobian, a sparse matrix of dimension $\mathcal{N} \times \mathcal{N}$. Since the matrix is sparse, this also scales with either $\mathcal{O}(\mathcal{N})$ or $\mathcal{O}(N_q)$.

2. We then solve a system of linear equations associated with the Jacobian and the current estimate for the solution coefficients. The solution of a sparse linear system scales with $\mathcal{O}(\mathcal{N}^b)$, where $1 \leq b \leq 2$. This depends on the sparsity of the Jacobian, the quality of the preconditioner used, and the solution method used to solve the linear system. In this work, we use GMRES with a block ILU preconditioner with the minimum-discarded fill reordering [54, 50].

3. For linear problems (including all adjoint problems), one solution of the system is sufficient; but for nonlinear primal problems, many Newton iterations, sometimes $\mathcal{O}(100)$, are required to converge to the solution.

4. We are interested in this thesis in many-query scenarios, in which the entire problem described above must be repeatedly solved with different values of $\mu \in \mathcal{D}$.

For large-scale problems, $\mathcal{N}$ can be $\mathcal{O}(10^6)$ or higher, meaning simulations can take many hours or even days on a high performance computer running many cores in parallel. This can put parametric studies or optimization out of reach very quickly even for companies or researchers with moderate to large computational resources.

Research is ongoing to reduce the cost of each of these items. For example, the adaptive mesh refinement strategy of Section 2.3.3 seeks to decrease $\mathcal{N}$ and $N_q$ by reducing the number of elements needed to obtain a given accuracy on the output (item 1). The solution of sparse linear systems of equations (item 2) is important for a wide range of applications, and is therefore well-researched, as is the solution of nonlinear systems with Newton-like methods (item 3). In optimization, research focuses on reducing the number of function evaluations required for convergence (item 4).

In this thesis, however, we focus on dramatically reducing $\mathcal{N}$ and $N_q$ (i.e., by several orders of magnitude) by leveraging the fact that we do solve the problem multiple times for different values of $\mu$. Rather than rely on the HFM solution for each parameter value, we use data from only a few HFM solutions to build an empirical (but physics-informed) surrogate model but with minimal loss of accuracy. Reduction of $\mathcal{N}$ is discussed in Section 2.4 and reduction of $N_q$ is discussed in Section 2.5.

## 2.4 Reduced Order Modelling

In the FEM, there is a tradeoff between efficiency and geometric convenience, as the DOF are related to spatial elements that must conform to $\Omega$. AMR improves resemblance of possible functions in $\mathcal{V}_h$ to the expected shape of the solution, but to a limited extent. There is significant redundancy, parts of $\mathcal{V}_h$ that are highly unlikely to contain solutions to the PDE. An unnecessarily rich space $\mathcal{V}_h$ makes the problem of finding $u_h(\mu) \in \mathcal{V}_h$ (implying a full search) such that (2.10) is satisfied more costly. The element-wise compact support of the FEM basis functions does produce sparsity in the quadrature integration and residual Jacobian; however, the sheer number of basis functions required results in large computational costs. Globally supported basis functions that bear some

relation to the primal solution itself (or more accurately, to the parametrically induced manifold of primal solutions) would be able to represent the solution much more efficiently. In projection-based ROM, these reduced basis (RB) functions are empirically determined by examining orthogonal components of the solution data for a small collection of existing HFM solution snapshots. The theory of ROM is covered extensively in [48, 12, 35]. ROM has been successfully used in a wide range of engineering applications, including applications from heat transfer [51], structural dynamics [69], topology optimization [32], carbon capture and storage [1], and aerodynamic design [70]. Many more citations could be made, but the works cited here use ROM in the context of optimization.

### 2.4.1 Primal State and Output Approximation

To use the RB method to estimate the primal solution, we construct a low-dimensional space $\mathcal{V}_N^{\text{pr}}$ that is still capable of accurately approximating the primal solution manifold: $(u_N(\mu) \in \mathcal{V}_N^{\text{pr}}) \approx (u_h(\mu) \in \mathcal{V}_h)$. This new approximation space is constructed with primal RB functions: $\mathcal{V}_N^{\text{pr}} \equiv \text{span}\{\zeta_i^{\text{pr}}\}_{i=1}^{N_{\text{pr}}}$. We define $N_{\text{pr}} \equiv \dim(\mathcal{V}_N^{\text{pr}})$, and is usually $\mathcal{O}(10)$, making it much less than $\mathcal{N}_h$, which is usually $\mathcal{O}(10^4)$–$\mathcal{O}(10^6)$. Any function $v_N \in \mathcal{V}_N^{\text{pr}}$ is represented by

$$v_N = \sum_{i=1}^{N_{\text{pr}}} \hat{v}_{N,i} \zeta_i^{\text{pr}}. \tag{2.17}$$

Unlike the FEM space $\mathcal{V}_h$, which is designed to approximate functions in $\mathcal{V}$ using piecewise polynomials, the RB space $\mathcal{V}_N^{\text{pr}}$ is tailored to only represent functions in the primal parametric manifold. This restriction means we can use an extremely low number of basis functions tailored to capture the dominant modes of behaviour of $u_h(\mu)$ in the parametric manifold of solutions.

To solve for the primal state in this reduced space, we once again use a Galerkin projection. We find $u_N(\mu) \in \mathcal{V}_N^{\text{pr}}$ such that

$$r_h(u_N(\mu), v_N; \mu) = 0 \quad \forall v_N \in \mathcal{V}_N^{\text{pr}}. \tag{2.18}$$

The same residual operator from the FEM formulation (2.13) is used. This nonlinear system of equations of dimension $N_{\text{pr}}$ is solved for the solution coefficients $\hat{u}_{N,j}(\mu)$ using a Newton-like method, using the residual Jacobian $r_h'[u_N(\mu)](\zeta_j^{\text{pr}}, \zeta_i^{\text{pr}}; \mu)$ (which is no longer sparse).

For linear coercive problems, Cea's Lemma can be invoked to show that the RB method achieves Galerkin optimality, meaning that the solution error $\|u_h(\mu) - u_N(\mu)\|_{\mathcal{V}}$, is a fixed fraction $C$ (independent of $\mathcal{V}_N^{\text{pr}}$) of the minimum possible error relative to $u_h(\mu)$ for any function in $\mathcal{V}_N^{\text{pr}}$ (see proof

in [48]). In other words, for linear coercive problems the solution $u_N(\mu)$ to (2.18) guarantees that

$$\|u_h(\mu) - u_N(\mu)\|_{\mathcal{V}} = C \inf_{w_N \in \mathcal{V}_N^{\mathrm{pr}}} \|u_h(\mu) - w_N\|_{\mathcal{V}}. \tag{2.19}$$

While it is not possible to prove the same theoretical guarantee for all nonlinear and non-coercive problems, a similar behaviour generally is observed [13].

In order to exactly compute the gradient of the objective function evaluated using $u_N(\mu)$, we must solve the "discrete" adjoint problem. In other words, we must approximate the adjoint in the same space as the primal solution. This is not guaranteed, or indeed likely at all, to yield a good approximation of the high-fidelity dual or the continuous dual, since the primal space was not designed to approximate dual solutions. However, it does yield the exact gradient of the RB-computed output. We seek $z_N^{\mathrm{pr}}(\mu) \in \mathcal{V}_N^{\mathrm{pr}}$ such that

$$r_h'[u_N(\mu)](v_N, z_N^{\mathrm{pr}}(\mu); \mu) = q'[u_N(\mu)](v_N; \mu) \quad \forall v_N \in \mathcal{V}_N^{\mathrm{pr}}. \tag{2.20}$$

Using this approximation of the dual in the primal RB space, we compute

$$\frac{D}{D\mu}\big(q(u_N(\mu); \mu)\big) = -\partial_\mu r_h(u_N(\mu), z_N^{\mathrm{pr}}(\mu); \mu) + \partial_\mu q(u_N(\mu); \mu) \tag{2.21}$$

We shall explain in detail in Section 3.1 how the RB functions are selected. Three general points are mentioned here, which apply in virtually any projection-based ROM setup.

- The basis functions are constructed based on pre-computed training solutions (snapshots).
- The basis functions are orthogonal (not required, but improves conditioning).
- The RB space $\mathcal{V}_N^{\mathrm{pr}}$ can be progressively enriched by adding more and more basis functions.

### 2.4.2 DWR Error Estimation

The concept of progressively enriching the ROM during the procedure of interest, such as optimization, is very similar to the concept of AMR. In both cases, we introduce further degrees of freedom to the approximation space in order to improve the accuracy of the solution. However, in the case of AMR, the additional degrees of freedom come from improved *spatial* resolution. In the ROM, spatial resolution is fixed to that of the underlying HFM, as the basis functions are already globally defined. Rather, additional degrees of freedom come from improved *modal* resolution, as we add global orthogonal basis functions.

Extending the analogy further, in AMR, we use a DWR method to determine where in space further degrees of freedom are necessary. Here we describe a similar DWR method to determine "where" in the parametrically induced solution manifold an additional training point needs to be added. Practically speaking, in the context of optimization and building the ROM on-the-fly, the DWR method determines "when" during the course of traversing the manifold the RB space must be enriched.

As in AMR, the dual solution used for the DWR error estimation cannot be computed in the same approximation space as is used for computing the primal solution. This is because the primal solution is defined as the function for which the residual is zero for all test functions in that space. The most obvious choice of space for representing the dual $\mathcal{V}_N^{\mathrm{du}}$ is to generate it in the same manner as $\mathcal{V}_N^{\mathrm{pr}}$, from a set of training snapshots of $z_h(\mu)$. This space is tailored to represent the parametrically induced manifold of dual solutions, and consists of $N_{\mathrm{du}}$ basis functions: $\mathcal{V}_N^{\mathrm{du}} \equiv \{\zeta_i^{\mathrm{du}}\}_{i=1}^{N_{\mathrm{du}}}$.

Then in a similar manner to (2.11) and (2.15), we approximate the HFM dual (given by (2.11)) in the dual RB space, seeking $z_N^{\mathrm{du}}(\mu) \in \mathcal{V}_N^{\mathrm{du}}$ such that

$$r_h'[u_N(\mu)](v_N, z_N^{\mathrm{du}}(\mu); \mu) = q'[u_N(\mu)](v_N; \mu) \quad \forall v_N \in \mathcal{V}_N^{\mathrm{du}}. \tag{2.22}$$

Using this approximation of the dual in the dual RB space, we construct a ROM output error estimator $\eta_N$ as follows:

$$|q(u_h(\mu); \mu) - q(u_N(\mu); \mu)| \approx \eta_N(\mu) = \left|r_h(u_N(\mu), z_N^{\mathrm{du}}(\mu); \mu)\right|. \tag{2.23}$$

## 2.5 Hyperreduction by an Empirical Quadrature Procedure

Although the RB method described in Section 2.4 dramatically reduces the number of DOF (and thus the cost associated with solving for the solution update in each Newton iteration), each residual and Jacobian evaluation still requires an integration over the entire domain, using the quadrature rule (2.13). The compact support of the HFM basis functions means cost of the HFM does not scale with $\mathcal{N} \times N_q$. But having lost compact support, reducing $\mathcal{N}$ to either $N_{\mathrm{pr}}$ or $N_{\mathrm{du}}$ does not significantly affect the cost of residual evaluations, which still scales with $N_q$. Each quadrature point in the mesh still needs to be visited, and the flux and source terms from (2.4) evaluated. For this reason, we refer to a ROM implemented with full quadrature as a "naive ROM."

If the residual operator is affine in the solution and affine with respect to functions of the

parameter, we can then rewrite (2.18) as follows:

$$r_h(u(\mu), \zeta_i^{\mathrm{pr}}; \mu) = \sum_{a=0}^{n_a} \Theta_a(\mu) \sum_{j=1}^{N_{\mathrm{pr}}} \hat{u}_{N,j}(\mu) r_a(\zeta_j^{\mathrm{pr}}, \zeta_i^{\mathrm{pr}}) = \sum_{b=0}^{n_b} \bar{\Theta}_b(\mu) \bar{r}_b(\zeta_i^{\mathrm{pr}}), \quad i = 1, ..., N_{\mathrm{pr}}, \tag{2.24}$$

where $r_a(\cdot, \cdot)$ and $\bar{r}_b(\cdot)$ arise from affine separation of the residual $r_h(\cdot, \cdot; \mu)$ into $n_a$ and $n_b$ (typically small numbers) parameter-independent residual bilinear and linear forms and parameter-dependent functions $\Theta_a : \mathcal{D} \to \mathbb{R}$ and $\bar{\Theta}_b : \mathcal{D} \to \mathbb{R}$. This is affine parameter decomposition. Since $r_h(\cdot, \cdot; \mu)$ is affine in the first argument, we can separate the RB functions from their coefficients. We can then evaluate $(n_a + n_b) \cdot N_{\mathrm{pr}}^2$ residual terms, with cost dependent on $N_q$, and then regardless of the number of ROM solves no further integration is required. There are also methods for applying naive ROM when the PDE is nonlinear but a polynomial in the state, and still separable in a manner similar to (2.24) [63, 62]. But the number of residual terms that must be evaluated offline scales by $(n_a + n_b) \cdot (N_{\mathrm{pr}})^t$, where $t - 1$ is the maximum degree of the polynomial applied to $u$ in the PDE. To extend ROM beyond this limited class of problems, we must employ a hyperreduction technique.

The empirical quadrature procedure (EQP) is a technique used to approximate integrals using a sparse quadrature rule [49], which has been used in the context of hyperreduction [67]. This involves empirically selecting a much smaller set of $\tilde{N}_q$ reduced quadrature points $\{\tilde{x}_q\}_{q=1}^{\tilde{N}_q}$ and weights $\{\tilde{\rho}_q\}_{q=1}^{\tilde{N}_q}$, where $\tilde{N}_q \ll N_q$, such that the integral of the form (2.13) approximated using this quadrature rule is approximately equal to the result yielded by the full quadrature rule. We represent the EQP-related quantities with a tilde; throughout this thesis, any quantity $\tilde{\cdot}$ is computed using an empirical quadrature rule. As before, if we treat $f : \Omega \to \mathbb{R}$ as an integrand of interest, we desire:

$$\int_\Omega f(x)\, dx \approx \sum_{q=1}^{N_q} \rho_q f(x_q) \approx \sum_{q=1}^{\tilde{N}_q} \tilde{\rho}_q f(\tilde{x}_q). \tag{2.25}$$

Since $\tilde{N}_q \ll N_q$, the cost associated with this integration is much lower.

It would not have been possible to find a sparse quadrature approximation for solving the HFM problem (2.10). The FEM basis functions already have compact support, but since the problem has $\mathcal{N}$ degrees of freedom, each basis function is integrated separately, as this is how we solve (2.10) $\forall v_h \in \mathcal{V}_h$. Because we now only need to integrate $N_{\mathrm{pr}}$ or $N_{\mathrm{du}}$ basis functions, all of which have global support, it is more realistic to expect a sparse quadrature rule to perform well.

The problem of finding a sparse set of quadrature points that meet certain criteria is a challenging one. To simplify the problem, we assume that the new points are a subset of the original points.

Then we can recast the problem as including all the original points, but finding a set of weights of size $N_q$ such that most of the weights are zero. We shall describe this set of weights as $\{\hat{\rho}_q\}_{q=1}^{N_q}$, where $\{\tilde{\rho}_q\}_{q=1}^{\tilde{N}_q} = \{\hat{\rho}_q \mid \hat{\rho}_q \neq 0,\ q = 1, ..., N_q\}$, the non-zero entries of $\hat{\rho}_q$. We then select a set of integrands $\{f_i(x)\}_{i=1}^{n_c}$ we wish to integrate accurately, where $n_c$ describes the number of accuracy constraints we place on our EQP. (See Section 3.2 for the selection of constraints.) We then have the minimization problem:

$$\begin{aligned} \hat{\rho} &= \underset{\rho \in \mathbb{R}_{\geq 0}^{N_q}}{\arg\min} \|\rho\|_0 \\ \text{subject to } &\left| \sum_{q=0}^{N_q} \rho_q f_i(x_q) - \int_\Omega f_i(x)\ dx \right| \leq \varepsilon_{\text{EQP}},\ \text{for } i = 1, ..., n_c, \end{aligned} \tag{2.26}$$

where $\varepsilon_{\text{EQP}} \in \mathbb{R}_{\geq 0}$ is the EQP tolerance. The integration in (2.26) is performed using the original HFM quadrature rule. This optimization problem is still combinatorially hard, so we approximate it by solving a sparse non-negative least squares (NNLS) problem following [29, 30]. This algorithm adds weights one by one until all accuracy constraints are met. Since we add weights one by one, this involves solving a sequence of systems of overdetermined algebraic equations using the least squares method. See Appendix A for more details and discussion on solution of the NNLS problem, or refer to [38, 18].

It is theoretically possible (ignoring issues of machine precision and conditioning) to drive the error in (2.26) down to 0 if we add weights until the system is fully determined (i.e. full rank, rank $n_c$) and then solve the linear system exactly. However, a non-zero tolerance $\varepsilon_{\text{EQP}}$ promotes sparsity of the reduced quadrature rule. Enforcing non-negativity of the weights is intended to yield a stable quadrature rule, one that works for similar integrals not included in the constraint set. This maintains some connection with the intuitive understanding of a quadrature rule — sampling the data at characteristic locations.

As noted in Section 2.3, the residual operator is split into three parts: a volume part, a boundary part, and an interior facet part. In the description of the EQP just given, we have simply referred to $N_q$. It is possible to combine all weights into one NNLS problem, or it is possible to pose the accuracy constraints for the contribution from each integral and solve three separate NNLS problems. The number of reduced quadrature points will be higher, but the cost associated with the NNLS problem may not be much greater as each problem has fewer degrees of freedom. Splitting the problem like this results in a more reliable quadrature rule, and sometimes demonstrates greater stability in evaluating integrals other than those used for accuracy constraints.

Approximating the primal and dual states are then straightforward, but we introduce them for notational completeness. We find $\tilde{u}_N(\mu)$ such that $\tilde{r}_h(\tilde{u}_N(\mu), v_N; \mu) = 0 \; \forall v_N \in \mathcal{V}_N^{\text{pr}}$, where $\tilde{r}_h(\cdot, \cdot; \mu)$ is an integral of the same form as $r_h(\cdot, \cdot; \mu)$ only evaluated using the new empirical quadrature rule. We find $\tilde{z}_N^{\text{pr}}(\mu)$ such that $\tilde{r}_h'[\tilde{u}_N(\mu)](v_n, \tilde{z}_N^{\text{pr}}(\mu); \mu) = q'[\tilde{u}_N(\mu)](v_N; \mu) \; \forall v_N \in \mathcal{V}_N^{\text{pr}}$. In this thesis, we still use full quadrature to evaluate the output $q(\cdot; \mu)$, as well as to compute $z_N^{\text{du}}(\mu)$ for DWR error estimation. However, we may compute both of these using $\tilde{u}_N(\mu)$.

## 2.6 Geometry Transformation

The focus of this thesis is on the application of ROM and the EQP to many-query problems. So far, we have presented these methods without special consideration for the nature of the parameter $\mu$. In ROM literature, special attention has been paid to linear problems with affine parameter dependence, as these permit the use of a RB method without the need for hyperreduction. However, because of hyperreduction, the method we propose does not require this kind of simple parameter dependence. We demonstrate results for examples with general parameteric dependence, including free-form deformation of the domain $\Omega(\mu)$. This demonstrates that the method can handle virtually any kind of general parametric dependence, and can be used in advanced problems such as aerodynamic shape optimization. This section describes the type of transformations considered and how they interact with the HFM and the ROM.

### 2.6.1 Generalized Geometry Transformation Definition

We deform the mesh rather than generate a new topologically different mesh for each new geometry. This avoids two expensive steps at each iteration: mesh generation, and transfer of both previous high-fidelity solutions and RB functions into the new mesh. It also guarantees that the optimization objective function and constraints are continuous, provided that the geometry transformation is continuous and that the corresponding changes to the PDE solution are continuous. Topological changes to the mesh would cause discrete jumps in the objective function value, causing additional challenges for optimization.

The free-form deformation (FFD) technique proposed by [55] provides a continuous geometry mapping, making it appealing for use in gradient-based shape optimization. In this technique, the geometry is placed inside a lattice of control points. An example showing a control lattice and deformed geometry in 2D is shown in Figure 2.2. The parameters represent the $x$-, $y$-, and $z$-displacements of the control points. Some of these values may be excluded from the parameter

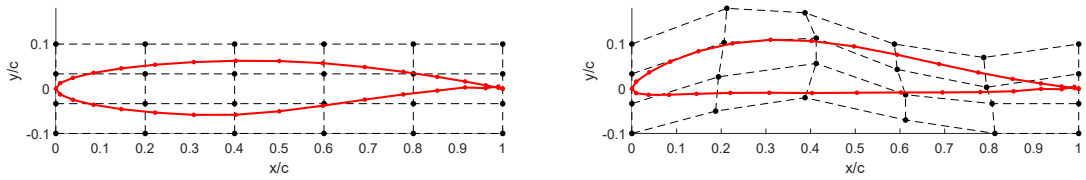



Figure 2.2: RAE2822 airfoil, undeformed and subject to a Bézier curve-based FFD transformation, with control lattice shown

space, meaning that control point is fixed in that axis. From here, there are several possible ways of transforming the geometry, the details of which are explained in Appendix B.

While the exact geometry transformation scheme used does not affect our FEM and ROM formulations and can be treated as a "black box," here we define the interface needed for that "black box." The transform operator $\mathcal{T}$ maps from the original, untransformed domain $\Omega_0$ to $\Omega(\mu)$ so that $x(\mu) = \mathcal{T}(x_0; \mu)$. The transformation is thus formally described as $\mathcal{T}(\cdot; \mu) : \Omega_0 \to \Omega(\mu)$. However, the transformation operator must provide three other crucial pieces of information. The tensor field $J_\mathcal{T}(\mu) : \Omega_0 \to \mathbb{R}^{d\times d}$ is the transformation Jacobian, where $J_\mathcal{T}(\mu)_{ij} = \frac{\partial \mathcal{T}(\cdot,\mu)_i}{\partial x_{0,j}}$, $i, j = 1, ..., d$. We also require the sensitivity of both the points and the Jacobian to the parameter, $\frac{\partial x}{\partial \mu} : \Omega_0 \to \mathbb{R}^{d\times n_p}$ and $\frac{\partial J_\mathcal{T}}{\partial \mu} : \Omega_0 \to \mathbb{R}^{d\times d\times n_p}$.

### 2.6.2 Integration of FFD with FEM

The transform Jacobian $J_\mathcal{T}$ (dropping the "$(\mu)$" for brevity) is used to describe both the change in basis function derivatives, infinitesimal volume $dV$ and edge length $dS$ (used in integration), and wall normal $\hat{n}$ as follows:

$$\begin{aligned}
\frac{\partial \phi_i}{\partial x} &= J_\mathcal{T}^{-T} \frac{\partial \phi_i}{\partial x_0}, \ i = 1, ..., \mathcal{N} \\
dV &= \det(J_\mathcal{T}) dV_0, \\
dS &= J_\delta dS_0, \text{ where } J_\delta \equiv \| \det(J_\mathcal{T}) J_\mathcal{T}^{-T} \hat{n}_0 \|_2, \\
\hat{n} &= \frac{\det(J_\mathcal{T}) J_\mathcal{T}^{-T} \hat{n}_0}{J_\delta}.
\end{aligned} \tag{2.27}$$

In the RB context, $\{\zeta_{N,i}^{\text{pr}}\}_{i=1}^{N_{\text{pr}}}$ is substituted for $\{\phi_i\}_{i=1}^{\mathcal{N}}$ in (2.27). Here $J_\delta$ is the change in length of an infinitesimal line segment along the boundary. We also compute the sensitivity of $J_\delta$ and $\hat{n}$ with

respect to the parameter $\mu$:

$$\begin{aligned}\frac{\partial J_\delta}{\partial \mu} &= \frac{1}{J_\delta}\hat{n}^T\bigg(\frac{\partial\big(\det(J_\mathcal{T})\big)}{\partial\mu}J_\mathcal{T}^{-T}\hat{n}_0 + \det(J_\mathcal{T})\frac{\partial\big(J_\mathcal{T}^{-T}\big)}{\partial\mu}\hat{n}_0\bigg),\\ \frac{\partial \hat{n}}{\partial \mu} &= \frac{1}{J_\delta}\bigg(\frac{\partial\big(\det(J_\mathcal{T})\big)}{\partial\mu}J_\mathcal{T}^{-T}\hat{n}_0 + \det(J_\mathcal{T})\frac{\partial\big(J_\mathcal{T}^{-T}\big)}{\partial\mu}\hat{n}_0 - \frac{\partial J_\delta}{\partial\mu}\hat{n}\bigg).\end{aligned} \tag{2.28}$$

We are primarily interested here in how the FFD transform affects the residual (2.13), the sensitivity of the residual to the control parameters, $\partial_\mu r_h(u,v;\mu)$. Since all the terms in (2.27) depend on the Jacobian, the change in the residual due to the transformation $\partial_\mu r_h(u,v;\mu)$ depends heavily on $\frac{\partial J_\mathcal{T}}{\partial\mu}$. The result is that we can algebraically compute the sensitivity of the residual to the parameter, through many applications of the chain rule.

Following [53, 59], we use a map-then-discretize approach, in which state variables such as $w, v : \Omega(\mu) \to \mathbb{R}^{n_k}$ are mapped onto $\Omega_0$ and designated $\check{w}, \check{v} : \Omega_0 \to \mathbb{R}^{n_k}$. In other words, $w = \check{w}\circ\mathcal{T}^{-1}(\cdot,\mu)$. Note that $w_h$ and $\check{w}_h$ have the same FEM coefficients, and $w_N$ and $\check{w}_N$ have the same RB coefficients. But the spatial gradients must be modified by $J_\mathcal{T}^{-T}$. In other words, $\nabla w = J_\mathcal{T}^{-T}\check{\nabla}\check{w}$. With this in mind, we repeat (2.4) but integrating over the reference domain:

$$\begin{aligned}r_h(w,v;\mu) = &\int_{\partial\Omega_0}\check{v}\big(F(\check{w};\mu) + G(\check{w}, J_\mathcal{T}^{-T}\check{\nabla}\check{w};\mu)\big)\cdot\hat{n}(\mu)\ J_\delta dS_0\\ &+\int_{\Omega_0} -\bigg(J_\mathcal{T}^{-T}\check{\nabla}\check{v}\cdot\big(F(\check{w};\mu) + G(\check{w}, J_\mathcal{T}^{-T}\check{\nabla}\check{w};\mu)\big) - \check{v}S(\check{w}, J_\mathcal{T}^{-T}\check{\nabla}\check{w};\mu)\bigg)\ \det(J_\mathcal{T})dV_0.\end{aligned} \tag{2.29}$$

Note that $\hat{n}$ is still the transformed normal, so we write it as $\hat{n}(\mu)$ to avoid confusion. Then differentiating with respect to $\mu$,

$$\begin{aligned}\partial_\mu r_h(w,v;\mu) = &\int_{\partial\Omega_0}\bigg[\check{v}\bigg(\frac{\partial F(\check{w};\mu)}{\partial\mu} + \frac{\partial G(\check{w}, J_\mathcal{T}^{-T}\check{\nabla}\check{w};\mu)}{\partial\mu} + \frac{\partial G(\check{w}, J_\mathcal{T}^{-T}\check{\nabla}\check{w};\mu)}{\partial(J_\mathcal{T}^{-T}\check{\nabla}\check{w})}\frac{\partial\big(J_\mathcal{T}^{-T}\big)}{\partial\mu}\check{\nabla}\check{w}\bigg)\cdot\hat{n}(\mu)\\ &\quad+\check{v}\big(F(\check{w};\mu) + G(\check{w}, J_\mathcal{T}^{-T}\check{\nabla}\check{w};\mu)\big)\cdot\frac{\partial\hat{n}}{\partial\mu}\bigg]J_\delta\\ &\quad+\check{v}\big(F(\check{w};\mu) + G(\check{w}, J_\mathcal{T}^{-T}\check{\nabla}\check{w};\mu)\big)\cdot\hat{n}(\mu)\frac{\partial J_\delta}{\partial\mu}\ dS_0\\ &+\int_{\Omega_0} -\bigg[\frac{\partial\big(J_\mathcal{T}^{-T}\big)}{\partial\mu}\check{\nabla}v\cdot\big(F(\check{w};\mu) + G(\check{w}, J_\mathcal{T}^{-T}\check{\nabla}\check{w};\mu)\big)\\ &\quad+J_\mathcal{T}^{-T}\check{\nabla}\check{v}\cdot\bigg(\frac{\partial F(u;\mu)}{\partial\mu} + \frac{\partial G(\check{w}, J_\mathcal{T}^{-T}\check{\nabla}\check{w};\mu)}{\partial\mu} + \frac{\partial G(\check{w}, J_\mathcal{T}^{-T}\check{\nabla}\check{w};\mu)}{\partial(J_\mathcal{T}^{-T}\check{\nabla}\check{w})}\frac{\partial\big(J_\mathcal{T}^{-T}\big)}{\partial\mu}\check{\nabla}\check{w}\bigg)\\ &\quad-\check{v}\bigg(\frac{\partial S(\check{w}, J_\mathcal{T}^{-T}\check{\nabla}\check{w};\mu)}{\partial\mu} + \frac{\partial S(\check{w}, J_\mathcal{T}^{-T}\check{\nabla}\check{w};\mu)}{\partial(J_\mathcal{T}^{-T}\check{\nabla}\check{w})}\frac{\partial\big(J_\mathcal{T}^{-T}\big)}{\partial\mu}\check{\nabla}\check{w}\bigg)\bigg]\det(J_\mathcal{T})\\ &\quad-\bigg(\check{\nabla}\check{v}\cdot\big(F(\check{w};\mu) + G(\check{w}, J_\mathcal{T}^{-T}\check{\nabla}\check{w};\mu)\big) - \check{v}S(\check{w}, J_\mathcal{T}^{-T}\check{\nabla}\check{w};\mu)\bigg)\frac{\partial\big(\det(J_\mathcal{T})\big)}{\partial\mu}\ dV_0.\end{aligned} \tag{2.30}$$

There are many terms in (2.30), but broadly speaking they reflect the four ways in which $\mu$ affects the residual.

- If any flux or source terms $F$, $G$, or $S$ are *directly parameter-dependent*, it is straightforward to see how they affect the residual. This includes terms which are coordinate-dependent, which would then also involve $\frac{\partial x}{\partial \mu}$.
- The transform changes function gradients, reflecting the *change in shape* of the domain. This affects both the gradient-dependent flux and source terms and the inner product with $\nabla v$, the test function derivatives.
- The transform changes the $dS$ and $dV$ terms in the integral in (2.4), reflecting the *change in size* of the domain.
- The transform also results in a *change in direction* of the outward normal of surfaces. Often boundary source and flux terms also depend on the normal.

An example of a coordinate-dependent flux term, which fits into the first category above, arises when we consider the RANS equations with the Spalart-Allmaras turbulence model [58]. This model has a source term $S$ dependent on the distance to the nearest wall. Clearly this would be affected not only by the position of the point of interest, but also by the position of the wall, as well as the change in the solution to the minimization problem of finding the nearest wall point. Of these, we neglect the last term, as we anticipate its contribution to the overall residual sensitivity to be small. But the other two terms, associated with the movement of the point of interest and the supposed nearest point on the wall, are computed using $\frac{\partial x}{\partial \mu}$. Direct dependence can also include non-geometric parameters such as material properties – these are the only terms used in our first test case in Section 4.1.

In some cases, it must be determined whether a term "follows" the deformed geometry. For example, if a boundary term arises from ambient heat flux (and the Biot number is assumed to stay constant), then perhaps the fourth class of terms, sensitivity to change in direction, does not arise. However, in other cases, the source term does not move, or only moves in one direction. For example, the point of application of a gravity-induced load on a structural part might change with the geometry, but the direction would always point down, meaning the angle relative to the deformed geometry changes.

The way $J_{\mathcal{T}}^{-T}$ depends on $\mu$ and then goes on to affect the gradient of the state means the residual does not permit affine decomposition. It cannot be written in the form of (2.24). This

complex parameter dependence necessitates the use of hyperreduction in order to fully detatch online computational cost from the size of the mesh. By demonstrating the use of our method on geometric transformations, we hope to emphasize the generality of our method to any continuous and differentiable parametric manifold.

## 2.7 Optimization Methods

ROM is most useful in "many-query" scenarios, when we wish to solve a PDE under a range of similar conditions. This encompasses a broad range of problems, but in this thesis we focus on one subset of those applications: optimization. Optimization is the process of minimizing (or maximizing, but the two can be seen as equivalent) some objective function, and is usually performed by evaluating it many times with different parameter values and then deciding on an optimal solution. In the language of optimization, we speak of "function evaluations" to refer to evaluations of the objective function. In the context of interest here, a function evaluation requires approximating the solution to a PDE using the FEM and thus is computationally costly.

Deterministic gradient-based optimization algorithms are widely known to surpass stochastic algorithms in problems such as ours where the objective function is presumed to be relatively smooth, but function evaluations are cost-intensive [46]. Our intention is to use a hyperreduced ROM (HROM), which will be progressively updated on-the-fly, as a surrogate for the HFM. This suggests the use of trust-region (TR) methods. In the TR context, then, "function evaluations" will refer to HROM solves and "model updates" will refer to the process of solving the HFM and updating the HROM based on the newly acquired HFM data. The majority of the cost comes from model updates. The HROM objective function is cheap to evaluate, but interspersed throughout the HROM solves are model updates requiring solution of the HFM.

In general, we wish to solve the constrained optimization problem:

$$\begin{aligned} \mu_{\text{opt}} &= \underset{\mu \in \mathbb{R}^{n_p}}{\arg\min}\, f(\mu) \\ \text{subject to } & c_E(\mu) = 0, \\ & c_I(\mu) \geq 0, \end{aligned} \tag{2.31}$$

where $f : \mathcal{D} \to \mathbb{R}$ is the objective function, $\mathcal{D}$ being the subset of $\mathbb{R}^{n_p}$ that satisfies the constraints. The equality and inequality constraints are given by $c_E : \mathbb{R}^{n_p} \to \mathbb{R}^l$ and $c_I : \mathbb{R}^{n_p} \to \mathbb{R}^m$, where $l$ and $m$ are the numbers of equality and inequality constraints, respectively.

The two broad classes of methods for enforcing inequality constraints are active set methods and interior point (IP) methods [46]. Active set methods work by traversing the edges of the parameter domain, whereas IP methods converge to the optimal solution following the so-called "central path." This roughly minimizes the distance travelled, and therefore the area of the parameter domain where the surrogate needs to model the underlying objective function well. The fact that we are trying to minimize model updates motivates the use of an optimization algorithm that requires *minimal exploration of the parameter domain.* In other words, an algorithm that uses few function evaluations but takes large steps is not as good as an algorithm that uses more function evaluations but travels a shorter total distance to reach the optimal point. (We know that the accuracy of the HROM diminishes as we move away from training points.) This suggests the use of an IP method, described in detail in Chapter 19 of [46]. In this section, we first introduce the IP method we use and then demonstrate how it fits into the general setting of TR methods.

### 2.7.1 Interior Point Method

The IP method used in this thesis involves the application of a quasi-Newton method with BFGS-style Hessian updates to an objective function with a barrier term that ensures the inequality constraints are satisfied [46]. The solution to (2.31) is approximated as follows:

$$\begin{aligned} \{\mu_{\text{opt}}, s_{\text{opt}}\} = \underset{\mu \in \mathbb{R}^{n_p}, s \in \mathbb{R}^m_{>0}}{\arg\min} \; & f(\mu) - \vartheta \sum_{i=1}^{m} \log s_i, \\ \text{subject to } & c_E(\mu) = 0, \\ & c_I(\mu) - s = 0, \end{aligned} \tag{2.32}$$

where the scalar $\vartheta \in \mathbb{R}_{>0}$ is referred to as the "barrier parameter" and $s \in \mathbb{R}^m_{>0}$ is a set of slack variables associated with the inequality constraints. In the limit of $\vartheta \to 0$, (2.32) yields the solution to (2.31). In optimization literature, the barrier parameter is often referred to as $\mu$ and the parameters as $x$, but for notational consistency, we continue to use $\mu \in \mathbb{R}^{n_p}$ to represent the parameter and $x \in \mathbb{R}^d$ to represent the spatial variable. Whenever the constraints are close to being violated, $s$ is close to 0, and $-\log s_i$ grows rapidly. At $\vartheta = 0$, the barrier function would be discontinuous; however, for finite values of $\vartheta$ it is continuous. The use of a barrier function avoids the combinatorial aspect associated with active set methods.

We define a Lagrangian:

$$\mathcal{L}(\mu) \equiv f(\mu) - c_E^T(\mu)y - c_I^T(\mu)z, \tag{2.33}$$

where $y \in \mathbb{R}^l$ and $z \in \mathbb{R}^m$ are Lagrange multipliers associated with $c_E(\mu)$ and $c_I(\mu)$ respectively. The Karush-Kuhn-Tucker (KKT) optimality conditions associated with this problem are

$$\begin{aligned} \nabla\mathcal{L}(\mu) = \nabla f(\mu) - A_E^T(\mu)y - A_I^T(\mu)z &= 0, \\ -\frac{\vartheta e}{s} + z &= 0, \\ c_E(\mu) &= 0, \\ c_I(\mu) - s &= 0. \end{aligned} \tag{2.34}$$

Assume the fraction operator is performed element-wise and $e \in \mathbb{R}^m$ is a vector full of 1's. The Jacobians of the constraint functions are indicated by $A_E \equiv \frac{\partial c_E}{\partial \mu}$ and $A_I \equiv \frac{\partial c_I}{\partial \mu}$. Throughout this section we also use $\nabla(\cdot) \equiv \frac{\partial(\cdot)}{\partial \mu}$ and $\nabla^2(\cdot) \equiv \frac{\partial^2(\cdot)}{\partial \mu^2}$.

As described in [46], Newton's method applied to (2.34) can be decoupled and simplified to solving the linear system:

$$\begin{aligned} \begin{bmatrix} \nabla^2\mathcal{L}(\mu) + A_I^T(\mu)\mathrm{diag}(\frac{s}{z})A_I(\mu) & A_E^T(\mu) \\ A_E(\mu) & 0 \end{bmatrix} \begin{bmatrix} p_\mu \\ p_y \end{bmatrix} &= \begin{bmatrix} -\nabla\mathcal{L}(\mu) - A_I^T(\mu)(c_I(\mu) \odot \frac{z - \vartheta e}{s}) \\ -c_E(\mu) \end{bmatrix}, \\ p_z &= \left( -c_I(\mu) + \frac{\vartheta e}{z} - A_I(\mu)p_\mu \right) \odot \frac{z}{s}, \\ p_s &= -s + \frac{\vartheta e - s \odot p_z}{z}. \end{aligned} \tag{2.35}$$

Note that $\odot$ denotes the Hadamard (element-wise) product. The Hessian $\nabla^2\mathcal{L}(\mu) \in \mathbb{R}^{n_p \times n_p}$ is approximated by $B_k$ using the damped BFGS strategy in Chapter 18 of [46] (see Algorithm 12 in Appendix C). The subscript $k$ indicates that this estimate is updated at each step of the IP algorithm.

We then use the computed steps to perform a line search, where we take proportionally equal steps in $\mu$ and $s$, and proportionally equal steps in $y$ and $z$, according to the computed step directions $p_\mu$, $p_s$, $p_y$, and $p_z$. First we compute the maximum allowable steps using a fraction to the boundary

rule:

$$\begin{aligned} \alpha_s^{\max} &= \max\{\alpha \in (0,1] : s + \alpha p_s \geq 0.005s\}, \\ \alpha_z^{\max} &= \max\{\alpha \in (0,1] : z + \alpha p_z \geq 0.005z\}. \end{aligned} \tag{2.36}$$

We then set $\alpha_s = \alpha_s^{\max}$ and $\alpha_z = \alpha_z^{\max}$ and perform a backtracking line search subject to the Armijo conditions (Algorithm 1), with a merit function of the form:

$$\varphi(\mu) \equiv \begin{cases} \nu\|c_E(\mu)\|_1 - \|z\|_\infty \sum_{i=1}^m \min\{0, c_{I,i}(\mu)\}, & \min\{c_I(\mu)\} < 0, \\ \nu\|c_E(\mu)\|_1 + f(\mu) - \vartheta \sum_{i=1}^m \log s_i, & \min\{c_I(\mu)\} \geq 0. \end{cases} \tag{2.37}$$

The first expression only applies at the beginning of the algorithm if the initial guess is infeasible, and drives the algorithm towards the feasible region. Once inside the feasible region, the barrier function is applied, which prevents the algorithm from ever leaving the feasible region again. The equality constraints are loosely enforced by a simple penalty function with a penalty weight $\nu$ which is computed according to the rule:

$$\nu = \left| \frac{\nabla f \cdot p_\mu}{0.8\|c_E(\mu)\|_\infty} \right|. \tag{2.38}$$

**Algorithm 1:** Armijo Line Search

**Data:** $\alpha_s^{\max}, \alpha_z^{\max}, p_\mu, p_s, p_y, p_z, \mu, s, y, z, f(\mu), \nabla f(\mu)$ (current iterate)
**Result:** $\mu^+, s^+, y^+, z^+, f(\mu^+), \nabla f(\mu^+)$ (next iterate)

$\gamma \leftarrow 0.6$ (backtracking factor), $\beta \leftarrow 0.01$ (sufficient decrease factor)
$\alpha_s \leftarrow \alpha_s^{\max}$, $\alpha_z \leftarrow \alpha_z^{\max}$
$\nu \leftarrow \left| \frac{\nabla f \cdot p_\mu}{0.8\|c_E(\mu)\|_\infty} \right|$
$\varphi_{\text{Armijo}} \leftarrow \varphi(\mu) + \beta \nabla\varphi(\mu) \cdot p_\mu$, computed from $f(\mu)$ and $\nabla f(\mu)$ using (2.37)
$\varphi(\mu^+) \leftarrow \varphi_{\text{Armijo}}$ (to force start)
**while** $\varphi(\mu^+) \geq \varphi_{Armijo}$ **do**
    $\mu^+ \leftarrow \mu + \alpha_s p_\mu$, $s^+ \leftarrow s + \alpha_s p_s$
    $\alpha_s \leftarrow \gamma\alpha_s$, $\alpha_z \leftarrow \gamma\alpha_z$
    Compute constraints $c_I(\mu^+)$ and $c_E(\mu^+)$ and evaluate feasibility.
    **if** $\mu^+$ *is not feasible* **then**
        (Abort linesearch iteration before evaluating objective function.)
        **continue**
    **end**
    Compute objective function $f(\mu^+)$ and merit function $\varphi(\mu^+)$.
**end**
Compute objective function gradient $\nabla f(\mu^+)$.
$y^+ \leftarrow y + \alpha_z p_y$, $z^+ \leftarrow z + \alpha_z p_z$
Reset the slack variable, as current $s^+$ is just an estimate: $s^+ \leftarrow \max(c_I(\mu), 0)$

In practice, we terminate the line search after a finite number of iterations (usually around 10-20),

as failure to converge is a signal that either $p_\mu$ is not a direction of descent or the entire optimization algorithm has converged and only extremely small steps can result in further improvement. However, this is very rare. Since BFGS Hessian updates ensure that the Hessian is positive definite, the step direction computed using (2.35) will always be a valid descent direction if $\nabla f$ is computed exactly. The IP algorithm is equipped with its own convergence conditions which should be satisfied before the objective function becomes sufficiently "flat" for a line search to fail.

The Lagrange multipliers are initialized using a least squares approach, which solves the following least squares algebraic problem:

$$\begin{bmatrix} A_E(\mu) & 0 \\ A_I(\mu) & -\text{diag}(s) \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} \nabla f(\mu) \\ -\vartheta e \end{bmatrix}. \tag{2.39}$$

We then correct $z$ in case the formula above yields negative values for $z$ by setting

$$z_i \leftarrow \max(|\vartheta/s_i|, z_i), \quad i = 1, ..., m. \tag{2.40}$$

There are many methods to update the barrier parameter $\vartheta$. The one observed to perform best in practice is based on (19.19) and (19.20) in [46] and is as follows:

$$\begin{aligned} \vartheta_{k+1} &= \sigma_k \frac{s_k^T z_k}{m}, \\ \sigma_k &= 0.1 \min\left(0.05 \frac{1-\xi_k}{\xi_k}, 2\right), \\ \xi_k &= \frac{\min_i [s_k]_i [z_k]_i}{s_k^T z_k / m}. \end{aligned} \tag{2.41}$$

We now summarize the entire IP optimization algorithm in Algorithm 2. The algorithm is run until the first-order optimality $\tau_{\text{opt}}$ is less than a tolerance $\epsilon_{\text{opt}}$.

### 2.7.2 Trust Region Method

In the trust region (TR) method, we solve a series of successive optimization sub-problems, restricting the domain of interest to a small region, a trust region. The general approach to TR-based optimization is covered in Chapter 4 of [46]. For the $k$th sub-problem, we replace $f(\mu)$ with a surrogate $\tilde{f}_k(\mu)$ (which is much cheaper to evaluate). When one sub-problem terminates, the true function $f(\mu)$ is evaluated, the surrogate model is updated, and that point becomes the center of the next TR. We may solve the sub-problem with any optimization method of our choice, so we use the IP method. The formal definition of the $k$th sub-problem is

**Algorithm 2:** Interior Point Method

**Data:** Initial guess $\mu_0$, $n_{\max}$, $\epsilon_{\text{opt}}$, `warm_start`, and optionally $f(\mu_0)$, $\nabla f(\mu_0)$, $B_0$
**Result:** $\mu_{\text{opt}}$, $f(\mu)_{\text{opt}}$, $\tau_{\text{opt}}$

```
if NOT(warm_start) then
    Evaluate f(μ_0), ∇f(μ_0).
    Initialize Lagrange multipliers (y, z) according to (2.39).
    B_0 ← I
end
k ← 0
for k = 1, ..., n_max do
    Compute step direction (p_μ, p_s, p_y, p_z) by solving (2.35).
    Compute the maximum steps α_s^max and α_z^max using (2.36).
    Perform a line search (Algorithm 1) to obtain μ_k.
    τ_k ← ‖∇L(μ)‖_∞
    if τ_k ≤ ε_opt then
        break
    end
    Update the estimate of the Hessian B_k (Algorithm 12 in Appendix C).
    Update the value of the barrier parameter ϑ using (2.41).
end
τ_opt ← τ_k
μ_opt ← μ_k
```

$$\begin{aligned}\{\mu_{\text{opt},k}, s_{\text{opt},k}\} = \underset{\mu\in\mathcal{D}_{\text{TR},k}, s\in\mathbb{R}^m_{>0}}{\arg\min}\ & \tilde{f}_k(\mu) - \mu\sum_{i=1}^{m}\log s_i, \\ \text{subject to } & c_E(\mu) = 0, \\ & c_I(\mu) - s = 0,\end{aligned} \tag{2.42}$$

where $\mathcal{D}_{\text{TR},k} \subset \mathbb{R}^{n_p}$ is the trust region associated with the $k$th iterate. The trust region is defined as the region where a certain metric $\theta$ (for example, the Euclidean distance to the TR center) is less than a certain value $\Delta_k$ (the TR size); e.g., $\mathcal{D}_{\text{TR},k} \equiv \{\mu \in \mathbb{R}^{n_p} \mid \theta(\mu) \leq \Delta_k\}$. The TR size is initialized by the user, but adapts based on how accurate it measures the surrogate model to be in the previous sub-problem. This adaptive adjustment of the TR size minimizes dependency on the initial TR size. The TR size for the next optimization sub-problem is then adapted as follows:

$$\Delta_{k+1} = \begin{cases} 0.25\Delta_k & \text{if } \varrho_k < 0.25, \\ 2.0\Delta_k & \text{if } \varrho_k > 0.75, \end{cases} \tag{2.43}$$

where $\varrho$ is the actual reduction over the expected reduction, in terms of the merit function:

$$\varrho_k \equiv \frac{\varphi(\mu_k) - \varphi(\mu_{\text{opt},k})}{\varphi(\mu_k) - \tilde{\varphi}(\mu_{\text{opt},k})}. \tag{2.44}$$

**Algorithm 3:** Trust Region Method

**Data:** Initial guess $\mu_0$, $n_{\text{TRmax}}$, $\epsilon_{\text{opt}}$, $\Delta_0$
**Result:** $\mu_{\text{opt}}$, $f(\mu)_{\text{opt}}$, $\tau_{\text{opt}}$
1 Evaluate $f(\mu_0)$, $\nabla f(\mu_0)$.
2 Initialize Lagrange Multipliers according to (2.39).
3 $B_0 \leftarrow I$
4 $k \leftarrow 0$
5 Construct the surrogate model $\tilde{f}_0$.
6 **for** $k = 1, ..., n_{TRmax}$ **do**
7 Solve the sub-problem (2.42) using Algorithm 2 with $f(\mu_k)$, $\nabla f(\mu_k)$, and $B_k$ for a warm start. Terminate if $\theta(\mu_{\text{opt},k}) > 0.9\Delta_k$. Returns $\mu_{\text{opt},k}$.
8 Compute $f(\mu_{\text{opt},k})$, $\nabla f(\mu_{\text{opt},k})$, and $\nabla \mathcal{L}(\mu_{\text{opt},k})$.
9 $\tau_k \leftarrow \|\nabla \mathcal{L}(\mu)\|_\infty$
10 **if** $\tau_k \leq \epsilon_{opt}$ **then**
11 **break**
12 **end**
13 Compute $\varrho_k$ using (2.44) and compute $\Delta_{k+1}$ using (2.43).
14 **if** $\varrho_k > 0$ **then**
15 $\mu_{k+1} \leftarrow \mu_{\text{opt},k}$
16 Compute $B_{k+1}$ using Algorithm 12.
17 **else**
18 $\mu_{k+1} \leftarrow \mu_k$ and $B_{k+1} \leftarrow B_k$
19 **end**
20 Construct the new surrogate model $\tilde{f}_{k+1}$.
21 $k \leftarrow k + 1$
22 **end**
23 $\tau_{\text{opt}} \leftarrow \tau_k$
24 $\mu_{\text{opt}} \leftarrow \mu_k$

The merit function is roughly equivalent to the objective function. See (2.37), and substitute $\tilde{f}_k(\mu)$ instead of $f(\mu)$ to obtain $\tilde{\varphi}(\mu)$.

The sub-problem need not be solved exactly; that is, we do not need to treat the TR as a constraint the way we treat the other problem constraints, with a slack variable. This would involve traversing the TR boundary. Instead, we can choose to terminate the sub-problem when $\theta$ is within 90% of $\Delta_k$. Line search backtracking must also backtrack until the TR constraint is satisfied.

Algorithm 3 summarizes the TR approach. Note Line 14, in which the algorithm conducts a final check on the value of $\varrho$. If $\varrho < 0$, this indicates that the merit function actually increased due to inaccuracies in $\tilde{f}_k$. In this case, the new suggested TR center $\mu_{\text{opt},k}$ is rejected, the surrogate model is reconstructed at $\mu_k$, and the TR size is reduced.

The Hessian approximation is reset to the approximation at the end of the previous sub-problem. During the sub-problem, Hessian updates would be performed, but this would approximate the Hessian of the surrogate model. The initial guess for the Hessian at each sub-problem does not use the Hessian estimate at the end of the previous sub-problem, but a separate outer BFGS scheme

considering each sub-problem solve as a step is being performed here in Algorithm 3. This is because if the sub-problem did indeed converge (rather than terminate due to the TR constraint), then very small steps might have led to poor conditioning in BFGS updates, and the Hessian estimate would therefore be a bad initial guess for the next sub-problem.

In Chapter 4 of [46], it is assumed that the surrogate model will be a second-order polynomial approximation of the objective function. However, this is not necessary. Any general surrogate model used in trust region methods is guaranteed to converge to a local minimum of the underlying HFM if the following statements hold:

- Zero-order consistency (ZOC):

$$f(\mu_k) = \tilde{f}_k(\mu_k), \tag{2.45}$$

- First-order consistency (FOC):

$$\nabla f(\mu_k) = \nabla \tilde{f}_k(\mu_k), \tag{2.46}$$

- and a sufficient decrease condition on each sub-problem:

$$\tilde{f}_k(\mu_{\text{opt},k}) \leq \tilde{f}_k(\mu_k) + \beta \nabla \tilde{f}_k(\mu_k) \cdot (\mu_{\text{opt},k} - \mu_k). \tag{2.47}$$

A convergence guarantee with these conditions is proven in [2]. The third condition is guaranteed already by the use of the sufficient decrease condition during Armijo line searches in the solution of the sub-problem. Therefore, the ZOC and FOC conditions (2.45) and (2.46) are the only conditions we require of the surrogate model.

# Chapter 3

# Integrated Framework

The purpose of this chapter is to develop an integrated optimization framework that builds on the ingredients introduced in Chapter 2. We start by formally stating the problem of interest, combining notation from Sections 2.3 and 2.7. We wish to solve the following PDE-constrained optimization problem: find $\mu_{\text{opt}}$ such that

$$\begin{aligned} \mu_{\text{opt}} &= \arg\min_{\mu \in \mathcal{D}} C(q(u_h(\mu);\mu);\mu), \\ \text{where } u_h(\mu) \text{ satisfies} \quad & r_h(u_h(\mu), v_h; \mu) = 0 \quad \forall v_h \in \mathcal{V}_h. \end{aligned} \tag{3.1}$$

Constraints of the form discussed in Section 2.7 are used to define $\mathcal{D}$. The output $q(u_h(\mu);\mu)$ may be submitted to other low-cost operations before becoming the optimization cost function $C(q(u_h(\mu);\mu);\mu)$. We often abbreviate this to simply $C(\mu)$. But note that "evaluating $C(\mu)$" involves solving for $u_h(\mu)$, computing $q(u_h(\mu);\mu)$, and then computing $C(q(u(\mu);\mu);\mu)$. Solving for the state is assumed to be by far the largest source of computational expense in evaluating $C(\mu)$. We shall also refer to $\tilde{C}(\mu) \equiv C(q(\tilde{u}_N(\mu);\mu);\mu)$, the HROM surrogate for $C(\mu)$.

Although we use the surrogate model to accelerate the optimization process, we ultimately seek the solution to the HFM-based problem (3.1). Thus in Sections 3.1 and 3.2, we focus on ensuring that the HROM satisfies the ZOC and FOC conditions from Section 2.7.2. This informs the selection of RB functions and EQP training constraints. Then in Section 3.3 we discuss how the DWR error estimator can be used to define a TR. Lastly, in Section 3.4 the entire HROM-accelerated PDE-constrained optimization framework is presented in algorithmic form.

## 3.1 Reduced Basis Selection

The RB functions are generated by analyzing solution data from a collection of high-fidelity solutions for a set of training parameters $\Xi_{\text{train}} \equiv \{\mu_i\}_{i=1}^{n_{\text{train}}} \subset \mathcal{D}$. We solve the HFM for all $\mu \in \Xi_{\text{train}}$ to construct a set of solution snapshots $U_{\text{train}} \equiv \{u_h(\mu) \mid \mu \in \Xi_{\text{train}}\}$. We then wish to construct a space $\mathcal{V}_N^{\text{pr}}$ such that $\mathcal{V}_N^{\text{pr}} \equiv \text{span}\{\zeta_i^{\text{pr}}; i = 1, ..., N_{\text{pr}}\} \approx \text{span}(U_{\text{train}})$.

We know that for a given training point $\mu_i$ if $u_h(\mu_i)$ exists in the span of $\mathcal{V}_N^{\text{pr}}$, then $r_h(u_h(\mu_i), v; \mu_i) = 0 \ \forall v \in \mathcal{V}_N^{\text{pr}}$, since $\mathcal{V}_N^{\text{pr}} \subset \mathcal{V}_h$. This means that unless the problem has multiple roots, solving the ROM problem (2.18) will yield $u_N(\mu_i) = u_h(\mu_i)$ exactly. Since we exactly reproduce the HFM solution, if we then compute the output $q(u_N(\mu_i); \mu_i)$ using full quadrature, then the output will also be identical to $q(u_h(\mu_i); \mu_i)$. Enforcing this at the TR center $\mu_{\text{TR}}$ guarantees that we satisfy the ZOC condition (2.45) needed for optimization convergence.

However, we also wish the ROM surrogate to satisfy the FOC condition (2.46), so that $\tilde{C}(\mu)$ is an acceptable TR surrogate that guarantees convergence to the solution of the underlying HFM-based optimization problem. As shown in (2.21), the output gradient depends on the solution to the dual problem in the primal space, (2.20). This means that if $z_h(\mu_{\text{TR}})$ is included in the span of $\mathcal{V}_N^{\text{pr}}$ and (as we have already guaranteed) $u_N(\mu_{\text{TR}}) = u_h(\mu_{\text{TR}})$, then we would have the same result of exact reproduction: $z_N^{\text{pr}}(\mu_{\text{TR}}) = z_h(\mu_{\text{TR}})$. Combined with the ZOC result already obtained, all terms in (2.21) are now equal, and so $\frac{D}{D\mu}\big(q(u_N(\mu_{\text{TR}}); \mu_{\text{TR}})\big) = \frac{D}{D\mu}\big(q(u_h(\mu_{\text{TR}}); \mu_{\text{TR}})\big)$, satisfying the FOC condition. In summary, we have two requirements of $\mathcal{V}_N^{\text{pr}}$:

- $\mathcal{V}_N^{\text{pr}} \supseteq \text{span}\{u_h(\mu_{\text{TR}}), z_h(\mu_{\text{TR}})\}$ to guarantee exact reproduction at $\mu_{\text{TR}}$, and
- $\mathcal{V}_N^{\text{pr}} \approx \text{span}(U_{\text{train}})$ to promote accuracy for all regions of $\mathcal{D}$ near elements of $\Xi_{\text{train}}$.

Using the sample solutions directly as basis functions would result in very poor conditioning, as the solutions are expected to be similar to one another, and therefore nearly linearly dependent. This motivates the selection of an orthogonal (orthonormal) basis set with an equivalent span [48]. However, many orthogonal basis sets can be chosen to represent a given span. Two orthogonalization techniques are discussed in the following sections: proper orthogonal decomposition (POD) and Gram-Schmidt orthogonalization (GSO). We review these two methods, along with demonstrating how the ZOC and FOC conditions are enforced.

### 3.1.1 Proper Orthogonal Decomposition

Algorithm 4 shows the procedure for generating basis functions from a generic training set using proper orthogonal decomposition (POD). The primary function of performing POD on $U_{\text{train}}$ is in fact compression, and not orthogonalization. The orthogonality of the basis $\{\zeta_i^{\text{pr}}\}_{i=1}^{N_{\text{pr}}}$ is a secondary result of POD. This compressive property makes it ideal in the context of ROM-accelerated optimization, since it identifies the RB set that best approximates $U_{\text{train}}$ with a minimal number of basis functions — hence its use in [1, 51, 70]. The first basis function identified by POD contains the the mean solution across $U_{\text{train}}$, the second basis function contains the next most energetic mode, and so on. For this reason, we may truncate — take the first $N$ basis functions, or simply use all basis functions for maximum accuracy: $N = n_{\text{train}}$ (in which case $\mathcal{V}_N^{\text{pr}} = \text{span}(U_{\text{train}})$ exactly). The eigenvalues of the correlation matrix between all solutions in $U_{\text{train}}$ in Line 3 are each associated with a basis function, and should exhibit exponential decay, indicating diminishing returns in accuracy upon enriching the space $\mathcal{V}_N^{\text{pr}}$.

**Algorithm 4:** Proper Orthogonal Decomposition

**Data:** Solution dataset $D_{\text{train}} \equiv \{v_{h,i}\}_{i=1}^{n_{\text{train}}}$
**Result:** RB set $\{\zeta_i\}_{i=1}^{N}$ and eigenvalues $\{\lambda_i\}_{i=1}^{N}$
1 **for** $i = 1$ **to** $n_{train}$ **do**
2  **for** $j = 1$ **to** $n_{train}$ **do**
3  $C_{i,j} = \frac{1}{n_{\text{train}}}(v_{h,i}, v_{h,j})_\chi$
4  **end**
5 **end**
6 Solve the eigenvalue problem $C\Psi = \Psi\Lambda$, yielding $\Psi \in \mathbb{R}^{n_{\text{train}} \times n_{\text{train}}}$ right eigenvectors and $\Lambda \equiv \text{diag}(\lambda_i)$, a diagonal matrix of eigenvalues.
7 Truncate all eigenvalues and eigenvectors where $\frac{\lambda_i}{\lambda_1} \leq \varepsilon_{\text{ROM}}$.
8 $N \leftarrow$ number of remaining eigenvalues. Now $\Psi \in \mathbb{R}^{n_{\text{train}} \times N}$.
9 **for** $i = 1$ **to** $N$ **do**
10  $\zeta_i \leftarrow 0$
11  **for** $j = 1$ **to** $n_{train}$ **do**
12  $\zeta_i \leftarrow \zeta_i + \frac{1}{n_{\text{train}}}\Psi_{j,i} v_{h,j}$
13  **end**
14 **end**

Any valid inner product can be used for $(\cdot, \cdot)_\chi$. In this work, we use the broken (element-wise) $H_1(\Omega_0)$ inner product, where $\Omega_0$ is the untransformed domain, requiring the solutions to be mapped back to $\Omega_0$ to perform the inner product. (This operation is cost-free, since the FEM coefficients are unchanged.) The choice of inner product used here dictates the norm in which the error is minimized when reproducing the training states using the truncated RB set [48].

In the original works that use the RB method, the training data set was generated first, running

the HFM for a pre-selected $\Xi_{\text{train}}$. However, it is also possible to use a sequential approach, in which we start with an extremely small data set, perhaps even for just one parameter value $\mu_0$ which alone comprises $\Xi_{\text{train}}$, and then incrementally update "on-the-fly" as required. For this, we could use incremental POD, for which we refer to [28, 70]. However, incremental POD is roughly as expensive as simply redoing POD from scratch, unless the cost of the eigenvalue problem dominates. For our case, when $\mathcal{N} \gg n_{\text{train}}$, most of the cost comes from constructing the correlation matrix $C$ and then computing the basis functions at the end.

To guarantee satisfying the FOC condition, we can include $z_h(\mu_{\text{TR}})$, the HFM dual at the current TR center $\mu_{\text{TR}}$, in the dataset that is subjected to Algorithm 4: $U_{\text{train}} \cup z_h(\mu_{\text{TR}})$. Note that $\mu_{\text{TR}} \in \Xi_{\text{train}}$. This makes a maximum of $n_{\text{train}} + 1$ basis functions available.

However, the use of truncation eliminates any guarantee of satisfying both the ZOC condition and the FOC condition. For this reason, we cannot apply truncation in this context. Additionally, including $z_h(\mu_{\text{TR}})$ in the dataset and then subjecting it to POD violates the philosophy of POD, which is to look at dominant modes of solutions which are *known to be similar*. For example, the first mode would be the "average" of $U_{\text{train}} \cup z_h(\mu_{\text{TR}})$, which has no physical meaning. A solution to both of these problems would be to perform POD on the dataset, excluding $u_h(\mu_{\text{TR}})$ and $z_h(\mu_{\text{TR}})$, truncate if desired, and then add these snapshots in afterwards using GSO (see Section 3.1.2 below). However, this approach is not used in this thesis. When we use POD, we simply do not use truncation (i.e., use all $n_{\text{train}} + 1$ basis functions).

The space $\mathcal{V}_N^{\text{du}}$, which is used to represent the dual for the purpose of DWR error estimation, can be constructed by performing POD on the dataset $Z_{\text{train}} \equiv \{z_h(\mu) \mid \mu \in \Xi_{\text{train}}\}$ to obtain the basis set $\{\zeta_i\}_{i=1}^{N_{\text{du}}}$. For $n_{\text{train}} = 1$, this will mean that $N_{\text{pr}} = 2$ and $N_{\text{du}} = 1$, making $\mathcal{V}_N^{\text{du}} \subset \mathcal{V}_N^{\text{pr}}$. There is only one dual solution in the training set and it is included in the span of both spaces. This will mean $\eta_N(\mu)$ will be 0 everywhere in $\mathcal{D}$, because $r_h(u_N(\mu), \zeta_i^{\text{pr}}; \mu) = 0, \forall \zeta_i^{\text{pr}} \in \mathcal{V}_N^{\text{pr}} \supset \mathcal{V}_N^{\text{du}}$. We simply cannot use $\eta_N(\mu)$ in this case. However, for $n_{\text{train}} > 1$, $\eta_N(\mu)$ will be non-zero and can be used as an error estimator. The error estimator will also improve as the dual space $\mathcal{V}_N^{\text{du}}$ is enriched.

Another effect of including $z_h(\mu_{\text{TR}})$ in the dataset for constructing both $\mathcal{V}_N^{\text{pr}}$ and $\mathcal{V}_N^{\text{du}}$ is that we are guaranteed that $\eta_N(\mu_{\text{TR}}) = 0$. This is good, because we know that the actual error here is also 0, due to the ZOC condition. This property will be referred to in Section 3.3 when we use $\eta_N(\mu)$ to define the TR.

### 3.1.2 Gram-Schmidt Orthogonalization

Gram-Schmidt orthogonalization (GSO) is another approach to obtain an orthogonal basis. Unlike POD, GSO does not provide an optimal compression of training solution snapshots; instead, GSO successively updates orthogonal bases. Here the first basis function represents the first solution (normalized), the second represents whatever in the second solution was orthogonal to the first, and so on. GSO is computationally cheaper than POD, because adding new data does not change the existing basis functions. This approach has been used in ROM-accelerated optimization literature in [32, 70].

Algorithm 5 demonstrates the GSO procedure for augmenting a reduced basis $\{\zeta_i\}_{i=1}^N$ with the solution at a new training point $v_h$.

**Algorithm 5:** Augment Reduced Basis Functions by GSO

**Data:** Existing RB set $\{\zeta_i\}_{i=1}^N$, new solution to add $v_h$
**Result:** Updated RB set, final RB coefficient $c$

$\bar{\zeta}_{N+1} \leftarrow v_h$
**for** $i = 1$ **to** $N$ **do**
    $\bar{\zeta}_{N+1} \leftarrow \bar{\zeta}_{N+1} - (\zeta_i, v_h)_\chi \zeta_i$
**end**
$c \leftarrow \|\bar{\zeta}_{N+1}\|_\chi$
$\zeta_{N+1} \leftarrow \frac{\bar{\zeta}_{N+1}}{c}$
$N \leftarrow N + 1$

The first step is an orthogonalizing step to compute an orthogonal basis function $\bar{\zeta}$, and then we normalize the basis function to obtain $\zeta$. The inner product $(\cdot, \cdot)_\chi$ (and associated norm, $\|\cdot\|_\chi$) used for orthogonalization must be a suitable inner product. We use the broken $H_1(\Omega_0)$ inner product, as in POD. This procedure can be used even with no existing RB set ($N = 0$), in which case the orthogonalizing loop is not entered and the basis function is simply the normalized solution.

Algorithm 5 does not compute the eigenvalue associated with the new basis function, as the data is not "compressed" here, but merely orthogonalized. The final norm $c \equiv \|\zeta_{N+1}\|_\chi$ would normally be expected to decrease as the space is enriched, as the portion of the new solution $u_h$ not already represented by the existing basis functions decreases, given that we are sampling from a presumably limited parametric manifold. Although it is not an eigenvalue, we do use it as a metric for the significance of new information added by the latest snapshot $u_h$, which is why it is stored and returned. As in POD, we can use this metric to decide whether to truncate basis functions that yield no observable improvement to the quality of the ROM, and might result in conditioning problems in the solution of the ROM problem.

As long as truncation is not used, GSO and POD on the same dataset yield exactly the same approximation space $\mathcal{V}_N$, though with different basis functions. As soon as truncation is used, however, the resulting spaces are different. Since each basis function is affected by all solutions in POD, removing any one basis function loses all exact reproduction. With GSO, truncation effectively occurs at the level of the dataset, so we lose exact reproduction of only one solution. As already mentioned, it would be possible to combine POD and GSO in an attempt to get the best of both worlds, but we did not do this for any cases in Chapter 4.

To formally describe the RB generation strategy that preserves FOC through truncation, we introduce the operator $\text{GSO}(\cdot)$ which maps a set of raw HFM data to a set of orthogonal basis functions of the same length. The expression $\{\zeta_i\}_{i=1}^{n_{\text{train}}} = \text{GSO}(\{u_h(\mu_1), ..., u_h(\mu_{n_{\text{train}}})\})$ indicates that $u_h(\mu_1)$ has been normalized following Algorithm 5 to yield $\zeta_1$, the RB set has been augmented by $u_h(\mu_2)$ using Algorithm 5 to yield $\zeta_2$, and so on up to $\zeta_{n_{\text{train}}}$. We emphasize that in this notation, the set of data in the argument of $\text{GSO}(\cdot)$ is an *ordered* set. In other words, $\text{GSO}(\{u_h(\mu_1), u_h(\mu_2), u_h(\mu_3)\}) \neq \text{GSO}(\{u_h(\mu_1), u_h(\mu_3), u_h(\mu_2)\})$, although the span of the resulting RB set would be the same.

It is assumed from this point on that the incremental nature of GSO will be leveraged to compute $\text{GSO}(\{u_h(\mu_1), ..., u_h(\mu_{n_{\text{train}}})\})$ given $\text{GSO}(\{u_h(\mu_1), ..., u_h(\mu_{n_{\text{train}}-i})\})$ for any integer $i > 0$ using $i$ applications of Algorithm 5. Likewise, $\text{GSO}(\{u_h(\mu_1), ..., u_h(\mu_{n_{\text{train}}}), u_h(\mu_B)\})$ can be efficiently computed given $\text{GSO}(\{u_h(\mu_1), ..., u_h(\mu_{n_{\text{train}}}), u_h(\mu_A)\})$ by removing the basis function with index $n_{\text{train}} + 1$ associated with $u_h(\mu_A)$ and then applying Algorithm 5 with $u_h(\mu_B)$. We neglect these implementation details from now on for notational simplicity. With this in mind, the total update strategy is given in Algorithm 6.

The boolean "`rejection`" is true if the new point is *not* the desired FOC point, but the state is still to be used to update the RB functions. This occurs when the TR algorithm decides to reject a point due to a discovered increase in the objective function, when a decrease was predicted (see Line 14 of Algorithm 3). Based on `rejection`, the swap operation in Line 4 of Algorithm 6 ensures that $\mu_{\text{TR}}$, the center of the *next* TR, is always at the end of the training set. This simplifies the notation, but the crucial point is that $u_h(\mu_{\text{TR}})$ and $z_h(\mu_{\text{TR}})$ are included in the construction of $\mathcal{V}_N^{\text{pr}}$.

The truncation operations in Lines 8 and 13 are to avoid conditioning issues associated with too many basis functions. Notationally and indeed conceptually, GSO effects truncation at the level of the dataset, as each basis function is associated with a particular solution. However, in practice, this can also be done directly at the level of basis functions. Since this algorithm only adds one basis function at a time, only one would ever need to be removed. However, removing the basis function

**Algorithm 6:** First-Order Consistent Reduced Basis Update Strategy Using GSO

**Data:** New training point $\mu_{n_{\text{train}}}$, $u_h(\mu_{n_{\text{train}}})$, $z_h(\mu_{n_{\text{train}}})$, existing RB sets $\{\zeta_i^{\text{pr}}\}_{i=1}^{N_{\text{pr}}}$ and $\{\zeta_i^{\text{du}}\}_{i=1}^{N_{\text{du}}}$, existing training sets $\Xi_{\text{train}}^{\text{pr}}$ and $U_{\text{train}}^{\text{pr}} \equiv \{u_h(\mu) \mid \mu \in \Xi_{\text{train}}^{\text{pr}}\}$, $\Xi_{\text{train}}^{\text{du}}$ and $Z_{\text{train}}^{\text{du}} \equiv \{z_h(\mu) \mid \mu \in \Xi_{\text{train}}^{\text{du}}\}$, ROM tolerance $\varepsilon_{\text{ROM}}$, boolean `rejection`

**Result:** Updated $\{\zeta_i^{\text{pr}}\}_{i=1}^{N_{\text{pr}}}$ and $\{\zeta_i^{\text{du}}\}_{i=1}^{N_{\text{du}}}$.

1 Add $\mu_{n_{\text{train}}}$ to $\Xi_{\text{train}}^{\text{pr}}$, add $u_h(\mu_{n_{\text{train}}})$ to $U_{\text{train}}^{pr}$, $n_{\text{train}}^{\text{pr}} \leftarrow n(\Xi_{\text{train}}^{\text{pr}})$.
2 Add $\mu_{n_{\text{train}}}$ to $\Xi_{\text{train}}^{\text{du}}$, add $z_h(\mu_{n_{\text{train}}})$ to $Z_{\text{train}}^{\text{du}}$, $n_{\text{train}}^{\text{du}} \leftarrow n(\Xi_{\text{train}}^{\text{du}})$.
3 **if** *rejection* **then**
4  Swap $\mu_{n_{\text{train}}} - 1$ and $\mu_{n_{\text{train}}}$ in both $\Xi_{\text{train}}^{\text{pr}}$ and $\Xi_{\text{train}}^{\text{du}}$ (change order, also in $U_{\text{train}}^{\text{pr}}$, $Z_{\text{train}}^{\text{pr}}$).
5 **end**
6 $\{\zeta_i^{\text{pr}}\}_{i=1}^{N_{\text{pr}}} \leftarrow \text{GSO}(U_{\text{train}}^{\text{pr}})$
7 **if** *Coefficient $c < \varepsilon_{ROM}$ from last update in Algorithm 5* **then**
8  Remove $\mu_{n_{\text{train}}^{\text{pr}}-1}$ from $\Xi_{\text{train}}^{\text{pr}}$ and $u_h(\mu_{n_{\text{train}}^{\text{pr}}-1})$ from $U_{\text{train}}^{\text{pr}}$, meaning $n_{\text{train}}^{\text{pr}} \leftarrow n_{\text{train}}^{\text{pr}} - 1$.
9 **end**
10 $\{\zeta_i^{\text{pr}}\}_{i=1}^{N_{\text{pr}}} \leftarrow \text{GSO}(U_{\text{train}}^{\text{pr}} \cup z_h(\mu_{n_{\text{train}}^{\text{pr}}}))$
11 $\{\zeta_i^{\text{du}}\}_{i=1}^{N_{\text{du}}} \leftarrow \text{GSO}(Z_{\text{train}}^{\text{du}})$
12 **if** *Coefficient $c < \varepsilon_{ROM}$ from last update in Algorithm 5* **then**
13  Remove $\mu_{n_{\text{train}}^{\text{du}}-1}$ from $\Xi_{\text{train}}^{\text{du}}$ and $z_h(\mu_{n_{\text{train}}^{\text{du}}-1})$ from $Z_{\text{train}}^{\text{du}}$, meaning $n_{\text{train}}^{\text{du}} \leftarrow n_{\text{train}}^{\text{du}} - 1$.
14 **end**
15 $\{\zeta_i^{\text{du}}\}_{i=1}^{N_{\text{du}}} \leftarrow \text{GSO}(Z_{\text{train}}^{\text{du}})$

just added would not result in ZOC at $\mu_{n_{\text{train}}^{\text{pr}}}$. We must delete the last two basis functions and then re-add one associated with the current solution. Simply deleting the last one would lose ZOC, and would mean the ROM had not changed at all.

There is an assumption here that, for example, in the case of the primal solutions, $u_h(\mu_{n_{\text{train}}^{\text{pr}}-1})$ is the most similar solution to $u_h(\mu_{n_{\text{train}}^{\text{pr}}})$. We justify this assumption by pointing out that this algorithm is being used in the context of optimization. If we ignore rejection for the moment (an assumption we will return to), the swapping operation in Line 4 guarantees that $C(\mu_1) > ... > C(\mu_{n_{\text{train}}^{\text{pr}}-1}) > C(\mu_{n_{\text{train}}^{\text{pr}}})$. In other words, the closest function value to $C(\mu_{n_{\text{train}}^{\text{pr}}})$ is $C(\mu_{n_{\text{train}}^{\text{pr}}-1})$, meaning the solution most similar to $u_h(\mu_{n_{\text{train}}^{\text{pr}}})$ is *likely* to be $u_h(\mu_{n_{\text{train}}^{\text{pr}}-1})$. Of course, this is not guaranteed, since a proof would require some in-depth knowledge about the relationship between $C(\mu)$ and $u_h(\mu)$. But in practice this works well because $\varepsilon_{\text{ROM}}$ is low and this scenario really only arises when the optimizer "barely moves" in $\mathcal{D}$ from one TR center to the next.

Regarding rejection, Algorithm 6 already requires the following assumption: if `rejection` is true, the truncation operations in Lines 8 and 13 will not be invoked. If they were, then in fact the entire algorithm would leave $\mathcal{V}_N^{\text{pr}}$ and $\mathcal{V}_N^{\text{du}}$ completely unchanged, leading to a potential infinite loop. In practice, we make this scenario give the user a warning, because it is extremely unlikely to occur. If rejection occurs, this is an indication that the RB space does not currently represent a good approximation of $C(\mu)$ in that part of $\mathcal{D}$. More data is required. If adding that data does not meet

the threshold of $\varepsilon_{\text{ROM}}$, the only reasonable explanation is that $\varepsilon_{\text{ROM}}$ is too high, and this must be fixed by the user. Because of rejection, our assumption that $C(\mu_1) > ... > C(\mu_{n^{\text{pr}}_{\text{train}}-1}) > C(\mu_{n^{\text{pr}}_{\text{train}}})$ may not necessarily hold. For example, $C(\mu_3)$ may be greater than $C(\mu_2)$. Nevertheless, if rejection and truncation never coincide, when truncation does occur both $C(\mu_2)$ and $C(\mu_3)$ are higher than $C(\mu_{n^{\text{pr}}_{\text{train}}-1})$. A very high objective function has not just been placed in the second-last position.

We do not truncate after adding $z_h(\mu_{\text{TR}})$ to the primal RB space, because we do not anticipate the dual solution to be close to a linear combination of primal solutions. If it is, this would be an indication that the problem may be self-adjoint (something the user should know *a priori*). In the case of self-adjoint problems, $z_h(\mu_{\text{TR}})$ is already in $\mathcal{V}_N^{\text{pr}}$, and Line 10 should be omitted completely from Algorithm 6.

There is no reason to believe that including $z_h(\mu_{\text{TR}})$ in the primal space improves accuracy in estimating the solution globally. The RB coefficient (used in the reconstruction of (2.17)) associated with the last basis function, which is always associated with $z_h(\mu_{\text{TR}})$, is likely to be near zero in most cases. However, it will likely be non-zero in primal solutions near $\mu_{\text{TR}}$, as it is responsible for ensuring FOC at $\mu_{\text{TR}}$, so it must have some minor corrective influence on the solution there.

The POD algorithm is relatively concise, because it simply restarts every time a new ROM is desired. At each HFM solution, all we need to do is to store $u_h(\mu_k)$ and $z_h(\mu_k)$. Then whenever we want a ROM, we call Algorithm 4 with the desired dataset. Due to both the successive nature of GSO and the more detailed consideration of truncation, the way it is actually used in practice is more complicated to describe in notation, but requires less floating point operations in total.

### 3.1.3 Reduced Basis Storage

Whether we use POD or GSO, each basis function $\zeta_i^{\text{pr}}$ belongs to $\mathcal{V}_h$, as each one is a linear combination of prior solutions. This means each one can be represented by a vector of FE coefficients, a column of the matrix $\hat{Y}^{\text{pr}} \in \mathbb{R}^{\mathcal{N} \times N_{\text{pr}}}$, as follows:

$$\zeta_i^{\text{pr}} = \sum_{j=1}^{\mathcal{N}} \hat{Y}_{j,i}^{\text{pr}} \phi_j. \tag{3.2}$$

In practice, this means both POD and GSO can be reduced to algebraic problems, resulting in sparse matrix-vector multiplications and vector inner products of size $\mathcal{N}$ and in the case of POD an eigenvalue problem of size $n_{\text{train}} \times n_{\text{train}}$. This means that regardless of which strategy is used, the cost of constructing the basis functions is quite small (though not negligible) relative to both computing $u_h$ and EQP training, which we will consider shortly. To store the RB set, we must store

$\hat{Y}^{\mathrm{pr}} \in \mathbb{R}^{\mathcal{N} \times N_{\mathrm{pr}}}$. The same applies for $\zeta_i^{\mathrm{du}}$.

Additionally, we can compute and store $\zeta_i^{\mathrm{pr}}(x_q), i = 1, ..., N_{\mathrm{pr}}, q = 1, ..., N_q$ for easier usage during online residual and Jacobian evaluations, to avoid having to repeatedly evaluate the FEM polynomials. This requires additional memory of size $N_q \times N_{\mathrm{pr}}$, but reduces computational cost.

## 3.2 First-Order Consistent EQP

Once the RB functions have been constructed, the next step is to update the reduced quadrature rule. This requires the primal and dual solutions at the chosen TR center $\mu_{\mathrm{TR}}$, $u_N(\mu_{\mathrm{TR}})$ $(= u_h(\mu_{\mathrm{TR}}))$ and $z_N^{\mathrm{pr}}(\mu_{\mathrm{TR}})$ $(= z_h(\mu_{\mathrm{TR}}))$. The EQP allows us to tailor the accuracy of hyperreduction with respect to any quantity of interest that is defined as an integral. We leverage this property to ensure that our model still satisfies the ZOC condition (2.45) and the FOC condition (2.46) at the current TR center $\mu_{\mathrm{TR}}$. This extends the convergence guarantee of the optimizer to cases when the surrogate model is the hyperreduced ROM. We indicate first which constraints *must* be applied to retain FOC, and then we suggest other constraints that might also be enforced to improve the accuracy of the reduced quadrature but are not necessary.

### 3.2.1 Selection of FOC-Critical Accuracy Constraints

We ensure the HROM satisfies the ZOC and FOC conditions by setting the following constraints on the NNLS problem (2.26) (with $i = 1, ..., N_{\mathrm{pr}}$):

$$\left| r_h(u_N(\mu_{\mathrm{TR}}), \zeta_i^{\mathrm{pr}}; \mu_{\mathrm{TR}}) - \tilde{r}_h(u_N(\mu_{\mathrm{TR}}), \zeta_i^{\mathrm{pr}}; \mu_{\mathrm{TR}}) \right| \leq \varepsilon_{\mathrm{FOC}}, \tag{3.3}$$

$$\left| r_h'[u_N(\mu_{\mathrm{TR}})](\zeta_i^{\mathrm{pr}}, z_N^{\mathrm{pr}}(\mu_{\mathrm{TR}}); \mu_{\mathrm{TR}}) - \tilde{r}_h'[u_N(\mu_{\mathrm{TR}})](\zeta_i^{\mathrm{pr}}, z_N^{\mathrm{pr}}(\mu_{\mathrm{TR}}); \mu_{\mathrm{TR}}) \right| \leq \varepsilon_{\mathrm{FOC}}, \tag{3.4}$$

$$\left| \partial_\mu r_h(u_N(\mu_{\mathrm{TR}}), z_N^{\mathrm{pr}}(\mu_{\mathrm{TR}}); \mu_{\mathrm{TR}}) - \partial_\mu \tilde{r}_h(u_N(\mu_{\mathrm{TR}}), z_N^{\mathrm{pr}}(\mu_{\mathrm{TR}}); \mu_{\mathrm{TR}}) \right| \leq \varepsilon_{\mathrm{FOC}}. \tag{3.5}$$

We set $\varepsilon_{\mathrm{FOC}}$ very low, close to machine precision. There are a total of $2N_{\mathrm{pr}} + n_p$ constraints. The first constraint guarantees consistency of the residual at $\mu_{\mathrm{TR}}$, and hence of the solution $\tilde{u}_N(\mu_{\mathrm{TR}})$ (provided the HROM problem is well-posed), and hence of the output. In other words, $\tilde{u}_N(\mu_{\mathrm{TR}}) = u_N(\mu_{\mathrm{TR}})$, and so $q(\tilde{u}_N(\mu_{\mathrm{TR}}); \mu_{\mathrm{TR}}) = q(u_N(\mu_{\mathrm{TR}}); \mu_{\mathrm{TR}})$ (ZOC). We compute the output $q(\cdot, \mu)$ using full order quadrature, as the operations required to do so are much cheaper than to evaluate the residual, and it is only required once per solution evaluation (unlike the residual, which is evaluated at each Newton iteration). The second constraint guarantees consistency of the dual approximated in the primal space: $\tilde{z}_N^{\mathrm{pr}}(\mu_{\mathrm{TR}}) = z_N^{\mathrm{pr}}(\mu_{\mathrm{TR}})$. The use of full quadrature to evaluate $q(\cdot, \mu)$ is also

important here. The third constraint guarantees consistency of the dual-weighted residual sensitivity, and thus of the output gradient (complete FOC). Together with the first two conditions, this guarantees (for $\varepsilon_{\text{FOC}} = 0$),

$$\partial_\mu r_h(u_N(\mu_{\text{TR}}), z_N^{\text{pr}}(\mu_{\text{TR}}); \mu_{\text{TR}}) = \partial_\mu \tilde{r}_h(\tilde{u}_N(\mu_{\text{TR}}), \tilde{z}_N^{\text{pr}}(\mu_{\text{TR}}); \mu_{\text{TR}}). \tag{3.6}$$

The three tildes in (3.6) correspond to consistency guaranteed by the three constraints. Clearly this demonstrates FOC with (2.21):

$$\begin{aligned} \frac{D(q(u_N(\mu_{\text{TR}}); \mu_{\text{TR}}))}{D\mu} &= -\partial_\mu r_h(u_N(\mu_{\text{TR}}), z_N^{\text{pr}}(\mu_{\text{TR}}); \mu_{\text{TR}}) + \partial_\mu q(u_N(\mu_{\text{TR}}); \mu_{\text{TR}}) \\ = \frac{D(q(\tilde{u}_N(\mu_{\text{TR}}); \mu_{\text{TR}}))}{D\mu} &= -\partial_\mu \tilde{r}_h(\tilde{u}_N(\mu_{\text{TR}}), \tilde{z}_N^{\text{pr}}(\mu_{\text{TR}}); \mu_{\text{TR}}) + \partial_\mu q(\tilde{u}_N(\mu_{\text{TR}}); \mu_{\text{TR}}). \end{aligned} \tag{3.7}$$

### 3.2.2 EQP Performance and Additional Accuracy Constraints

In practice, stability issues of the nonlinear solver were sometimes encountered using the hyperreduced model obtained using only the constraints (3.3)–(3.5). This is likely due to using a small number of constraints with a very tight accuracy tolerance, yielding a quadrature rule that does not apply well to other parameter values near $\mu_{\text{TR}}$. The constraints (3.3)–(3.5) are the bare minimum required to achieve first-order consistency at $\mu_{\text{TR}}$. Additional EQP constraints can be used to improve the quality of the EQP results.

One commonly included constraint is a constant constraint:

$$\begin{aligned} \Big||\Omega(\mu_{\text{TR}})| - \sum_{i=1}^{N_q^v} \hat{\rho}_q^v\Big| &\leq \varepsilon, \\ \Big||\partial\Omega(\mu_{\text{TR}})| - \sum_{i=1}^{N_q^b} \hat{\rho}_q^b\Big| &\leq \varepsilon. \end{aligned} \tag{3.8}$$

This constraint ensures that constant functions are integrated correctly over the domain and boundary — that the empirical quadrature rule does not artificially change the size of the domain. We must specify the parameter since in geometry optimization problems the domain itself depends on the parameter. The constant integration constraint does not guarantee that constants will be integrated accurately over other domains for $\mu \neq \mu_{\text{TR}}$.

Additional constraints may include a Jacobian constraint (3.9), a more generalized residual sensitivity constraint (3.10), or a Lagrangian solution constraint that considers HFM data at previous training points (3.11):

$$\left|r'_h[u_N(\mu_{\rm TR})](\zeta_i^{\rm pr},\zeta_j^{\rm pr});\mu_{\rm TR}) - \tilde{r}'_h[u_N(\mu_{\rm TR})](\zeta_i^{\rm pr},\zeta_j^{\rm pr});\mu_{\rm TR})\right| \le \varepsilon, \qquad i,j = 1...N_{\rm pr}, \tag{3.9}$$

$$\left|\partial_\mu r_h(u_N(\mu_{\rm TR}),\zeta_i^{\rm pr};\mu_{\rm TR}) - \partial_\mu \tilde{r}_h(u_N(\mu_{\rm TR}),\zeta_i^{\rm pr};\mu_{\rm TR})\right| \le \varepsilon, \qquad i = 1...N_{\rm pr}, \tag{3.10}$$

$$\left|r_h(u_N(\mu_t),\zeta_i^{\rm pr};\mu_t) - \tilde{r}_h(u_N(\mu_t),\zeta_i^{\rm pr};\mu_t)\right| \le \varepsilon, \quad i = 1...N_{\rm pr}, \mu_t \in \Xi_{\rm train}. \tag{3.11}$$

These additional constraints do not need to be solved to the same level of accuracy, as they are not needed for the FOC guarantee. This is why we have indicated them as being solved to a general tolerance $\varepsilon$, which is not necessarily equal to $\varepsilon_{\rm FOC}$. However, this requires the NNLS problem to prioritize some constraints over others. This is possible, but difficult in extreme cases due to conditioning challenges.

The constraints from (3.3), (3.9), (3.10), and (3.11) may also be weighted by the inverse of the residual Jacobian. The weighting is useful if we desire a finite tolerance on the *solution* error (rather than the residual) [66]. However, if $\varepsilon_{\rm FOC}$ is small enough, the weighting should not be relevant. The finalized EQP process is shown in Algorithm 7.

**Algorithm 7:** First-Order Consistent EQP Training

**Data:** Point at which the naive model is FOC $\mu_{\rm TR}$, primal RB functions $\{\zeta_i^{\rm pr}\}_{i=1}^{N_{\rm pr}}$, HFM-consistent primal and dual solutions $u_N(\mu_{\rm TR})$, $z_N(\mu_{\rm TR})$

**Result:** Quadrature points $\{x_q\}$ and associated weights $\hat{\rho}_q$

1 Add constraints from (3.3)–(3.5) at $\mu_{\rm TR}$.
2 Add any optional constraints (3.8)–(3.11) to improve EQP quality (can be at a lighter tolerance).
3 Solve the NNLS problem to select $\{x_q\}$ and $\hat{\rho}_q$.

## 3.3 Trust Region Definition

In this work, we consider three variants of trust regions (TRs). A TR $\mathcal{D}_{\rm TR}$ is a subset of the parameter domain $\mathcal{D}$ which acts as the domain for each optimization sub-problem (2.42) (see also (3.1)), in which the cheap surrogate model is being used instead of the underlying function. The surrogate model is assumed to be most accurate near the latest unrejected HFM solve at $\mu_{\rm TR}$ (the TR center), because this is the data from which the surrogate model is constructed. As mentioned in Section 2.7.2, we do not solve the sub-problem exactly. We simply terminate the TR sub-problem once we have reached 90% of the way to the boundary, when $\theta(\mu) \ge 0.9\Delta_{\rm TR}$ [69]. (See Section 2.7.2 and (2.43) for how the TR size $\Delta_{\rm TR}$ is selected and updated.)

The standard TR used in TR-based optimization is a geometric TR (GTR), where the metric is simply distance (in the Euclidian norm) from the TR center:

$$\mathcal{D}_{\text{GTR}} \equiv \{\mu \in \mathcal{D} \mid \theta_{\text{GTR}}(\mu) \equiv \|\mu - \mu_{\text{TR}}\|_2 \leq \Delta_{\text{GTR}}\}. \tag{3.12}$$

This TR forms a hypersphere around the TR center, illustrated in Figure 3.1a. A hypercube would also be possible with the infinity norm. Elliptical TRs are also sometimes used to reflect different scales in different parameter dimensions using some weighted norm. However these require *a priori* knowledge about the scaling of the problem.

We also introduce two other TRs based on the *a posteriori* error estimator $\eta_N(\mu)$. First, we consider an error-based TR (ETR) defined as,

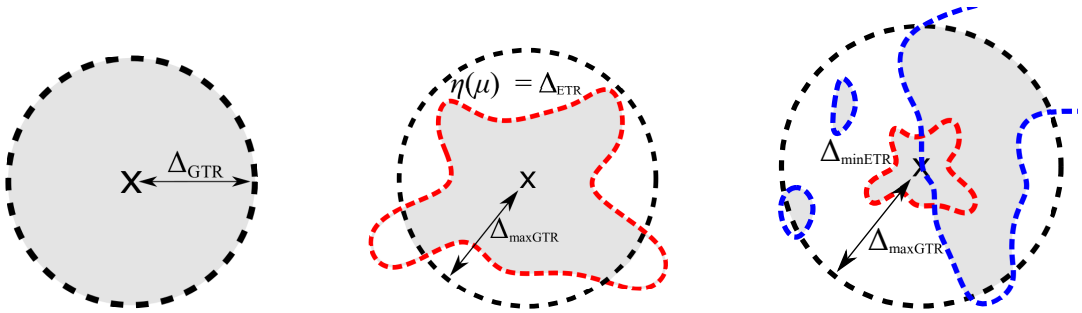
$$\mathcal{D}_{\text{ETR}} \equiv \{\mu \in \mathcal{D} \mid \theta_{\text{ETR}}(\mu) \equiv \eta_N(\mu) \leq \Delta_{\text{ETR}}\} \cap \mathcal{D}_{\text{maxGTR}}, \tag{3.13}$$

where $\mathcal{D}_{\text{maxGTR}}$ is defined by (3.12) using a fixed, user-defined maximum GTR size $\Delta_{\text{maxGTR}}$, and $\eta_N(\mu)$ is defined in (2.23). The ETR is illustrated in Figure 3.1b. The TR "size" $\Delta_{\text{ETR}}$ is an upper bound on the error estimator. This TR is designed to use the error estimator to reflect the true purpose of the TR: estimate where the surrogate model can and cannot be trusted. However, in case the error estimator breaks down, we do place a (very large, and usually inactive) GTR as a safety net. Because $z_h(\mu_{\text{TR}})$ is exactly reproduced in both the primal and dual RB spaces, and because we are not using hyperreduction to evaluate the error estimator, we are guaranteed that $\eta_N(\mu_{\text{TR}}) = 0$. This ensures the ETR is not an empty set for $\Delta_{\text{ETR}} > 0$.

The last trust region we simply term an "advanced TR" (ATR) for lack of a more concise term, and is illustrated in Figure 3.1c. It is defined as,

$$\begin{aligned}\mathcal{D}_{\text{ATR}} \equiv \big(\{\mu \in \mathcal{D} \mid \theta_{\text{ATR}}(\mu) \equiv \eta_N(\mu) \leq (\tilde{C}(\mu_{\text{TR}}) - \tilde{C}(\mu)) + \beta(\mu - \mu_{\text{TR}})^T \nabla_\mu \tilde{C}(\mu_{\text{TR}})\} \\ \cup\, \mathcal{D}_{\text{minETR}}\big) \cap \mathcal{D}_{\text{maxGTR}},\end{aligned} \tag{3.14}$$

where $\beta \in \mathbb{R}_{>0}$ comes from the sufficient decrease conditions used in line searches (Algorithm 1). The ATR seeks to guarantee that the new training point will satisfy the sufficient decrease conditions. In cases where the error is large but an anticipated reduction in the (hyperreduced) cost function $\tilde{C}(\mu)$ is even larger, it might still be advantageous to accept this point. This accounts for the fact that near the beginning, where cost reductions are usually easier to obtain, the model does not need to be as accurate as near the end when the gradient of the objective function is nearly 0. We also




(a) GTR (b) ETR (c) ATR

Figure 3.1: Various TRs visualized for $n_p = 2$, $\mathcal{D} \subset \mathbb{R}^2$. The black lines correspond to a GTR, the red lines to the contours of $\eta_N(\mu) = \Delta_{\text{ETR}}$ (an ETR), and the blue lines to the intersection contours of $\eta_N(\mu)$ with the sufficient decrease condition (an ATR).

include a minimum ETR. Without the minimum ETR, if the error estimate grows faster than the function decreases the ATR may be empty, at least in the suggested step direction. We choose a small $\Delta_{\text{minETR}}$, because if this happens, this means the model is certainly not accurate enough to determine whether a given step results in increase or decrease in the objective function. Note that $\Delta_{\text{maxGTR}}$ and $\Delta_{\text{minETR}}$ do not dynamically update following (2.43).

As illustrated in Figure 3.1, the ATR is not guaranteed to be contiguous. In fact, neither is the ETR, although it is more likely to be contiguous, as $\eta_N(\mu)$ grows as we move away from $\mu_{\text{TR}}$, whereas the ATR also depends on the behaviour of the objective function.

The ETR and the ATR both capture differences in scaling between dimensions in the parameter space. The ETR prioritizes local accuracy of the surrogate model, while the ATR prioritizes taking the largest possible step with some reduction still expected. It is not possible to guarantee which will yield better performance. Both the ETR and the ATR face challenges when $\eta_N(\mu)$ is not reliable. Also, they both require evaluation of the function (or at least computation of the RB solution) before feasibility with respect to the TR can be assessed. If the IP algorithm suggests a very large step, perhaps due to an inaccurate Hessian, it is very cheap for the GTR to backtrack until it is feasible, and only then solve the HROM problem. This is not possible with the ETR and the ATR, requiring more HROM solves.

## 3.4 Integrated Optimization Algorithm

All of the ingredients above were integrated into an algorithm to solve (3.1) that constructs the ROM on-the-fly, or as the optimizer progresses. We present Algorithm 8 and then highlight some important features. The algorithm follows the basic structure of Algorithm 3 in Section 2.7.

**Algorithm 8:** Trust Region Optimization with On-The-Fly HROM Construction

**Data:** Initial guess $\mu_0$, $n_{\max}$, $n_{\text{warmup}}$, $\epsilon_{\text{opt}}$
**Result:** $\mu_{\text{opt}}$, $C(\mu_{\text{opt}})$, $\tau_{\text{opt}}$

1 Evaluate $C(\mu_0)$, $\nabla C(\mu_0)$.
2 Initialize Lagrange multipliers $(y, z)$ according to (2.39).
3 $B_0 \leftarrow I$, unless a more accurate estimate is available (warm start).
4 $k \leftarrow 0$
`/* Optional HFM warmup */`
5 **while** $n_{HF} < n_{warmup}$ **do**
6 Compute step direction $(p_\mu, p_s, p_y, p_z)$ by solving (2.35).
7 Compute the maximum steps $\alpha_s^{\max}$ and $\alpha_z^{\max}$ using (2.36).
8 Perform a line search using Algorithm 1 to obtain $\mu_k$ using $C(\mu)$.
9 Update the estimate of the Hessian $B_k$ using Algorithm 12.
10 $\tau_k \leftarrow \|\nabla \mathcal{L}(\mu)\|_\infty$
11 **if** $\tau_k \leq \epsilon_{opt}$ **then**
12 **break** and go to Line 33 (extremely unlikely).
13 **end**
14 Update the value of the barrier parameter using (2.41).
15 $k \leftarrow k + 1$
16 **end**
17 Initialize the RB sets (Algorithm 6) and run EQP (Algorithm 7) to initialize the HROM surrogate $\tilde{C}_k(\mu)$.
`/* Main HROM-accelerated outer optimization loop */`
18 **while** $n_{HF} < n_{max}$ **do**
19 Solve the TR sub-problem using the HROM surrogate $\tilde{C}_k(\mu)$ using Algorithm 2.
20 Compare $\mu_k$ to all previous TR centers and perturb if necessary.
21 Solve the HFM problem at $\mu_k$, $n_{\text{HF}} \leftarrow n_{\text{HF}} + 1$.
22 Re-initialize Lagrange multipliers according to (2.39).
23 $\tau_k \leftarrow \|\nabla \mathcal{L}(\mu)\|_\infty$
24 **if** $\tau_k \leq \epsilon_{opt}$ **then**
25 **break**
26 **end**
27 Compute $\varrho$ using (2.44), and `rejection` $\leftarrow (\varrho < 0)$.
28 Update the TR size using (2.43).
29 Incorporate latest HFM data into $\mathcal{V}_N^{\text{pr}}$ and $\mathcal{V}_N^{\text{du}}$ (Algorithm 6).
30 $k \leftarrow k + 1$
31 Rerun the EQP (Algorithm 7), yielding an updated surrogate $\tilde{C}_k(\mu)$.
32 **end**
33 $\tau_{\text{opt}} \leftarrow \tau_k$
34 $\mu_{\text{opt}} \leftarrow \mu_k$

We present Algorithm 8 in the context of using GSO to construct the basis functions. If the POD method is preferred for generating RB functions, use Algorithm 4 on the datasets specified in Section 3.1.1 instead of GSO (Algorithm 6).

After initialization, we must perform at least one HFM solve, the minimum required to set up an HROM surrogate for use in the main optimization loop. However, if the user desires (i.e., by setting $n_{\text{warmup}} > 0$), further HFM solves can optionally be performed (see Lines 5–16). This improves the quality of the initial ROM, which if it is formed based on only one HFM solution may be quite

rudimentary. Every time we evaluate $C(\mu)$ in Line 8, we increment $n_{\text{HFM}}$ by 1 to keep track of the number of HFM solves. We terminate the line search if $n_{\text{HFM}} = n_{\text{warmup}}$, and if this leaves the line search unconverged we return $\mu$ to the lowest $\mu$-value observed so far. Note that optimization is a secondary purpose of this section of the algorithm (and convergence of the entire optimization problem is therefore unlikely, unless $n_{\text{warmup}}$ is high). The primary purpose is to build a good ROM while respecting the user's cost request.

Lines 18–32 describe the HROM-accelerated TR-based approach to solving the optimization problem (3.1). Each solution of the sub-problem in Line 19 makes use of the IP algorithm (Algorithm 2) to find the solution to (2.42), using the HROM surrogate $\tilde{C}_k(\mu)$. A warm start is used, using data from $C(\mu_{\text{TR}})$. In line searches, we backtrack until TR constraints are satisfied. Error-based TR constraints must be evaluated after $\tilde{C}_k(\mu)$ has been evaluated.

Regarding the perturbation in Line 20, if the suggested next training point is very close to any of the previous training points and the sub-problem converges due to the TR constraint rather than optimality conditions, we solve for a new step using (2.35), backtrack until problem constraints are satisfied, and use this as the next training point instead. This avoids having two HFM snapshots that are very similar in the training set, which would lead to almost no new information being added to the ROM. Although this could be handled by $\varepsilon_{\text{ROM}}$, it is wasteful to compute the HFM solution if we can predict ahead of time that a very similar one is already in the span of $\mathcal{V}_N^{\text{pr}}$. We do not perturb if in fact the solutions are close because the sub-problem terminated due to satisfying the optimality conditions, rather than reaching the edge of the TR, because this means we do actually need a very small adjustment to the ROM to converge to $\mu_{\text{opt}}$. In this case, conditioning will be handled by $\varepsilon_{\text{ROM}}$.

# Chapter 4

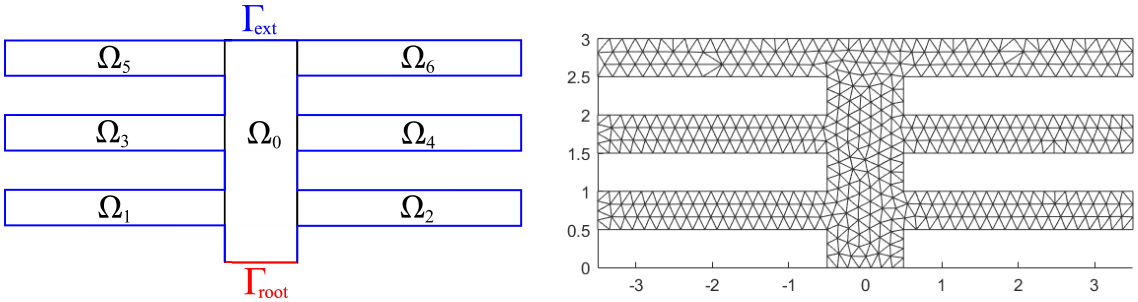
# Results

In this chapter, we present results for three optimization problems of the form (3.1) solved using the approach summarized in Section 3.4. In each case, we compare the results to a baseline which solves the problem without the use of TR methods and the HROM surrogate, simply applying the IP algorithm and using the HFM at each function evaluation. The three model problems are a thermal fin conductivity optimization governed by a nonlinear heat equation, pressure matching on a deformed NACA0012 airfoil subject to compressible Euler flow, and then pressure matching on a deformed NACA0012 airfoil subject to compressible RANS flow.

In each case, besides comparing results to the HFM-only baseline, we also take a closer look at the behaviour of specific aspects of the algorithm. With the thermal fin problem, we assess the performance of the DWR error estimator $\eta_N(\mu)$, and we also look at behaviour near $\mu_{\text{TR}}$ and verify that the HROM surrogate satisfies the FOC condition. In the Euler problem, we analyze the effect of user-defined settings such as the number of warmup solves and the initial TR size. In the RANS problem, we consider how the optimality tolerance $\epsilon_{\text{opt}}$ affects the opportunity for cost savings, as well as the advantages and disadvantages of the error-based TR definitions compared to the GTR.

## 4.1 Nonlinear Heat Equation: Thermal Fin Optimization

In this section, we apply the HROM-accelerated optimization framework to a thermal fin subject to a *nonlinear* heat equation. We build on the work by Qian et al. [51], who showed the application of ROM to a thermal fin optimization problem subject to the standard heat equation, a linear PDE. Additionally, their parametrization, which described the conductivity of different regions of the fin and natural convection away from the surface, permitted affine decomposition of the stiffness matrix.




(a) Designation of regions and boundaries, $\Gamma_{\text{root}}$ in red and $\Gamma_{\text{ext}}$ in blue

(b) Mesh for HFM ($\mathcal{N} = 2{,}029$ degrees of freedom)

Figure 4.1: Model thermal fin problem

This permitted the use of ROM without hyperreduction, while still obtaining $\mathcal{N}$-independence in computation of the objective function. It also permitted the construction of strict *a posteriori* error bounds on the solution and compliance output (temperature at the root of the fin). Here we introduce nonlinearity by considering temperature-dependent conductivity. Besides allowing us to demonstrate the behaviour of the hyperreduced model in an experiment with a linear analogue in the literature, this problem is also of practical relevance, since some materials do in fact exhibit temperature-dependent conductivity. The small scale of the problem allows the use of a prototype code (written in MATLAB) that can run at low cost on a commodity laptop, facilitating testing and verification of various aspects of the proposed algorithm. We focus specifically on the performance of the *a posteriori* error estimator and on the local FOC guarantee.

### 4.1.1 Problem Definition

The problem consists of a thermal fin split into seven regions (see Figure 4.1a). A (dimensionless) heat flux is applied at $\Gamma_{\text{root}}$. In mathematical form, the governing equation is

$$\begin{aligned} -\nabla \cdot (k(u(\mu);\mu)\nabla u(\mu)) &= 0 && \text{on } \Omega, \\ -(k(u(\mu);\mu)\nabla u(\mu)) \cdot \hat{n} &= -1 && \text{on } \Gamma_{\text{root}}, \\ -(k(u(\mu);\mu)\nabla u(\mu)) \cdot \hat{n} &= \text{Bi}(\mu)u(\mu) && \text{on } \Gamma_{\text{ext}}. \end{aligned} \tag{4.1}$$

The thermal conductivity of the fin is described by $k(u(\mu);\mu)$. The conductivity of $\Omega_0$, the trunk of the fin, does not depend on $\mu$, but only on $u(\mu)$, while for $i = 1, ..., 6$, $\mu_i$ affects the conductivity of region $\Omega_i$ as follows:

$$k(u(\mu);\mu) \equiv \begin{cases} e^{u(\mu)} & \text{on } \Omega_0, \\ 10^{\mu_i} e^{u(\mu)} & \text{on } \Omega_i,\ i = 1, ..., 6. \end{cases} \tag{4.2}$$

The Biot number describes natural convection away from the surface $\Gamma_{\text{ext}}$, and is computed by $\text{Bi}(\mu) = 10^{\mu_7}$. The parameter domain $\mathcal{D} \equiv \{\mu_i \in [-1.0, 1.0], i = 1, ..., 6; \mu_7 \in [-2.0, -1.25]\}$. The integral output, or quantity of interest, $q(u(\mu))$ (not self-adjoint) is the average temperature over the domain. This output contains no direct dependence on $\mu$, so we drop the $\mu$ in the argument for $q$. The optimization objective $C(q(u(\mu)); \mu)$ combines the output with an algebraic cost to model the tradeoff between more favourable material properties and external conditions with increased cost:

$$\begin{aligned} q(u(\mu)) &\equiv \int_\Omega u(\mu)\ dx, \\ C(q(u(\mu)); \mu) &\equiv q(u(\mu)) + \sum_{i=1}^{n_p} 0.1 e^{\mu_i}. \end{aligned} \tag{4.3}$$

The mesh used (shown in Figure 4.1b) has $\mathcal{N} = 2,029$ degrees of freedom. Output-based adaptive mesh refinement was not performed, as this mesh was tested against much finer meshes and the output was found to not vary by more than 0.1% at $\mu_{\text{opt}}$. We use a continuous Galerkin method.

Throughout this example, we follow the ROM-accelerated optimization algorithm, Algorithm 8. We use incremental POD to generate the RB functions and use no additional EQP constraints beyond (3.3)–(3.5), the bare minimum to satisfy the FOC condition. The EQP problem is solved until the algebraic system in NNLS is fully determined, meaning the number of non-zero weights will be equal to the number of constraints, and the EQP tolerance is limited only by the conditioning of the NNLS problem. Only volume integrals were hyperreduced. There are no interior facet integrations (since we used a continuous Galerkin method), and boundary facet integrations are deemed cheap enough to be easily computed using full quadrature. We do not perform any extra initial HFM warmup solves (i.e., $n_{\text{warmup}} = 0$ in Algorithm 8).

### 4.1.2 Assessment of the *A posteriori* Error Estimator

In this subsection, we compare error estimates with the true error (i.e., difference with the HFM) in both the naive ROM and the hyperreduced ROM. We randomly generate a set of test parameters $\Xi_{\text{test}} \subset \mathcal{D}$ of size $n_{\text{test}}$ and another set of training parameters $\Xi_{\text{train}} \subset \mathcal{D}$ of size $n_{\text{train}}$ such that $\Xi_{\text{train}} \cap \Xi_{\text{test}} = \emptyset$. We then perform the test routine in Algorithm 9, in which we progressively enrich a ROM by adding the solutions at parameters $\mu_i \in \Xi_{\text{train}}$, and then test the efficacy of this

ROM at test points $\mu_j \in \Xi_{\text{test}}$. The efficacy of the ROM is determined by how well it predicts the output, described by $\Delta_N(\mu)$. We also compare the actual error to the DWR error estimator $\eta_N(\mu)$. The effectivity $H_N(\mu) = \eta_N(\mu)/\Delta_N(\mu)$ is an indicator of the performance of the error estimator. Throughout this section, we denote $\eta_N(\tilde{u}_N(\mu);\mu)$ by $\tilde{\eta}_N(\mu)$ for brevity. The solution $\tilde{u}_N(\mu)$ is obtained using reduced quadrature, but $\eta_N$ itself is evaluated using full quadrature (see (2.23)). The same applies to $\tilde{\Delta}_N(\mu)$ and $\tilde{H}_N(\mu)$. We also abbreviate $N_{\text{du}} = n_{\text{train}}$ as $N$. Note that $N_{\text{pr}} = N + 1$ because of the inclusion of a single dual in the training of $\mathcal{V}_N^{\text{pr}}$.

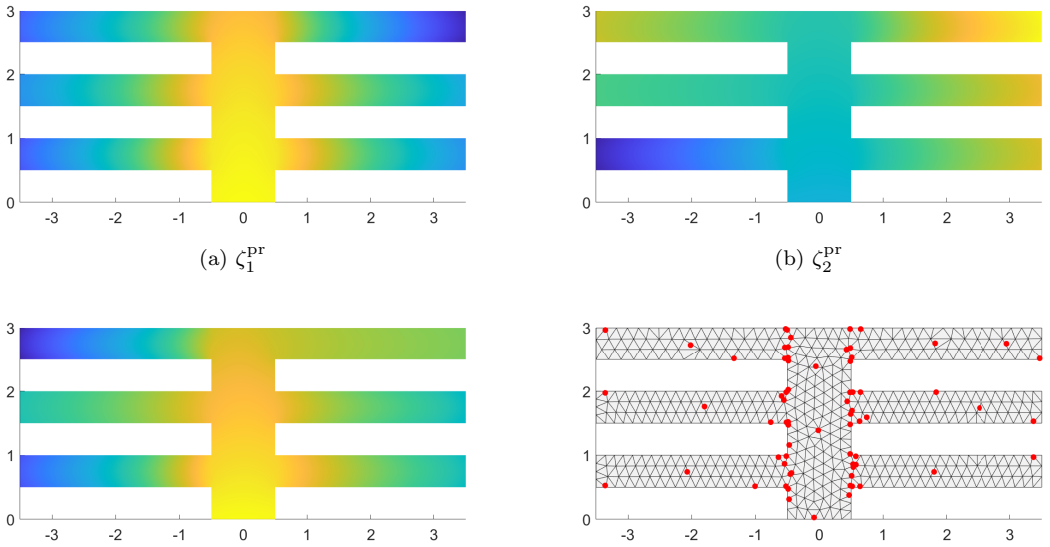
**Algorithm 9:** *A posteriori* Error Estimator Performance Test

**Data:** $\Xi_{\text{test}}$, $\Xi_{\text{train}}$
**Result:** Various error metrics for the naive ROM and HROM
**for** $j = 1, ..., n_{test}$ **do**
  Compute $u_h(\mu_j)$ and $q(u_h(\mu_j))$ (will be used later).
**end**
**for** $i = 1, ..., n_{train}$ **do**
  Compute $u_h(\mu_i)$ and $z_h(\mu_i)$.
  Update (or construct, for $i = 1$) ROM for computing $u_N(\mu)$. Note: $N = i$.
  Use the EQP to generate a reduced quadrature rule for computing $\tilde{u}_N(\mu)$.
  **for** $j = 1, ..., n_{test}$ **do**
    Compute $u_N(\mu_j)$, and $\tilde{u}_N(\mu_j)$.
    Compute the output $q(u_N(\mu_j))$ and $q(\tilde{u}_N(\mu_j))$.
    Evaluate $\Delta_N(\mu_j) = q(u_h(\mu_j)) - q(u_N(\mu_j))$, $\tilde{\Delta}_N(\mu_j) = q(u_h(\mu_j)) - q(\tilde{u}_N(\mu_j))$.
    Compute the DWR error estimators $\eta_N(\mu_j)$, and $\tilde{\eta}_N(\mu_j)$.
    Compute the effectivities $H_N(\mu_j) \equiv \eta_N(\mu_j)/\Delta_N(\mu_j)$ and $\tilde{H}_N(\mu_j) \equiv \tilde{\eta}_N(\mu_j)/\tilde{\Delta}_N(\mu_j)$.
  **end**
  Compute the average errors across $\Xi_{\text{test}}$: $\bar{\Delta}_N$, $\bar{\tilde{\Delta}}_N$, $\bar{\eta}_N$, and $\bar{\tilde{\eta}}_N$. Observe behaviour vs $N$.
  Compute the average effectivities across $\Xi_{\text{test}}$: $\bar{H}_N$ and $\bar{\tilde{H}}_N$. Observe behaviour vs $N$.
**end**
Select one value of $N$ and examine the effectivity of the error estimators for all parameters in $\Xi_{\text{test}}$.

First, to establish visual familiarity with the RB method, we consider the ROM constructed using Algorithm 9 for $n_{\text{test}} = 10$ and $n_{\text{train}} = 30$. We present the first three primal basis functions and the location of quadrature points on the domain in Figure 4.2.

Since these basis functions were generated using POD, the physical interpretation is that they are modes. The modes capture the parametric variation of $u_h(\mu)$, most of which depends on the difference in the conductivity of the fins. This is why the main difference between the basis functions is which fins have high temperature and which fins have low temperature. The second mode versus the first mode also seems to reflect the difference between the trunk and the tips of the fins, which depends on $\text{Bi}(\mu)$.

The reduced quadrature points selected by the EQP are clustered near the boundary between



(a) $\zeta_1^{\text{pr}}$ (b) $\zeta_2^{\text{pr}}$

(c) $\zeta_3^{\text{pr}}$ (d) Mesh overlayed with quadrature points (red)

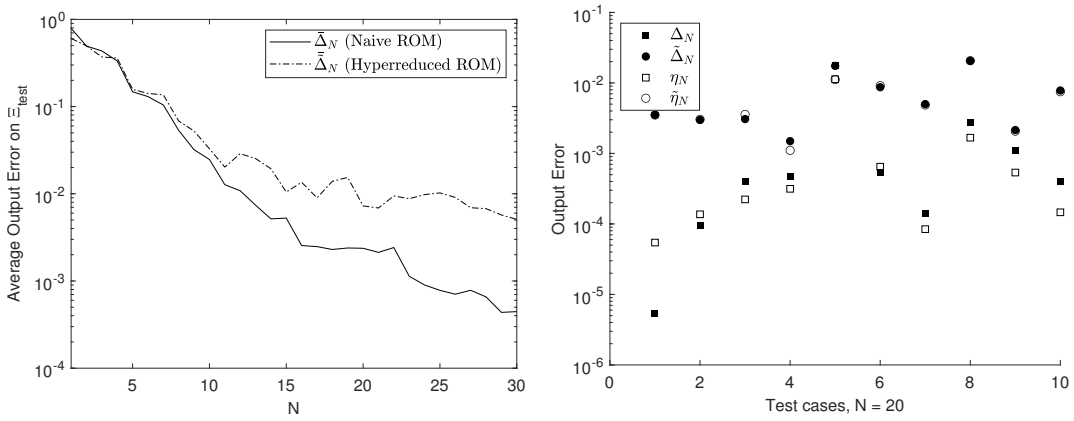
Figure 4.2: Reduced basis and reduced quadrature visualization

the trunk and the fins, indicating that this is an important place to capture parametric variation. Because we solved the NNLS problem until it was fully constrained, the number of points equals the number of constraints: $\tilde{N}_q = 2N_{\text{pr}} + n_p = 2(31) + 7 = 69$. There are fewer points earlier in the test routine, where $N_{\text{pr}}$ is lower. In the HFM baseline, there are six quadrature points in each element, and $N_q = 5{,}256$ in total.

Turning now to quantitative results from Algorithm 9, we expect in general all error estimates to decrease as $N$ increases. The hyperreduction error in general might compound with the ROM error, so it is expected to be higher (unless there is some lucky cancellation). This can be seen in Figure 4.3a, where our expectation is validated. Beyond $N \approx 15$, hyperreduction error dominates. This is because the EQP training strategy used is focused on local FOC, not global ZOC, as will be discussed in the Section 4.1.3.

We also expect $\eta_N(\mu)$ to give a good approximation of the true error $\Delta(\mu)$. The second test, shown in Figure 4.3b, uses an isolated value of $N = 20$. It does not average across $\Xi_{\text{test}}$, as comparing the average error and average error estimate does not reveal how well the error estimator captures variations in the actual error. Generally we see that the error estimator provides a good indication of the true error. As in Figure 4.3a, the error from the hyperreduced model is almost always greater than the error from the naive model.

We next quantify the performance of the error estimator in terms of the effectivity, defined as $H_N(\mu) \equiv \eta_N(\mu)/\Delta_N(\mu)$. In Figure 4.4, the effectivity averaged over $\Xi_{\text{test}}$ is shown to approach 1




(a) Comparison of naive and hyperreduced ROM errors.

(b) Comparison of local naive and hyperreduced ROM errors and error estimators. The solid shapes correspond to true error, and the outlined shapes estimate the error.

Figure 4.3: Evaluation of a posteriori error estimator performance

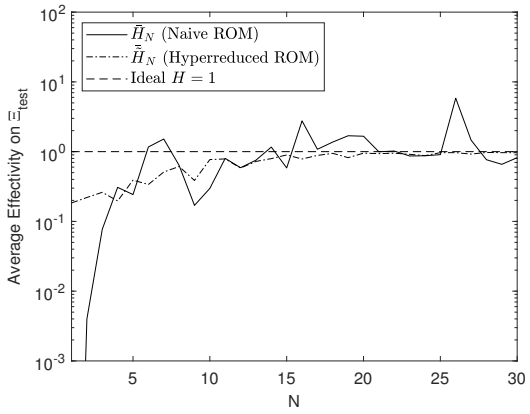



Figure 4.4: Comparison of the average effectivity of error estimators with naive and hyperreduced ROM solutions

as $N$ increases. Ideally $H_N(\mu) = 1$, meaning the error estimator exactly predicts the true error. A reliable error estimator would yield $H_N(\mu) \geq 1$, which might be preferable to an overly optimistic estimator with $H(\mu) < 1$. The error estimators are shown to be too optimistic for a small number of basis functions, but the effectivity generally approaches 1 as basis functions are added.

The effectivity data is noisy, even as $N$ increases in the naive ROM case. This is because of the trend observed in Figure 4.3a, where the error itself gets smaller, making the denominator in the definition of $H_N(\mu)$ approach 0. In other words, the error becomes harder to estimate as it grows lower. In cases where $\Delta_N(\mu)$ happens to be very low, outliers can strongly affect the effectivity, despite averaging. This problem is not as pronounced in the hyperreduced case, since as Figure 4.3a shows the error does not converge as quickly.

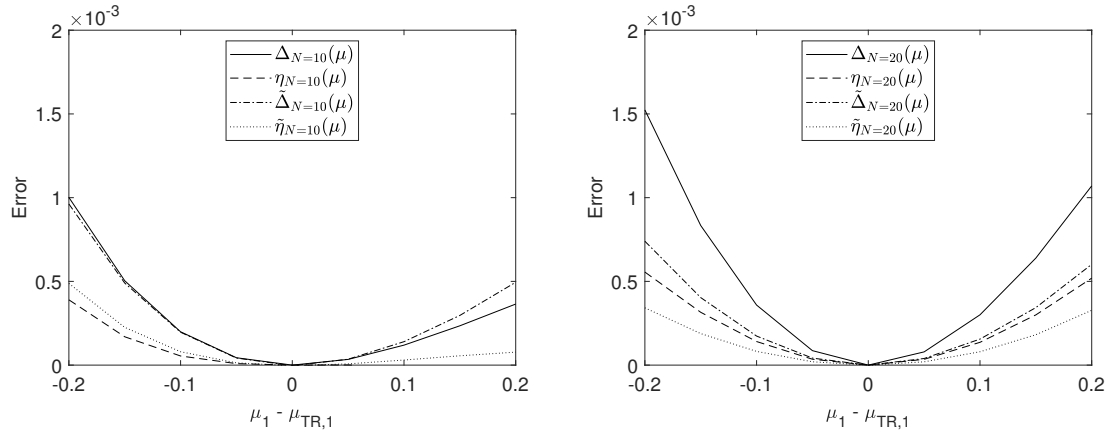
### 4.1.3 Performance Near $\mu_{\mathbf{TR}}$

The suite of tests above demonstrates the ability of the ROM to give a global (in $\mathcal{D}$) approximation of $u_h(\mu)$, since we use randomly selected test points from anywhere in $\mathcal{D}$. However, the naive ROM is designed to also give FOC near $\mu_{\text{TR}}$. This is not tested in Algorithm 9. Additionally, our EQP is designed to provide FOC in the HROM near $\mu_{\text{TR}}$, and consequently is expected to exhibit near-ZOC for a finite region around $\mu_{\text{TR}}$, but none of the training constraints considered other regions of the parameter space. This is why in Figure 4.3a, once the hyperreduction error dominates, increasing the number of basis functions does little to further reduce the error. The increased number of EQP constraints associated with more basis functions does not reduce the hyperreduction error, as these constraints are not directed towards global approximation of $u_h(\mu)$. This motivates the test given in Algorithm 10, in which we build a ROM at a point $\mu_{\text{TR}}$ and then take very small steps of length $h_{\text{step}}$ on either side of it to see if the slope is as predicted. (We are not actually doing TR-based optimization here, but we call the point $\mu_{\text{TR}}$ to emphasize that the HROM construction treats the training point as a TR center.) We restrict ourselves to one dimension of the parameter space for ease of visualization, specifically $\mu_1$ since it has a larger impact on the output.

**Algorithm 10:** Test of HROM surrogate model near the TR center $\mu_{\text{TR}}$

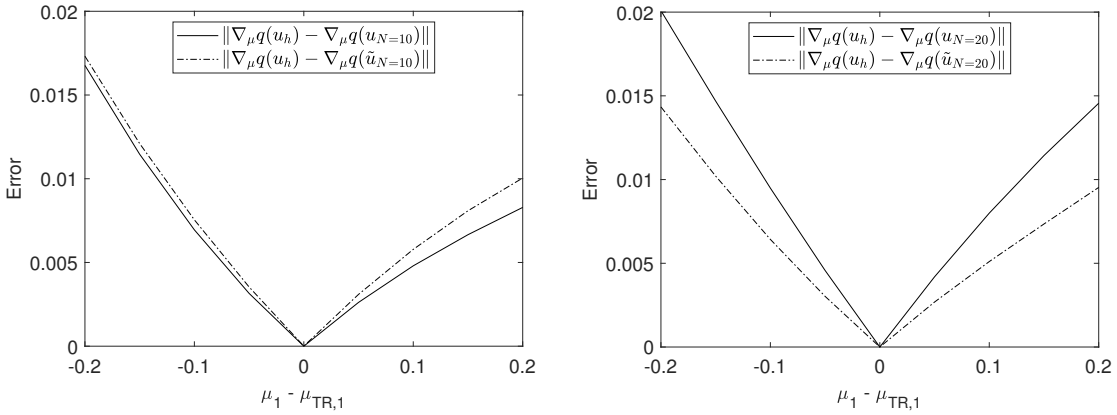
**Data:** $\Xi_{\text{train}}$, $n_{\text{step}}$, $h_{\text{step}}$
**Result:** Observations of error in a region around $\mu_{\text{TR}}$
**for** $i = 1, ..., n_{train}$ **do**
  Compute $u_h(\mu_i)$, $z_h(\mu_i)$ $q(u_h(\mu_i))$ and $\nabla_\mu q(u_h(\mu_i))$.
  Update (or construct, for $i = 1$) an HROM for computing $u_N(\mu)$, and $\tilde{u}_N(\mu)$ with $N = i$ (Algorithm 8).
  Compute $u_N(\mu_i)$, $\tilde{u}_N(\mu_i)$, and evaluate the function and gradient. These should be exactly consistent.
  Generate a set of test points $\Xi_{\text{test}}$ around $\mu_{\text{TR}} = \mu_i$ by stepping $n_{\text{step}}$ times along the $\mu_1$ axis in each direction, with a step size of $h_{\text{step}}$.
  **for** $\mu_t$ *in* $\Xi_{test}$ **do**
    Compute $u_h(\mu_t)$, $u_N(\mu_t)$, and $\tilde{u}_N(\mu_t)$.
    Compute the output $q$ and gradient $\nabla_\mu q$ at each solution above.
    Evaluate $\Delta_N(\mu_t) = q(u_h(\mu_t)) - q(u_N(\mu_t))$, $\tilde{\Delta}_N(\mu_t) = q(u_h(\mu_t)) - q(\tilde{u}_N(\mu_t))$.
    Compute the DWR error estimators $\eta_N(\mu_t)$, and $\tilde{\eta}_N(\mu_t)$.
  **end**
  Construct a plot showing output error and error estimator and a plot showing output gradient error along $\Xi_{\text{test}}$ for each selected value of $N$.
**end**

Figure 4.5 shows the actual and estimated error around $\mu_{\text{TR}}$ (along the $\mu_1$ axis) for both the naive and hyperreduced ROMs, for $N = 10$ and $N = 20$. Several things are of interest here. Firstly, FOC is obvious from the fact that both the error and the derivative of the error are equal to 0 at




(a) $N = 10$ (b) $N = 20$

Figure 4.5: Naive and hyperreduced ROM output errors and error estimators near $\mu_{\text{TR}}$

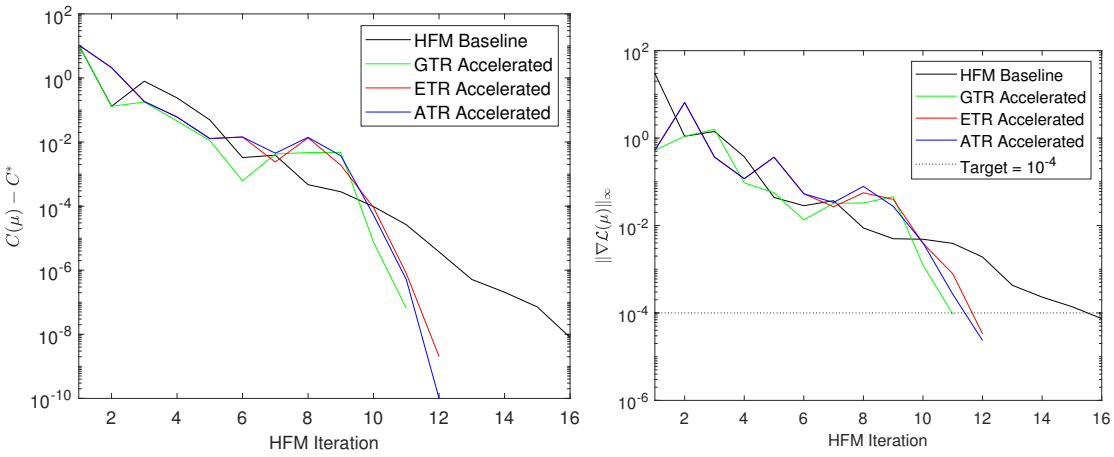



(a) $N = 10$ (b) $N = 20$

Figure 4.6: Naive and hyperreduced ROM output gradient errors near $\mu_{\text{TR}}$

$\mu_{\text{TR}}$. The same is true of the error estimator. The effectivity for either model can be inferred from the match between error and error estimate. Lastly, note that from $N = 10$ to $N = 20$, we see no discernible reduction in the error in the region around $\mu_{\text{TR}}$. (A different trust region center is used in each case, namely, the latest training point.) While data from $\mu_{\text{TR},i=10}$ was used to construct the ROM at $\mu_{\text{TR},i=20}$, it does not have a significant impact on lowering the error there locally. Adding more basis functions decreases the *global average* error, as demonstrated in Figure 4.3a; however, it is not guaranteed to improve *local, non-averaged* approximations near $\mu_{\text{TR}}$.

FOC is demonstrated again in Figure 4.6, where the norm of the error in the gradient is shown for both the naive and hyperreduced ROMs. This error goes to 0 at $\mu_{\text{TR}}$, satisfying the FOC condition. (The model is clearly not second-order consistent, but it was not expected to be so because this was




(a) Objective function (b) First order optimality

Figure 4.7: Convergence of the thermal fin model problem

not part of the design of the training algorithm.) Again, no clear trend is identified with increased RB size. We do not have an *a posteriori* estimate for error in the gradient.

### 4.1.4 Optimization

The tests above all involve expensive evaluations of the HFM on $\Xi_{\text{test}}$, a step which would never be taken in a real-world application since the cost of doing so is greater than simply using the HFM to solve the problem of interest. We did it simply to validate important characteristics of the HROM. However, here we demonstrate performance of the HROM as it would be used in application. This tests the integrated framework of Algorithm 8 to minimize the cost function (4.3).

In Figure 4.7, we show the convergence of the algorithm with HFM evaluations along the horizontal axis. Figure 4.7a shows the decrease in the objective function (offset by a constant $C^*$ for logarithmic scaling). The fact that the decrease is non-monotonic for the HFM baseline is due to failed line search iterations, and for the HROM-accelerated cases is due to rejection of the solution of a sub-problem (much more frequent). The first-order optimality is shown in Figure 4.7b. The problem is terminated when the optimality passes the threshold of $\epsilon_{\text{opt}} = 10^{-4}$. First order optimality of $\tau_{\text{opt}} \leq 10^{-4}$ was obtained in all cases.

Figure 4.7 demonstrates that the benefit of HROM acceleration starts taking effect only after a certain point, in this case around the tenth iteration. Before that, the ROM is insufficiently accurate to achieve much acceleration, and the rate of convergence is roughly the same as the HFM baseline. Once the optimizer passes about the tenth iteration, we may truly categorize this as a "many-query scenario," in which it is possible to see tangible acceleration.

Table 4.1: Results for Nonlinear Thermal Fin Optimization

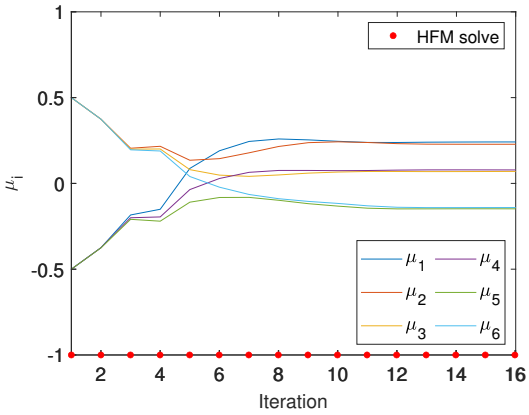
| **Case** | **# HFM** | **# HROM** | $t_{\mathbf{HFM}}$ **[s]** | $t_{\mathbf{HROM}}$ **[s]** | $t_{\eta}$ **[s]** | $t_{\mathbf{train}}$ **[s]** | $t_{\mathbf{total}}$ **[s]** |
|---|---|---|---|---|---|---|---|
| HFM | 16 | - | 6.4 | - | - | - | 6.4 |
| GTR | 11 | 60 | 4.0 | 0.1 | - | 0.2 | 4.6 |
| ETR | 12 | 79 | 4.5 | 0.2 | 0.5 | 0.3 | 6.0 |
| ATR | 12 | 59 | 4.6 | 0.1 | 0.4 | 0.3 | 6.0 |

Table 4.1 shows the number of HFM solves, the number of HROM solves, and the timing breakdown for optimization. Although the number of HFM solves is reduced, the additional cost of EQP training means that substantial speedup in terms of CPU time is observed only with the GTR. This is perhaps not surprising for a problem this small. Requiring only 16 HFM solves to reach optimality provides little opportunity for improvement, and the relatively small problem with only 2,029 DOF has low solution times. The results were obtained on a commodity laptop in MATLAB. In the next section, we present a larger case running in C++, where we see more reliable speedup.

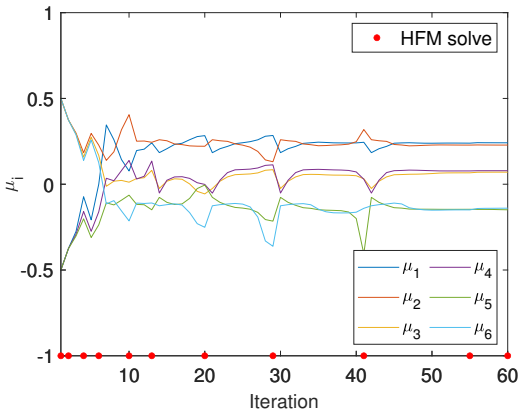
Figure 4.8a demonstrates the optimization "trajectory" (i.e., path through the parameter space to the optimal solution) for solving the optimization problem using the HFM only. Figures 4.8b–4.8d show the results using our proposed HROM-accelerated framework (Algorithm 8). We do not show $\mu_7$, since it scales differently; however, it was also included as a design parameter in the optimization. Model update positions are represented by the red dots along the bottom axis.

We have demonstrated the convergence of the method generally, regardless of the choice of TR. However, by examining the trajectories in Figure 4.8 a little further we can understand their performance in more detail. Unsurprisingly, most HFM solves are clustered near the beginning, and become increasingly less frequent as the optimizer progresses. This is because the accuracy of the ROM increases and the step size decreases as the optimizer approaches the optimal solution while the TR does not necessarily get smaller. Near the end, there is another cluster of HFM solves. Here, it is not the TR causing the sub-problem to terminate, but the fact that the sub-problem has satisfied the optimality conditions. When the HFM problem is solved, it is discovered that the true optimality conditions are not quite satisfied.

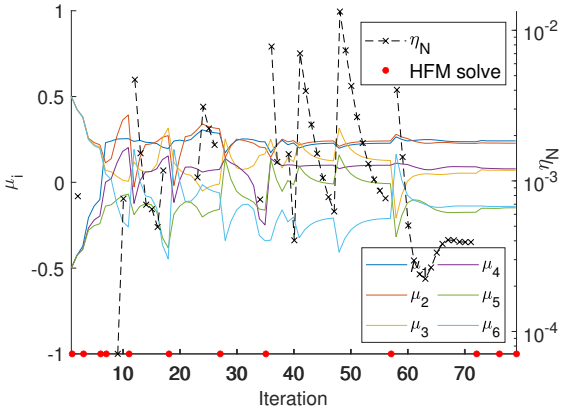
The three TR variants use approximately the same number of HFM and HROM solves. With the settings used here, the ETR took the highest number of ROM solves. This depends partly on the initial TR size chosen. (Section 4.2.5 investigates the effect of the initial TR size.) In general, we expect to see more HROM solves with the ETR and ATR than with the GTR, because the GTR algorithm is able to back-track into the TR *prior* to evaluating the function, and in Figure 4.8 we plot only points for which the function was evaluated (including points that were rejected during Armijo



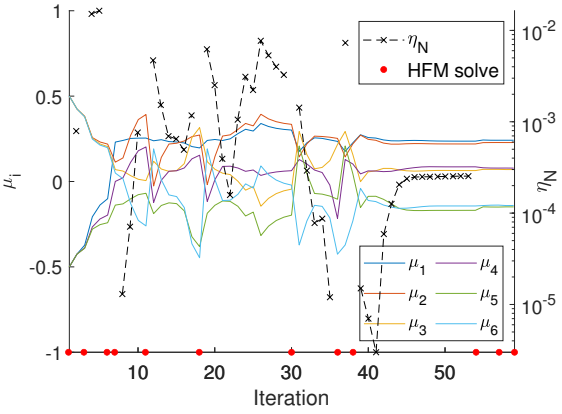

(a) HFM Trajectory




(b) HROM Trajectory with GTR




(c) HROM Trajectory with ETR




(d) HROM Trajectory with ATR

Figure 4.8: Comparison of optimization methods for thermal fin model problem

backtracking). The ETR and ATR algorithms can only evaluate the TR constraint after evaluating the function, which is why their trajectories demonstrate significant backtracking. Despite this, the number of HFM and HROM solves are about equal in the GTR and ATR cases. However, the GTR has the cost advantage of a) not having to construct the dual RB space and b) not having to evaluate the error estimator, which requires full quadrature integration.

These results cast doubt on whether error-based TRs are worth the extra cost. The advantage of the error-based TRs is supposed to stem from the fact that defining the TR size is more closely tied to the accuracy of the HROM. However, even with the GTR the algorithm is equipped with a size adjustment based on the success of the previous ROM (actual reduction vs predicted reduction). This eventually functions as a *de facto* error estimator, despite being only evaluated at each TR update rather than at each function evaluation. However, the objective function behaviour in this study is not expected to be that complex. Inclusion of $\mathrm{Bi}(\mu)$ was supposed to make the behaviour

of the problem more interesting; however, $\mathrm{Bi}(\mu)$ converged to near its optimal solution almost immediately, minimizing exploration in that dimension. In other problems, error may vary quite differently between dimensions, or there may be significant coupling between parameters, making a simple spherical TR potentially ill-suited.

## 4.2 Euler Equations: Pressure Matching Inverse Design

In this section, we consider the compressible Euler equations as the governing PDE. This tests our method on a system of hyperbolic PDEs (whereas the heat equation was scalar elliptic). It also allows us to demonstrate the method in an aerodynamics application without the added complication of turbulence. We are aware of several studies in which naive ROM was applied to an optimization problem constrained by the Euler equations [70, 68]. Both works specifically acknowledged the need for hyperreduction. Whereas the purpose of Section 4.1 was mainly to demonstrate the performance of the error estimator and the behaviour of the ROM near the TR center, this section will now turn attention to the effect of certain hyperparameters in the HROM-accelerated optimization algorithm (Algorithm 8).

### 4.2.1 Problem Definition

In this problem, we start by generating a pressure distribution $p_{\mathrm{ref}}$ by solving for the flow over a deformed NACA0012 airfoil (corresponding to a parameter value $\mu_{\mathrm{ref}}$). The goal is then to reproduce the deformed geometry ($\mu_{\mathrm{ref}}$) by solving an optimization problem. We use the undeformed NACA0012 geometry as the initial guess and the difference between the current estimate and the reference pressure $p_{\mathrm{ref}}$ as the comparison metric. The output of interest is thus:

$$q(u(\mu);\mu) \equiv \int_{\Gamma_{\mathrm{foil}}(\mu)} (p(u(\mu)) - p_{\mathrm{ref}})^2 \, dS. \tag{4.4}$$

This problem therefore follows the inverse design procedure sometimes used in airfoil design, in which a pressure distribution is prescribed and an airfoil yielding that pressure distribution is sought. However, in our case the exact solution is known *a priori*, as $q(u(\mu_{\mathrm{ref}});\mu_{\mathrm{ref}}) = 0$. We can check that the problem has been solved by comparing $\mu_{\mathrm{ref}}$ and $\mu_{\mathrm{opt}}$ — these should match exactly. No algebraic manipulations are needed for the optimization objective function — $C(q(u(\mu);\mu);\mu) \equiv q(u(\mu);\mu)$.

The governing equations are the standard steady-state compressible Euler equations with conservative variables. Conservation of mass, momentum, and energy are enforced throughout $\Omega$. Dirichlet

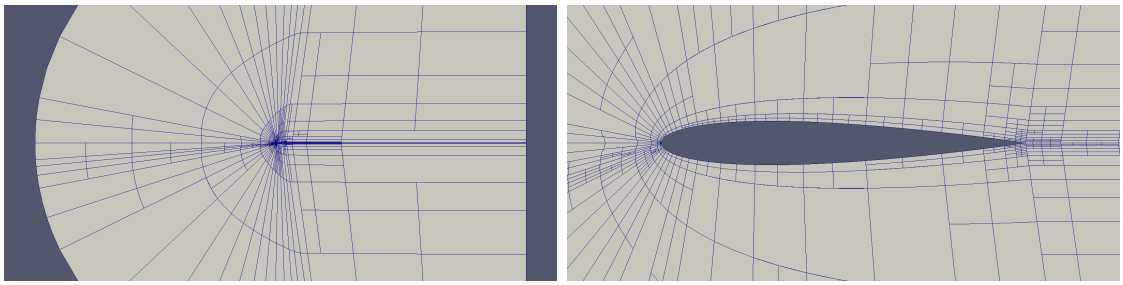


Figure 4.9: Mesh used for solution of the Euler equation pressure-matching problem

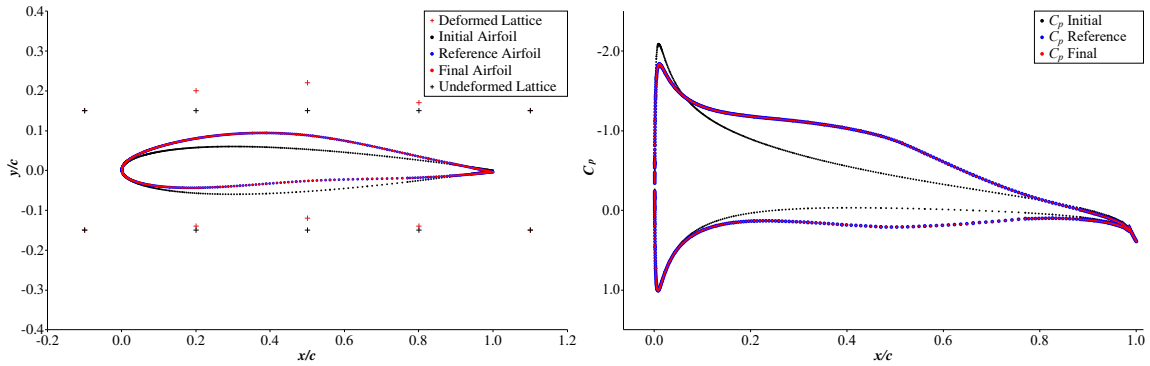
boundary conditions specifying the free stream flow properties are enforced on the external boundaries, and flow-tangency is enforced around the airfoil. The flow conditions are freestream Mach number $M_\infty = 0.2$ and angle of attack $\alpha = 5°$ (to make the pressure profile non-symmetric). These are both held constant, and $\alpha = 5°$ is enforced by specifying the two components of the velocity at the far-field boundary condition.

The adaptively refined mesh used to approximate the solution to the Euler equations is shown in Figure 4.9. A DG method was used, and polynomial approximations of degree $p = 2$ are used to construct the solution on each element, resulting in a total of $\mathcal{N} = 20{,}496$ DOF.

The parametrization is done using FFD B-splines (see Appendix B), placing the airfoil in a 2 x 5 lattice of FFD points. All the corner points are fixed (to prevent translation), and the six left-right interior points are only allowed to move vertically. Figure 4.10a shows the FFD lattice and the geometry.

### 4.2.2 High Fidelity Baseline

We start by directly applying the IP algorithm (Algorithm 2) using only the HFM model. The optimization required 20 HFM solves to reproduce the reference geometry with a first-order optimality tolerance of $\epsilon_{\text{opt}} = 10^{-3}$. This corresponded to an objective function value of $\mathcal{O}(10^{-8})$. The resulting geometry and pressure distributions are shown in Figure 4.10. The optimizer nearly exactly reproduces the geometry and the pressure distribution, reproducing $\mu_{\text{ref}}$ to within $10^{-4}$. (The target and final lattice control nodes are not both shown, as they would be indistinguishable.) Along with all the cases in this section, the case was run on a single node containing 40 cores of the Niagara cluster at SciNet. It took 33.6 seconds to solve the optimization problem.




(a) Geometry and FFD parametrization

(b) Reference $C_p$ distribution

Figure 4.10: Geometry and $C_p$ distribution for Euler-based pressure matching optimization, with $M = 0.2$ and $\alpha = 5°$

### 4.2.3 Baseline ROM Settings

We wish to analyze the effect of certain hyper-parameters; however, it would be impossible to try all possible combinations at any meaningful resolution. We therefore decide on some baseline settings, which for this problem were found to yield good results.

- The default number of warmup HFM solves is $n_{\text{warmup}} = 0$ (although we shall investigate the effect of this setting in Section 4.2.4).
- The default initial trust region size is $\Delta_0 = 0.05$, for both the ETR and the GTR (also investigated in Section 4.2.5).
- The optimization problem is solved to first-order optimality of $\epsilon_{\text{opt}} = 10^{-3}$.
- The initial Hessian estimate is based on an initial step size of 0.1.
- The maximum GTR size is $\Delta_{\text{maxGTR}} = 0.5$ and the minimum ETR size is $\Delta_{\text{minETR}} = 10^{-4}$.
- The RB functions are constructed using GSO and the ROM tolerance is set to $\varepsilon_{\text{ROM}} = 10^{-8}$.
- The EQP tolerance is set to $\varepsilon_{\text{FOC}} = 10^{-6}$ for FOC-critical constraints (3.3)–(3.5) and $\varepsilon = 10^{-4}$ for non-FOC-critical constraints.
- Non-FOC-critical constraints include the Jacobian at $\mu_{\text{TR}}$ (3.9) and the total residual sensitivity (3.10).
- Residual constraints are not weighted by the inverse of the residual Jacobian.

- The EQP problem is *not* solved separately for elements, interior facets, and boundary facets. However, unlike the thermal fin problem, we do use reduced quadrature for boundary and interior facets.

The additional EQP constraints are expected to increase the computational cost associated with EQP training, as they increase the size of the NNLS problem (however, with a looser tolerance, they are expected to be satisfied with only a few more iterations than would have been needed without them). The relatively loose EQP tolerance of $\varepsilon_{\text{FOC}} = 10^{-6}$ relieves the pressure on the NNLS. This may result in a loss of the FOC guarantee, but since we are solving only to an optimality of $\epsilon_{\text{opt}} = 10^{-3}$ we have found that it is not necessary to go tighter.

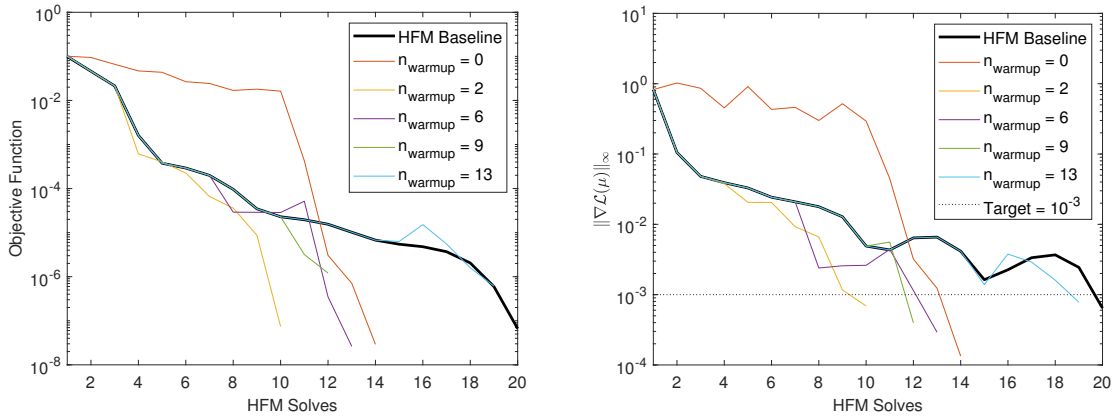
It would be interesting to mathematically predict the maximum allowable NNLS tolerance to achieve a certain optimality convergence guarantee, as well as to explore why more constraints are necessary and how tightly they must be enforced. However, it was not possible to study the effects of all these hyperparameters within the scope of this thesis project. This certainly would provide fertile ground for future research.

### 4.2.4 Effect of Number of Initial High-Fidelity Solutions

First we investigate the effect of $n_{\text{warmup}}$ on the optimizer for the GTR, ETR, and ATR cases. As shown in Algorithm 8, this indicates how many HFM solves we do before starting to apply HROM acceleration (minus the initial solve, which is not optional). We saw from the thermal fin results in Section 4.1 (specifically Figure 4.7) that using the HROM had little to no advantage for the first few iterations, where there was not enough data available to build a sufficiently accurate HROM. Inaccuracy of both the HROM itself and the DWR error estimator, especially near the beginning, can lead to increased exploration and a higher number of HROM solves and training points. We increase $n_{\text{warmup}}$ from 0 to 14, meaning up to 75% of the number of HFM solves required in the baseline are done before applying HROM-acceleration.

In Figure 4.11 we demonstrate some sample results obtained using the ATR, which demonstrate typical convergence trends of the objective function and first-order optimality. As was shown for the thermal fin, in the beginning not much acceleration is seen with $n_{\text{warmup}} = 0$. However, once acceleration does start to occur, it occurs very quickly, rapidly overtaking the HFM baseline.

The number of HFM and HROM solves as well as the computational time for these sample cases are shown in Table 4.2. It shows that the HROM training time, which is dominated by the solution of the EQP problem, limits the computational savings, and in some cases even results in higher cost,




(a) Objective function

(b) First order optimality

Figure 4.11: Effect of number of initial HFM solves $n_{\text{warmup}}$ on convergence of the pressure matching problem using HROM acceleration with an ATR

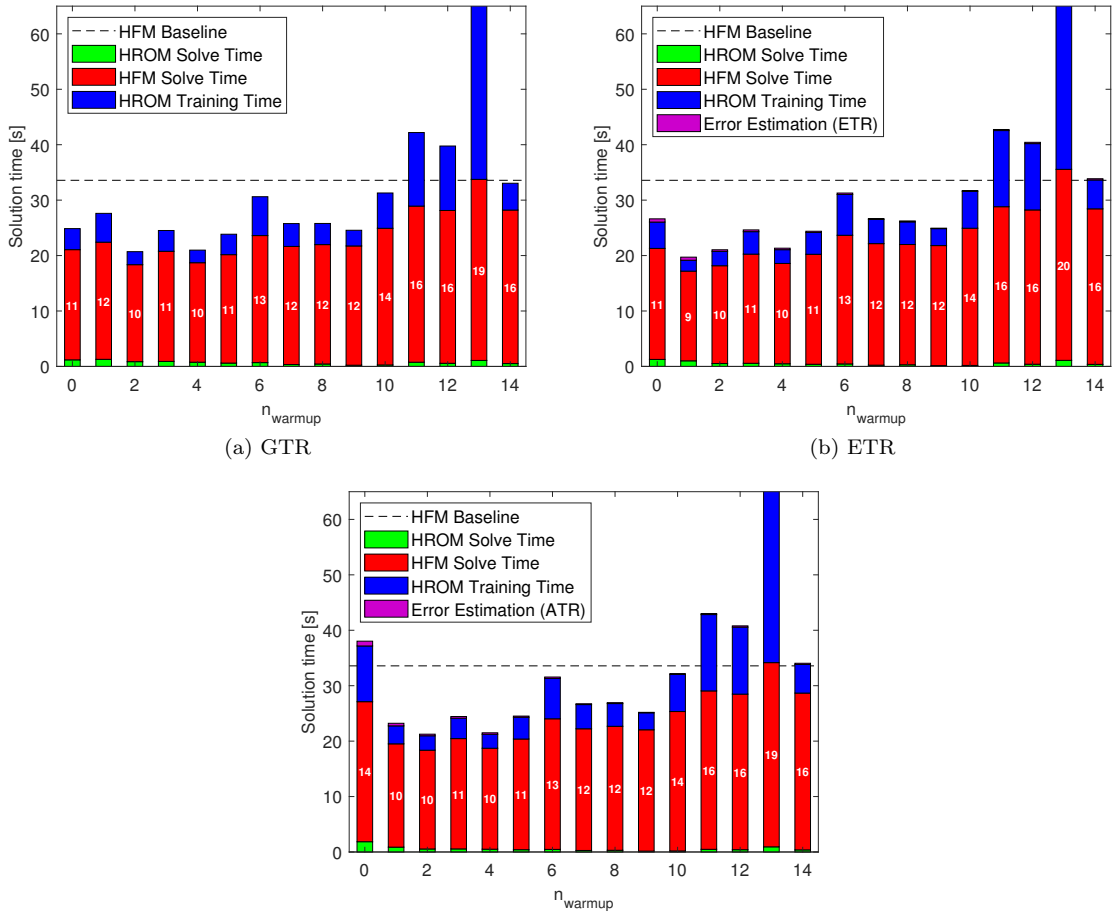
Table 4.2: Results for Euler Inverse Aerodynamic Design

| **Case** | **# HFM** | **# HROM** | $t_{\mathbf{HFM}}$ [s] | $t_{\mathbf{HROM}}$ [s] | $t_\eta$ [s] | $t_{\mathbf{train}}$ [s] | $t_{\mathbf{total}}$ [s] |
|---|---|---|---|---|---|---|---|
| HFM | 20 | - | 33.6 | - | - | - | 33.6 |
| $n_{\text{warmup}} = 0$ | 14 | 198 | 25.3 | 1.9 | 0.9 | 10.1 | 38.1 |
| $n_{\text{warmup}} = 2$ | 10 | 62 | 17.8 | 0.5 | 0.3 | 2.6 | 21.3 |
| $n_{\text{warmup}} = 6$ | 13 | 46 | 23.6 | 0.4 | 0.3 | 7.3 | 31.6 |
| $n_{\text{warmup}} = 9$ | 12 | 14 | 21.9 | 0.1 | 0.1 | 3.1 | 25.2 |
| $n_{\text{warmup}} = 13$ | 19 | 73 | 33.2 | 0.9 | 0.5 | 64.4 | 99.1 |

even if the number of HFM solves is reduced. The particularly extreme case of $n_{\text{warmup}} = 13$, where nearly 2/3 of the time is spent on training, indicates that after a certain number of HFM solves, the EQP constraints chosen are ill-suited, and the NNLS problem has difficulty converging.

We only show a sample of results in Figure 4.11 and Table 4.2. However, results for a complete search of all three TR variants for $n_{\text{warmup}} \in [0, 14]$ are presented more concisely in Figure 4.12, focusing on the CPU time breakdown. In all cases, the optimizer does indeed converge to the correct solution; the question is whether it does so more or less quickly than the high fidelity baseline, and what elements contribute the most to the computational cost. Each bar represents a single solution of the optimization problem, with various contributions to the computational cost indicated by color. The total cost is compared to the baseline, meaning that bars under the baseline are truly accelerated by the HROM framework.

As expected, the computational cost increases with $n_{\text{warmup}}$ beyond a certain value, for the simple reason that the HROM acceleration strategy is not being used until later in the optimization process. Again, there is random variation, but a rough trend is visible. The HROM model is not trained between initial HFM solves. The EQP constraints required for a high number of basis




(a) GTR

(b) ETR

(c) ATR

Figure 4.12: Effect of number of initial HFM solves $n_{\text{warmup}}$ on computational time with HROM-accelerated optimization. The red bar shows time spent on HFM solves, and the number on each bar indicates the number of HFM solves.

functions means late NNLS solves are much more expensive than early ones, resulting in nonlinear dependence of training cost on $N_{\text{pr}}$, as was also seen in Table 4.2.

However, also as expected, it is sometimes beneficial to set $n_{\text{warmup}}$ slightly larger than 0. This is most pronounced in the case of the ATR. This is because a ROM built with only one solution snapshot is quite inaccurate, and potentially even counterproductive. Trying to use it for optimization may be a waste of resources. This applies especially to the error-based trust regions, since $\eta_N(\mu)$ is not even available with $n_{\text{train}} = 1$, and is unreliable in the early stages even once it does become available, as was shown in Section 4.1 and specifically Figure 4.4. We may use the HFM in the initial phase, where large increases in the objective function may be taken with large steps in the parameter space $\mathcal{D}$; and the cheap HROM to do smaller adjustments later in the optimization process.

There is also some variability, as there is an element of "chance" involved as well. A small

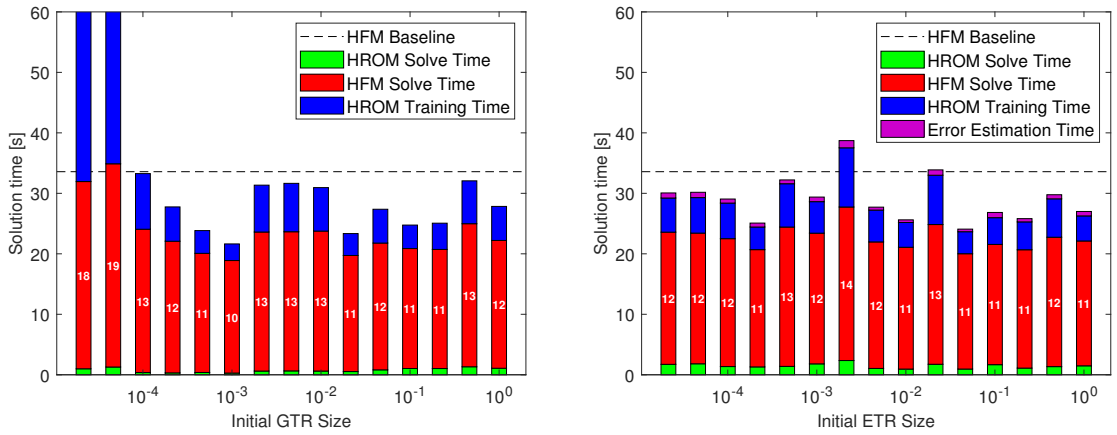



Figure 4.13: Effect of initial TR size on computational time with HROM-accelerated optimization

difference in trajectory can leave the optimizer very close to the true minimum, but just outside the convergence criteria, requiring one more HFM solve. With these solutions requiring 10–15 HFM solves, one extra HFM solve is significant in terms of percentage.

The evaluation of the error estimator itself takes some computational time, although this also contributes very little to the computational cost. We are evaluating the error estimator using full quadrature, as the EQP did not contain any accuracy constraints related to the error estimator. We would expect the HROM training time to also be slightly larger on average, as more GSO is required to construct $\mathcal{V}_N^{\mathrm{du}}$. However, the cost of GSO is very small compared to the cost of constructing the EQP constraints and solution of the NNLS problem.

In this particular problem, all parameters referred to positions of independent control points in the FFD lattice. Like the thermal fin problem, this limits the degree of global parameter coupling. This explains why using an error-based TR yields no significant advantage. In other problems, there may be complex interplay between two or more parameters or a substantial difference in scale, and for this ETRs or ATRs might perform better than spherical GTRs. We shall see this in Section 4.3.

### 4.2.5 Effect of Initial TR Size

The cases in Section 4.2.4 use an initial TR size of $\Delta_0 = 0.05$, both for the GTR and the ETR. (The ATR does not require an initial TR size.) In this section, we keep $n_{\mathrm{warmup}} = 0$ and investigate the effect of the initial TR size. A wide range of initial GTR and ETR sizes were tested, and the results are reported in the same style as in Section 4.2.4. Once again, all cases converge, but we are interested in the impact on performance. Figure 4.13 shows the results of this analysis.

The GTR results indicate that for a wide range of initial GTR sizes, some cost reduction can be

expected (up to nearly 40% in the best case). The relative stability for a wide range of initial TR sizes can be attributed to the fact that the TR changes size as the optimizer progresses based on the accuracy of the ROM in the previous case, meaning that near the end of the optimization routine the TR size is virtually independent of the initial size.

In extreme cases, the update rule (2.43), which either multiplies the size by 2, divides by 4, or keeps the region the same size, is not able to adapt quickly enough. With very small TRs, the optimizer relies too heavily on HFM solves, and extra burden is placed on the EQP training. Little difference was observed for particularly large trust regions, especially in the GTR case, when the initial GTR is larger even than the parameter domain. In this problem, the optimizer itself uses inequality constraints to bound the parameters at $\pm 0.2$ (in the infinity norm).

The ETR also obtains cost reduction in most cases, although with less reliability and more random variation. There is less correlation between initial TR size and computational cost, and we see random spikes. This indicates that error-based TRs are not as robust as GTRs. Relying on $\eta_N(\mu)$ can be dangerous. If $\eta_N(\mu)$ is overly pessimistic, a given ROM does not get used to its full potential and expensive training goes to waste. If instead it is overly optimistic, it will be more exploratory, and time is subsequently wasted training the ROM for parts of $\mathcal{D}$ that would not otherwise have been visited. Both scenarios potentially increase the number of HFM solves required.

More HROM solves are also required with the ETR. The optimizer is unable to assess whether it is currently located inside or outside the trust region until after it has evaluated the HROM solution. However, this has almost no effect on the final cost.

### 4.2.6 Comparison with Naive ROM (Full Quadrature)

Given that a substantial portion of the cost in Figures 4.12 and 4.13 comes from HROM training, which is dominated by the EQP problem, it is worth considering whether the speedup achieved by hyperreduction is worth the cost of training. Does the cost benefit on the HROM solves justify the extra training cost? Here we report results for the same tests run in Section 4.2.5, but using naive ROM (i.e., integrals were evaluated using full quadrature). Results are shown in Figure 4.14.

For both the GTR and the ETR, median computational cost is slightly higher with naive ROM as compared with the hyperreduced ROM. In most cases, there is still an advantage over the HFM baseline. However, the biggest change is that the cost has shifted from training to ROM solves. Because we only need to construct the RB set and not reduced quadrature, there is no EQP problem. Training consists only of performing orthogonalization, which is relatively cheap; however, the

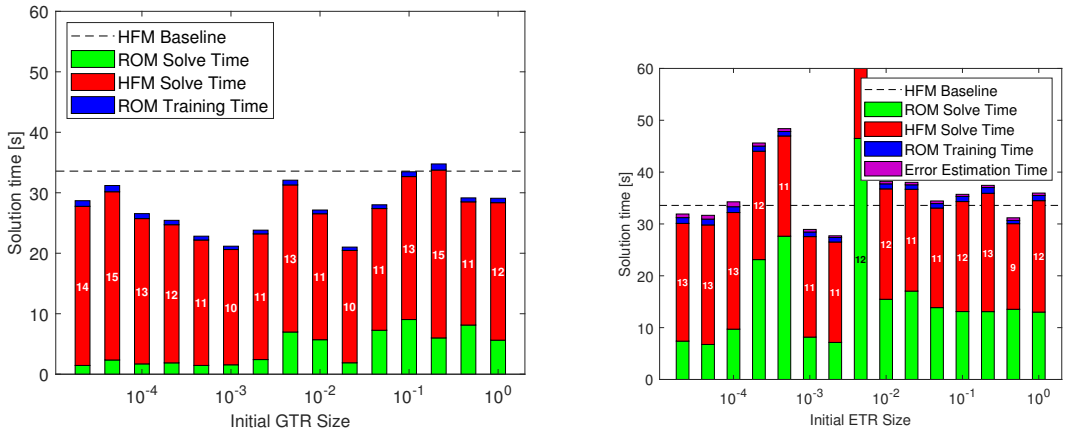



Figure 4.14: Effect of initial TR size on computational time with naive ROM

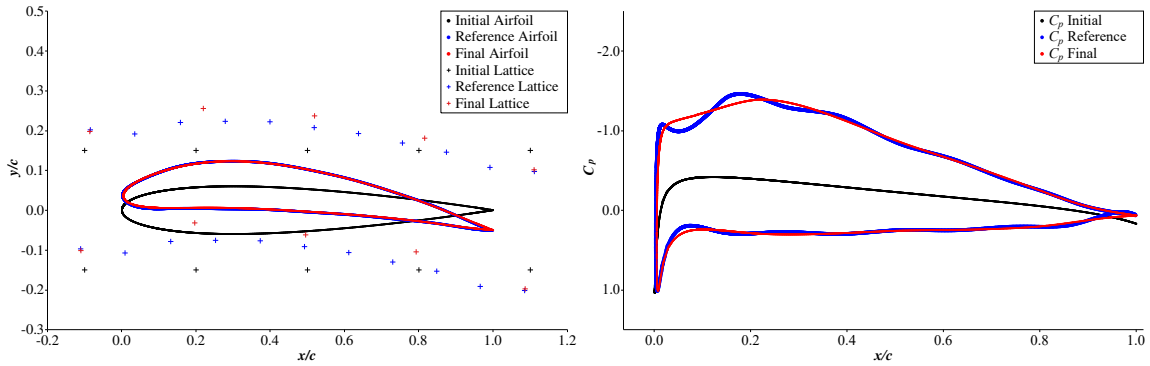
resulting problems require full quadrature evaluation for each residual and Jacobian evaluation in Newton-like iteration when solving the nonlinear problem. This scales with $N_q$.

The trend of cost vs initial TR size is also less obvious, as the cost is not purely associated with model updates. The number of low-fidelity model solves is highly variable, as before; however, it now has a substantial impact on the cost. In one ETR case, one particular sub-problem has difficulty converging. This results in a very high cost, despite the number of HFM solves being normal. This happened in hyperreduced cases as well (for example, see Table 4.2, for $n_{\text{warmup}} = 0$), but it had a negligible effect due to the cheap cost of hyperreduced function evaluations.

This shows that HROM does offer some speedup over naive ROM, and is able to more consistently bring the computational cost below the HFM baseline. However, the cost of the EQP significantly detracts from the improved cost associated with the HROM. The trade-offs between a large number of HROM solves over a small number of HFM solves, or between expensive EQP and expensive full quadrature integration, will be highly problem-dependent. Regardless, we have conclusively shown that both variants of the ROM framework, with and without hyperreduction, do consistently converge to the optimal solution, as expected for a surrogate model that satisfies the ZOC and FOC conditions and hence provides a convergence guarantee.

## 4.3 RANS Equations: Pressure Matching Inverse Design

In this section, we consider a similar problem to the inverse design problem in Section 4.2, only with several additional complications, most notably that the governing equations are now the Reynolds-averaged Navier-Stokes (RANS) equations. The goal here is to push the method to the limit of what has been achieved in the ROM-accelerated optimization literature so far.




(a) Reference, initial, & HFM-based optimal geometry (b) Reference, initial, & HFM-based optimal $C_p$ distribution

Figure 4.15: RANS-based pressure matching optimization, $\mathrm{Re} = 2 \times 10^5$, $\epsilon_{\mathrm{opt}} = 10^{-3}$.

### 4.3.1 Problem Definition

The objective here is to reproduce the geometry for NACA6412 starting with NACA0012, based on the pressure. Once again, this is an example of an inverse design problem. The output is the same as in Section 4.2 (see (4.4)), with $p_{\mathrm{ref}}$ generated on the NACA6412 geometry.

The governing equations are the steady-state, fully turbulent RANS equations using the Spalart-Allmaras (SA) turbulence model [58, 3]. In total we have five coupled unknown fields: density $\bar{\rho}$, momentum $\bar{\rho u}$ in $x$- and $y$-directions, energy $\bar{E}$, and the SA variable $\nu$ which is used to compute the turbulent viscosity.

Figure 4.15a shows the geometry of the airfoil and the parametrization. To avoid the optimizer taking advantage of mesh dependency, we use the same mesh for optimization as we used to generate the reference solution. To this end, we do not use the exact geometry of NACA6412, but apply an FFD transformation to a NACA0012 which we know to give the best approximation of NACA6412, in the Euclidean sense of geometry comparison. (This involves the solution of another very simple optimization problem.) We use a much richer FFD parametrization to generate the reference (with a 2 x 11 lattice) than we use for the main optimization problem (a 2 x 5 lattice), meaning that we get a much better approximation to generate the reference than will be attainable later. Since we are using cubic B-splines to construct this airfoil, we refer to it as Cu-NACA6412, to distinguish it from the true analytical NACA6412 profile. In other words, we have one reference mesh which we continuously deform, but multiple FFD lattices – one for generating $p_{\mathrm{ref}}$ (visualized in Figure 4.15b), and one for solving the optimization problem.

The end points were fixed relative to the chord line in both cases, and only vertical motion relative to the chord line is permitted for the interior control points. The angle of attack $\alpha$ is also

now a design variable. The angle of attack is controlled not by controlling the flow direction at the far-field boundary, as was done in Section 4.2, but by rotating the entire FFD lattice, as controlled by a single variable. The other variables are the motion of the lattice points in the direction normal to the chord line, not to the freestream direction.

The reference pressure distribution on the right in Figure 4.15b shows some variability. This could be due to imperfections in the solution of the geometry optimization problem with the rich B-spline parametrization. However, this only serves to make our optimization problem more challenging so we accept it.

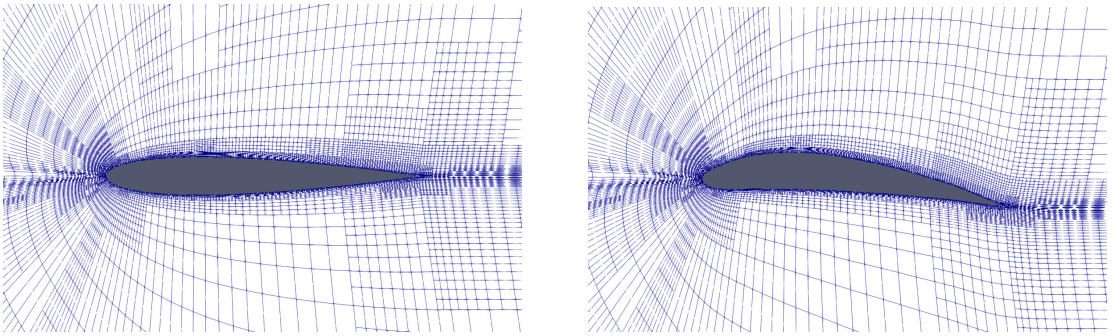
As seen in Figure 4.15a, the final geometry as determined by the HFM-baseline problem converges to a slightly different rotation angle of the lattice, with the remaining points more displaced in the positive vertical direction, in an attempt to induce the same curvature. The geometry of the airfoil shapes were therefore offset, and for the purposes of the figure are shifted vertically to be super-imposed for straightforward comparison. The lattices were not shifted. The profiles were not corrected for angle of attack relative to the initial NACA0012 guess, as our interest is how well the geometry was reproduced, including the angle. In other words, it is not surprising that pressure matching does not guarantee that the airfoil will be in the same *position*, but it should yield the same *shape* and *orientation*.

This problem may seem similar to the problem in Section 4.2, but there are three additional complications. Firstly, the RANS equations introduce anisotropic flow features, both in the boundary layer and wake regions. These features are much more difficult to capture with a ROM. Approximating sharp features (e.g., wake) whose location depends on the parameter poses a challenge for ROM, and is an area of ongoing reserach [44].

Secondly, the reference pressure distribution is generated by a geometry which is not possible to reproduce exactly in the optimization parameter space. It is difficult to know what effect this has on the difficulty of the optimization problem. The lack of an exact solution may make the region near the global minimum, commonly known as the "basin of Newton attraction", less pronounced.

Thirdly, the inclusion of the angle of attack as a parameter has several effects on the optimization problem:

- The design variables do not all scale equally.
- There is more global parameter coupling — the final position of a control point depends on its own local parameter and on $\alpha$.
- The wake may be considered "moving", although it does not move as much as if we were



(a) NACA0012 at $\alpha = 0°$ (b) Cu-NACA6412 at $\alpha = 5°$

Figure 4.16: Mesh used for RANS pressure matching problem with $\text{Re} = 2 \times 10^5$, $\mathcal{N} = 77{,}940$

to rotate the boundary condition instead of rotating the FFD lattice. This mitigates the challenges to ROM, and is the reason why we use this approach to parametrizing $\alpha$.

Higher Reynolds number flows will make the problem more difficult to reduce. For this reason, we consider two Reynolds numbers: $\text{Re} = 2 \times 10^5$ and $\text{Re} = 1 \times 10^6$. We shall compare the effect this has on the HROM. The freestream Mach number is fixed at $M_\infty = 0.2$, and the flow is fully turbulent (i.e., we do not model transition from laminar to turbulent flow).

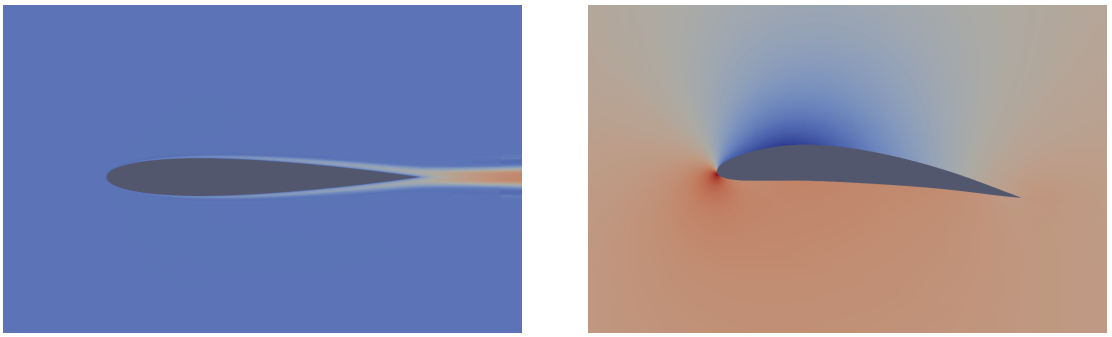
### 4.3.2 High Fidelity Baseline

The RANS equations require a much finer mesh in order to capture the boundary layer and wake regions. We once again apply an $h$-adaptive DG method with piecewise quadratic basis functions described in Section 2.3, using the AMR strategy described in Section 2.3.3. For $\text{Re} = 2 \times 10^5$, we obtain a mesh with $\mathcal{N} = 77{,}940$ DOF. Figure 4.16 shows the undeformed and deformed meshes. The angle of attack induced by the FFD transform is also visible. The whole domain is not shown – a C-shaped domain with a far-field radius of 30 chord lengths is used similar to the one used in Section 4.2.

Sample results of state variables are shown in Figure 4.17. The turbulent wake is clearly visible behind the trailing edge. The initial, reference, and final geometry and $C_p$ distributions have already been shown in Figure 4.15b, which uses the mesh shown in Figure 4.16.

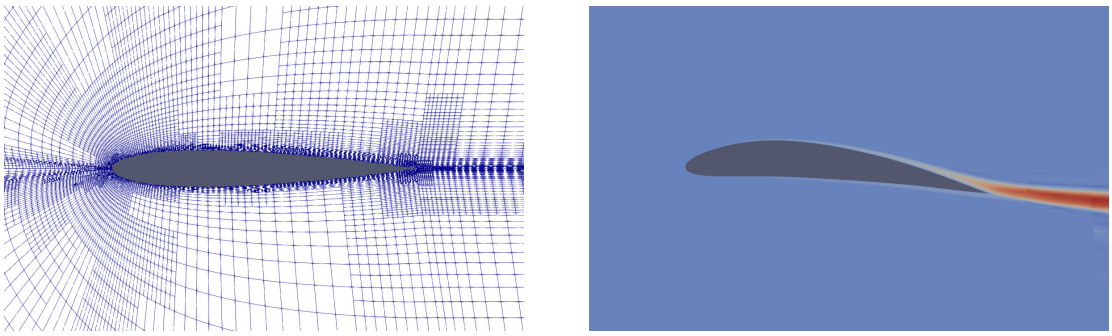
The mesh for $\text{Re} = 1 \times 10^6$ and one of the state variables is shown in Figure 4.18. This mesh has $\mathcal{N} = 138{,}960$ DOF and is generated in the same way as the one used for $\text{Re} = 2 \times 10^5$. The additional resolution is required because of the thinner boundary layer.

It is difficult to comment on the accuracy of both meshes in quantitative terms, for several reasons:



(a) SA variable $\nu$ for NACA0012 at $\alpha = 0°$ (b) Pressure for Cu-NACA6412 at $\alpha = 5°$

Figure 4.17: State visualization with $\mathrm{Re} = 2 \times 10^5$

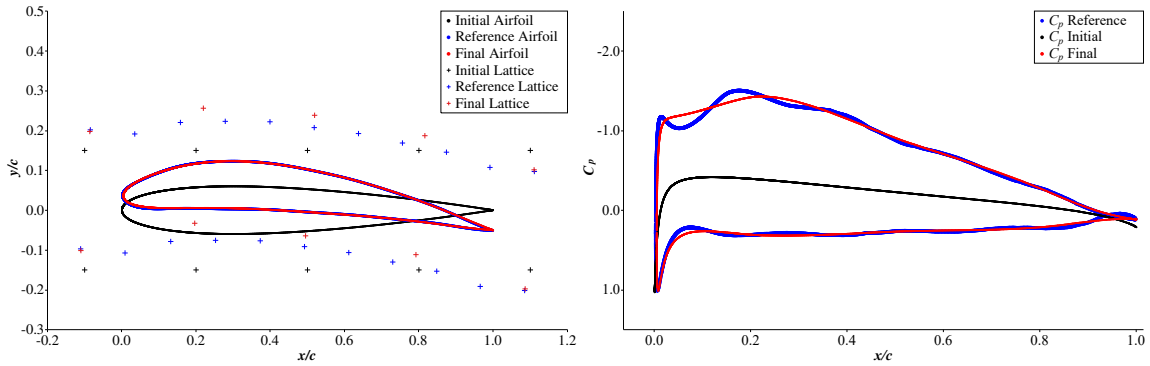


(a) Mesh used for NACA0012 with $\alpha = 0°$ (b) SA variable $\nu$ for Cu-NACA6412 with $\alpha = 5°$

Figure 4.18: Mesh and state visualization for $M_\infty = 0.2$, $\mathrm{Re} = 1 \times 10^6$, $\mathcal{N} = 138{,}960$

- We are interested in parametric geometry studies where we use the same mesh for all parameters. An accuracy figure reported for one value of the parameter does not necessarily hold for other values of the parameter.
- Most applied aerodynamicists are interested in the accuracy of the mesh in computing the drag, whereas here we are interested in the pressure difference.
- The mesh must be refined *before* the reference solution $p_{\mathrm{ref}}$ can be obtained. This means it is impossible to refine with $(p - p_{\mathrm{ref}})^2$ as the quantity of interest.

Specifically due to the last point, the mesh is refined based on absolute pressure. However, the accuracy of the meshes shown in Figures 4.16 and 4.18a in calculating the drag using the meshes used were compared to much finer meshes adapted specifically to target drag. As a result, we can say with some confidence that the meshes used are within at most 5% accuracy in computing drag for most parameter values, and more likely within 1%. Unfortunately more concrete statements cannot be provided.




(a) Reference, initial, & HFM-based optimal geometry (b) Reference, initial, & HFM-based optimal $C_p$ distribution

Figure 4.19: RANS-based pressure matching optimization, $\text{Re} = 1 \times 10^6$, $\epsilon_{\text{opt}} = 10^{-2}$

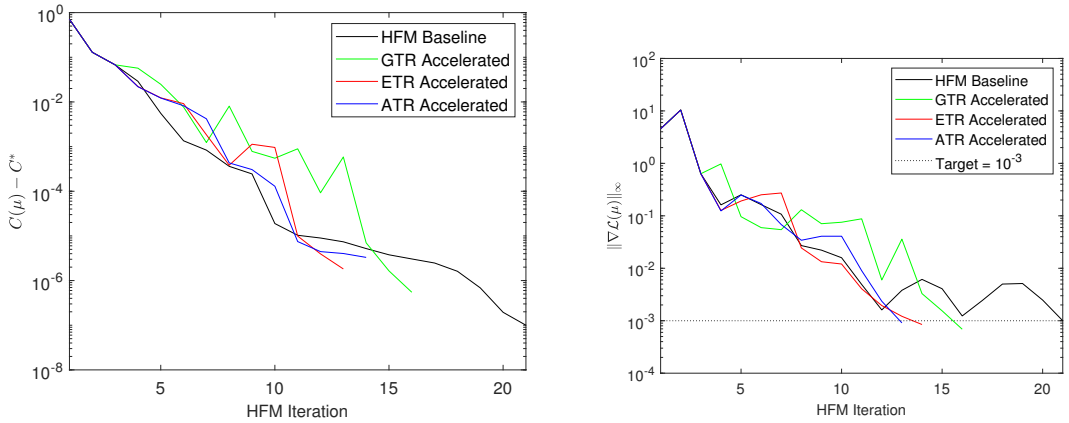
The results shown in Figure 4.15 are for when we solve the optimization problem to a first-order optimality of $\epsilon_{\text{opt}} = 10^{-3}$. It is interesting to ask whether it is necessary to progress the optimizer this far, given the fact that the resulting airfoil is nearly an exact match. The same problem is solved to $\epsilon_{\text{opt}} = 10^{-2}$, but the results are virtually indistinguishable from Figure 4.15 and so the figure is not repeated.

Figure 4.19 shows a similar result for the $\text{Re} = 1 \times 10^6$ case, with the optimization problem solved to a tolerance of $\epsilon_{\text{opt}} = 10^{-2}$. All computations in this section are performed on a single node of the Niagara cluster at SciNet with 40 CPU cores.

### 4.3.3 TR-Based HROM-Accelerated Optimization

We now apply our newly-developed HROM-accelerated optimization framework (Algorithm 8) to this problem. The algorithm settings are as follows:

- An NNLS tolerance of $\varepsilon_{\text{FOC}} = 10^{-6}$ is used for FOC-critical constraints, and $\varepsilon = 10^{-4}$ for non-critical constraints.
- Additional EQP constraints includes the Jacobian (3.9), weighted by the inverse Jacobian, and the direct residual sensitivity (3.10) at $\mu_{\text{TR}}$.
- The GSO tolerance is set to $\varepsilon_{\text{ROM}} = 10^{-6}$. GSO was used rather than POD.
- All three TR variants are tested. Both the GTR and ETR use an initial TR size of $\Delta_0 = 0.01$, with a maximum GTR size of $\Delta_{\text{maxGTR}} = 0.07$ and a minimum ETR size of $\Delta_{\text{minETR}} = 10^{-4}$.




(a) Objective function (b) First order optimality

Figure 4.20: Convergence of the output and optimality for $\text{Re} = 2 \times 10^5$, $\epsilon_{\text{opt}} = 10^{-3}$ for the HFM baseline and the three TR variants

Table 4.3: Results for RANS Pressure-Matching Optimization, $\text{Re} = 2 \times 10^5$, $\epsilon_{\text{opt}} = 10^{-3}$

| **Case** | **# HFM** | **# HROM** | $t_{\textbf{HFM}}$ [s] | $t_{\textbf{HROM}}$ [s] | $t_\eta$ [s] | $t_{\textbf{train}}$ [s] | $t_{\textbf{total}}$ [s] |
|---|---|---|---|---|---|---|---|
| HFM | 21 | - | 341.3 | - | - | - | 341.3 |
| GTR | 16 | 91 | 272.3 | 49.5 | - | 122.6 | 444.9 |
| ETR | 13 | 127 | 222.5 | 69.4 | 12.3 | 64.5 | 369.1 |
| ATR | 14 | 142 | 246.2 | 81.0 | 14.1 | 127.6 | 469.1 |

- The EQP is applied separately to elements and facets, resulting in two smaller NNLS problems for each model update.

The convergence plot for the $\text{Re} = 2 \times 10^5$, $\epsilon_{\text{opt}} = 10^{-3}$ case is shown in Figure 4.20, including both objective function (offset by a constant $C^*$ for logarithmic scaling) and first order optimality versus HFM solves. The equivalent plot for the thermal fin problem in Figure 4.8 shows that the benefit of ROM only starts taking effect after a certain threshold number of HFM solves. The same is true here, with that threshold being higher, making reduction quite difficult. We show results for $\epsilon_{\text{opt}} = 10^{-3}$, but results are the same, only truncated, for $\epsilon_{\text{opt}} = 10^{-2}$. In that case, the number of HFM solves is not reduced at all.

The time breakdown and number of HFM and HROM evaluations for all three cases are shown in Tables 4.3, 4.4, and 4.5. To summarize the results, this algorithm in fact took longer to solve the problem than the HFM-baseline. As the algorithm currently stands, then, we cannot present this as a "successful" result in terms of HROM-accelerated optimization.

For the $\text{Re} = 2 \times 10^5$ case, the algorithm converged to the same optimal solution, to the first order optimality requested, the solution shown in Figure 4.15, regardless of the type of TR used. The convergence guarantee of our algorithm has extended to this RANS problem.

Table 4.4: Results for RANS Pressure-Matching Optimization, $\mathrm{Re} = 2 \times 10^5$, $\epsilon_{\mathrm{opt}} = 10^{-2}$

| **Case** | **# HFM** | **# HROM** | $t_{\mathbf{HFM}}$ [s] | $t_{\mathbf{HROM}}$ [s] | $t_\eta$ [s] | $t_{\mathbf{train}}$ [s] | $t_{\mathbf{total}}$ [s] |
|---|---|---|---|---|---|---|---|
| HFM | 11 | - | 184.5 | - | - | - | 184.5 |
| GTR | 13 | 82 | 220.7 | 50.3 | - | 87.0 | 358.1 |
| ETR | 11 | 72 | 192.7 | 39.0 | 6.4 | 47.3 | 285.6 |
| ATR | 11 | 177 | 198.3 | 98.5 | 14.3 | 63.8 | 375.1 |

Table 4.5: Results for RANS Pressure-Matching Optimization, $\mathrm{Re} = 1 \times 10^6$, $\epsilon_{\mathrm{opt}} = 10^{-2}$

| **Case** | **# HFM** | **# HROM** | $t_{\mathbf{HFM}}$ [s] | $t_{\mathbf{HROM}}$ [s] | $t_\eta$ [s] | $t_{\mathbf{train}}$ [s] | $t_{\mathbf{total}}$ [s] |
|---|---|---|---|---|---|---|---|
| HFM | 11 | - | 703.7 | - | - | - | 703.7 |
| GTR | 13 | 59 | 851.8 | 61.7 | - | 132.0 | 1,045.6 |
| ETR | 13 | 166 | 859.5 | 142.4 | 32.6 | 142.4 | 1,209.3 |
| ATR | (did not converge) | | | | | | |

Although the algorithm failed to reduce the total computational time, it did reduce the number of HFM solves in the case of solving to $\epsilon_{\mathrm{opt}} = 10^{-3}$. As already noted, this benefit does not extend to the case of solving $\epsilon_{\mathrm{opt}} = 10^{-2}$. It is reasonable to ask in this problem whether $\epsilon_{\mathrm{opt}} = 10^{-3}$ is necessary, since the pressure match is quite satisfactory with $\epsilon_{\mathrm{opt}} = 10^{-2}$. However, this depends on the user's judgement, and would be highly context-dependent.

Although in many cases the HFM baseline and TR methods use the same number of HFM solves, the TR method takes more computational time. The additional time comes from both HROM function evaluations and training time. Error estimation time was negligible. The high cost associated with HROM solves can be primarily attributed to the FFD transformation. The cost of FFD was not taken into account during algorithm development. This still scales with the number of quadrature points $N_q$. Further effort should be devoted to increasing the speed of the FFD transformation. This will reduce both the HROM and HFM evaluation times, but will have a more substantial effect on HROM solve time. The average time for a single HROM evaluation (0.57 s) is about 30× lower than the average time for a single HFM evaluation (16.8 s) on the mesh with $\mathcal{N} = 77,940$ DOF.

The second source of increased cost is associated with model training, which predominantly comes from the EQP. This is a subject of ongoing research, and provides the largest possible area for improvement of HROM-accelerated methods. Our inclusion of the inverse Jacobian EQP constraint means training cost is expected to rise with at least $N_{\mathrm{pr}}^2$, which is why it is so much larger for the ATR case than for the ETR case. This $N_{\mathrm{pr}}^2$-dependency of EQP training should be removed by selecting better auxiliary accuracy constraints. These constraints are added to improve the stability of the HROM, so stability of HROM itself would need to be better understood in order to identify

more intelligent constraints on the EQP.

Comparing the $\mathrm{Re} = 2 \times 10^5$ case with the $\mathrm{Re} = 1 \times 10^6$ case (specifically Tables 4.4 and 4.5, which are for the same $\epsilon_{\text{opt}}$), we see that the HFM baseline and the GTR method are unaffected by the change in Reynolds number, but the ETR method requires two more HFM solves and stability issues led the ATR method to fail (i.e., to not converge within 30 HFM solves). It is somewhat by chance that the GTR cost remained unchanged, but overall it is not surprising that the HROM had more difficulty modelling the higher Reynolds number case.

Comparing the GTR, ETR, and ATR, the ATR is the least robust, failing to converge for $\mathrm{Re} = 1 \times 10^6$. We attribute this to inaccuracy in the error estimator. The ATR heavily relies on quantitative accuracy of $\eta_N$ compared to $q(u_h) - q(\tilde{u}_N)$, given that it does not have an adaptive TR size update rule. The adaptive TR size update rule allows the ETR to use $\eta_N$ as a relative indication of error, without putting too much faith in the exact value of $\eta_N$. If $N_{\text{pr}}$ becomes too high, the EQP training becomes more difficult. This may be the reason for the failure to converge.

The ETR performed better than the GTR, and so did the ATR when it was stable. This may be due to the fact that we have global coupling in the parameter domain and parameters that scale differently (due to the inclusion of $\alpha$ as a parameter). Tying the TR specifically to the PDE output error does suggest a more meaningful TR. However, we also advise the reader to take these explanations and results as provisional, because with such a low number of HFM solves to begin with, some of the variation could be purely due to chance (similar to the variation observed in Section 4.2).

Despite the failure to see concrete computational improvement in a problem at this scale, we have demonstrated that the method works at all in this context. We emphasize that our comparison is with respect to a state-of-the-art PDE constrained optimization method based on an adaptive high-order DG method and interior point method, which minimizes the number of HFM solves needed to generate the baseline. We believe these results present fertile ground for continued research, and push the boundaries of what has been achieved to date in the ROM-based optimization literature.

# Chapter 5

# Conclusion and Future Work

## 5.1 Summary

In Chapter 2, we reviewed the key mathematical ingredients of the FEM and projection-based ROM. We described one type of hyperreduction, namely the EQP, which enables model order reduction of nonlinear PDEs. We described FFD-based geometry transformation methods for both FEM and ROM, with an emphasis on the structure required for gradient-based optimization. We derived the residual sensitivity terms needed to compute the output gradient using the adjoint method. We described a trust region method based on the interior point method for optimization.

In Chapter 3, we then integrated these various ingredients into an HROM-accelerated optimization framework. We devised an approach to construct the two key ingredients of HROM — reduced basis and reduced quadrature — to ensure the ZOC and FOC conditions required for the convergence of the TR method are satisfied. To construct the RB, we apply POD or GSO to a judiciously chosen set of snapshots to ensure that the dual of the reduced problem also exists in the primal RB space, ensuring exact reproduction at the TR center. To construct the reduced quadrature rule, we apply the EQP with a set of constraints which are also chosen to guarantee ZOC and FOC directly. We then demonstrated two possible ways of using the *a posteriori* error estimator $\eta_N(\mu)$ to construct a trust region, besides the conventional geometric trust region.

In Chapter 4, we then demonstrated computational speedup resulting from this integrated framework on a nonlinear thermal fin problem and a pressure matching problem with the Euler equations. The thermal fin problem has been studied before, but with a linear PDE and therefore without hyperreduction [51]. The pressure matching problem with Euler equations had also been studied, but

again without hyperreduction, leaving $N_q$-dependency in the residual evaluation [70]. In addition, we investigated the local properties of the HROM in the region near the TR center. We investigated the effect of different tuning parameters, such as the TR definition, the number of initial HFM "warmup" solves, and the initial TR size. We then sought to push the algorithm to its limits, by applying it to the context of RANS-constrained flow over a 2D airfoil with a variable angle of attack. The method was demonstrated to be functional, although not yet preferable to HFM-based optimization from a cost perspective.

## 5.2 Future Work

We noted in Chapters 3–4 several areas of the current form of the ROM-accelerated optimization algorithm (Algorithm 8) that need further work. Integrating both POD and GSO to get the benefits of compression while still retaining exact ZOC and FOC should be investigated. More research is required on what extra constraints beyond the FOC-criticial constraints are best to place on the EQP problem. The failure of the ATR for the RANS case demonstrates the need to ensure that the HROM is stable for high values of $N_{\text{pr}}$. Faster FFD transformations would improve the cost of the HROM solves, making them almost negligible.

In Section 4.3, we observed that the current method is incapable of accelerating the solution of pressure-matching inverse design problem governed by the RANS equations; however, it is capable of solving the problem at least to a certain optimality level. In the remainder of this section, we inform the reader of problems we attempted to solve with HROM and trust regions, and were completely unsuccessful. We hope that this provides a starting point for future research, and outlines some of the challenges ahead, beyond those already discussed in Section 4.3.

One problem attempted was the Liebeck pressure matching problem, where $p_{\text{ref}}$ is generated analytically [61]. The goal of the Liebeck design problem is to maximize the lift over drag ratio at the point just before flow separation. For this problem, we found the optimizer to eventually stall, even in the HFM-only case. This demonstrates the need to carefully define the objective function, and carefully select a suitable parameter space capable of generating the kind of solution sought. More fine-tuning is required on the IP optimizer as well.

Another problem attempted was lift-constrained drag minimization, as well as lift-over-drag maximization, of a 2D airfoil in RANS flow, given a fixed minimum thickness. While the HFM baseline converged, the new HROM-accelerated method failed. Lift-over-drag maximization is a difficult application for surrogate models, due to the accuracy required on drag. HROM sometimes

produces slightly negative drag, a non-physical result that the optimizer will certainly attempt to exploit. Also, the optimizer is not robust enough to be able to handle the asymptotic nature of the objective function. Very large numbers of HROM solves are conducted trying to find the point of "0 drag", eventually resulting in an ill-suited training point. The model updates, and the process repeats.

Both lift-constrained drag minimization and lift-over-drag maximization require the solution of two adjoint problems. The formulation and algorithm we presented here extend easily to the case of two adjoints – all adjoint-related steps are simply duplicated. However, in practice this was found to adversely affect the stability of the ROM. Inclusion of more basis functions in $\mathcal{V}_N^{\text{pr}}$ which bear no resemblance to $u_h$ seems to result in instability. More research is required on why this stability problem arises and how to correct it or compensate for it.

We also tried variable fidelity optimization, where the AMR-driven mesh generation works concurrently with the HROM-based optimization. However, this often results in the HROM finding training points that take advantage of the coarse mesh. The mesh then refines in that area, and the HROM moves somewhere else in $\mathcal{D}$ where the mesh is still poor, resulting in a game of "whack-a-mole". The richer the parameter space, the more opportunity for this arises. The training points end up being of airfoils that are extremely thick, twisted, with a bulge in one particular spot, and so on. Ideally the mesh refinement would cause the solver to realize that this is not a good point, but this results in much more mesh refinement than is necessary if we fix the parameters while refining the mesh. Problems of this sort are ubiquitous in aerodynamic optimization, and are not unique to this method [22].

Much state-of-the-art computational aerodynamic optimization research being done currently by groups who are prepared to expend large computational resources using HFM-based optimization are already studying 3D wing configurations [26], full aircraft configurations [33], aerostructrual optimization [37], and noise reduction [71]. The problems we have considered in this thesis are a far cry from these real-world problems. But the methods used in those solvers have been in development for many years. While ROM will never displace those well-established methods, it is ready to complement them. Our work helps add maturity to HROM-accelerated optimization, advancing it closer to the level of usefulness in industrially relevant optimization problems.

# Bibliography


[1] A. Agarwal and L. Biegler. A trust-region framework for constrained optimization using reduced order modeling. *Optimization in Engineering*, 14:3–35, 2011.

[2] N. M. Alexandrov, J. E. Dennis, R. Lewis, and V. Torczon. A trust-region framework for managing the use of approximation models in optimization. *Structural Optimization*, 15:16–23, 1998.

[3] S. R. Allmaras, F. T. Johnson, and P. R. Spalart. Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model. 7th International Conference on Computational Fluid Dynamics ICCFD7-1902, ICCFD, 2012.

[4] D. Amsallem, M. Zahr, Y. Choi, and C. Farhat. Design optimization using hyper-reduced-order models. *Struct. Multidiscip. Optim.*, 51(4):919–940, 2015.

[5] S. S. An, T. Kim, and D. L. James. Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. Graph.*, 27(5):165:1–165:10, 2008. ISSN: 0730-0301.

[6] G. Anderson, M. Aftosmis, and M. Nemec. Parametric deformation of discrete geometry for aerodynamic shape design. In *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*. American Institute of Aeronautics and Astronautics, 2012.

[7] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini. Unified analysis of discontinuous Galerkin methods for elliptical problems. *SIAM Journal on Numerical Analysis*, 39(5):1749–1779, 2002.

[8] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera. An "empirical interpolation" method: application to efficient reduced-basis discretization of partial differential equations. *C. R. Acad. Sci. Paris, Ser. I*, 339:667–672, 2004.

[9] T. J. Barth. Numerical methods for gasdynamic systems on unstructured meshes. In D. Kröner, M. Ohlberger, and C. Rohde, editors, *An Introduction to Recent Developments in Theory and Numerics for Conservation Laws*, pages 195–282. Springer-Verlag, 1999.

[10] F. Bassi, A. Crivellini, S. Rebay, and M. Savini. Discontinuous Galerkin solution of the Reynolds-averaged Navier-Stokes and $k$-$\omega$ turbulence model equations. *Computers & Fluids*, 34(4-5):507–540, 2005.

[11] R. Becker and R. Rannacher. An optimal control approach to a posteriori error estimation in finite element methods. *Acta Numerica*, 10:1–102, 2001. A. Iserles, editor.

[12] P. Benner, S. Gugercin, and K. Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 57(4):483–531, 2015.

[13] F. Brezzi, J. Rappaz, and P. A. Raviart. Finite dimensional approximation of nonlinear problems. Part I: Branches of nonsingular solutions. *Numerische Mathematik*, 36:1–25, 1980.

[14] T. Bui-Thanh, M. Damodaran, and K. Willcox. Proper orthogonal decomposition extensions for parametric applications in compressible aerodynamics. In *21st AIAA Applied Aerodynamics Conference*, number 2003-4213. American Institute of Aeronautics and Astronautics, 2003.

[15] K. Carlberg, C. Bou-Mosleh, and C. Farhat. Efficient non-linear model reduction via a least-squares Petrov-Galerkin projection and compressive tensor approximations. *International Journal for Numerical Methods in Engineering*, 86(2):155–181, 2011. ISSN: 1097-0207.

[16] K. Carlberg, C. Farhat, J. Cortial, and D. Amsallem. The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows. *Journal of Computational Physics*, 242:623–647, 2013.

[17] M. Ceze and K. J. Fidkowski. Constrained pseudo-transient continuation. *International Journal for Numerical Methods in Engineering*, 102(11):1683–1703, 2015. ISSN: 1097-0207. nme.4858.

[18] T. Chapman, P. Avery, P. Collins, and C. Farhat. Accelerated mesh sampling for the hyper reduction of nonlinear computational models. *International Journal for Numerical Methods in Engineering*, 109(12):1623–1654, 2017.

[19] S. Chaturantabut and D. C. Sorensen. Nonlinear model reduction via Discrete Empirical Interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010.

[20] B. Cockburn. Discontinuous Galerkin methods. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 83(11):731–754, 2003.

[21] C. Daversin-Catty and C. Prud'homme. Simultaneous empirical interpolation and reduced basis method for non-linear problems. *C. R. Acad. Sci. Paris, Ser. I*, 353:1105–1109, 12, 2015.

[22] M. Drela. *Frontiers of computational fluid dynamics*. In D. Caughey and M. Hafez, editors. World Scientific, 1998. Chapter Pros and cons of aerodynamic optimization.

[23] Z. Drmax and S. Gugercin. A new selection operator for the discrete interpolation method - improved a priori error bound and extensions. *SIAM Journal on Scientific Computing*, 38(2):A631–A648, 2016.

[24] E. Du. *A model reduction framework with the empirical quadrature procedure for high-dimensional shape-parameterized partial differential equation*. Master's thesis, University of Toronto, 2020.

[25] E. Du, M. Sleeman, and M. Yano. Adaptive discontinuous-Galerkin reduced-basis reduced-quadrature method for many-query CFD problems. In *AIAA AVIATION 2021 FORUM*. American Institute of Aeronautics and Astronautics, 2021.

[26] B. Epstein, A. Jameson, S. Peigin, D. Roman, N. Harrison, and J. Vassberg. Compartive study of three-dimensional wing drag minimization by different optimization techniques. *Journal of Aircraft*, 46(2):526–541, 2009.

[27] R. Everson and L. Sirovich. Karhunen-Loève procedure for gappy data. *Journal of the Optical Society of America A, Optics and Image Science*, 12(8):1657–1664, 1995.

[28] H. Fareed. *Incremental proper orthogonal decomposition for PDE simulation data: algorithms and analysis*. Doctoral dissertation, Missouri University of Science and Technology, 2018.

[29] C. Farhat, P. Avery, T. Chapman, and J. Cortial. Dimensional reduction of nonlinear finite element dynamic models with finite rotations and energy-based mesh sampling and weighting for computational efficiency. *International Journal for Numerical Methods in Engineering*, 98(9):625–662, 2014. ISSN: 1097-0207.

[30] C. Farhat, T. Chapman, and P. Avery. Structure-preserving, stability, and accuracy properties of the energy-conserving sampling and weighting method for the hyper reduction of nonlinear finite element dynamic models. *International Journal for Numerical Methods in Engineering*, 102(5):1077–1110, 2015. ISSN: 1097-0207. nme.4820.

[31] K. Fidkowski and D. Darmofal. Review of output-based error estimation and mesh adaptation in computational fluid dynamics. *AIAA Journal*, 49(4):673–694, 2011.

[32] C. Gogu. Improving the efficiency of large scale topology optimization through on-the-fly reduced order model construction. *International Journal for Numerical Methods in Engineering*, 101(4):281–304, 2015.

[33] F. Götten, D. F. Finger, M. Havermann, C. Braun, M. Marino, and C. Bil. Full configuration drag estimation of short-to-medium range fixed-wing uavs and its impact on initial sizing optimization. *CEAS Aeronautical Journal*, 12:586–603, 1969.

[34] R. Hartmann and P. Houston. Error estimation and adaptive mesh refinement for aerodynamic flows. In H. Deconinck, editor, *VKI LS 2010-01: 36$^{th}$ CFD/ADIGMA course on hp-adaptive and hp-multigrid methods, Oct. 26-30, 2009*. Von Karman Institute for Fluid Dynamics, Rhode Saint Genèse, Belgium, 2009.

[35] J. S. Hesthaven, G. Rozza, and B. Stamm. *Certified reduced basis methods for parametrized partial differential equations*. Springer, 2016.

[36] C. T. Kelley and D. E. Keyes. Convergence analysis of pseudo-transient continuation. *SIAM Journal on Numerical Analysis*, 35(2):508–523, 1998.

[37] S. Khosravi and D. W. Zingg. Aerostructural optimization of drooped wings. *Journal of Aircraft*, 55(3):1261–1268, 2018.

[38] C. L. Lawson and R. J. Hanson. *Solving least squares problems*. Prentice Hall, 1974. ISBN: 0138225850.

[39] P. LeGresley and J. Alonso. Airfoil design optimization using reduced order models based on proper orthogonal decomposition. In *Fluids 2000 Conference and Exhibit*. American Institute of Aeronautics and Astronautics, 2000.

[40] P. A. LeGresley and J. J. Alonso. Investigation of non-linear projection for POD based reduced order models for aerodynamics. In *39th Aerospace Sciences Meeting and Exhibit*, number 2001–0926. American Institute of Aeronautics and Astronautics, 2001.

[41] Y. Maday, O. Mula, A. Patera, and M. Yano. The generalized empirical interpolation method: stability theory on Hilbert spaces with an application to the Stokes equation. *Computer Methods in Applied Mechanics and Engineering*, 287:310–334, 2015.

[42] A. Manzoni, A. Quarteroni, and G. Rozza. Shape optimization for viscous flows by reduced basis methods and free-form deformation. *International Journal for Numerical Methods in Fluids*, 70(5):646–670, 2012.

[43] M. Meyer and H. G. Matthies. Efficient model reduction in non-linear dynamics using the Karhunen-Loève expansion and dual-weighted-residual methods. *Computational Mechanics*, 31(1-2):179–191, 2003.

[44] N. J. Nair and M. Balajewicz. Transported snapshot model order reduction approach for parametric, steady-state fluid flows containing parameter-dependent shocks. *International Journal for Numerical Methods in Engineering*, 117(12):1234–1262, 2019.

[45] N. C. Nguyen, A. T. Patera, and J. Peraire. A 'best points' interpolation method for efficient approximation of parametrized functions. *International Journal for Numerical Methods in Engineering*, 73(4):521–543, 2008. ISSN: 1097-0207.

[46] J. Nocedal and S. Wright. *Numerical optimization*. Springer-Verlag GmbH, 2006. ISBN: 0387303030.

[47] M. Ohlberger and S. Rave. Reduced basis methods: success, limitations and future challenges. In *Proceedings of the Conference Algoritmy*, pages 1–12, 2016.

[48] A. T. Patera and G. Rozza. Reduced basis approximation and a posteriori error estimation for parametrized partial differential equations. MIT Pappalardo Graduate Monographs in Mechanical Engineering, MIT, 2006.

[49] A. T. Patera and M. Yano. An LP empirical quadrature procedure for parametrized functions. *Comptes Rendus Mathematique*, 355(11):1161–1167, 2017.

[50] P.-O. Persson and J. Peraire. Newton-GMRES preconditioning for discontinuous Galerkin discretizations of the Navier-Stokes equations. *SIAM Journal on Scientific Computing*, 30(6):2709–2733, 2008.

[51] E. Qian, M. Grepl, K. Veroy, and K. Willcox. A certified trust region reduced basis approach to PDE-constrained optimization. *SIAM Journal on Scientific Computing*, 39(5):S434–S460, 2017.

[52] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced basis methods for partial differential equations*. Springer, 2016.

[53] G. Rozza, D. B. P. Huynh, and A. T. Patera. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations — Application to transport and continuum mechanics. *Archives of Computational Methods in Engineering*, 15(3):229–275, 2008.

[54] Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[55] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques - SIGGRAPH 86*. ACM Press, 1986.

[56] M. Sleeman. *Goal-oriented model reduction for time-dependent nonlinear parametrized partial differential equations*. Master's thesis, University of Toronto, 2020.

[57] M. Sleeman and M. Yano. Goal-oriented model reduction for parametrized time-dependent nonlinear partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 388:114206, 2022.

[58] P. R. Spalart and S. R. Allmaras. A one-equation turbulence model for aerodynamics flows. *La Recherche Aérospatiale*, 1:5–21, 1994.

[59] T. Taddei and L. Zhang. A discretize-then-map approach for the treatment of parameterized geometries in model order reduction. *Computer Methods in Applied Mechanics and Engineering*, 384:113956, 2021.

[60] P. Tiso and D. J. Rixen. Discrete empirical interpolation method for finite element structural dynamics. In *Topics in Nonlinear Dynamics, Volume 1*, pages 203–212. Springer New York, 2013.

[61] J. C. Vassberg and A. Jameson. Test cases for inverse aerodynamic design. *Computers and Fluids*, 223:104923, 2021.

[62] K. Veroy and A. T. Patera. Certified real-time solution of the parametrized steady incompressible Navier-Stokes equations: rigorous reduced-basis *a posteriori* error bounds. *International Journal for Numerical Methods in Fluids*, 47:773–788, 2005.

[63] K. Veroy, C. Prud'homme, D. Rovas, and A. Patera. A posteriori error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic partial differential equations. In *16th AIAA Computational Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics, 2003.

[64] K. Washabaugh, M. J. Zahr, and C. Farhat. On the use of discrete nonlinear reduced-order models for the prediction of steady-state flows past parametrically deformed complex geometries. AIAA 2016-1814, AIAA, 2016.

[65] M. Yano. Goal-oriented model reduction of parametrized nonlinear PDEs; application to aerodynamics. *International Journal for Numerical Methods in Engineering*, accepted, 2020.

[66] M. Yano. Discontinuous Galerkin reduced basis empirical quadrature procedure for model reduction of parametrized nonlinear conservation laws. *Advances in Computational Mathematics*, 45(5-6):2287–2320, 2019.

[67] M. Yano and A. T. Patera. An LP empirical quadrature procedure for reduced basis treatment of parametrized nonlinear PDEs. *Computer Methods in Applied Mechanics and Engineering*, 344:1104–1123, 2019.

[68] W. Yao, S. Marques, T. Robinson, C. Armstrong, and L. Sun. A reduced-order model for gradient-based aerodynamic shape optimisation. *Aerospace Science and Technology*, 106, 2020.

[69] Y. Yue and K. Meerbergen. Accelerating optimization of parametric linear systems by model order reduction. *Society for Industrial and Applied Mathematics Journal of Optimization*, 23(2):1344–1370, 2013.

[70] M. J. Zahr and C. Farhat. Progressive construction of a parametric reduced-order model for PDE-constrained optimization. *International Journal for Numerical Methods in Engineering*, 102(5):1111–1135, 2015.

[71] B. Y. Zhou, T. Albring, N. R. Gauger, C. R. da Silva, T. D. Economon, and J. J. Alonso. Efficient airframe noise reduction framework via adjoint-based shape optimization. *AIAA Journal*, 59(2):580–595, 2021.

[72] R. Zimmermann and S. Görtz. Non-linear reduced order models for steady aerodynamics. *Procedia Computer Science*, 1(1):165–174, 2012. ISSN: 1877-0509. ICCS 2010.

# Appendix A

# Non-Negative Least Squares (NNLS) Algorithm

In this appendix, we describe the basic operation of the NNLS algorithm, which we use to approximately solve the optimization problem,

$$\begin{aligned} \hat{\rho} &= \underset{\rho \in \mathbb{R}^n}{\arg\min} \|\rho\|_0, \\ \text{subject to } & |A\rho - b|_i \leq \varepsilon, \text{ for } i = 1, ..., m, \end{aligned} \tag{A.1}$$

where $A$ and $b$ come from the target integrands and integrals, respectively. In reality, the optimization problem being solved is the quadratic problem,

$$\hat{\rho} = \underset{\rho \in \mathbb{R}^n_{>0}}{\arg\min} \left( \rho^T A^T A \rho - A^T b \right) \tag{A.2}$$

This problem is guaranteed to be convex because $A^T A$ is positive semidefinite. The solution is typically sparse because an iterative method is used to solve the problem. The simple problem $A\rho = b$ is under-constrained, assuming $n > m$, so in theory the maximum number of non-zero entries in $\rho$ to satisfy the constraints is $m$, which would yield a fully constrained (i.e., square) system if we ignored the entries of $\rho$ set to 0. The algorithm to solve this problem is presented in Algorithm 11.

The expression $(A_P^T A_P)^{-1} A_P^T b$ in Line 8 is sometimes denoted $A_P^\dagger b$, involving the Moore-Penrose pseudo-inverse, the solution of a least-squares problem, etc.

The algorithm presented here is the "vanilla" NNLS algorithm. Various modifications are possible

**Algorithm 11:** Non-Negative Least Squares (NNLS)

**Data:** $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $\varepsilon$
**Result:** $\hat{\rho}$, sparse, non-negative solution to $A\rho - b \leq \varepsilon$

1 $\rho \leftarrow 0$
2 $w \leftarrow A^T(b - A\rho)$
3 $P \leftarrow \emptyset$
4 $R \leftarrow 1, ..., n$
5 **while** $R \neq \emptyset$ *AND* $\max(w_R) > \varepsilon$ **do**
6  $j \in R \leftarrow$ index of $\max(w_R)$ in $w$.
7  Add $j$ to $P$ and remove $j$ from $R$.
8  Define $s \in \mathbb{R}^n$ and let $s_P = (A_P^T A_P)^{-1} A_P^T b$.
9  $s_R \leftarrow 0$
10  **while** $\min(s_P) \leq 0$ **do**
11  $\alpha \leftarrow \min(\frac{\rho_i}{\rho_i - s_i})$, for $i \in P$ and $s_i \leq 0$
12  $\rho \leftarrow \rho + \alpha(s - x)$
13  Move to $R$ all indices $j$ in $P$ where $\rho_j \leq 0$
14  $s_P \leftarrow (A_P^T A_P)^{-1} A_P^T b$
15  $s_R \leftarrow 0$
16  **end**
17  $\rho \leftarrow s$
18  $w \leftarrow A^T(b - A\rho)$
19 **end**
20 $\hat{\rho} \leftarrow \rho$

to both improve efficiency and enhance functionality. For example, a series of successive pseudo-inverse problems are being solved, with columns being added one by one. This can be efficiently done using a QR factorization that successively updates with a column-adding algorithm. QR factorization can also improve the speed of calculating the residual in Line 18.

Additionally, it is possible to enforce each constraint, each of which corresponds to a row in $A$, to a different level of accuracy. To do this, the entire system is multiplied row-wise by a scaling vector of $\varepsilon_{\text{const}}/\varepsilon_{\text{desired}}$. Constraints we desire to satisfy with tighter accuracy are scaled to be larger, so they have a larger effect in the max operation in Line 5. This rescaled problem is then solved using a single value of $\varepsilon_{\text{const}}$.

# Appendix B

# FFD Transformation

The exact nature of the geometry transform is not important to the formulation of the HROM-accelerated optimization framework presented in the main body of the thesis, as discussed in Section 2.6 — the geometry transformation can be treated as a "black box", as long as it provides the required sensitivity outputs. However, in this appendix we provide the mathematical details of two types of transformation. For both the test cases in this thesis that made use of geometry transformations, we used the B-spline version; however, we did perform tests using the original Bézier Curve FFD during our research as well so we present the details of both. We also present the full 3D definitions of the transforms, despite the fact that we only considered 2D geometry in the test cases.

## B.1 Bézier Curves

The Bézier curves variant of FFD was the one used in the original publication on the subject [55]. Each point in the geometry is mapped using Bernstein polynomials, making any straight line in the original geometry a Bézier curve, with polynomial degree equal to the number of control points in that direction. An example showing the control lattice and deformed geometry is shown in Figure B.1.

The domain is first represented in normalized ($s$, $t$, $u$) coordinates, such that any point inside the control lattice in the undeformed geometry $\vec{x}_0$ can be represented as

$$\vec{x}_0 = \vec{O} + s\hat{S} + t\hat{T} + u\hat{U}. \tag{B.1}$$

It is possible to have the scaling vectors $\hat{S}$, $\hat{T}$, and $\hat{U}$ point in different directions than the original

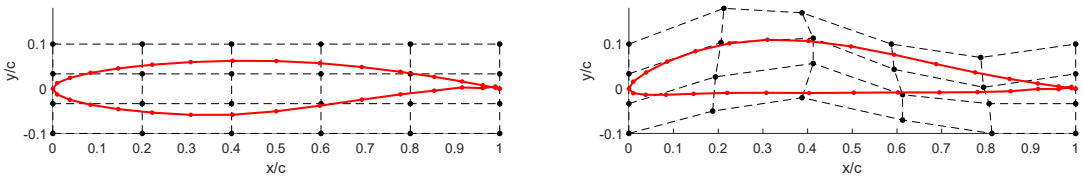



Figure B.1: RAE2822 airfoil, undeformed and subject to an FFD transformation, with control lattice shown.

coordinates, and in practice this offers a way to parametrize the rotation of the entire lattice, which we used to control the angle of attack. So these could also be parametrized; however, for simplicity, we ignore that in the derivation here. The point $\vec{O}$ contains the untransformed coordinates of the bottom left rear point in the control lattice. We can find the $s$-, $t$-, and $u$-coordinates as

$$s = \frac{\hat{T} \times \hat{U} \cdot (\vec{x}_0 - \vec{O})}{\hat{T} \times \hat{U} \cdot \hat{S}}, \; t = \frac{\hat{S} \times \hat{U} \cdot (\vec{x}_0 - \vec{O})}{\hat{S} \times \hat{U} \cdot \hat{T}}, \; u = \frac{\hat{S} \times \hat{T} \cdot (\vec{x}_0 - \vec{O})}{\hat{S} \times \hat{T} \cdot \hat{U}}. \tag{B.2}$$

We then select the number of control points for the lattice in each direction, $l$, $m$, and $n$. The location of the undeformed control points are

$$\vec{P}^0_{ijk} = \vec{O} + \frac{i}{l-1}\hat{S} + \frac{j}{m-1}\hat{T} + \frac{k}{n-1}\hat{U}. \tag{B.3}$$

Note that $\mathbf{P^0}$ is therefore a 4-D tensor, containing $d \times l \times m \times n$ scalars. If the coordinates of the deformed lattice are then contained in $\vec{P}_{ijk}$, we define $\mu$ as a flattened vector containing the difference between these two:

$$\mu \equiv I_m\big(\mathrm{vec}(\mathbf{P}) - \mathrm{vec}(\mathbf{P^0})\big). \tag{B.4}$$

The mask matrix $I_m$ is an identity matrix with rows removed corresponding to coordinates of control points that we wish to hold constant.

The Cartesian coordinates of the deformed point $\vec{x}$ are then computed using a Bernstein polynomial constructed using binomial expansions. First define a general $i$th term from a binomial expansion with $p$ terms as

$$B(x, p, i) = \binom{p}{i} (p - x)^{p-i} x^i. \tag{B.5}$$

Then we compute the coordinates of the deformed point as

$$\vec{x} = \sum_{i=0}^{l} B(s,l,i) \sum_{j=0}^{m} B(t,m,j) \sum_{k=0}^{n} B(u,n,k) \vec{P}_{ijk}. \tag{B.6}$$

Our formulation also requires the Jacobian associated with the FFD transformation $J_{\mathcal{T}}$. We may compute this as

$$J_{\mathcal{T},ij} = \frac{\partial x_i}{\partial x_{0,j}} = \frac{\partial x_i}{\partial s}\frac{\partial s}{\partial x_{0,j}} + \frac{\partial x_i}{\partial t}\frac{\partial t}{\partial x_{0,j}} + \frac{\partial x_i}{\partial u}\frac{\partial u}{\partial x_{0,j}}. \tag{B.7}$$

The first terms require differentiation of the Bernstein polynomials, and thus of the binomial terms:

$$\frac{dB(x,p,i)}{dx} = \binom{p}{i} \left( (i-p)(1-x)^{p-i-1}x^i + i(p-x)^{p-i}x^{i-1} \right). \tag{B.8}$$

We then use this to differentiate (B.6) with respect to $s$, $t$, and $u$:

$$\frac{\partial \vec{x}}{\partial s} = \sum_{i=0}^{l} \frac{dB(s,l,i)}{ds} \sum_{j=0}^{m} B(t,m,j) \sum_{k=0}^{n} B(u,n,k) \vec{P}_{ijk}, \tag{B.9}$$

$$\frac{\partial \vec{x}}{\partial t} = \sum_{i=0}^{l} B(s,l,i) \sum_{j=0}^{m} \frac{dB(t,m,j)}{dt} \sum_{k=0}^{n} B(u,n,k) \vec{P}_{ijk}, \tag{B.10}$$

$$\frac{\partial \vec{x}}{\partial u} = \sum_{i=0}^{l} B(s,l,i) \sum_{j=0}^{m} B(t,m,j) \sum_{k=0}^{n} \frac{dB(u,n,k)}{du} \vec{P}_{ijk}. \tag{B.11}$$

The second terms in (B.7) require differentiation of the transformation into normalized coordinates in (B.2):

$$\frac{\partial s}{\partial x_{0,j}} = \frac{(\hat{T} \times \hat{U})_j}{\hat{T} \times \hat{U} \cdot \hat{S}}, \quad \frac{\partial t}{\partial x_{0,j}} = \frac{(\hat{S} \times \hat{U})_j}{\hat{S} \times \hat{U} \cdot \hat{T}}, \quad \frac{\partial u}{\partial x_{0,j}} = \frac{(\hat{S} \times \hat{T})_j}{\hat{S} \times \hat{T} \cdot \hat{U}}. \tag{B.12}$$

As indicated in Section 2.6, this Jacobian will be used to transform basis function gradients (via $J_{\mathcal{T}}^{-T}$) and element areas (via $\det(J_{\mathcal{T}})$).

Both the transformed position and the Jacobian are linear in the control points (and therefore $\mu$), making it straightforward to compute $\frac{\partial \vec{x}}{\partial \mu}$ and $\frac{\partial J_{\mathcal{T}}}{\partial \mu}$. We simply do not perform the summations indicated in (B.6) and (B.9), and use the values of $i$, $j$, $k$ that correspond to the parameter (control point) of interest.

Notice several features of the FFD transform:

- Any control point affects the position of every point in the lattice.
- Moving the lattice points for example in the $x$-direction only affects the $x$-coordinates of the deformed points (coordinate independence).
- The transformation is defined by a continuous polynomial, making it smooth everywhere inside the lattice.
- If the geometry of interest lies partially outside the control lattice and we wish to maintain $C^k$ continuity in the geometry along a certain direction, we can ensure this by keeping the first $k+1$ control points along this direction fixed in that direction [55].

## B.2 B-Splines

Another FFD transformation gaining popularity in aerodynamic shape optimization uses B-splines. This method is used in Blender, an open-source graphics software used in several aerodynamic shape optimization studies [6, 64]. Rather than having the entire geometry be defined by a continuous polynomial of arbitrarily high degree (determined by the number of control points), this method uses piecewise polynomials of *fixed* degree. The result is that between lattice points, or inside a lattice "cell", a straight line maps to a Bézier curve, but not across lattice cells. Across cells, the curves are spliced with a certain level of continuity guaranteed by the choice of polynomial degree ($C = (p-1)/2$). In this work we use cubic ($p = 3$) B-splines to control the geometry, which ensure $\mathbb{C}^1$ continuity of smooth geometry both between lattice cells and outside the lattice. The coefficients are the deformation of the two control points before and after it in each dimension. (For higher dimensions, more points/more coefficients would be used).

Mathematically, we describe the process as follows. The cubic B-spline construction uses four basis functions:

$$B_0(t) = \frac{1}{6}(1-t)^3, \tag{B.13}$$

$$B_1(t) = \frac{1}{6}(3t^3 - 6t^2 + 4), \tag{B.14}$$

$$B_2(t) = \frac{1}{6}(-3t^3 + 3t^2 + 3t + 1), \tag{B.15}$$

$$B_3(t) = \frac{1}{6}(t^3). \tag{B.16}$$

Then to transform a point $\vec{x}_0 \to \vec{x}$, we define $I$, $J$, and $K$ as the index of the control point *before* the point of interest $\vec{x}$ in each direction. Even if $\vec{x}$ is outside the lattice and to the left of the 0th

node, we can say formally:

$$\{I, J, K\} = \left\lfloor \frac{\vec{x}_0 - \vec{P}_{000}}{\vec{P}_{lmn} - \vec{P}_{000}} \right\rfloor, \tag{B.17}$$

where $\lfloor \cdot \rfloor$ denotes the floor operator and all operations are performed element-wise. The transform is applied to the entire domain, not just points inside the lattice, so these numbers may be less than 0. Then let $x_0$, $y_0$, and $z_0$ represent the three components of the original point $\vec{x}_0$. We then perform

$$\vec{x} = \vec{x}_0 + \sum_{k=\max(0,K-1)}^{\min(n,K+2)} \sum_{j=\max(0,J-1)}^{\min(m,J+2)} \sum_{i=\max(0,I-1)}^{\max(l,I+2)} B_i(x_0) B_j(y_0) B_k(z_0) (\vec{P}_{ijk} - \vec{P}^0_{ijk}). \tag{B.18}$$

Further, the Jacobian associated with this transformation is as follows (with indices removed for brevity, but as in (B.18)):

$$J_{\mathcal{T}} = I + \sum_{k=\max(0,K-1)}^{\min(n,K+2)} \sum_{j=\max(0,J-1)}^{\min(m,J+2)} \sum_{i=\max(0,I-1)}^{\max(l,I+2)} (\vec{P}_{ijk} - \vec{P}^0_{ijk}) \begin{bmatrix} \frac{\partial B_i(x_0)}{\partial x_0} B_j(y_0) B_k(z_0) \\ B_i(x_0) \frac{\partial B_j(y_0)}{\partial y_0} B_k(z_0) \\ B_i(x_0) B_j(y_0) \frac{\partial B_k(z_0)}{\partial z_0} \end{bmatrix}^T \tag{B.19}$$

Once again, the sensitivities $\frac{\partial \vec{x}}{\partial \mu}$ and $\frac{\partial J_{\mathcal{T}}}{\partial \mu}$ are trivial, as $\vec{x}$ and $J_{\mathcal{T}}$ are linear in $\vec{P}_{ijk}$.

We summarize with a few observations:

- This transformation ensures $\mathbb{C}^1$ continuity across the boundary of the lattice (due to the definition of the B-spline basis functions), making it useful for cases when we wish to deform only a small part of the mesh without having to put the entire domain inside a lattice and only move a small number of control points.
- It also is known to introduce more local changes inside the lattice (up to two control points in both directions along all axes), as opposed to the classic FFD described in Section B.1, where each control point introduces changes all over the lattice.
- Again, the $x$-position of the transformed points is only affected by the $x$-movement of the control points, and so on for $y$ and $z$ (i.e., coordinate independence).
- As mentioned, the transformation affects the entire domain. The max and min operations in (B.18) describe how points outside the FFD control lattice are handled.

# Appendix C

# BFGS Hessian Update

The Broyden-Fletcher-Goldfarb-Shanno (BFGS) Hessian update algorithm provides an estimate to the Hessian, and is commonly used in optimization [46]. If given a positive definite initial Hessian estimate, the BFGS algorithm is guaranteed to always produce a positive definite Hessian estimate, even if the true Hessian is not positive definite. This is important, because it guarantees that the step direction selected will be a direction of descent.

A variation on the algorithm, which is used in this thesis, is the damped BFGS update algorithm, given in Algorithm 12 [46]. The regular BFGS algorithm can be obtained by always setting $\theta \leftarrow 1$ below. The damped BFGS algorithm is also guaranteed to generate a positive definite Hessian estimate.

**Algorithm 12:** Damped BFGS Hessian update

**Data:** $\mu_{k+1}, \mu_k, \nabla f(\mu_{k+1}), \nabla f(\mu_k), B_k$ (use $B_0 = I$)
**Result:** $B_{k+1}$, Hessian estimate at the next iterate

$s_k \leftarrow \mu_{k+1} - \mu_k$
$g_k \leftarrow \nabla f(\mu_{k+1}) - \nabla f(\mu_k)$
**if** $s_k^T g_k \geq 0.2 s_k^T B_k s_k$ **then**
    $\theta_k \leftarrow 1$
**else**
    $\theta_k \leftarrow \frac{0.8 s_k^T B_k s_k}{s_k^T B_k s_k - s_k^T g_k}$
**end**
$r_k \leftarrow \theta_k g_k + (1 - \theta_k) B_k s_k$
$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{r_k r_k^T}{s_k^T r_k}$