Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Efficient hyperreduction by empirical quadrature procedure with constraint reduction for large-scale parameterized nonlinear problems

Adrian Humphry | **Masayuki Yano**

[1]Institute for Aerospace Studies, University of Toronto, Ontario, Canada

Correspondence
Corresponding author Adrian Humphry, Apt 3107 - 1001 Bay St, Toronto, ON, M5S 3A6, Canada.
Email: humphryadrian@gmail.com

Present address
Apt 3107 - 1001 Bay St, Toronto, ON, M5S 3A6, Canada

Abstract

We develop efficient hyperreduction methods for projection-based model reduction of nonlinear partial differential equations (PDEs) with a large number of parameters and/or large parametric extents. Our formulation is based on the empirical quadrature procedure (EQP), which solves an optimization problem that involves "residual-matching constraints" over a training parameter set to find a sparse quadrature rule that yields rapid yet accurate approximations of the PDE residual, and solves the constrained optimization via non-negative least squares (NNLS). Specifically, we extend the EQP and NNLS to provide more efficient offline training for problems that (i) demand tight hyperreduction tolerances, (ii) involve a large number of residual-matching constraints, and/or (iii) involve a high-dimensional parameter space. To address (i), we develop second-order accurate constraints for EQP and a rounding-error stable NNLS formulation that efficiently provides a solution to the optimization problem with a tight tolerance. To address (ii), we develop NNLS with constraint reduction (NNLS-CR), which exploits the fact that many constraints are often redundant and systematically constructs a reduced orthogonal set of constraints that still represents all the original constraints. To address (iii), we introduce an EQP method that adaptively constructs the training parameter set and solves the associated constrained optimization problem using a version of NNLS-CR that admits incremental constraint update. We demonstrate the offline efficiency of the methods, as well as the parametric robustness of the resulting ROMs, using parameterized Navier–Stokes and Reynolds-averaged Navier–Stokes equations in four different contexts: shape parameter sweep; flight parameter sweep; ensemble-based data assimilation; and forward uncertainty quantification.

KEYWORDS
model reduction, hyperreduction, empirical quadrature procedure, non-negative least squares, high-dimensional problems

1 | INTRODUCTION

Many engineering tasks—such as parametric studies, design optimization, and uncertainty quantification—require accurate approximation of parameterized partial differential equations (PDEs) for many different parameter values. One approach to provide rapid and reliable solutions in these many-query scenarios is projection-based model-order reduction (MOR) [5, 33, 28]. MOR leverages offline–online computational decomposition: in the offline stage, which is expensive but performed only once, we invoke the full-order model (FOM) for select parameter values to learn a low-dimensional approximation of the solution manifold and construct a reduced-order model (ROM); in the online stage, we rapidly evaluate the ROM for many different parameter values. For nonlinear PDEs, MOR must also incorporate *hyperreduction*, which constructs a reduced approximation of the PDE residual. The offline training cost of hyperreduction can be significant for problems with many parameters and/or significant parametric extent; moreover, many hyperreduction methods involve user-defined parameters that must be tuned to

obtain optimal performance. The goal of this work is to develop a hyperreduction method that is more efficient (i.e., reduces offline training cost to yield a ROM of given accuracy) and automated (i.e., requires minimal user input).

Many hyperreduction methods have been developed over the last two decades. The goal of these methods is to provide a rapidly evaluatable approximation of the ROM residual, which typically arises from a (Petrov–)Galerkin projection of the (weak form of the) FOM residual onto the N-dimensional space spanned by the reduced basis (RB). Hyperreduction methods can be broadly classified into two categories. The first class of methods first approximates the functions that constitutes the residual using interpolation or regression, and then (exactly) integrates the approximated functions. Hyperreduction methods in this class include (but are not limited to) gappy POD [21, 9], the empirical interpolation method (EIM) [3, 26]; its discrete variant DEIM [13]; unassembled DEIM [38] tailored for finite element (FE) methods; QDEIM [17], which incorporates QR factorization to improve scalability of DEIM for large-scale problems; and the Gauss–Newton with approximation tensor method [11].

The second class of methods, which is the focus of the present work, directly approximates the ROM residual using a sparse quadrature rule. Hyperreduction methods in this class include (but are not limited to) the optimal cubature formulation [2]; the energy conserving sampling and weighting (ECSW) method [23, 22]; the empirical cubature method [27]; and the empirical quadrature procedure (EQP) [32, 42]. The offline stage of these direct-integration hyperreduction methods proceeds as follows: we first introduce a set of training parameter values and the associated solution training states; we next construct a set of "residual matching constraints", which measures the extent by which the reduced-quadrature (RQ) approximation violates the original full-quadrature residual for the training configurations; we then solve the constraint minimization problem that seeks to minimize the number of non-zero entries in the RQ rule while satisfying the residual-matching constraints to the target tolerance. In the online stage, the residual is approximated using the RQ rule to provide rapid approximation of the PDEs. Mathematically, the set of residual-matching constraints takes on the form $|\mathbf{A}\rho - \mathbf{b}| \leq \delta$, where $\rho \in \mathbb{R}^{K_h}$ is the set of K_h (but sparse) RQ weights sought; $\mathbf{A} \in \mathbb{R}^{m\times K_h}$ is the array of quadrature-point-wise residuals, such that $\mathbf{A}\rho \in \mathbb{R}^m$ is the RQ approximation of the m residual values; $\mathbf{b} \in \mathbb{R}^m$ is the full-quadrature residual values; and $\delta \in \mathbb{R}^m$ is the array of residual-matching tolerances.

The solution to the constrained minimization problem can be approximated using linear programming (LP) [32, 40] or non-negative least squares (NNLS) [23, 22, 12, 36]. A scalable and parallel implementation of NNLS, which also incorporates efficient incremental QR factorization, presented in [12] is the baseline method used in the present work. The NNLS algorithm employs an iterative active set method to determine a minimal set of positive quadrature weights that satisfies the given constraints [30]. Previous improvements to the NNLS algorithm include the sequential NNLS method [6], which employs QR factorization to efficiently update the solution on each iteration of NNLS, and the fast NNLS method [7], which accelerates the NNLS solution for problems with many more constraints than variables (which is not the case for hyperreduction).

In the present work, we propose three improvements to the standard NNLS algorithm for the cases that (i) demand tight tolerance δ, (ii) involve a large number of constraints, and (iii) requires sampling in a high-dimensional parameter space. The first extension involves a reformulation of the residual $\mathbf{A}\rho - \mathbf{b}$, so that the NNLS algorithm behaves numerically more stably, especially when we wish to achieve very tight tolerance. The second extension, which is arguably the key contribution of the present work, is the development of NNLS with constraint reduction (NNLS-CR) for problems that involve many constraints. The method builds on the assumption that, just like the parametric solution manifold, the residual associated with parameterized PDEs is also amenable to "low-dimensional approximation". Building on this assumption, we devise a systematic procedure to reduce the m constraints in $|\mathbf{A}\rho - \mathbf{b}| \leq \delta \in \mathbb{R}^m$ to $\tilde{m}$ orthogonalized constraints $|\mathbf{Q}\rho - \mathbf{b}_\mathbf{Q}| \leq \mathbf{b}_\mathbf{Q}$ for $\tilde{m} \ll m$ which still represent all the original constraints. We then apply NNLS to the reduced set of constraints. The third extension is the development of an incremental version of the NNLS-CR, which we name NNLS-CRi, where "i" stands for incremental. Specifically, the algorithm provides efficient NNLS solution when the set of constraints are updated/augmented incrementally in some iterative process. As we discuss next, the NNLS-CRi meshes particularly well with adaptive/incremental update of the training parameter set often demanded in high-dimensional problems.

The offline cost of RB and RQ construction (via NNLS), as well as the parametric robustness of the resulting ROM, depend on the choice of the training parameter set. The training parameter set must provide sufficient coverage of the entire parameter space to yield a parametrically robust ROM, but the set must not be excessively large to avoid excessive offline training cost. The selection of an appropriate training parameter set can be challenging, especially in high-dimensional parameter spaces. For the RB construction, one approach to provide effective sampling is a weak greedy algorithm [39], which uses an online-efficient error estimate/bound of the ROM to efficiently assess the quality of the ROM over a large set of sampling points and then chooses the next snapshot evaluation location. Extensions of the weak greedy algorithm to high-dimensional problems include that based on a "saturation assumption" [29], Hessian-based methods [10, 14], and their low-rank approximations [25, 31].

The aforementioned works on RB construction for high-dimensional parameter spaces are for linear PDEs, which do not require hyperreduction. Weak greedy algorithms have been extended to treat nonlinear PDEs by incorporating hyperreduction via EIM [15, 18] and EQP [41]; however, they do not necessarily scale well to high-dimensional problems unless an appropriate training set is somehow identified *a priori*. To this end, we propose an EQP method for high-dimensional problems that adaptively enriches the training set. The formulation leverages the idea of a "low-dimensional approximation" of the residual manifold developed for NNLS-CR, and attempts to find a minimal training set required to well represent the manifold under the saturation assumption (cf. [29] for RB). The adaptive EQP methods compliments NNLS-CR; the former attempts to restrict the size of the training set *a priori*, while the latter further reduces the constructed constraints *a posteriori*. The key computational ingredient of adaptive EQP is NNLS-CRi, which allows efficient augmentation of constraints.

We summarize the sixfold contributions of the present work:

1. We develop a rounding-error stable variant of NNLS which provides faster and robust convergence to tighter error tolerances.
2. We appeal to the concept of reducible constraint manifold and develop NNLS with constraint reduction (NNLS-CR)—which first constructs a reduced set of orthogonalized constraints that represent the original constraints and then solves the reduced problem using NNLS—to reduce the NNLS cost for problems with a larger number of constraints.
3. We develop NNLS-CRi, a variant of NNLS-CR that provides an incremental and adaptive update of the reduced constraints for problems where the original constraints are incrementally introduced.
4. We develop a simultaneous RB and RQ greedy training procedure for high-dimensional problems, which incorporates an adaptive EQP method that incrementally enriches the training set to provide sufficient, but not excessive, coverage of the parameter space based on a saturation assumption.
5. We introduce second-order constraints for the EQP for primal residual, which, when combined with rounding-error stable NNLS and NNLS-CR, provides tighter output error control for a moderate offline training cost.
6. We demonstrate the efficacy of the proposed NNLS-CR/i and adaptive EQP for compressible Navier–Stokes and Reynolds-averaged Navier–Stokes (RANS) equations in four different contexts: shape parameter sweep; flight parameter sweep; ensemble-based data assimilation; and forward uncertainty quantification (UQ).

It is worth noting here that contributions 1–3 are applicable in general contexts requiring the solution to a NNLS problem, such as other hyperreduction methods or sparse nonlinear regression. In particular, these methods will perform well in the presence of tight tolerances and a large number of constraints.

The remainder of this paper is organized as follows. Section 2 reviews various components of projection-based model reduction. Section 3 reviews the EQP for primal residual, output functional, and DWR, and in addition introduces the EQP for primal residual that provides second-order error control. Section 4 reviews the scalable NNLS solver introduced in [12], which is used as the baseline NNLS solver in the present work. Section 5 introduces the rounding-error stable variant of the NNLS solver. Section 6 introduces NNLS-CR, which provides constraint reduction to reduce the NNLS cost for problems with many constraints. Section 7 introduces NNLS-CRi, adaptive EQP, and simultaneous RB–RQ greedy training algorithms. Section 8 assesses the NNLS-CR and adaptive EQP using four different aerodynamics problems governed by the compressible Euler and RANS equations. Section 9 summarizes the work and provide additional perspectives.

2 | PROJECTION-BASED MODEL REDUCTION

In this section, we review the projection-based ROM formulation that builds on RBs and RQs and set the notations used throughout this work. We first state the general form of the parameterized problems considered in this work (Section 2.1) and then introduce the finite element (FE) approximation (Section 2.2), RB approximation (Section 2.3), RB–RQ approximation (Section 2.4), and the associated error estimate (Section 2.5).

2.1 | Problem statement

We introduce the general form of nonlinear systems of parameterized PDEs considered throughout this work. For simplicity, we consider time-independent problems, though the formulation readily extends to time-dependent problems. We first introduce a d-dimensional spatial domain $\Omega \subset \mathbb{R}^d$ and a P-dimensional parameter space $\mathcal{D} \subset \mathbb{R}^P$. We next introduce a vector-valued Hilbert

space $\mathcal{V}$ on Ω. Our problem is as follows: given $\mu \in \mathcal{D}$, find $u(\mu) \in \mathcal{V}$ such that

$$r(u(\mu), v; \mu) = 0 \quad \forall v \in \mathcal{V}, \tag{1}$$

where $r : \mathcal{V} \times \mathcal{V} \times \mathcal{D} \to \mathbb{R}$ is a semi-linear form associated with the system of PDEs. Given the solution $u(\mu)$, we evaluate the output

$$s(\mu) = q(u(\mu); \mu), \tag{2}$$

for some output functional $q : \mathcal{V} \times \mathcal{D} \to \mathbb{R}$. Our goal is to rapidly and accurately approximate the parameter-to-output map $\mu \mapsto u(\mu) \mapsto s(\mu)$ in many-query scenarios.

2.2 | Finite element (FE) method

We now consider finite element approximation of the output prediction problem given by (1) and (2). To begin, we first introduce a finite element space $\mathcal{V}_h$ of dimension N_h. We next introduce a FE semi-linear form $r_h : \mathcal{V}_h \times \mathcal{V}_h \times \mathcal{D} \to \mathbb{R}$ and a FE output form $q_h : \mathcal{V}_h \times \mathcal{D} \to \mathbb{R}$ that result from an approximation of the integrals in the forms $r(\cdot, \cdot; \cdot)$ and $q(\cdot; \cdot)$, respectively, using a piecewise Gauss-like quadrature rule[†]: i.e.,

$$r_h(w, v; \mu) \coloneqq \sum_{i=1}^{K_h} \rho_i r_{h,i}(w, v; \mu) \quad \text{and} \quad q_h(w; \mu) \coloneqq \sum_{i=1}^{K_h} \rho_i q_{h,i}(w; \mu),$$

where K_h is the number of quadrature of quadrature points, $\{\rho_i\}_{i=1}^{K_h}$ are the quadrature weights, and $\{r_{h,i} : \mathcal{V} \times \mathcal{V} \times \mathcal{D} \to \mathbb{R}\}_{i=1}^{K_h}$ and $\{q_{h,i} : \mathcal{V} \times \mathcal{D} \to \mathbb{R}\}_{i=1}^{K_h}$ are the integrands evaluated at the associated quadrature points $\{x_i\}_{i=1}^{K_h}$. For instance, if $r(w, v; \mu) \coloneqq \int_\Omega \nabla v \cdot a(\mu) \nabla w dx$, then $r_{h,i}(w, v; \mu) \coloneqq \nabla v(x_i) \cdot a(x_i; \mu) \nabla w(x_i)$. The quadrature rule may be associated with the boundary $\partial\Omega$ to enforce boundary conditions or element interfaces for stabilized FE methods. Accurate integration in $r_h(\cdot, \cdot; \cdot)$ and $q_h(\cdot; \cdot)$ requires $K_h = \mathcal{O}(N_h)$. In this work, we use the DG discretization and its point-wise decomposition considered in [20], but the methods we develop apply to any discretization that admits a point-wise decomposition of residual and output forms.

We also introduce the associated algebraic form of the FE problem. To this end, we introduce a FE basis operator $\Phi^{\text{fe}} : \mathbb{R}^{N_h} \mapsto \mathcal{V}_h$ such that $\Phi^{\text{fe}} \mathbf{w}_h = \sum_{j=1}^{N_h} \phi_j^{\text{fe}} \mathbf{w}_{h,j} = w_h$, where $\{\phi_j^{\text{fe}}\}_{j=1}^{N_h}$ is the FE basis for $\mathcal{V}_h$, and $\mathbf{w}_h \in \mathbb{R}^{N_h}$ are the FE basis coefficients associated with the FE function $w_h \in \mathcal{V}_h$. We then introduce the associated residual operator $\mathbf{r}_h : \mathbb{R}^{N_h} \times \mathcal{D} \to \mathbb{R}^{N_h}$ such that $\mathbf{r}_h(\mathbf{w}_h; \mu)_i \coloneqq r_h(\Phi^{\text{fe}} \mathbf{w}_h, \phi_i^{\text{fe}}; \mu)$, $i = 1, \ldots, N_h$, for all $\mathbf{w}_h \in \mathbb{R}^{N_h}$. We similarly introduce the output functional $\mathbf{q}_h : \mathbb{R}^{N_h} \times \mathcal{D} \to \mathbb{R}$ such that $\mathbf{q}_h(\mathbf{w}_h; \mu) = q_h(\Phi^{\text{fe}} \mathbf{w}_h; \mu)$ for all $\mathbf{w}_h \in \mathbb{R}^{N_h}$. The associated quadrature-point-wise decomposition of the operators are given by

$$\mathbf{r}_h(\mathbf{w}_h; \mu) = \sum_{i=1}^{K_h} \rho_i \mathbf{r}_{h,i}(\mathbf{w}_h; \mu) \quad \text{and} \quad \mathbf{q}_h(\mathbf{w}_h; \mu) = \sum_{i=1}^{K_h} \rho_i \mathbf{q}_{h,i}(\mathbf{w}_h; \mu), \tag{3}$$

where the quadrature-point-wise operators satisfy $\mathbf{r}_{h,i}(\mathbf{w}_h; \mu)_i \coloneqq r_{h,i}(\Phi^{\text{fe}} \mathbf{w}_h, \phi_i^{\text{fe}}; \mu)$, $i = 1, \ldots, N_h$, and $\mathbf{q}_{h,i}(\mathbf{w}_h; \mu) \coloneqq q_{h,i}(\Phi^{\text{fe}} \mathbf{w}_h; \mu)$ for all $\mathbf{w}_h \in \mathbb{R}^{N_h}$. We will use this algebraic form of the problem to describe various concepts from hereon.

We now introduce the FE problem: given $\mu \in \mathcal{D}$, find $\mathbf{u}_h(\mu) \in \mathbb{R}^{N_h}$ such that

$$\mathbf{r}_h(\mathbf{u}_h(\mu); \mu) = 0 \quad \text{in } \mathbb{R}^{N_h} \tag{4}$$

and evaluate the associated output

$$s_h(\mu) \coloneqq \mathbf{q}_h(\mathbf{u}_h(\mu); \mu). \tag{5}$$

We use a Newton-like method to solve the FE problem (4); the process requires the evaluation of the Jacobian $\mathbf{J}_h(\cdot; \mu) : \mathbb{R}^{N_h} \to \mathbb{R}^{N_h \times N_h}$ associated with $\mathbf{r}_h(\cdot; \mu) : \mathbb{R}^{N_h} \to \mathbb{R}^{N_h}$.

2.3 | RB method

To accelerate the solution of the FE problem (4) and the evaluation of the FE output (5), we consider a RB approximation of the FE problem. Specifically, we assume that the parametric solution manifold $\mathcal{U}_h \coloneqq \{\mathbf{u}_h(\mu)\}_{\mu \in \mathcal{D}}$ is amenable to a low-dimensional

[†] The forms $r_h(\cdot, \cdot; \cdot)$ and $q_h(\cdot; \cdot)$ may also include additional stabilization terms. This is the case for the DG method considered in this work.

approximation, introduce a $\mathcal{V}$-orthonormal RB $\{\phi_i \in \mathbb{R}^{N_h}\}_{i=1}^N$ for the manifold $\mathcal{U}_h$ where $N \ll N_h$, and construct the associated RB operator (for the FE coefficients) $\mathbf{\Phi}_N : \mathbb{R}^N \to \mathbb{R}^{N_h}$ such that $\mathbf{\Phi}_N \mathbf{w}_N = \sum_{i=1}^N \phi_i \mathbf{w}_{N,i}$. The RB may be formed using, for example, proper orthogonal decomposition or the Greedy algorithm [33]. We then define the associated RB (Galerkin) residual $\mathbf{r}_N : \mathbb{R}^N \times \mathcal{D} \to \mathbb{R}^N$ and RB output functional $\mathbf{q}_N : \mathbb{R}^N \times \mathcal{D} \to \mathbb{R}$ such that, $\forall \mathbf{w}_N \in \mathbb{R}^N$,

$$\mathbf{r}_N(\mathbf{w}_N; \mu) = \mathbf{\Phi}_N^T \mathbf{r}_h(\mathbf{\Phi}_N \mathbf{w}_N; \mu) \coloneqq \sum_{i=1}^{K_h} \rho_i \mathbf{r}_{N,i}(\mathbf{w}_N; \mu), \tag{6}$$

$$\mathbf{q}_N(\mathbf{w}_N; \mu) = \mathbf{q}_h(\mathbf{\Phi}_N \mathbf{w}_N; \mu) \coloneqq \sum_{i=1}^{K_h} \rho_i \mathbf{q}_{N,i}(\mathbf{w}_N; \mu), \tag{7}$$

where $\mathbf{r}_{N,i}(\mathbf{w}_N; \mu) \coloneqq \mathbf{\Phi}_N^T \mathbf{r}_{h,i}(\mathbf{\Phi}_N \mathbf{w}_N; \mu)$ and $\mathbf{q}_{N,i}(\mathbf{w}_N; \mu) = \mathbf{q}_{h,i}(\mathbf{\Phi}_N \mathbf{w}_N; \mu)$, $i = 1, \ldots, K_h$. Our RB problem is as follows: given $\mu \in \mathcal{D}$, find $\mathbf{u}_N(\mu) \in \mathbb{R}^N$ such that

$$\mathbf{r}_N(\mathbf{u}_N(\mu); \mu) = 0 \quad \text{in } \mathbb{R}^N \tag{8}$$

and evaluate the associated output

$$s_N(\mu) \coloneqq \mathbf{q}_N(\mathbf{u}_N(\mu); \mu). \tag{9}$$

We again use a Newton-like method to solve the RB problem, which requires the evaluation of the Jacobian $\mathbf{J}_N(\cdot; \mu) : \mathbb{R}^N \to \mathbb{R}^{N \times N}$ associated with $\mathbf{r}_N(\cdot; \mu) : \mathbb{R}^N \to \mathbb{R}^N$.

The RB problem (8) and RB output (9) requires the evaluation of the FE forms (3), which use the quadrature rules with $K_h = \mathcal{O}(N_h)$ points. Hence, even though the dimension of the approximation space has been reduced to $N \ll N_h$, the evaluation of the solution of the RB problem still requires $\mathcal{O}(N_h)$ operations.

2.4 | RB–RQ method

We now hyperreduce the RB problem (8) and RB output evaluation (9). Namely, we approximate the FE residual and output forms (3) using RQ rules specifically designed for each problem. We will defer the discussion of the construction of the RQ rules using EQP to Section 3; in this section, we assume that RQ rules are given and present the evaluation of the solution, output, and error estimate using the RQ rules.

To hyperreduce the RB problem (8), we introduce an RQ approximation $\tilde{\mathbf{r}}_N(\cdot; \cdot)$ of the residual form $\mathbf{r}_N(\cdot; \cdot)$ in (6):

$$\tilde{\mathbf{r}}_N^r(\mathbf{w}_N; \mu) \coloneqq \sum_{i=1}^{K_h} \rho_i^r \mathbf{r}_{N,i}(\mathbf{w}_N; \mu), \tag{10}$$

where $\{\rho_i^r\}_{i=1}^{K_h}$ is a set of sparse quadrature weights such that $\|\rho^r\|_0 \coloneqq K^r \ll K_h$. We then solve the RB–RQ problem: given $\mu \in \mathcal{D}$, find $\tilde{\mathbf{u}}_N(\mu) \in \mathbb{R}^N$ such that

$$\tilde{\mathbf{r}}_N^r(\tilde{\mathbf{u}}_N(\mu); \mu) = 0 \quad \text{in } \mathbb{R}^N. \tag{11}$$

The solution is again obtained using a Newton-like method, which requires the evaluation of the Jacobian $\tilde{\mathbf{J}}_N^r(\cdot; \mu) : \mathbb{R}^N \to \mathbb{R}^{N \times N}$ associated with $\tilde{\mathbf{r}}_N^r(\cdot; \mu) : \mathbb{R}^N \to \mathbb{R}^N$. As $K^r \ll K_h$, the RQ residual form (10) can be evaluated in $\mathcal{O}(K^r)$ operations, and the solution cost of the RB–RQ problem (11) is independent of K_h (and N_h).

To hyperreduce the RB output evaluation (9), we analogously introduce an RQ approximation $\tilde{\mathbf{q}}_N(\cdot; \cdot)$ of the output form $\mathbf{q}_N(\cdot; \cdot)$ in (7):

$$\tilde{\mathbf{q}}_N^q(\mathbf{w}_N; \mu) \coloneqq \sum_{i=1}^{K_h} \rho_i^q \mathbf{q}_{N,i}(\mathbf{w}_N; \mu), \tag{12}$$

where $\{\rho_i^q\}_{i=1}^{K_h}$ is a set of sparse quadrature weights such that $\|\rho^q\|_0 \coloneqq K^q \ll K_h$. We then evaluate the RB–RQ output $\tilde{s}_N(\mu) \coloneqq \tilde{q}^q(\tilde{u}_N(\mu); \mu)$.

2.5 | Output error estimation: dual-weighted residual (DWR) method

We also wish to equip our ROM with an output error estimate. To this end, we use the DWR method [4] adopted to the ROM error estimation [41]. To begin, we introduce a dual FE problem: given $\mu \in \mathcal{D}$, find $\mathbf{z}_h(\mu) \in \mathbb{R}^{N_h}$ such that

$$\mathbf{J}_h(\mathbf{u}_h(\mu);\mu)^T \mathbf{z}_h(\mu) = \mathbf{g}_h(\mathbf{u}_h(\mu)) \quad \text{in } \mathbb{R}^{N_h}, \tag{13}$$

where $\mathbf{g}_h(\mathbf{u}_h(\mu);\mu) \in \mathbb{R}^{N_h}$ is the gradient of $\mathbf{q}_h(\cdot;\mu)$ evaluated at $\mathbf{u}_h(\mu)$. We next assume that the parametric dual solution manifold $\mathcal{Z}_h \coloneqq \{\mathbf{z}_h(\mu)\}_{\mu\in\mathcal{D}}$ is amenable to low-dimensional approximation, introduce a $\mathcal{V}$-orthonormal dual RB $\{\boldsymbol{\phi}_i^{\mathrm{du}} \in \mathbb{R}^{N_h}\}_{i=1}^N$ for the manifold $\mathcal{Z}_h$, and construct the associated dual RB operator $\boldsymbol{\Phi}_N^{\mathrm{du}} : \mathbb{R}^N \to \mathbb{R}^{N_h}$ such that $\boldsymbol{\Phi}_N^{\mathrm{du}} \mathbf{v}_N = \sum_{i=1}^N \boldsymbol{\phi}_i^{\mathrm{du}} \mathbf{v}_{N,i}$. We then introduce the associated dual RB problem: given $\mu \in \mathcal{D}$, find $\mathbf{z}_N^{\mathrm{du}}(\mu) \in \mathbb{R}^N$ such that

$$\mathbf{J}_N^{\mathrm{du}}(\mathbf{u}_N(\mu);\mu)^T \mathbf{z}_N^{\mathrm{du}} = \mathbf{g}_N^{\mathrm{du}}(\mathbf{u}_N(\mu);\mu) \quad \text{in } \mathbb{R}^N, \tag{14}$$

where $\mathbf{J}_N^{\mathrm{du}}(\mathbf{w}_N;\mu) \coloneqq \boldsymbol{\Phi}_N^{\mathrm{du}T} \mathbf{J}_h(\boldsymbol{\Phi}_N \mathbf{w}_N;\mu) \boldsymbol{\Phi}_N^{\mathrm{du}} \in \mathbb{R}^{N\times N}$ and $\mathbf{g}_N^{\mathrm{du}}(\mathbf{w}_N;\mu) \coloneqq \boldsymbol{\Phi}_N^{\mathrm{du}T} \mathbf{g}_h^{\mathrm{du}}(\boldsymbol{\Phi}_N \mathbf{w}_N;\mu) \in \mathbb{R}^N$. Note that $\mathbf{J}_N^{\mathrm{du}}(\mathbf{w}_N;\mu)$ is *not* the Jacobian of $\mathbf{r}_N(\cdot;\mu)$ (i.e., $\mathbf{J}_N^{\mathrm{du}}(\mathbf{w}_N;\mu) \neq \mathbf{J}_N(\mathbf{w}_N;\mu)$) since it is obtained through the Galerkin projection of the dual problem onto the *dual* RB space. Similarly, $\mathbf{g}_N^{\mathrm{du}}(\mathbf{w}_N;\mu)$ is *not* the gradient of $\mathbf{q}_N(\cdot;\mu)$. We then evaluate the DWR error estimate

$$\eta_N(\mu) \coloneqq \mathbf{z}_N^{\mathrm{du}T} \mathbf{r}_N^{\mathrm{du}}(\mathbf{u}_N(\mu);\mu), \tag{15}$$

where $\mathbf{r}_N^{\mathrm{du}}(\mathbf{w}_N;\mu) \coloneqq \boldsymbol{\Phi}_N^{\mathrm{du}T} \mathbf{r}_h(\boldsymbol{\Phi}_N \mathbf{w}_N;\mu) \in \mathbb{R}^N$, $\forall \mathbf{w}_N \in \mathbb{R}^N$. The error estimate $\eta_N(\mu)$ approximates the output error $s_N(\mu) - s_h(\mu)$. The solution of the RB dual problem (14) and the RB error estimate (15) requires the evaluation of the FE forms (3) with $K_h = \mathcal{O}(N_h)$ quadrature points, and hence the evaluation of the error estimate also requires $\mathcal{O}(N_h)$.

We now reduce the computational cost of the DWR error estimate given by (14) and (15) through hyperreduction. To this end, we introduce RQ approximations of the residual $\mathbf{r}_N^{\mathrm{du}}(\cdot;\cdot)$, Jacobian $\mathbf{J}_N^{\mathrm{du}}(\cdot;\cdot)$, and the output gradient $\mathbf{g}_N^{\mathrm{du}}(\cdot;\cdot)$ used in DWR:

$$\tilde{\mathbf{r}}_N^{\mathrm{du},\eta}(\mathbf{w}_N;\mu) \coloneqq \sum_{i=1}^{K_h} \rho_i^\eta \mathbf{r}_{N,i}^{\mathrm{du}}(\mathbf{w}_N;\mu), \quad \tilde{\mathbf{J}}_N^{\mathrm{du},\eta}(\mathbf{w}_N;\mu) \coloneqq \sum_{i=1}^{K_h} \rho_i^\eta \mathbf{J}_{N,i}^{\mathrm{du}}(\mathbf{w}_N;\mu), \quad \tilde{\mathbf{g}}^{\mathrm{du},\eta}(w_N;\mu) \coloneqq \sum_{i=1}^{K_h} \rho_i^\eta \mathbf{g}_{N,i}^{\mathrm{du}}(w;\mu), \tag{16}$$

where $\{\rho_i^\eta\}_{i=1}^{K_h}$ is a set of sparse quadrature weights such that $\|\rho^\eta\|_0 \coloneqq K^\eta \ll K_h$, and the quadrature forms are given by $\mathbf{r}_{N,i}^{\mathrm{du}}(\mathbf{w}_N;\mu) \coloneqq \boldsymbol{\Phi}_N^{\mathrm{du}T} \mathbf{r}_{h,i}(\boldsymbol{\Phi}_N \mathbf{w}_N;\mu)$, $\mathbf{J}_{N,i}^{\mathrm{du}}(\mathbf{w}_N;\mu) \coloneqq \boldsymbol{\Phi}_N^{\mathrm{du}T} \mathbf{J}_{h,i}(\boldsymbol{\Phi}_N \mathbf{w}_N;\mu) \boldsymbol{\Phi}_N^{\mathrm{du}}$, and $\mathbf{g}_{N,i}^{\mathrm{du}} = \boldsymbol{\Phi}_N^{\mathrm{du}T} \mathbf{g}_{h,i}(\boldsymbol{\Phi}_N \mathbf{w}_N;\mu)$. (We again defer the discussion of the construction of the RQ rules to Section 3 and assume that they are given.) To compute the hyperreduced DWR error estimate, we first solve the dual RB–RQ problem: given $\mu \in \mathcal{D}$, find $\tilde{\mathbf{z}}_N^{\mathrm{du}}(\mu) \in \mathbb{R}^N$ such that $\mathbf{J}^{\mathrm{du},\eta}(\tilde{\mathbf{u}}_N(\mu);\mu)^T \tilde{\mathbf{z}}_N^{\mathrm{du},\eta}(\mu) = \mathbf{g}^\eta(\tilde{\mathbf{u}}_N(\mu);\mu)$. We then evaluate the error estimate $\tilde{\eta}_N(\mu) \coloneqq \tilde{\mathbf{z}}_N^{\mathrm{du}T} \mathbf{r}^{\mathrm{du},\eta}(\tilde{u}_N(\mu);\mu)$. The cost to evaluate this error estimate is independent of K_h (and N_h).

We have now defined the RB–RQ problem and its associated RB–RQ output error estimation method, which require three distinct RQ rules: $\{\rho_i^r\}_{i=1}^{K_h}$, $\{\rho_i^q\}_{i=1}^{K_h}$, and $\{\rho_i^\eta\}_{i=1}^{K_h}$. The following section will discuss how to construct these RQ rules such that the error in the output and our error estimator are controlled.

3 | EMPIRICAL QUADRATURE PROCEDURE

In this section, we review the EQP, which we use to construct RQ rules. We first state the general form of the constrained optimization problem (Section 3.1). We then review the EQP constraints for first-order output control (Section 3.2) and DWR error estimation (Section 3.4). We also propose new EQP constraints for second-order output error control (Section 3.3). We then conclude the section with a discussion of computational cost (Section 3.5).

3.1 | EQP: general form

We use the EQP to construct RQ rules $\rho^\bullet \in \mathbb{R}^{K_h}$ for $\bullet \in \{r, q, \eta\}$ so that we can control the error in the RB–RQ output $\tilde{s}_N(\mu)$ and error estimate $\tilde{\eta}_N(\mu)$ due to hyperreduction. As discussed in the introduction, the EQP recasts the sparse quadrature identification

problem as an optimization problem that involves "(weighted-)residual matching conditions" over a training set $\Xi^{\mathrm{EQP}} \subset \mathcal{D}$ of the size N^{EQP}. The general form of the EQP is as follows:

Definition 1 (Generic EQP: $\texttt{EQP}^\bullet(\Xi^{\mathrm{EQP}}, \delta^\bullet)$)**.** Given a set of parameter values $\Xi^{\mathrm{EQP}} \subset \mathcal{D}$, find a sparse set of quadrature weights $\rho^\bullet \in \mathbb{R}^{K_h}$ such that

$$\rho^\bullet = \arg\min_{\rho \in \mathbb{R}^{K_h}} \|\rho\|_0,$$

subject to the non-negativity constraint $\rho \in \mathbb{R}^{K_h}_{\geq 0}$, constant-function accuracy constraint $\left|\left|\Omega\right| - \sum_{i=1}^{K^h} \rho_i\right| < \delta^c$ for $\delta^c \in \mathbb{R}_{>0}$, and manifold accuracy constraints, or "(weighted-)residual matching conditions", of the form

$$\left|c^\bullet(\rho;\mu)\right|_i < \delta_i^\bullet, \quad i = 1,\ldots,N_c^\bullet, \quad \forall \mu \in \Xi^{\mathrm{EQP}}, \tag{17}$$

where $c^\bullet : \mathbb{R}^{K^h} \times \mathcal{D} \to \mathbb{R}^{N_c^\bullet}$ is a parameterized constraint function that is linear in the first argument.

The choice of the manifold accuracy constraints (17) dictates the behavior of EQP. We now introduce specific manifold accuracy constraints that we use to control the error in the RB–RQ output $\tilde{s}_N(\mu)$ and the RB–RQ error estimate $\tilde{\eta}_N(\mu)$.

3.2 | EQP for first-order output error control

The goal of output-based EQP is to provide direct control of output (as opposed to solution field) error. In this section, as a preliminary to introduce the "second-order" output error control in Section 3.3, we first briefly review the "first-order" output error control developed in [41]; for a complete presentation, we refer to [41]. To begin, we decompose the hyperreduction output error $|s_N(\mu) - \tilde{s}_N(\mu)|$ into two parts:

$$\left|s_N(\mu) - \tilde{s}_N(\mu)\right| \leq \left|\mathbf{q}_N(\mathbf{u}_N(\mu);\mu) - \mathbf{q}_N(\tilde{\mathbf{u}}_N(\mu);\mu)\right| + \left|\mathbf{q}_N(\tilde{\mathbf{u}}_N(\mu);\mu) - \tilde{\mathbf{q}}_N(\tilde{\mathbf{u}}_N(\mu);\mu)\right|;$$

the two terms are (i) the output error due to the approximation of the RB solution by the RB–RQ solution (i.e., $\mathbf{u}_N(\mu)$ associated with $r_N(\cdot;\cdot)$ vs $\tilde{\mathbf{u}}_N(\mu)$ associated with $\tilde{\mathbf{r}}^r(\cdot;\mu)$) and (ii) the output error due to the evaluation of the output functional using an RQ rule (i.e., $\mathbf{q}_N(\cdot;\mu)$ vs $\tilde{\mathbf{q}}_N^q(\cdot;\mu)$).

We first address (i) using the DWR relation for output errors. To this end, we first define a "modified" dual solution coefficient, which serves as the weight in our DWR-based EQP:

Definition 2. Let $\mu \in \mathcal{D}$ and $\hat{\mathbf{u}}_N(\mu) \in \mathbb{R}^N$ be the RB solution $\mathbf{u}_N(\mu)$ given by (8) or its approximation (of arbitrary fidelity). Let $\hat{\mathbf{z}}_N^{\mathrm{pr}}(\mu) \in \mathbb{R}^N$ be the solution to the dual RB problem (14) linearized about $\hat{\mathbf{u}}_N(\mu)$ (instead of $\tilde{\mathbf{u}}_N(\mu)$) and solved using the primal RB $\mathbf{\Phi}_N$ (instead of $\mathbf{\Phi}_N^{\mathrm{du}}$). We then cap the minimum value of the dual solution coefficient to obtain the modified dual solution coefficient: $\hat{\mathbf{z}}_N^{\mathrm{pr,mod}}(\mu)_i \coloneqq \max\{|\hat{\mathbf{z}}_N^{\mathrm{pr}}(\mu)_i|, z^{\min}(\mu)\}$, $i = 1,\ldots,N$, where $z^{\min}(\mu) \coloneqq N^{1/2}\sqrt{\delta^r}\|\hat{\mathbf{z}}_N^{\mathrm{pr}}(\mu)\|_2$.

We now restate a proposition from [41] that motivates the choice of our manifold constraints:

Proposition 3.1 (Output error due to RB–RQ approximation $\tilde{\mathbf{r}}_N(\cdot;\cdot) \approx \mathbf{r}_N(\cdot;\cdot)$)
Let $\hat{\mathbf{u}}_N(\mu) \in \mathbb{R}^N$ be the RB solution $\mathbf{u}_N(\mu)$ given by (8) *or its approximation, and $\hat{\mathbf{z}}_N^{\mathrm{pr,mod}}(\mu) \in \mathbb{R}^N$ be the associated modified dual solution given by Definition 2. Suppose*

$$|\hat{\mathbf{z}}_N^{\mathrm{pr,mod}} \circ (\mathbf{r}_N(\hat{\mathbf{u}}_N(\mu);\mu) - \tilde{\mathbf{r}}_N(\hat{\mathbf{u}}_N(\mu);\mu)|_i \leq \frac{2\delta^r}{3N}, \quad i = 1,\ldots,N, \tag{18}$$

$$|\mathbf{I} - \mathbf{J}_N(\hat{\mathbf{u}}_N(\mu);\mu)^{-1}\tilde{\mathbf{J}}_N(\hat{\mathbf{u}}_N(\mu);\mu)|_{ij} \leq \delta^J, \quad i,j = 1,\ldots,N, \tag{19}$$

where $\mathbf{I} \in \mathbb{R}^{N\times N}$ is the identity matrix, $\circ$ denotes the Hadamard product, and the absolute values in the left hand sides are applied to each entry of the vector or matrix. Then

$$\left|\mathbf{q}_N(\mathbf{u}_N(\mu);\mu) - \mathbf{q}_N(\tilde{\mathbf{u}}_N(\mu);\mu)\right| \leq \delta^r + \mathcal{O}((\delta^J)^2) + \mathcal{O}(\hat{\delta}^2) + \mathcal{O}(\tilde{\delta}^2), \tag{20}$$

where $\hat{\delta} \equiv \|\mathbf{u}_N(\mu) - \hat{\mathbf{u}}_N(\mu)\|_2$ and $\tilde{\delta} \equiv \|\mathbf{u}_N(\mu) - \tilde{\mathbf{u}}_N(\mu)\|_2$.
Proof. See [41]. □

From Proposition 3.1, we obtain the EQP constraints proposed in [41] aimed to control the leading (i.e., first-order) term of the output error $\left|\mathbf{q}_N(\mathbf{u}_N(\mu);\mu) - \mathbf{q}_N(\tilde{\mathbf{u}}_N(\mu);\mu)\right|$.

Definition 3 (EQP for primal residual with first-order output error control: $\texttt{EQP}^r(\Xi^{\text{EQP}}, \delta^r)$)**.** The EQP constraints $c^r : \mathbb{R}^{K_h} \times \mathcal{D} \to \mathbb{R}^{N_c^r := N}$ to find the RQ weights $\{\rho_i^r\}_{i=1}^{K_h}$ for the primal residual that controls the first-order output error is given by

$$|c^r(\rho;\mu)|_i = |\hat{\mathbf{z}}_N^{\text{pr,mod}} \circ (\mathbf{r}_N(\hat{\mathbf{u}}_N(\mu);\mu) - \sum_{j=1}^{K_h} \rho_j \mathbf{r}_{N,j}(\hat{\mathbf{u}}_N(\mu);\mu))|_i \le \frac{2\delta^r}{3N}, \quad i = 1,\dots,N, \tag{21}$$

where $\hat{\mathbf{u}}_N(\mu) \in \mathbb{R}^N$ is RB solution $\mathbf{u}_N(\mu)$ or its approximation, and $\hat{\mathbf{z}}_N^{\text{pr,mod}}(\mu) \in \mathbb{R}^N$ is the associated modified dual solution given by Definition 2. The number of constraints per training parameter value is $N_c^r := N$.

Remark 1. The constraints (21) corresponds to (18), which controls the *first-order* error δ^r in the output error bound (20) but not the *second-order* error $\mathcal{O}(\delta^J)^2$ associated with the integration of the Jacobian. The choice was justified in [41] as (i) the error is of higher order, (ii) we expect that the residual RQ control by (18) controls the Jacobian RQ error to some degree, and (iii) the explicit control of Jacobian RQ error requires N^2 constraints (instead of N for the residual error control). Section 3.3 will discuss situations when the Jacobian RQ error must be explicitly controlled.

Having controlled the error source (i), we now present the EQP constraints proposed in [41] aimed to find $\{\rho_i^q\}_{i=1}^{K_h}$ that controls the error source (ii): $|\mathbf{q}_N(\tilde{\mathbf{u}}_N(\mu);\mu) - \tilde{\mathbf{q}}_N(\tilde{\mathbf{u}}_N(\mu);\mu)|$.

Definition 4 (EQP for output functional: $\texttt{EQP}^q(\Xi^{\text{EQP}}, \delta^q)$)**.** The EQP constraints $c^q : \mathbb{R}^{K_h} \times \mathcal{D} \to \mathbb{R}^{N_c^q := 1}$ to find the RQ weights $\{\rho_i^q\}_{i=1}^{K_h}$ for the output functional are given by

$$|c^q(\rho;\mu)| = |\mathbf{q}_N(\tilde{\mathbf{u}}_N(\mu);\mu) - \sum_{j=1}^{K^q} \rho_j \mathbf{q}_{N,j}(\tilde{\mathbf{u}}_N(\mu);\mu)| < \delta^q, \tag{22}$$

where $\tilde{\mathbf{u}}_N(\mu) \in \mathbb{R}^N$ is the RB–RQ solution given by (11). The number of constraints per training parameter value is $N_c^q := 1$.

3.3 | EQP for second-order output error control

As discussed in Remark 1, the EQP^r constraint in Definition 3 explicitly controls the first-order term but not the second-order terms. In cases that demand a very tight error tolerance, or in cases where the parameter space is sparsely sampled, the constraints given by (21) can be insufficient to control the output error. The inadequacy of the first-order constraints will be demonstrated in Section 8.2.2. In order to train our model for these cases, we add second-order constraints such that we control δ^J to an acceptable value. We thereby introduce an updated EQP procedure that controls second-order errors.

Definition 5 (EQP for primal residual with second-order output error control: $\texttt{EQP}^{r2}(\Xi^{\text{EQP}}, \delta^r)$)**.** The EQP constraints $c^{r2} : \mathbb{R}^{K_h} \times \mathcal{D} \to \mathbb{R}^{N_c^r := N+N^2}$ to find the RQ weights $\{\rho_i^r\}_{i=1}^{K_h}$ for the primal residual that controls the second-order output error are given by $c^{r2}(\rho;\mu) = \left(c_1^{r2}(\rho;\mu)^T \;\; c_2^{r2}(\rho;\mu)^T \right)^T$ where

$$|c^{r2}(\rho;\mu)|_i = |\hat{\mathbf{z}}_N^{\text{pr,mod}} \circ (\mathbf{r}_N(\hat{\mathbf{u}}_N(\mu);\mu) - \sum_{j=1}^{K_h} \rho_j \mathbf{r}_{N,j}(\hat{\mathbf{u}}_N(\mu);\mu))|_i \le \delta^r, \quad i = 1,\dots,N, \tag{23}$$

$$|c_2^{r2}(\rho;\mu)|_{ij} = |\mathbf{I} - \mathbf{J}_N(\hat{\mathbf{u}}_N(\mu);\mu)^{-1} \sum_{j=1}^{K_h} \rho_j \mathbf{J}_{N,j}(\hat{\mathbf{u}}_N(\mu);\mu)|_{ij} \le \delta^r, \quad i,j = 1,\dots,N, \tag{24}$$

where $\hat{\mathbf{u}}_N(\mu) \in \mathbb{R}^N$ is the RB solution $\mathbf{u}_N(\mu)$ or its approximation, and $\hat{\mathbf{z}}_N^{\text{pr,mod}}(\mu) \in \mathbb{R}^N$ is the associated modified dual coefficients given by Definition 2. The number of constraints per training parameter value is $N_c^{r^2} := N + N^2$.

Remark 2. The constraint tolerance in (24) is chosen to be δ^r. We were unable to determine *a priori* a strict relation between δ^r and δ^J, as presented in Proposition 3.1; however, given that the contributions to the output error are on the order of δ^r and $(\delta^J)^2$, we deem choosing $\delta^J = \delta^r$ is a reasonable approximation. We also note that this likely introduces some conservativeness in the error estimate.

3.4 | EQP for DWR error control

For completeness, we finally reproduce the EQP constraints for DWR introduced in [41].

Definition 6 (EQP for DWR error estimate: $\texttt{EQP}^\eta(\Xi^{\text{EQP}}, \delta^\eta)$)**.** The EQP constraints $c^\eta : \mathbb{R}^{K_h} \times \mathcal{D} \to \mathbb{R}^{N_c^\eta := 3N}$ to find the RQ weights $\{\rho_i^\eta\}_{i=1}^{K_h}$ for the DWR are given by $c^\eta(\rho;\mu) = \left(c_1^\eta(\rho;\mu)^T \;\; c_2^\eta(\rho;\mu)^T \;\; c_3^\eta(\rho;\mu)^T \right)^T$ where

$$|c_1^\eta(\rho;\mu)|_i = |\mathbf{r}^{\text{du,mod}}(\hat{\mathbf{u}}_N(\mu);\mu) \circ \mathbf{J}_N^{\text{du}}(\hat{\mathbf{u}}_N(\mu);\mu)^{-T}(\mathbf{J}_N^{\text{du}}(\hat{\mathbf{u}}_N(\mu);\mu)^T \hat{\mathbf{z}}_N^{\text{du}}(\mu) - \sum_{j=1}^{K^\eta} \rho_j \mathbf{J}_{N,j}^{\text{du}}(\hat{\mathbf{u}}_N(\mu);\mu)^T \hat{\mathbf{z}}_N^{\text{du}}(\mu))|_i \leq \frac{\delta^\eta}{4N}, \tag{25}$$

$$|c_2^\eta(\rho;\mu)|_i = |\mathbf{r}_N^{\text{du,mod}}(\hat{\mathbf{u}}_N(\mu);\mu) \circ \mathbf{J}_N^{\text{du}}(\hat{\mathbf{u}}_N(\mu);\mu)^{-T}(\mathbf{g}_N^{\text{du}}(\hat{\mathbf{u}}_N(\mu);\mu) - \sum_{j=1}^{K_h} \rho_j \mathbf{g}_{N,j}^{\text{du}}(\hat{\mathbf{u}}_N(\mu);\mu))|_i \leq \frac{\delta^\eta}{4N} \tag{26}$$

$$|c_3^\eta(\rho;\mu)|_i = |z^{\text{du,mod}} \circ (\mathbf{r}_N^{\text{du}}(\hat{\mathbf{u}}_N(\mu);\mu) - \sum_{j=1}^{K_h} \rho_j \mathbf{r}_{N,j}^{\text{du}}(\hat{\mathbf{u}}_N(\mu);\mu))|_i \leq \frac{\delta^\eta}{2N} \tag{27}$$

for $i = 1, \ldots, N$, where $\hat{\mathbf{u}}_N(\mu) \in \mathbb{R}^N$ is the RB solution $\mathbf{u}_N(\mu)$ or its approximation, and $\hat{\mathbf{z}}_N^{\text{pr,mod}}(\mu) \in \mathbb{R}^N$ is the associated modified dual coefficients given by Definition 2, $\mathbf{r}_N^{\text{du,mod}}(\hat{\mathbf{u}}_N(\mu);\mu) := \{|\mathbf{r}_N^{\text{du}}(\hat{\mathbf{u}}_N(\mu);\mu), r^{\text{du,min}}(\mu)\}$ and $\hat{\mathbf{z}}_N^{\text{du,mod}}(\mu) := \{|\hat{\mathbf{z}}_N^{\text{du}}(\mu)|, z^{\text{du,min}}(\mu)\}$ for $r^{\text{du,min}}(\mu) := \frac{1}{2}\sqrt{\frac{\delta^\eta}{\alpha(\mu)N}}$, $z^{\text{du,min}}(\mu) := \frac{1}{2}\sqrt{\frac{\alpha(\mu)\delta^\eta}{N}}$, and $\alpha(\mu) := \|\hat{\mathbf{z}}_N^{\text{du}}(\mu)\|_2 / \|\mathbf{r}_N^{\text{du}}(\hat{\mathbf{u}}_N(\mu))\|_2$. The number of constraints per training parameter value is $N_c^\eta := 3N$.

3.5 | Summary and computational cost

In summary, the EQP (Definition (1)) requires the solution of a constrained minimization problem, which takes on the form

$$\rho^\star = \arg \min_{\rho \in \mathbb{R}^{K^h}} \|\rho\|_0, \tag{28}$$

subject to K^h non-negativity constraints

$$\rho_i^\star \geq 0, \;\; i = 1, \ldots K^h, \tag{29}$$

and manifold constraints given by

$$\left|\mathbf{A}\rho - \mathbf{b}\right| < \delta, \tag{30}$$

where $\mathbf{A} \in \mathbb{R}^{m \times K^h}$, $\mathbf{b} \in \mathbb{R}^m$, and $\delta \in \mathbb{R}^m$ describe various manifold constraints for (first-order) residual (21); output functional (22); second-order residual (23) and (24); or DWR (25), (26) and (27). Note that the inequality in (30) is imposed on each of the m entries of $\left|\mathbf{A}\rho - \mathbf{b}\right|$. There are $N \times N_{\text{train}}^{\text{EQP}}$ constraints (given by (21)) for the first-order primal residual RQ rule, $(N + N^2) \times N_{\text{train}}^{\text{EQP}}$ constraints (given by (21) and(24)) for the second-order primal residual RQ rule, $N_{\text{train}}^{\text{EQP}}$ constraints (given by (22)) for the output RQ rule, and $3N^{\text{du}} \times N_{\text{train}}^{\text{EQP}}$ constraints (given by (25), (26) and (27)) for the DWR RQ rule. For time-dependent problems with N_t time-steps, the number of constraints for each of these methods will increase by a factor of the number of time steps N_t. In the subsequent sections, we will develop methods to efficiently solve these optimization problems with many constraints. We conclude the section with a few remarks.

Remark 3. The constraints constructed by Definitions 3, 4, 5, or 6 may potentially be ill-conditioned. For a parameter set Ξ^{EQP} that densely-samples $\mathcal{D}$, we construct a set of constraints for which the constraint manifold $\mathcal{C}$ is well-represented, as desired. However, the proximity of some of the parameter values in Ξ^{EQP} will result in a high degree of similarity between many of the constraints; i.e., the set of constraints contains many nearly redundant (or entirely redundant) constraints and is thus ill-conditioned. Section 6 will discuss the idea of ill-conditioned constraints in more detail.

Remark 4. In this work, we use the EQP to generate the "residual matching constraints" (30). However, as discussed in the introduction, the solution methods that will be developed in the subsequent sections are applicable to any other hyperreduction methods that require the solution of the constrained optimization problem of the form (28)–(30), such as ECSW.

4 | NNLS: STANDARD METHOD WITH INCREMENTAL QR

In this section, we review the baseline NNLS algorithm used in this work. We review the NNLS formulation (Section 4.1) and the incremental QR update (Section 4.2) and then discuss the computational cost (Section 4.3) to motivate the subsequent work.

4.1 | Standard NNLS algorithm

One way to approximate the solution to the constrained optimization problem (28)–(30) is the NNLS algorithm introduced by Lawson and Hanson [30]. This algorithm employs an active set approach to find the RQ rule, whereby at each iteration a single quadrature point (or equivalently, a single column of our constraint matrix $\mathbf{A}$) is added to the active set, and the process is repeated until all constraints are satisfied.

The NNLS algorithm [30] is reproduced in Algorithm 1. We first define two sets of quadrature points. The set $\mathcal{Z}$ contains all inactive quadrature points; i.e. quadrature points whose weight is set to zero. The set $\mathcal{P}$ contains all active quadrature points; i.e. quadrature points with nonzero (and positive) weight. The first step of the NNLS algorithm populates the set $\mathcal{Z}$ with all of the quadrature points and sets $\mathcal{P} = \emptyset$ and $\rho = \mathbf{0}$ (line 1). At each iteration, a Lagrange multiplier $\lambda = \mathbf{A}^\mathbf{T}(\mathbf{b} - \mathbf{A}\rho) \in \mathbb{R}^{K_h}$ for each candidate quadrature point is computed (lines 3 and 4). The quadrature point with the largest Lagrange multiplier is then removed from $\mathcal{Z}$ and added to $\mathcal{P}$ (lines 6 and 7). Given exact calculation of the Lagrange multipliers, this ensures that at each iteration, we select the quadrature point that results in the largest decrease in the ℓ^2 norm of the residual $\mathbf{b} - \mathbf{A}\rho$, while also ensuring that the associated quadrature weight will be positive. We then solve the least squares problem that minimizes $\mathbf{A}_\mathcal{P}\rho - \mathbf{b}$ to get $\tilde{\rho}$ (line 8), where $\mathbf{A}_\mathcal{P} \in \mathbb{R}^{m\times|\mathcal{P}|}$ comprises columns of $\mathbf{A}$ associated with the set $\mathcal{P}$ and $\tilde{\rho} \in \mathbb{R}^{|\mathcal{P}|}$ represents our temporary solution for the quadrature weights. This least squares solve uses the QR factorization $\mathbf{A}_\mathcal{P} = \mathbf{QR}$. While we do ensure that the weight of the most recently added quadrature point is positive, there is no guarantee that the other weights given by the least-squares solution will remain positive. In the case of negative weights, we note that somewhere in the space between our current solution (containing at least one negative weight) and our previous non-negative solution lies a non-negative solution with small residual. We then set our new solution to this intermediate solution, driving the negative weights to non-negative values (lines 10 and 11). Finally, we prune away any zero-valued weights, removing these quadrature points from $\mathcal{P}$ and adding them back to $\mathcal{Z}$ (lines 12 to 17). Our solution ρ is updated by $\tilde{\rho}$ (line 19). When all of the constraints are satisfied (i.e. $|\mathbf{A}\rho - \mathbf{b}| \leq \delta$), we terminate and return ρ. The NNLS algorithm is a guaranteed solution method [30]; i.e. given the solution exists, the NNLS algorithm is guaranteed to find ρ such that $|\mathbf{A}\rho - \mathbf{b}| \leq \delta$, at least in exact-precision arithmetic.

4.2 | Incremental QR

The majority of the cost of the NNLS algorithm is in the least-squares solve (line 8 of Algorithm 1), which requires QR factorization of $\mathbf{A}_\mathcal{P}$. To compute the factorization in an incremental manner, we perform the factorization using Householder reflectors; i.e. at the kth iteration, when a column $\mathbf{a}_k \in \mathbb{R}^m$ is added to $\mathbf{A}_\mathcal{P}$, we calculate $\mathbf{Q}^k \in \mathbb{R}^{m\times m}$ and $\mathbf{R}^k \in \mathbb{R}^{|\mathcal{P}|\times|\mathcal{P}|}$ such that

$$\mathbf{A}^k_\mathcal{P} = \mathbf{Q}^k \begin{bmatrix} \mathbf{R}^k \\ 0 \end{bmatrix},$$

where we compute the matrix $\mathbf{Q}^k$ by the product of Householder reflectors

$$\mathbf{Q}^k = \mathbf{H_1 H_2} \dots \mathbf{H}_{|\mathcal{P}|}, \tag{31}$$

for a Householder reflector $\mathbf{H}_{|\mathcal{P}|} \in \mathbb{R}^{m\times m}$ given by $\mathbf{H}_{|\mathcal{P}|} = I - \frac{2\mathbf{a}_k\mathbf{a}_k^\mathbf{T}}{\|\mathbf{a}_k\|^2}$. Thus, at each iteration, we simply update the matrix $\mathbf{Q}$ and $\mathbf{R}$ by $\mathbf{Q}^k = \mathbf{Q}^{k-1}\mathbf{H}_{|\mathcal{P}|}$ and

$$\mathbf{R}^k = \begin{bmatrix} \mathbf{R}^{k-1} & \mathbf{r_1} \\ 0 & \mathbf{r_2} \end{bmatrix}, \quad \text{where} \quad \begin{bmatrix} \mathbf{r_1} \\ \mathbf{r_2} \\ 0 \end{bmatrix} = (\mathbf{Q}^k)^\mathbf{T}\frac{\mathbf{a}_k}{\|\mathbf{a}_k\|}. \tag{32}$$

We then solve the least squares problem by solving for our temporary solution $\tilde{\rho}$ such that

$$\mathbf{R}^k\tilde{\rho} = (\mathbf{Q}^k)^\mathbf{T}\mathbf{b}, \tag{33}$$

Algorithm 1 Non-negative least squares [30]

Input: $\mathbf{A} \in \mathbb{R}^{m \times K^h}, \mathbf{b} \in \mathbb{R}^m, \delta \in \mathbb{R}^{K^h}$
Output: $\rho \in \mathbb{R}^{K^h}$
1: $\mathcal{Z} \leftarrow \{1, 2, \ldots, K^h\}$, $\mathcal{P} \leftarrow \emptyset$, $\rho \leftarrow \mathbf{0}$
2: **while** $\left|\mathbf{A}\rho - \mathbf{b}\right| \not< \delta$ **do**
3: $\quad \mathbf{r} \leftarrow \mathbf{b} - \mathbf{A}\rho$
4: $\quad \lambda \leftarrow \mathbf{A}^{\mathbf{T}}\mathbf{r}$
5: $\quad i \leftarrow \arg\max_{i \in \mathcal{Z}}(\lambda_i)$
6: $\quad \mathcal{Z} \leftarrow \mathcal{Z} \backslash \{i\}$
7: $\quad \mathcal{P} \leftarrow \mathcal{P} \cup \{i\}$
8: $\quad \tilde{\rho} \leftarrow \arg\min_{\mathbf{x} \in \mathbb{R}^{|\mathcal{P}|}} \|\mathbf{A}_{\mathcal{P}}\mathbf{x} - \mathbf{b}\|_2$
9: $\quad$ **if** any $\tilde{\rho} < 0$ **then**
10: $\quad\quad a \leftarrow \min_{i \in \mathcal{P}: \tilde{\rho}_i < 0}\{\rho_i/(\rho_i - \tilde{\rho}_i)\}$
11: $\quad\quad \tilde{\rho} \leftarrow \tilde{\rho} + a(\rho - \tilde{\rho})$
12: $\quad\quad$ **for** $i \in \mathcal{P}$ **do**
13: $\quad\quad\quad$ **if** $\tilde{\rho}_i = 0$ **then**
14: $\quad\quad\quad\quad \mathcal{Z} \leftarrow \mathcal{Z} \cup \{i\}$
15: $\quad\quad\quad\quad \mathcal{P} \leftarrow \mathcal{P} \backslash \{i\}$
16: $\quad\quad\quad$ **end if**
17: $\quad\quad$ **end for**
18: $\quad$ **end if**
19: $\quad \rho \leftarrow \tilde{\rho}$
20: **end while**

where the quantity $(\mathbf{Q}^k)^{\mathbf{T}}\mathbf{b}$ is updated simply by $(\mathbf{Q}^k)^{\mathbf{T}}\mathbf{b} = \mathbf{H}_{|\mathcal{P}|}{}^{\mathbf{T}}(\mathbf{Q}^{k-1})^{\mathbf{T}}\mathbf{b}$. This update scheme is more efficient than repeatedly performing the entire QR factorization; in each iteration, the cost to update the QR matrices is $\mathcal{O}(m^2)$ and the cost to solve (33) is $\mathcal{O}(|\mathcal{P}|^2)$.

Remark 5. The update scheme for the QR factorization described here applies only when we add new columns to the matrix $\mathbf{A}_{\mathcal{P}}$, and not when we remove columns in the inner loop on lines 9–18 of Algorithm 1. In the case where columns are pruned from $\mathbf{A}_{\mathcal{P}}$, we must recompute the QR factorization from scratch and cannot perform the simple update of $(\mathbf{Q}^k)^{\mathbf{T}}\mathbf{b}$. This motivates us to limit the number of inner loop iterations, which we attempt to do in Sections 5 and 6.

To compute the iterative QR update, as well as other matrix operations, in a parallel manner, we employ the ScaLAPACK library. Specifically, following [12] for NNLS in ECSW, we employ `pdormqr` to compute (32), `pdgeqrf` to compute (31), `pdormqr` to compute the right-hand-side of (33), and `pdtrsm` to solve (33).

4.3 | Computational requirements

We now derive the estimate of the computational complexity of NNLS (Algorithm 1). Let m and K^h be the total number of constraints and quadrature points, respectively, let K_i be the number of quadrature points in the nonzero set $\mathcal{P}$ on the ith iteration, and let K_f be the final number of points in the nonzero set. Then, in each iteration, we must compute the Lagrange multipliers $\mathbf{A}^{\mathbf{T}}(\mathbf{b} - \mathbf{A}\rho)$ and perform least squares on the columns of $\mathbf{A}$ that are in the nonzero set $\mathcal{P}$, which require $\mathcal{O}(mK^h)$ and $\mathcal{O}(m^2 + K_i^2)$ operations, respectively. Given that $m > K_i$, this simplifies to $\mathcal{O}(m^2)$ operations. All other operations during one NNLS iteration require negligible computation time. Thus, for an NNLS method taking n iterations, the computational complexity is $\mathcal{O}(nm^2 + nmK^h)$. The number of iterations, however, is not so easily determined. In the case of very few redundant constraints in $\mathbf{A}$—such as when constraint reduction is performed beforehand, as will be discussed in Section 6—, we typically observe that $n \approx m$; however, for systems with a poorly condition $\mathbf{A}$ (i.e., with many nearly redundant constraints) or tight tolerances δ, the number of iterations increase to much larger than m; i.e., $n \approx Cm$ for a potentially large C. Thus, in practice, the computational complexity of the NNLS algorithm is approximately $\mathcal{O}(m^3 + m^2K^h)$.

Remark 6. Due to the approximation $n \approx Cm$, and the difficulty of comparing the magnitude of the two terms in our cost expression, we cannot derive a strict relationship between the number of constraints m and the cost. We can, however, conclude that the computational cost scales superlinearly with m.

5 | NNLS WITH ROUNDING-ERROR STABLE RESIDUAL EVALUATION

The process of solving the EQP problem (28)–(30) with a very tight tolerance δ using NNLS often suffers from significant machine precision errors in the residual calculation ($\mathbf{r} = \mathbf{b} - \mathbf{A}_{\mathcal{P}}\rho$) due to the relative size of (large) $\mathbf{A}_{\mathcal{P}}\rho$ and (small) residual. When the rounding-errors are on the order of the residual itself, the algorithm makes a poor selection of columns (line 5 of Algorithm 1), which causes the algorithm to iterate many more times than is necessary or completely stall. Specifically, the algorithm often selects columns of $\mathbf{A}$ that will immediately be pruned due to a negative quadrature weight, wasting many iterations and preventing the use of the incremental QR algorithm described in Section 4.2 in the inner loop (lines 9 to 18); cf. Remark 5.

To mitigate the inefficiency, we propose a more stable form of residual evaluation that is less prone to rounding error, taking advantage of the already computed QR factorization. We first note that, for the least-squares solution ρ that satisfies $\mathbf{R}\rho = \mathbf{Q}^T\mathbf{b}$, $\mathbf{A}_{\mathcal{P}}\rho = \mathbf{Q}\mathbf{R}\rho = \mathbf{Q}\mathbf{Q}^T\mathbf{b}$, which represents the successive projection of $\mathbf{b}$ onto the orthonormal basis represented by the columns of $\mathbf{Q}$. We hence deduce that

$$\mathbf{r} = \mathbf{b} - \mathbf{A}_{\mathcal{P}}\rho = (\mathbf{I} - \mathbf{Q}\mathbf{Q}^T)\mathbf{b}. \tag{34}$$

The evaluation of $\mathbf{r}$ using $\mathbf{Q}$ requires two matrix multiplications, which is more expensive than the direct evaluation via $\mathbf{r} = \mathbf{b} - \mathbf{A}_{\mathcal{P}}\rho$. To avoid the higher cost, we employ the more stable residual evaluation (34) only when it is necessary; i.e., we use it only when the target tolerance δ is tight and when the residual is small, in later iterations of the NNLS algorithm. In order to detect the onset of significant rounding-errors, we simply wait until a column that results in a negative weight is selected and pruned in the same iteration. A theoretical comparison of the computational cost of the two residual evaluation methods is difficult, thus we defer comparison to numerical results in Section 8.1.2.

6 | NNLS WITH CONSTRAINT REDUCTION

In this section, we present an NNLS algorithm with constraint reduction. We first discuss the concept of reducibility of constraints (Section 6.1), then introduce three key ingredients of the algorithm (Sections 6.2–6.4), and finally present the algorithm (Section 6.5).

6.1 | Reducibility of constraints: similarity and redundancy

Given that our NNLS cost scales superlinearly with the number of constraints (Remark 6), the hyperreduction time can be significant for problems with many time-steps, large RB sizes, or many training points. To mitigate this issue, we first recall an observation that underpins many MOR ideas: the solution to our parameterized problem lies on a manifold that is amenable to low-dimensional approximation. We here extend the idea and presume that the parametric constraint manifold $\mathcal{C}$ is amenable to a low-dimensional approximation. Thus we should be able to identify a reduced number $\tilde{m} \ll m$ of constraints such that the solution to (28) with the $\tilde{m}$ constraints satisfies all m original constraints, which provides a good approximation for $\mathcal{C}$.

We recall from Remark 3 that if our parameter and temporal spaces are densely sampled, the set of constraints $\{\mathbf{A}, \mathbf{b}, \rho\}$ is ill-conditioned for the NNLS method. This is due to a high degree of similarity between constraints, often to the point of many completely redundant constraints. In this work, a redundant constraint refers to a constraint c_j that would be satisfied even if it were excluded from our set $\{\mathbf{A}, \mathbf{b}, \rho\}$ due to the similarity with other constraints in the set. This similarity is apparent when we consider that constraints are sampled from a parametric constraint manifold $\mathcal{C}$. For a densely-sampled parameter space or a finely discretized temporal domain, some of the parameter values or time steps will be close, resulting in similar or redundant constraints. In fact, if we consider the limit as $\Delta t \to 0$ or $\Xi^{\text{EQP}} \to \mathcal{D}$, we would have innumerable identical constraints. The similarity in constraints causes "interference"(see Remark 7) in the necessary quadrature weights, and causes our NNLS algorithm to often enter the inner loop (lines 9 to 18) and remove quadrature points from our nonzero set $\mathcal{P}$. This substantially increases the total number of iterations and the number of particularly expensive inner loop iterations (cf. Remark 5).

Remark 7. To describe what "interference" means in the context of EQP constraints and how it leads to increased inner loop NNLS iterations, we first note that we typically observe correlation between similarity in rows of $\mathbf{A}$ (i.e., constraints) and similarity in columns of $\mathbf{A}$. In other words, we have empirically observed that if the constraints are nearly colinear, then the columns of $\mathbf{A}$ are also nearly colinear. For such a set of similar constraints the fact that columns of $\mathbf{A}$ are nearly colinear results in "interference"; i.e. a newly introduced column of $\mathbf{A}$ may cause a previously determined quadrature weight to become negative, due to cancellation between the nearly colinear columns. We note that while this is not a theoretical result, we have observed in practice that, for "well-conditioned" matrices with a low amount of similarity between constraints, the NNLS algorithm is much less likely to require expensive inner iterations.

Remark 8. We note that the desire to have a well-conditioned constraint matrix presents a conflict in our objectives. To obtain a robust ROM we require a dense sampling of $\mathcal{C}$ in order to ensure accuracy of the model over $\mathcal{D}$, but a dense sampling of $\mathcal{C}$ will result in a poorly-conditioned constraint matrix. Luckily, the proposed constraint reduction methods will allow us to achieve both objectives: we take in a large set of ill-conditioned constraints that represent $\mathcal{C}$ adequately and then produce a smaller, well-conditioned set of constraints that represents $\mathcal{C}$. Effectively, the constraint reduction method extracts the important information and removes redundancy from a set of constraints.

The goal of constraint reduction is to construct a smaller and orthogonal (and thus well-conditioned) set of constraints $\{\mathbf{A}, \mathbf{b}, \tilde{\rho}\}$ (i) that can be solved efficiently by the NNLS method and (ii) whose solution satisfies the original constraints $\{\mathbf{A}, \mathbf{b}, \rho\}$, implying that the parametric constraint manifold $\mathcal{C}$ is well-approximated. Our constraint reduction method requires three ingredients to (I) generate constraints, (II) rank constraints, and (III) determine appropriate $\tilde{m} < m$. We presents the three ingredients in the following sections.

6.2 | Generation of constraints

To generate a set of reduced and orthogonal constraints, we employ QR factorization. To this end, we first define "row-wise" QR factorization.

Definition 7 (Row-wise QR factorization)**.** For a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, define the "row-wise" QR factorization by the matrices $\mathbf{Q} \in \mathbb{R}^{m\times n}$ and $\mathbf{R} \in \mathbb{R}^{m\times m}$ such that $\mathbf{A} = \mathbf{RQ}$; here, $\mathbf{Q} = \mathbf{Q}^{\star\mathbf{T}}$ and $\mathbf{R} = \mathbf{R}^{\star\mathbf{T}}$, where $\mathbf{Q}^\star \in \mathbb{R}^{n\times m}$ and $\mathbf{R}^\star \in \mathbb{R}^{m\times m}$ are the standard (column-wise) QR factorization matrices that satisfy $\mathbf{A}^\mathbf{T} = \mathbf{Q}^\star\mathbf{R}^\star$.

The row-wise QR factorization $\mathbf{A} = \mathbf{RQ}$ generates an orthonormal basis for the rows of $\mathbf{A}$ (i.e., the constraints), which is represented by the rows of $\mathbf{Q}$. We now replace the (potentially ill-conditioned) $\mathbf{A}$ with orthogonal $\mathbf{Q}$ in the NNLS problem using the following proposition.

Proposition 6.1 (QR-based constraints for NNLS)
Given a matrix $\mathbf{A} \in \mathbb{R}^{m\times K^h}$, vectors $\mathbf{b} \in \mathbb{R}^m$, and $\delta \in \mathbb{R}^m$, define $\mathbf{Q} \in \mathbb{R}^{m\times K^h}$ and $\mathbf{R} \in \mathbb{R}^{m\times m}$ by a row-wise QR factorization $\mathbf{A} = \mathbf{QR}$. Suppose $\mathbf{b}_\mathbf{Q} \in \mathbb{R}^m$ and $\delta_\mathbf{Q} \in \mathbb{R}^m_{>0}$ satisfy

$$\mathbf{Rb}_\mathbf{Q} = \mathbf{b}, \tag{35}$$

$$\mathbf{R}_{\text{abs}}\delta_\mathbf{Q} \leq \delta, \tag{36}$$

where $\mathbf{R}_{\text{abs}}$ denote the element-wise absolute value of $\mathbf{R}$; i.e., $(\mathbf{R}_{\text{abs}})_{ij} = |\mathbf{R}_{ij}|$. If there exists $\rho^\star \in \mathbb{R}^{K^h}$ such that

$$\left|\mathbf{Q}\rho^\star - \mathbf{b}_\mathbf{Q}\right| < \delta_\mathbf{Q}, \tag{37}$$

then $\rho^\star$ also satisfies

$$\left|\mathbf{A}\rho^\star - \mathbf{b}\right| < \delta.$$

Proof. We first write (37) as

$$\mathbf{b}_\mathbf{Q} - \delta_\mathbf{Q} < \mathbf{Q}\rho^\star < \mathbf{b}_\mathbf{Q} + \delta_\mathbf{Q}.$$

We then multiply by $\mathbf{R}$ to get

$$\mathbf{Rb}_\mathbf{Q} - \mathbf{R}\delta_\mathbf{Q} < \mathbf{RQ}\rho^\star = \mathbf{A}\rho^\star < \mathbf{Rb}_\mathbf{Q} + \mathbf{R}\delta_\mathbf{Q},$$

We next invoke (35) to obtain

$$\left|\mathbf{A}\rho^\star - \mathbf{b}\right| < \mathbf{R}\delta_\mathbf{Q}.$$

We note that, for positive $\delta_\mathbf{Q}$, $\mathbf{R}\delta_\mathbf{Q} \leq \mathbf{R}_{\text{abs}}\delta_\mathbf{Q}$ and invoke (36) to obtain

$$\left|\mathbf{A}\rho^\star - \mathbf{b}\right| < \mathbf{R}\delta_\mathbf{Q} \leq \mathbf{R}_{\text{abs}}\delta_\mathbf{Q} \leq \delta,$$

which is the desired result. □

Remark 9. Note that the element-wise absolute value on $\mathbf{R}$ is necessary to ensure that our new tolerances $\delta_\mathbf{Q}$ are positive. This introduces some conservativeness in our tolerance selection in the sense that, even if our new tolerances are exactly met (i.e., $|\mathbf{Q}\rho - \mathbf{b}_\mathbf{Q}| = \delta_\mathbf{Q}$), some of our original constraints might be satisfied to a tolerance that is substantially smaller than the prescribed tolerance. In Section 6.4, we will introduce an *a posteriori* check of constraint satisfaction to ensure that the QR-based constraints are not overly conservative.

Informed by Proposition 6.1, we use (35) to find the target values $\mathbf{b}_\mathbf{Q}$ for reduced constraints. As for the tolerances $\delta_\mathbf{Q}$, the inequality in $\mathbf{R}_{\text{abs}}\delta_\mathbf{Q} \leq \delta$ allows for some choice in the selection of $\delta_\mathbf{Q}$. In an attempt to equidistribute tolerances $\delta_\mathbf{Q}$, we choose to evaluate each tolerance by

$$\delta_{\mathbf{Q},i} = \min_{j\in\{1,\ldots,i\}} \frac{1}{i\left|\mathbf{R}_{ji}\right|}\delta_j, \quad i = 1,\ldots,m. \tag{38}$$

The tolerance selection (38) is defined such that a constraint tolerance δ_j has approximately equal (or smaller) contributions from the tolerances $\delta_{\mathbf{Q},i}$, $i = 1,\ldots,m$. In other words, for a tolerance $\delta_j \geq \sum_{i=1}^m \left|\mathbf{R}_{ji}\right| \delta_{Q,i}$, the terms $\{\left|\mathbf{R}_{ji}\right| \delta_{Q,i}\}_{i=1}^m$ are approximately equal except when other constraints restrict $\delta_{Q,i}$ to a smaller value, as represented by the minimization in (38).

6.3 | Ranking of constraints

We now develop a method to rank the constraints represented by $\{\mathbf{Q}, \mathbf{b}_\mathbf{Q}, \delta_\mathbf{Q}\}$ so that important constraints can be identified and unnecessary constraints can be discarded. If we normalize our constraints such that all entries of $\delta_\mathbf{Q}$ are equal, we can consider the magnitude of any row of our constraint matrix $\mathbf{Q}$ as a measure of the amount of information introduced by that constraint. Thus we first select the constraint with the largest magnitude as our most important constraint. We then orthogonalize the rest of the constraints with respect to this first constraint and find our second most important constraint. We repeat the process for all m constraints. This is precisely QR factorization with reordering. Thus we make a small change to the procedure described in Section 6.2: we instead use the row-wise QR factorization with reordering given by $\mathbf{PA} = \mathbf{RQ}$, where $\mathbf{P}$ is a permutation matrix such that the constraints $\{\mathbf{Q}, \mathbf{b}_\mathbf{Q}, \delta_\mathbf{Q}\}$ are organized in descending order of importance. As the matrix $\mathbf{P}$ simply represents a reordering of the rows of $\left|\mathbf{Q}\rho^\star - \mathbf{b}_\mathbf{Q}\right| < \delta_\mathbf{Q}$, Proposition 6.1 still applies and the computational cost is not affected.

We recall that $\mathbf{R}_{ii}$ decreases with i for QR factorization with reordering. This implies that the tolerances $\delta_\mathbf{Q}$ given by (38) increase (i.e., loosen) with i. Given that the rows of $\mathbf{Q}$ are normalized and have equal magnitude, this implies that more important constraints (near the top of $\mathbf{Q}$) will have tighter tolerances, while less important constraints (near the bottom of $\mathbf{Q}$) will have looser tolerances, to the point that many of these later constraints will be unnecessary.

6.4 | Truncation of reduced constraints

Thus far, we have determined a set of m constraints that accurately (and conservatively) represent our m original constraints and that are ranked in order of importance. The next step is to truncate our constraint matrix $\mathbf{Q}$ to the first $\tilde{m} < m$ constraints needed to represent our original constraints. Here, we use the word "represent" to mean that the reduced set of constraints ensures that all original constraints are satisfied. It is unclear if we can determine *a priori* $\tilde{m}$ such that $|\mathbf{A}\rho - \mathbf{b}| < \delta$ is satisfied. However, given that the computational cost to evaluate $|\mathbf{A}\rho - \mathbf{b}| < \delta$ is small compared to the cost of the NNLS solve (which scales superlinearly with $\tilde{m}$), we can check *a posteriori* if original constraints are satisfied. Thus we propose an iterative method: in each iteration, we increase $\tilde{m}$, solve for ρ associated with the first $\tilde{m}$ constraints of $\{\mathbf{Q}, \mathbf{b}_\mathbf{Q}, \delta_\mathbf{Q}\}$, and then check if $|\mathbf{A}\rho - \mathbf{b}| < \delta$ holds. This method ensures that we select an acceptable value for $\tilde{m}$; however, the method involves many NNLS solves that becomes increasingly more expensive with $\tilde{m}$.

To increase the efficiency, we attempt to predict $\tilde{m}$ *a priori* at each iteration of our method, rather than blindly increasing $\tilde{m}$ by a predetermined value. To this end, we first denote the i-th row of $\mathbf{Q}$ and $\mathbf{A}$ by $\mathbf{q}_i$ and $\mathbf{a}_i$, respectively. Our goal is to find $\tilde{m}$ such that, for some solution $\rho \in \mathbb{R}^{K_h}$ that satisfies

$$|\mathbf{q}_i^{\mathbf{T}} \rho - \mathbf{b}_{\mathbf{Q},i}| < \delta_{\mathbf{Q},i} \quad i = 1, \ldots, \tilde{m},$$

our solution ρ will also satisfy

$$|\mathbf{a}_i^{\mathbf{T}} \rho - \mathbf{b}_i| < \delta_i \quad i = 1, \ldots, m.$$

To begin, we introduce $\rho_{\text{prev}} \in \mathbb{R}^{K_h}$ that satisfies the first $\tilde{m}_{\text{prev}} < \tilde{m}$ QR-based constraints from the previous iteration of the iterative constraint-enrichment algorithm. We then note that

$$\begin{aligned} |\mathbf{a}_i^T \rho - b_i| &= |\sum_{j=1}^{m} \mathbf{R}_{ij}(\mathbf{q}_j^T \rho) - \mathbf{b}_i| = |\sum_{j=1}^{\tilde{m}} \mathbf{R}_{ij}(\mathbf{q}_j^T \rho) + \sum_{j=\tilde{m}+1}^{m} \mathbf{R}_{ij}(\mathbf{q}_j^T \rho) - \mathbf{b}_i| \\ &\approx |\sum_{j=1}^{\tilde{m}} \mathbf{R}_{ij}\mathbf{b}_{\mathbf{Q},j} + \sum_{j=\tilde{m}+1}^{m} \mathbf{R}_{ij}(\mathbf{q}_j^T \rho) - \mathbf{b}_i| \approx |\sum_{j=1}^{\tilde{m}} \mathbf{R}_{ij}\mathbf{b}_{\mathbf{Q},j} + \sum_{j=\tilde{m}+1}^{m} \mathbf{R}_{ij}(\mathbf{q}_j^T \rho_{\text{prev}}) - \mathbf{b}_i|, \end{aligned} \tag{39}$$

where the first equality follows from $\mathbf{a}_i = \sum_{j=1}^{m} \mathbf{R}_{ij}\mathbf{q}_j$, and the second equality follows from splitting the sum at $\tilde{m}$. The last two approximations require two additional assumptions. The first approximation follows from the assumption that the tolerances $\delta_{\mathbf{Q},j}$, $j = 1, \ldots, \tilde{m}$, are negligible compared to the truth values $\mathbf{b}_{\mathbf{Q},j}$. The second approximation follows from the assumption that the residual associated with ρ is well approximated by ρ_{prev}: i.e., $\sum_{j=\tilde{m}+1}^{m} \mathbf{R}_{ij}(\mathbf{q}_j^T \rho) \approx \sum_{j=\tilde{m}+1}^{m} \mathbf{R}_{ij}(\mathbf{q}_j^T \rho_{\text{prev}})$; see Remark 10. The final expression of (39) contains only known values and is computable for any $\tilde{m}$. We find the smallest integer $\tilde{m}$ that satisfies

$$\left| \sum_{j=1}^{\tilde{m}} \mathbf{R}_{ij}\mathbf{b}_{\mathbf{Q},j} + \sum_{j=\tilde{m}+1}^{m} \mathbf{R}_{ij}(\mathbf{q}_j^{\mathbf{T}} \rho_{\text{prev}}) - \mathbf{b}_i \right| < \delta_i, \quad i = \tilde{m}+1, \tilde{m}+2, \ldots, \tilde{m} + \tilde{m}_{\text{extra}}, \tag{40}$$

and perform the NNLS solve. As the constraints are ranked in order of importance, we presume that the satisfaction of the $\tilde{m}_{\text{extra}}$ constraints implies the satisfaction of the rest of the constraints. In practice, we choose $\tilde{m}_{\text{extra}} = 5$.

Remark 10. In the last approximation in (39), while ρ may not be well-approximated by ρ_{prev}, we expect that the sum of the projection of ρ onto the constraints $\mathbf{q}_j$ is well-approximated using the previous solution ρ_{prev}. We again note that the *a posteriori* check will ensure satisfaction of all constraints, even if the approximations in (40) are poor.

Remark 11. The total number of constraints m is somewhat arbitrary, in that m represents the number of constraints we have sampled from the constraint manifold $\mathcal{C}$, but does not reflect the complexity of approximating $\mathcal{C}$ using a set of constraints. On the other hand, the number of reduced constraints $\tilde{m}$ reflects the complexity of $\mathcal{C}$, and approximates the minimal number of constraints needed to represent $\mathcal{C}$ to the degree of accuracy implicit in our selection of the original tolerances δ.

Remark 12. While $\tilde{m}$ hints at the complexity of $\mathcal{C}$, we also note that it is a conservative estimate; i.e., the minimal value of $\tilde{m}$ such that $\mathcal{C}$ is well-represented is likely lower than that determined by the constraint reduction methods due to various conservative choices.

6.5 | Algorithm: NNLS with constraint reduction

We put together the three ingredients of constraints reduction introduced in Sections 6.2–6.4 and present the NNLS with constraint reduction (NNLS-CR) in Algorithm 2. We start by normalizing the rows $\{\mathbf{A}, \mathbf{b}, \delta\}$ such that they all have the same tolerance (line 1). We then initialize $\tilde{m} = 0.1m$ (line 2), where m is the number of constraints in $\mathbf{A}$. On each iteration, we perform partial row-wise QR factorization with reordering, denoted by $\texttt{QRP}_{\text{row}}$, on $\mathbf{A}$ to obtain the first $\tilde{m} + \tilde{m}_{\text{extra}}$ orthogonal constraint vectors represented by $\mathbf{Q}^{(\tilde{m}+\tilde{m}_{\text{extra}})}$ (line 6). We then use (35) and (38) with row-reordering to calculate $\mathbf{b}_{\mathbf{Q}}$ and $\delta_{\mathbf{Q}}$ (lines 7 and 8). We iterate through increasing $\tilde{m}$, performing QR factorization and calculating $\mathbf{b}_{\mathbf{Q}}$ and $\delta_{\mathbf{Q}}$ until the termination condition (40) is satisfied (lines 4–9). Note that the QR factorization step represents an incremental update; i.e., we do not recalculate the already computed rows of $\mathbf{Q}$, but $\mathbf{b}_{\mathbf{Q}}$ and $\delta_{\mathbf{Q}}$ will be recalculated each iteration. We then perform NNLS on the $\tilde{m}$ constraints represented

by $\{\mathbf{Q}^{(\tilde{m})}, \mathbf{b}_{\mathbf{Q}}^{(\tilde{m})}, \delta_{\mathbf{Q}}^{(\tilde{m})}\}$ to get a solution ρ (line 10). If ρ satisfies $|\mathbf{A}\rho - \mathbf{b}| < \delta$, then we return ρ; otherwise, we iterate through this whole process, adding new constraints such that $\tilde{m} = \tilde{m} + 0.1m$, until ρ satisfies all constraints (lines 11 and 5).

Algorithm 2 Non-negative least squares with constraint reduction

Input: $\mathbf{A} \in \mathbb{R}^{m \times K^h}, \mathbf{b} \in \mathbb{R}^m, \delta \in \mathbb{R}^{K^h}$
Output: $\rho \in \mathbb{R}^{K^h}$

1: Normalize rows of $\mathbf{A}, \mathbf{b}, \delta$ by the tolerances δ
2: $m \leftarrow$ #rows($\mathbf{A}$), $\quad \tilde{m} \leftarrow 0$, $\quad \rho \leftarrow \mathbf{0}$, $\quad \tilde{m}_{\text{extra}} \leftarrow 5$
3: **while** $\tilde{m} < m$ **do**
4: **while** Condition (40) is not satisfied **do**
5: $\tilde{m} \leftarrow \tilde{m} + 0.1m$
6: Construct $\{\mathbf{Q}^{(\tilde{m}+\tilde{m}_{\text{extra}})}, \mathbf{R}^{(\tilde{m}+\tilde{m}_{\text{extra}})}, \mathbf{P}^{(\tilde{m}+\tilde{m}_{\text{extra}})}\} \leftarrow \text{QRP}_{\text{row}}(\mathbf{A}, \tilde{m} + \tilde{m}_{\text{extra}})$
7: $\mathbf{b}_{\mathbf{Q}}^{(\tilde{m}+\tilde{m}_{\text{extra}})} \leftarrow (\mathbf{R}^{(\tilde{m}+\tilde{m}_{\text{extra}})})^{-1}\mathbf{P}^{(\tilde{m}+\tilde{m}_{\text{extra}})}\mathbf{b}$
8: $\delta_{\mathbf{Q},i}^{(\tilde{m}+\tilde{m}_{\text{extra}})} \leftarrow \min_j \frac{1}{i|\mathbf{R}_{ji}|}(\mathbf{P}^{(\tilde{m}+\tilde{m}_{\text{extra}})}\delta)_j, \quad i = 1, \ldots, \tilde{m} + \tilde{m}_{\text{extra}}$
9: **end while**
10: Solve NNLS$(\mathbf{Q}^{(\tilde{m})}, \mathbf{b}_{\mathbf{Q}}^{(\tilde{m})}, \delta_{\mathbf{Q}}^{(\tilde{m})})$ for ρ
11: **if** $|\mathbf{A}\rho - \mathbf{b}| < \delta$ **then**
12: Terminate.
13: **end if**
14: **end while**

Remark 13. The constraint reduction in NNLS-CR occurs at the constraint-by-constraint level, which is finer than reducing the parameter set Ξ^{EQP}, since each parameter value yields many constraints. This results in a greater reduction in the number of constraints and more well-conditioned constraint matrix with less redundancy.

Remark 14. As the proposed method is purely algebraic, the constraint reduction method can be applied to any constrained minimization problem of the form (28) regardless of how $\{\mathbf{A}, \mathbf{b}, \delta\}$ are formed. Potential applications include other hyperreduction methods such as ECSW [23] as well as compressive sensing [24] and sparse nonlinear regression [8]; see Remark 4.

7 | HIGH-DIMENSIONAL PROBLEMS: ADAPTIVE EQP TRAINING AND GREEDY ALGORITHM

We have so far considered constraint *reduction*, where we start with a set of constraints $\{\mathbf{A}, \mathbf{b}, \delta\}$ associated with the EQP training parameter set Ξ^{EQP} and identify a smaller but representative set of constraints $\{\mathbf{Q}, \mathbf{b}_{\mathbf{Q}}, \delta_{\mathbf{Q}}\}$. An implicit assumption in the constraint *reduction* is that the original parameter set Ξ^{EQP} may be large, but not excessively large, so that we can form $\{\mathbf{A}, \mathbf{b}, \delta\}$. This assumption, however, may be violated in high-dimensional problem, where it is difficult to select *a priori* the set Ξ^{EQP} that provides (i) a sufficient coverage of the parameter space $\mathcal{D}$ but (ii) is not excessively large so that the constraints $\{\mathbf{A}, \mathbf{b}, \delta\}$ can be formed. To address the problem, we propose a formulation that adaptively and incrementally enriches the training parameter set Ξ^{EQP}, so that the resulting set meets both criteria (i) and (ii). We first present a key ingredient of the algorithm, NNLS for incrementally updated constraints, in Section 7.1. We then present a simultaneous RB–RQ training algorithm for high-dimensional problems in Section 7.2.

7.1 | NNLS-CRi for incrementally updated $\{\mathbf{A}, \mathbf{b}, \delta\}$

In order to develop an adaptive EQP sampling strategy, we first develop a modified version of the NNLS-CR algorithm for the cases where Ξ^{EQP} is incrementally enriched. To this end, we develop an NNLS-CRi algorithm, which incrementally update $\{\mathbf{Q}, \mathbf{b}_{\mathbf{Q}}, \delta_{\mathbf{Q}}\}$ with Ξ^{EQP}. To begin, suppose we have formed a reduced set of $\tilde{m}_{\text{prev}}$ constraints $\{\mathbf{Q}_{\text{prev}}, b_{\mathbf{Q},\text{prev}}, \delta_{\mathbf{Q},\text{prev}}\}$ associated with original m_{prev} constraints $\{\mathbf{A}_{\text{prev}}, \mathbf{b}_{\text{prev}}, \delta_{\text{prev}}\}$ in the previous iteration, and we are now given a new set of m_{new} constraints

$\{\mathbf{A}_{\text{new}}, \mathbf{b}_{\text{new}}, \delta_{\text{new}}\}$ that augment the original constraints. To find a new set of reduced constraints that represent also the new constraints, we first orthogonalize the new constraints $\mathbf{A}_{\text{new}}$ with respect to the previous reduced constraints $\mathbf{Q}_{\text{prev}}$, and make the associated adjustments to $\mathbf{b}_{\text{new}}$ and δ_{new}. To obtain the orthogonalized constraints $\{\hat{\mathbf{A}}, \hat{\mathbf{b}}, \hat{\delta}\}$ from $\{\mathbf{A}_{\text{new}}, \mathbf{b}_{\text{new}}, \delta_{\text{new}}\}$, we first orthogonalize and adjust the i-th constraint $\{\mathbf{a}_{\text{new},i}, \mathbf{b}_{\text{new},i}, \delta_{\text{new},i}\}$ as

$$\hat{\mathbf{a}}_i = \mathbf{a}_{\text{new},i} - \sum_{j=1}^{\tilde{m}_{\mathbf{Q},\text{prev}}} (\mathbf{a}_{\text{new},i}^T \mathbf{q}_{\text{prev},j}) \mathbf{q}_{\text{prev},j}, \tag{41}$$

$$\hat{\mathbf{b}}_i = \mathbf{b}_{\text{new},i} - \sum_{j=1}^{\tilde{m}_{\mathbf{Q},\text{prev}}} (\mathbf{a}_{\text{new},i}^T \mathbf{q}_{\text{prev},j}) \mathbf{b}_{\mathbf{Q},\text{prev},j}, \tag{42}$$

for $i = 1, \ldots, m_{\text{new}}$. As for the tolerances $\hat{\delta}$, we need to make two adjustments: first, we need to ensure that the tolerances $\delta_{\mathbf{Q},\text{prev}}$ are sufficiently small for the part of $\mathbf{a}_{\text{new},i}$ that they now represent; second, we need to adjust the tolerances δ_{new} to account for the orthogonalization. To handle the second adjustment we first consider a "naive" adaptation of the tolerances that follows (41) and (42), given by

$$\hat{\delta}_i^{\text{naive}} = \delta_{\text{new},i} - \sum_{j=1}^{\tilde{m}_{\mathbf{Q},\text{prev}}} |\mathbf{a}_{\text{new},i}^T \mathbf{q}_{\text{prev},j}| \delta_{\mathbf{Q},\text{prev},j}, \tag{43}$$

where the absolute value accounts for the fact that constraints can be satisfied to δ_{new} above or below $\mathbf{b}_{\text{new}}$. However, the naive tolerances may yield an ill-defined $\hat{\delta}^{\text{naive}}$ that contains negative values. We may need to tighten the previously determined tolerances $\delta_{\mathbf{Q}}$ to ensure the tolerance is positive, which ensures that the previously determined set of reduced constraints captures the information contained in the part of the constraint that was removed in (41): $\sum_{j=1}^{\tilde{m}_{\mathbf{Q},\text{prev}}} (\mathbf{a}_{\text{new},i}^T \mathbf{q}_{\text{prev},j}) \mathbf{q}_{\text{prev},j}$. We make this adjustment by setting

$$\hat{\delta}_{\mathbf{Q},\text{prev},i} = \min \left\{ \delta_{\mathbf{Q},\text{prev},i}, \min_j \left(\frac{\delta_j}{(\mathbf{a}_{\text{new},j}^T \mathbf{q}_{\text{prev},i}) \tilde{m}_{\mathbf{Q},\text{prev}}} \right) \right\}, \quad i = 1, \ldots, \tilde{m}_{\mathbf{Q},\text{prev}}. \tag{44}$$

We then make the second adjustment (to replace our naive attempt)

$$\hat{\delta}_i = \delta_i - \sum_{j=1}^{\tilde{m}_{\mathbf{Q},\text{prev}}} |\mathbf{a}_{\text{new},i}^T \mathbf{q}_{\text{prev},j}| \hat{\delta}_{\mathbf{Q},\text{prev},i}, \quad i = 1, \ldots, m_{\text{new}}. \tag{45}$$

By following (41), (42), (44), and (45), we obtain the orthogonalized constraints $\{\mathbf{A}, \mathbf{b}, \boldsymbol{\delta}\}$ that augment the reduced constraint set $\{\mathbf{Q}_{\text{prev}}, b_{Q,\text{prev}}, \hat{\delta}_{\mathbf{Q},\text{prev}}\}$ with adjusted tolerances $\boldsymbol{\delta}_{\mathbf{Q},\text{prev}}$. We then apply the NNLS-CR algorithm (Algorithm 2) to $\{\mathbf{A}_{\text{tot}}, \mathbf{b}_{\text{tot}}, \delta_{\text{tot}}\}$, where

$$\mathbf{A}_{\text{tot}} \coloneqq \begin{pmatrix} \mathbf{Q}_{\text{prev}} \\ \hat{\mathbf{A}} \end{pmatrix}, \quad \mathbf{b}_{\text{tot}} \coloneqq \begin{pmatrix} \mathbf{b}_{\mathbf{Q},\text{prev}} \\ \hat{\mathbf{b}} \end{pmatrix}, \quad \text{and} \quad \delta_{\text{tot}} \coloneqq \begin{pmatrix} \hat{\delta}_{\mathbf{Q},\text{prev}} \\ \hat{\delta} \end{pmatrix};$$

the only modification is that, in line 2, we set $\tilde{m} = \tilde{m}^{\text{prev}}$ so that the first $\tilde{m}_{\text{prev}}$ rows of the new reduced constraints is $\{\mathbf{Q}_{\text{prev}}, b_{Q,\text{prev}}, \hat{\delta}_{\mathbf{Q},\text{prev}}\}$. We name the resulting algorithm NNLS-CRi, where the "i" designates that it is designed for iterative update. The procedure is summarized in Algorithm 3. Note that, as we initialize $\tilde{m} = \tilde{m}^{\text{prev}}$, the iterative enrichment of the constraints using QR factorization with reordering (lines 4–9) is applied only to the subset of newly added constraints $\{\hat{\mathbf{A}}, \hat{\mathbf{b}}, \hat{\delta}\}$.

7.2 | Greedy algorithm and adaptive EQP for high-dimensional problems

We now present a simultaneous greedy RB–RQ training algorithm that uses an adaptive EQP to tackle high-dimensional problems. We provide an overview of the greedy algorithm in Algorithm 4 and details of the adaptive EQP in line 19 in Algorithm 5.

At high level, Algorithm 4 is a "standard" greedy algorithm but with two modifications: (i) the use of separate and additional accuracy testing (lines 9–13); and (ii) the use of adaptive EQP (line 19). Given a ROM of dimension N equipped with an online-efficient *a posteriori* error estimate, we first find the least-well-approximated parameter μ^{N+1} from a randomly constructed training set Ξ_J^{train} (line 7). We then check if the maximum error over the training set Ξ_J^{train} is less than the target tolerance. If the

Algorithm 3 Non-negative least squares with constraint reduction for iterative update

Input: $\mathbf{Q}_{\text{prev}} \in \mathbb{R}^{m_{\text{prev}} \times K^h}$, $\mathbf{b}_{\mathbf{Q},\text{prev}} \in \mathbb{R}^{m_{\text{prev}}}$, $\delta_{\mathbf{Q},\text{prev}} \in \mathbb{R}^{m_{\text{prev}}}$
$\mathbf{A}_{\text{new}} \in \mathbb{R}^{m_{\text{new}} \times K^h}$, $\mathbf{b}_{\text{new}} \in \mathbb{R}^{m_{\text{new}}}$, $\delta_{\text{new}} \in \mathbb{R}^{m_{\text{new}}}$
Output: $\rho \in \mathbb{R}^{K^h}$

Apply (41), (42), (44), and (45) to $\{\mathbf{A}_{\text{new}}, \mathbf{b}_{\text{new}}, \delta_{\text{new}}\}$ to obtain $\{\hat{\mathbf{A}}, \hat{\mathbf{b}}, \hat{\delta}\}$ and $\hat{\delta}_{\mathbf{Q},\text{prev}}$
Normalize $\{\hat{\mathbf{A}}, \hat{\mathbf{b}}, \hat{\delta}\}$
Form augmented constraints $\left\{ \mathbf{A}_{\text{tot}} = \begin{pmatrix} \mathbf{Q}_{\text{prev}} \\ \hat{\mathbf{A}} \end{pmatrix}, \mathbf{b}_{\text{tot}} = \begin{pmatrix} \mathbf{b}_{\mathbf{Q},\text{prev}} \\ \hat{\mathbf{b}} \end{pmatrix}, \delta_{\text{tot}} = \begin{pmatrix} \hat{\delta}_{\mathbf{Q},\text{prev}} \\ \hat{\delta} \end{pmatrix} \right\}$
Apply NNLS-CR (Algorithm 2) to $\{\mathbf{A}_{\text{tot}}, \mathbf{b}_{\text{tot}}, \delta_{\text{tot}}\}$ starting with $\tilde{m} = \tilde{m}_{\text{prev}}$

target tolerance is met over all Ξ_J^{train}, then we re-test the ROM over a separate and larger test set Ξ_K^{test} and terminate if the ROM meets the target tolerance over also the test set. If the ROM does not meet the target error tolerance (over Ξ_J^{train} or Ξ_K^{test}), then we solve the FOM problem at μ^{N+1} to obtain the primal and dual snapshot (line 16); we perform adaptive mesh refinement as necessary so that each snapshot meets the target error tolerance. We then update the RB parameter set Ξ_N^{RB}, primal RB $\mathbf{\Phi}_N$, and the dual RB $\mathbf{\Phi}_N^{\text{du}}$ (lines 17 and 18). We finally update the sparse RQ rules using the adaptive EQP (line 19).

Algorithm 4 Simultaneous greedy RB-RQ training

Input: ROM tolerance: δ^{ROM}
EQP tolerances: δ^r, δ^q, δ^η
FOM tolerance: δ^{FOM}
training and test set sizes: J, K
Output: primal and dual RBs: $\mathbf{\Phi}_N$, $\mathbf{\Phi}_N^{\text{du}}$
primal residual, output functional, and DWR RQ weights: ρ^r, ρ^q, ρ^η

1: **for** $N = 0, 1, 2, \ldots$ **do**
2: Reset training set Ξ_J^{train} with J random $\mu \in \mathcal{D}$
3: **if** $N = 0$ **then**
4: Choose centroidal parameter: $\mu^{N+1} = \text{Centroid}(\Xi_J^{\text{train}})$
5: **else**
6: Evaluate error estimate $\tilde{\eta}_N(\mu) = \tilde{\mathbf{z}}_N^{\text{du}T} \mathbf{r}^{\text{du},\eta}(\tilde{\mathbf{u}}_N(\mu); \mu)$ for all $\mu \in \Xi_J^{\text{train}}$
7: Find parameter that maximizes error estimate: $\mu^{N+1} = \arg\max_{\mu \in \Xi_J^{\text{train}}} \tilde{\eta}_N(\mu)$
8: **end if**
9: **if** $\eta_N(\mu^{N+1}) < \delta^{\text{ROM}}$ **then**
10: Populate test set Ξ_K^{test} with $K - J$ random $\mu \in \mathcal{D}$
11: Evaluate error estimate $\tilde{\eta}_N(\mu) = \tilde{\mathbf{z}}_N^{\text{du}T} \mathbf{r}^{\text{du},\eta}(\tilde{\mathbf{u}}_N(\mu); \mu)$ for all $\mu \in \Xi_K^{\text{test}}$
12: **if** $\max_{\mu \in \Xi_K^{\text{test}}} \tilde{\eta}_N(\mu) < \delta^{\text{ROM}}$ **then**
13: Terminate.
14: **end if**
15: **end if**
16: Solve primal and dual FOM problems: $\{\mathbf{u}_h(\mu^{N+1}), \mathbf{z}_h(\mu^{N+1})\} \leftarrow \text{FOM}(\mu^{N+1}; \delta^{\text{FOM}})$
17: Update the RB parameter set: $\Xi_{N+1}^{\text{RB}} \leftarrow \Xi_N^{\text{RB}} \cup \{\mu^{N+1}\}$
18: Update the primal and dual RBs: $\mathbf{\Phi}_N \leftarrow \text{GS}(\{\mathbf{\Phi}_N, \mathbf{u}_h(\mu^{N+1})\})$, $\mathbf{\Phi}_N^{\text{du}} \leftarrow \text{GS}(\{\mathbf{\Phi}_N^{\text{du}}, \mathbf{z}_h(\mu^{N+1})\})$
19: Update the RQ weights ρ^r, ρ^q, and ρ^η using adaptive EQP Algorithm 5.
20: **end for**

Given an online-efficient *a posteriori* error estimate, the "standard" greedy algorithm permits the use of a large training set Ξ_J^{train} without incurring a significant additional cost. For instance, if the cost of the ROM solve and error estimate is $\mathcal{O}(10^{-3})$ of a single FOM solve, then we can use $|\Xi_J^{\text{train}}| = \mathcal{O}(10^3)$ without making the error sampling a significant computational bottleneck. In Algorithm 4, we incorporate a separate and additional test set Ξ_K^{test} so that an even larger set can be used to validate the accuracy

of the ROM at convergence. In this sense, *if we did not have to consider the cost of EQP*, then the greedy method readily scales to high-dimensional problem that may require a large Ξ_J^{train} (and Ξ_K^{test}) to provide a sufficient coverage of $\mathcal{D}$. However, if we use the same Ξ_J^{train} as the training parameter set for $\texttt{EQP}^\bullet$ in (line 19), then the method would become computationally prohibitive for a large Ξ_J^{train} as the number of constraints scales with $|\Xi_J^{\text{train}}|$. We hence wish to devise an EQP that uses adaptively chosen, and ideally much smaller, subset $\Xi^{\text{EQP}} \subset \Xi_J^{\text{train}}$ as the EQP training set, so that the greedy algorithm scales to high-dimensional problems.

We present in Algorithm 19 an EQP that adaptively enriches the training set Ξ^{EQP} to ensure that the training set provides a sufficient, but not excessive, coverage of the parameter space $\mathcal{D}$. We first set our additional set size n^{add} to 25 and our "saturation" test set size n^{testSat} to 10 (line 1). We then iterate through the following steps. We first use $\texttt{EQP}^r$ (or $\texttt{EQP}^{r2}$) to construct the constraints $\{\mathbf{A}^{\text{new}}, \mathbf{b}^{\text{new}}, \delta^{\text{new}}\}$ associated with the additional parameter set Ξ^{add} (line 5). We then invoke NNLR-CRi (Algorithm 3) on the new constraints $\{\mathbf{A}^{\text{new}}, \mathbf{b}^{\text{new}}, \delta^{\text{new}}\}$ to updated the reduced constraints $\{\mathbf{Q}, \mathbf{b}_\mathbf{Q}, \delta_\mathbf{Q}\}$ and the primal residual RQ weights ρ^r (line 7). We next re-populate the sets Ξ^{add} and Ξ^{testSat} (lines 8 and 9). We finally test for saturation of constraints using Ξ^{testSat} (line 11). If the constraints are not satisfied, then we restart the loop, adding Ξ^{add} to the training set. If the constraints are satisfied, then we then construct the output functional and error estimate RQ rules using the training set Ξ^{EQP} found through the iterative process (lines 15–18).

Algorithm 5 EQP with adaptive selection of training set

Input: EQP tolerances: $\delta^r, \delta^q, \delta^\eta$
training parameter set: Ξ_J^{train}
RB parameters Ξ_N^{RB}
Output: EQP weights: ρ^r, ρ^q, ρ^η
EQP training set of parameters: Ξ^{EQP}

1: Set $n^{\text{add}} \leftarrow 25$, $n^{\text{testSat}} \leftarrow 10$
2: Initialize empty constraints $\{\mathbf{Q} \leftarrow, \mathbf{b}_\mathbf{Q} \leftarrow, \delta_\mathbf{Q} \leftarrow\}$.
3: Set $\Xi^{\text{add}} \leftarrow \Xi_N^{\text{RB}}$
4: **while** $\left|\Xi^{\text{EQP}}\right| < \left|\Xi_J^{\text{train}}\right|$ **do**
5: Construct constraints $\{\mathbf{A}^{\text{add}}, \mathbf{b}^{\text{add}}, \delta^{\text{add}}\}$ using $\texttt{EQP}^r(\Xi^{\text{add}}, \delta^r)$
6: Set $\Xi^{\text{EQP}} \leftarrow \Xi^{\text{EQP}} \cup \Xi^{\text{add}}$
7: Obtain new $\{\mathbf{Q}, \mathbf{b}_\mathbf{Q}, \delta_\mathbf{Q}\}$ and ρ^r using NNLS-CRi$(\{\mathbf{Q}, \mathbf{b}_\mathbf{Q}, \delta_\mathbf{Q}\}, \{\mathbf{A}, \mathbf{b}, \delta\})$
8: Populate Ξ^{add} with n^{add} random $\mu \in \Xi_J^{\text{train}} \backslash \Xi^{\text{EQP}}$
9: Populate Ξ^{testSat} with n^{testSat} random $\mu \in \Xi^{\text{add}}$
10: Construct constraints $\{\mathbf{A}^{\text{testSat}}, \mathbf{b}^{\text{testSat}}, \delta^{\text{testSat}}\}$ using $\texttt{EQP}^r(\Xi^{\text{testSat}}, \delta^r)$
11: **if** $\left|\mathbf{A}^{\text{testSat}}\rho - \mathbf{b}^{\text{testSat}}\right| < \delta^{\text{testSat}}$ **then**
12: Terminate.
13: **end if**
14: **end while**
15: Construct constraints $\{\mathbf{A}, \mathbf{b}, \delta\}$ using $\texttt{EQP}^q(\Xi^{\text{EQP}}, \delta^q)$
16: Obtain ρ^q using NNLS-CR$(\{\mathbf{A}, \mathbf{b}, \delta\})$
17: Construct constraints $\{\mathbf{A}, \mathbf{b}, \delta\}$ using $\texttt{EQP}^\eta(\Xi^{\text{EQP}}, \delta^\eta)$
18: Obtain ρ^η using NNLS-CR$(\{\mathbf{A}, \mathbf{b}, \delta\})$

Remark 15. Algorithm 5 uses the same set of EQP training parameters Ξ^{EQP} for $\texttt{EQP}^r$ (or $\texttt{EQP}^{r2}$), $\texttt{EQP}^q$ and $\texttt{EQP}^\eta$, with an implicit assumption that Ξ^{EQP} that provides sufficient coverage for $\texttt{EQP}^r$ is sufficient for the other two. Numerical studies in Section 8.4 support this assumption, at least for problems considered. For problems where the assumption may be violated, we could instaed apply the adaptive EQP training procedure separately to find each RQ rule for additional cost.

Remark 16. We have found through numerical experiments that the parameters n^{add} and n^{testSat} that specify the enrichment behavior do not have much of an effect on the overall performance of the greedy algorithm. We choose $n^{\text{add}} = 25$ based on an observation that $|\Xi^{\text{EQP}}| < 200$ for all test cases that we have considered; $n^{\text{add}} = 25$ provides a balance between minimizing the number of enrichment iterations in Algorithm 5 and providing sufficient granularity to not substantially overshoot the minimum

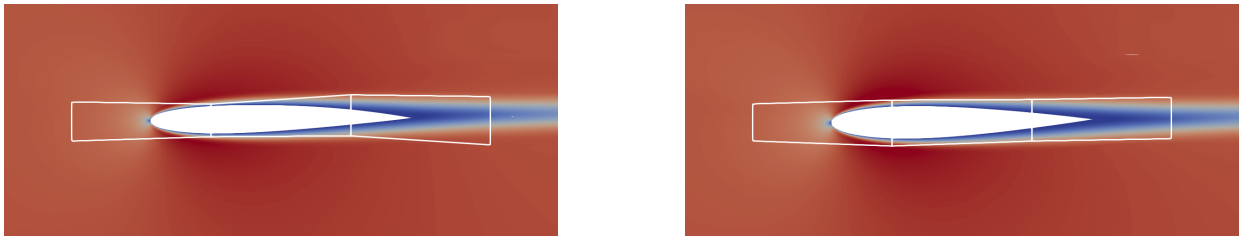


FIGURE 1 Deformed NACA0012 airfoil for two different lattice configurations [19].

required size of Ξ^{EQP}. For the second parameter, we set $n^{\text{testSat}} = 10$ heuristically to balance sufficient check and computational cost. We again note that the choice of n^{add} and n^{testSat} have little impact on the overall performance of the greedy algorithm.

8 | EXAMPLES

In this section, we assess the developed methods using four test cases: compressible Navier–Stokes flow over shape-parameterized airfoils (Section 8.1), the Reynolds-averaged Navier–Stokes (RANS) flow over the ONERA M6 wing (Section 8.2), data assimilation of unsteady Navier–Stokes flow (Section 8.3), and UQ of RANS flow over the RAE2822 airfoil (Section 8.4).

8.1 | Navier–Stokes flow over shape-parametrized airfoil

8.1.1 | Case description

We first consider model reduction of compressible Navier–Stokes flow over an airfoil whose shape is parameterized by free-form deformation (FFD) [35]. The undeformed airfoil is NACA0012, with the leading edge at (0,0) and normalized to have a unit chord length. We then introduce a 4×2 equispaced FFD control lattice over $[-1.1, 2.1] \times [-0.16, 0.16]$, where each lattice point can be deformed in the x_2 direction by $\Delta x \in [-0.02, 0.02]$. The eight variable lattice points define our parameter space, $\mathcal{D} \coloneqq [-0.02, 0.02]^8 \subset \mathbb{R}^8$. Figure 1 shows the NACA0012 airfoil, under FFD for different lattice configurations. For more detailed discussion of FFD and its incorporation in the RB–RQ method, we refer to [19, 20].

The flow condition is given by a fixed chord-based Reynolds number $Re_c = 4000$, freestream Mach number $M_\infty = 0.3$, and angle of attack $\alpha = 1^\circ$. Under these conditions, the flow remains laminar and, while compressibility effect is present, there are no shock waves. Our output quantity of interest is the drag coefficient on the airfoil, which, for the given flow condition and shape-parameter range, lies in $c_d \in [0.055, 0.065]$. We thus consider error tolerances in our output of 0.0006 for 1% error or 0.0003 for 0.5% error.

We obtain the "truth" FE solution using an adaptive high-order DG method. A $p = 2$ mesh is generating through anisotropic adaptive mesh refinement such that the DWR error estimate for all training parameter values is less than 0.5%; i.e., $\eta_h(\mu) \leq 0.0003$, $\forall \mu \in \Xi^{\text{train}}$. The resulting FE approximation space has $N_h = 10{,}968$ degrees of freedom with $K^h = 22{,}602$ quadrature points.

We demonstrate two important elements of the EQP, NNLS, and NNLS-CR methods: (i) the need for rounding-error stable residual computation (as described in Section 5) to achieve tight tolerances δ; and (ii) the substantial reduction in the number of constraints needed to determine our RQ rules, and the associated reduction in the offline training time achieved by employing NNLS-CR (as discussed in Section 6). To this end, we consider large predetermined parameter training sets $\Xi^{\text{train}} = \Xi^{\text{RB}} = \Xi^{\text{EQP}} = \{\mu_i \in \mathcal{D}\}_{i=1}^{N_{\text{train}}}$ of N_{train} parameter values drawn from a uniform random distribution.

8.1.2 | Rounding-error stable residual calculation

We first assess the rounding-error stable residual calculation introduced in Section 5. To this end, we set $N_{\text{train}} = 10$ and construct an RB of size $N = 10$. We then apply the rounding-error-stable and standard formulation to $\texttt{EQP}^{r2}(\Xi^{\text{train}}, \delta^r)$ to find the residual RQ rules. Output RQ rules are found using $\texttt{EQP}^{q}(\Xi^{\text{train}}, \delta^r/10)$ such that the output functional calculation error is an order of

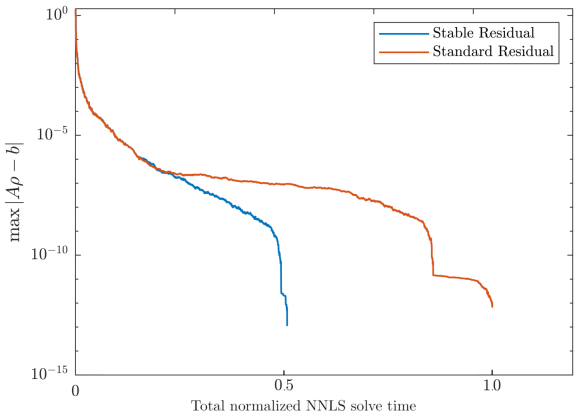


FIGURE 2 Comparison of the ℓ^∞ norm of the NNLS residual as a function of time for the rounding-error-stable and standard residual calculation methods for the FFD NACA case.

TABLE 1 Comparison of NNLS iterations and solve time for the standard and rounding-error stable residual calculation method for the FFD NACA case.

	outer loop iterations	inner loop iterations	normalized NNLS solve time
standard	14994	23247	1.0
rounding-error stable	7058	11693	0.51

magnitude smaller than the prescribed tolerance δ^r. To demonstrate the ability to achieve arbitrarily tight NNLS tolerances, we consider a (very) tight tolerance given by $\delta^r = 1 \times 10^{-10}$. While we recognize that this may be of a limited practical interest, the ability to achieve arbitrarily tight precision is (arguably) desired in any algorithm. All constrained minimization problems are solved using NNLS-CR.

Figure 2 compares the rounding-error-stable and standard NNLS residual calculation methods. Namely, we present the ℓ^∞ norm of the NNLS residual as a function of computation time for the final NNLS solve of the NNLS-CR method applied to $\texttt{EQP}^{r2}(\Xi^{\text{train}}, \delta^r)$. As discussed in Section 5, the stable residual calculation is activated after we encounter significant rounding error, which corresponds to the point where the two lines diverge. The rounding-error stable residual calculation method reduces the residual more rapidly after that point. Table 1 presents the number of iterations for the inner loop (lines 9–18 of Algorithm 1) and the outer loop (lines 2–20 of Algorithm 1) of the NNLS method, and the total NNLS solve time. The rounding-error residual calculation method reduces the number of inner and outer loop iterations, as well as the total solve time.

8.1.3 | Comparison of NNLS and NNLS-CR methods

We next compare the NNLS-CR method (Algorithm 2) to the standard NNLS method (Algorithm 1), in terms of the offline training time as well as the resulting RQ rules. In this study, we consider various values of $N_{\text{train}} = |\Xi^{\text{train}}|$. Both methods use POD on Ξ^{train} to construct an RB of size $N = 25$ and use $\texttt{EQP}^{r2}(\Xi^{\text{train}}, \delta_r)$ and $\texttt{EQP}^{q}(\Xi^{\text{train}}, \delta_r/10)$ to find the primal residual and output RQ rules, for a prescribed output error tolerance of 0.5%, i.e. $\delta^r = 0.0003$. Both methods are compared to the full quadrature RB solution (i.e. with the K^h FE quadrature points) for a random set of 40 test parameters Ξ^{test}_{40}.

Table 2 characterizes the constraint reduction and the ROMs obtained by employing these two methods. We see that the constraint reduction methods are able to reduce the number of constraints by up to almost 10 times (i.e., $\tilde{m} \approx m/10$), which results in a substantial reduction in the NNLS solve time. The size of the set of reduced constraints converges to ≈ 4000, from which we infer a key result: for the prescribed tolerance δ^r and RB size N, the entire continuous constraint manifold $\mathcal{C}$ is well represented by ≈ 4000 constraints.

Table 2 also shows that the output errors for the ROM constructed using NNLS-CR is smaller than those using NNLS, which indicates that our constraint tolerances $\delta_{\mathbf{Q}}$ for the NNLS-CR method are conservative compared to the tolerances for the original constraints. We also observe that we need a training set size of ≈ 50 to sufficiently sample our parameter space such that the

TABLE 2 Comparison of ROMs obtained by employing NNLS and NNLS-CR for the FFD NACA case. Online solve times are for a single core and are normalized by the single FOM evaluation time.

(a) NNLS

N_{train}	m	K^r	K^q	$\max_{\mu\in\Xi_{40}^{test}} \lvert\tilde{s}_N(\mu) - s_N(\mu)\rvert$	online solve time
25	16302	2122	17	2.21×10^{-4}	0.0074
30	19562	2024	18	2.57×10^{-3}	0.0084
35	22822	2270	18	2.84×10^{-4}	0.0043
40	26082	2067	17	9.38×10^{-4}	0.0090
45	29342	2252	19	4.09×10^{-4}	0.0057
50	32602	2157	19	3.33×10^{-4}	0.0075

(b) NNLS-CR

N_{train}	$\tilde{m}$	K^r	K^q	$\max_{\mu\in\Xi_{40}^{test}} \lvert\tilde{s}_N(\mu) - s_N(\mu)\rvert$	online solve time
25	3667	3647	17	2.12×10^{-4}	0.018
30	3423	3418	18	6.89×10^{-4}	0.017
35	3993	3971	18	7.45×10^{-5}	0.0097
40	3260	3253	17	4.50×10^{-4}	0.025
45	3667	3658	19	3.90×10^{-5}	0.016
50	4075	4065	19	4.59×10^{-5}	0.0091

TABLE 3 Comparison of hyperreduction time for the NNLS and NNLS-CR methods for the FFD NACA case, normalized by the total hyperreduction time for the NNLS method with $N_{\text{train}} = 50$. "Constraint reduction" refers to the row-wise QR factorization performed for constraint reduction.

(a) NNLS

N_{train}	constraint construction	NNLS solve	total
25	0.078	0.257	0.334
30	0.117	0.350	0.467
35	0.164	0.462	0.626
40	0.215	0.427	0.642
45	0.272	0.640	0.911
50	0.347	0.653	1.000

(b) NNLS-CR

N_{train}	constraint construction	constraint reduction	NNLS solve	total
25	0.078	0.0406	0.0576	0.176
30	0.115	0.0454	0.0554	0.216
35	0.163	0.0602	0.0689	0.293
40	0.217	0.0541	0.0284	0.299
45	0.279	0.0678	0.0473	0.394
50	0.344	0.0834	0.0729	0.501

desired output error is achieved for all test parameter values. (We recall that the prescribed error is an estimate and not an upper bound.) The online evaluation time for the ROMs based on NNLS-CR is greater than those based on NNLS, as the conservative choice of the $\delta_{\mathbf{Q}}$ tolerance results in RQ rules with nearly twice as many points.

Table 3 summarizes the training times. We normalize the times by the total offline hyperreduction time for the NNLS method applied to a training set of size $J = 50$. The NNLS-CR method is approximately twice as fast as the NNLS method. We further note that the constraint construction, which is identical for the NNLS and NNLS-CR methods, is the largest contributor to the hyperreduction time for NNLS-CR.

Remark 17. We were unable to obtain converged ROM solutions for few parameter values in $\mu \in \Xi_{40}^{\text{test}}$ in some cases. Both full quadrature solution and the RQ solution exhibited these numerical instabilities, and, for all parameter values for which the full-quadrature ROM converged, so did the RQ ROM. We thus conclude that these issues are not caused by the hyperreduction method, but rather by the Galerkin projection itself. Improving the stability of the projection is beyond the scope of this work and is not the focus of the present work, and hence we have excluded the results for unconverged cases from the error calculation.

8.2 | Reynolds-averaged Navier–Stokes flow over ONERA M6 wing

8.2.1 | Case description

We next consider turbulent flow over an ONERA M6 wing governed by the Reynolds-averaged Navier–Stokes (RANS) equation with the Spalart–Allmaras (SA) turbulence model [37]. The varied parameters are the freestream Mach number $M_\infty \in [0.3, 0.5]$ and angle of attack $\alpha \in [0°, 3°]$, which together define our parameter space $\mathcal{D} = [0.3, 0.5] \times [0°, 3°] \subset \mathbb{R}^2$. The Reynolds number is fixed to $Re = 10^6$. The quantity of interest is the drag coefficient, which, for the given flow conditions, lies approximately

TABLE 4 Comparison of ROMs obtained by employing $\texttt{EQP}^r$ and $\texttt{EQP}^{r2}$ for the ONERA case. For all tolerances smaller than 10^{-6}, the $\texttt{EQP}^r$ results are similar. Online solve times are normalized by the single FOM evaluation time.

(a) $\texttt{EQP}^r$

δ^r	K^r	K^q	$\max_{\mu\in\Xi_9^{test}} \lvert\tilde{s}_N(\mu) - s_N(\mu)\rvert$	online solve time
10^{-4}	67	7	9.28×10^{-6}	1.19×10^{-4}
10^{-5}	67	8	8.78×10^{-6}	1.17×10^{-4}
$< 10^{-6}$	67	9	2.06×10^{-6}	$\sim 1.2 \times 10^{-4}$

(b) $\texttt{EQP}^{r2}$

δ^r	K^r	K^q	$\max_{\mu\in\Xi_9^{test}} \lvert\tilde{s}_N(\mu) - s_N(\mu)\rvert$	online solve time
10^{-4}	460	7	9.22×10^{-6}	1.54×10^{-4}
10^{-5}	541	8	1.19×10^{-6}	1.61×10^{-4}
10^{-6}	622	9	2.66×10^{-7}	1.86×10^{-4}
10^{-7}	703	9	6.31×10^{-8}	1.87×10^{-4}
10^{-8}	784	9	2.11×10^{-9}	1.90×10^{-4}
10^{-9}	804	9	1.44×10^{-10}	1.69×10^{-4}

TABLE 5 Comparison of time to find primal residual RQ rule for the ONERA case, normalized by time for $\texttt{EQP}^r$ trial with $\delta^r = 10^{-9}$. For all tolerances, the $\texttt{EQP}^r$ incurred a similar hyperreduction time. "Constraint reduction" refers to the row-wise QR factorization performed for constraint reduction.

(a) $\texttt{EQP}^r$

δ^r	constraint construction	NNLS solve	total
all	~ 0.97	~ 0.031	~ 1.0

(b) $\texttt{EQP}^{r2}$

δ^r	constraint construction	constraint reduction	NNLS solve	total
10^{-4}	4.58	3.04	0.84	9.30
10^{-5}	4.57	4.04	1.57	11.0
10^{-6}	4.57	5.30	2.15	12.8
10^{-7}	4.56	6.76	2.59	14.7
10^{-8}	4.57	8.56	3.94	17.9
10^{-9}	4.62	9.03	5.37	19.9

within the range $C_D \in [0.02, 0.03]$. The "truth" FE solution is obtained using an adaptive high-order DG methods. A $p = 2$ mesh is generated through anisotropic adaptive mesh refinement such that the DWR error estimate for all training parameter values is less than 0.5%; i.e., $\eta_h(\mu) \leq 0.0001$, $\forall \mu \in \Xi^{\text{train}}$. The resulting mesh has $N_h = 864{,}720$ degrees of freedom with $K^h = 4{,}881{,}708$ quadrature points. This case involves a three-dimensional spatial domain and tests the ability of EQP to handle a large FE space size N_h and quadrature set size K^h. We also demonstrate that the second-order constraints discussed in Section 3.3 are necessary to achieve tight output error tolerances. For this problem, we use 200 cores for all calculations.

8.2.2 | Second-order constraints

We first illustrate the need for second-order constraints introduced in Section 3.3 to achieve tight error tolerances in large problems. To this end, we first generate $N = 9$ RB associated with a set Ξ_9^{train} of 3×3 parameter points equidistributed over the two-dimensional parameter space $\mathcal{D}$. We then find the RQ rules using the first-order $\texttt{EQP}^r(\Xi_9^{\text{train}}, \delta^r)$ and the second-order $\texttt{EQP}^{r2}(\Xi_9^{\text{train}}, \delta^r)$. $\texttt{EQP}^r$ is solved using the standard NNLS method as the small number of constraints ($m = 83$) does not require constraint reduction; $\texttt{EQP}^{r2}$, which involves many more constraints ($m = 812$), is solved by the NNLS-CR method. The output functional is reduced using $\texttt{EQP}^q(\Xi_9^{\text{train}}, \delta^r/10)$, such that the output functional evaluation error is an order of magnitude smaller than the prescribed output error tolerance δ^r. For this case, we set our test parameter set to be equal to the training set (i.e. $\Xi_9^{\text{test}} = \Xi_9^{\text{train}}$) to show that, even in the reproductive case, the second-order constraints are required to achieve very small tolerances δ^r.

Tables 4 and 5 summarize the ROMs and offline training time for $\texttt{EQP}^r(\Xi_9^{\text{train}}, \delta^r)$ and $\texttt{EQP}^{r2}(\Xi_9^{\text{train}}, \delta^r)$ for output error tolerances ranging from 0.5% ($\delta^r = 10^{-4}$) to 5×10^{-6}% ($\delta^r = 10^{-9}$). We also plot the output error in Figure 3. We first observe that the first-order method is unable to achieve a tolerance tighter than 2×10^{-6}, while the second-order method achieves the desired tolerances for each test. We conclude that the second-order constraints is needed to achieve very tight tolerances. We next observe that, due to an increase in the number of constraints from $m = 83$ to $m = 812$, both the offline training time and the RQ rule size are increased by about 10–20 times and 10 times, respectively, when the second-order constraints are used. We hence recommend using second-order constraints only when a tight tolerance necessitates it. In practice, as the solution of $\texttt{EQP}^r(\Xi_9^{\text{train}}, \delta^r)$ is relatively fast, we may first solve $\texttt{EQP}^r(\Xi_9^{\text{train}}, \delta^r)$, check whether the target tolerances are met, and solve $\texttt{EQP}^{r2}(\Xi_9^{\text{train}}, \delta^r)$ only if we fail to meet the target. In this study, we set the second-order tolerance δ^J equal to δ^r as discussed in Remark 2; however, the choice may be conservative for larger tolerances, leading to a large RQ rule size.

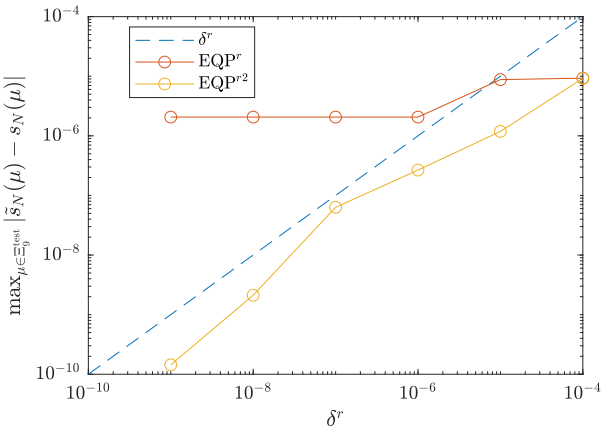


FIGURE 3 Comparison of the output error for the EQP^r and EQP^{r2} methods applied to the ONERA case.

TABLE 6 Comparison of ROMs obtained by employing NNLS and NNLS-CR for the ONERA case. Online solve times are for 200 cores and are normalized by the single FOM evaluation time.

	RQ rule size		maximum output errors for $\mu \in \Xi_{20}^{\text{test}}$			online time
	K^r	K^q	$\lvert\tilde{s}_N - s_N\rvert$	$\lvert s_N - s_h\rvert$	$\lvert\tilde{s}_N - s_h\rvert$	
NNLS	660	10	4.65×10^{-5}	1.23×10^{-4}	7.93×10^{-5}	2.07×10^{-4}
NNLS-CR	1179	10	4.53×10^{-5}	1.23×10^{-4}	8.10×10^{-5}	2.26×10^{-4}

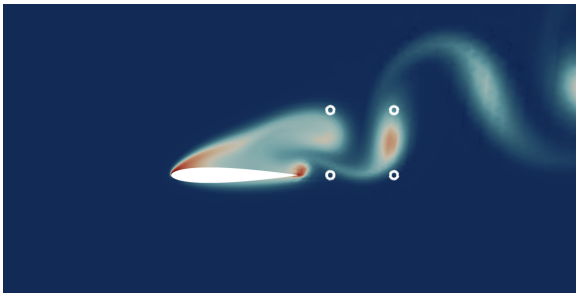
TABLE 7 Comparison of constraint reduction and hyperreduction time for the NNLS and NNLS-CR methods applied to the primal residual RQ rule for the ONERA case. All times are normalized by the single FOM evaluation time. “Constraint reduction" refers to the row-wise QR factorization performed for constraint reduction.

	m or $\tilde{m}$	constraint construction	constraint reduction	NNLS solve	total
NNLS	2498	5.99	—	6.28	12.3
NNLS-CR	1179	5.99	5.19	3.51	14.9

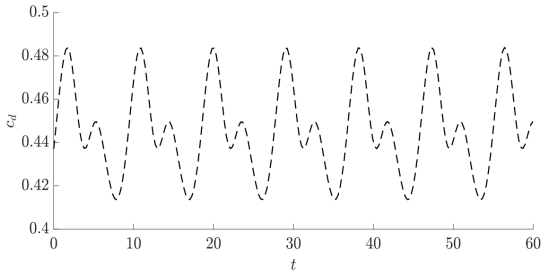
8.2.3 | Comparison of NNLS and NNLS-CR

We now compare the application of the NNLS and NNLS-CR methods to determine the RQ rules for the ONERA wing case. We apply POD on Ξ_{16}^{train} of 4×4 equispaced training points over $\mathcal{D}$ to find an RB of size $N = 12$; this value was picked such that we achieve a maximum RB error $\max_{\mu\in\Xi_{20}^{\text{test}}} |s_N(\mu) - s_h(\mu)| \approx 0.0001$. We then construct the primal residual RQ rule by solving EQP$^{r2}(\Xi_{16}^{\text{train}}, \delta^r)$ using NNLS and NNLS-CR for a prescribed output error tolerance of $\delta^r = 0.0001$ (0.5% error). In both cases, the output functional RQ rule is constructed using EQP$^q(\Xi_{16}^{\text{train}}, \delta^r/10)$ using NNLS; the constraint reduction is unnecessary for output functional since the number of constraints is only 16. We then assess the maximum output error $\max_{\mu\in\Xi_{20}^{\text{test}}} |\tilde{s}_N(\mu) - s_N(\mu)|$ for a set of 20 random test parameters Ξ_{20}^{test}. We employ the rounding-error stable residual calculation method for all NNLS and NNLS-CR.

Tables 6 and 7 summarize the ROMs constructed and the associated training times for NNLS and NNLS-CR. The NNLS and NNLS-CR methods both obtain RQ rules that achieve the desired output error tolerance for all test parameter values. Similarly to the previous case, NNLS-CR yields a larger RQ rule than NNLS; in this case, K^r is twice as large as for NNLS-CR. As for the hyperreduction time, we observe that the NNLS solve time itself is reduced by a factor of ~ 2 through constraint reduction; however, the constraint reduction requires an additional QR factorization, thus the total time is slightly higher for NNLS-CR. This is not discouraging for a few reasons: first, the hyperreduction times are similar, so NNLS-CR can be applied without a substantial increase in time; second, this case represents a small parameter space and thus a small constraint space, which is not amenable to much reduction in the number of constraints. For a problem with a “larger” parameter space (i.e., in higher dimension and/or with larger parametric ranges), we observe that NNLS-CR is more efficient, as we will see shortly in the subsequent examples.



(a) "true" solution field for a random initial state



(b) "true" drag coefficient history

FIGURE 4 Multi-fidelity ensemble Kalman filter study [16].

8.3 | Multi-fidelity ensemble Kalman filter

8.3.1 | Case description

As our third example, we consider data assimilation by a multi-fidelity ensemble Kalman filter (MF-EnKF) [16] for a NACA0012 airfoil under unsteady compressible Navier-Stokes flow. This multi-fidelity data assimilation method combines a FOM, a ROM, and observation data from probes to provide rapid and reliable state estimation. We refer to [36] for the RB–RQ formulation for unsteady problems and to [16] for the MF-EnKF formulation; we here provide only a brief case description that is particularly relevant to the present comparison of the NNLS and NNLS-CR techniques. Specifically, we note that MF-EnKF requires the solution of EQP problem for an ensemble of sets of snapshots associated with unsteady flow solutions. The flow condition is given by the chord-based Reynolds number of $Re_c = 700$, freestream Mach number $M_\infty = 0.2$, and angle of attack $\alpha = 20°$, which yields a separated, unsteady flow. The output is the drag coefficient which, for the given flow conditions, lies approximately within the range $c_d \in [0.4, 0.5]$. We thus consider error tolerances in our output of 0.002 for 0.5% error. Figure 4 shows the separated flow about the NACA airfoil, the location of the velocity probes and the periodic nature of the output.

The "truth" FE solution uses an adaptive $p = 2$ DG method for spatial discretization and a third-order diagonally-implicit Runge-Kutta (DIRK) formulation [1] for the time discretization. The POD and EQP training snapshots are associated with various trajectories, each starting with a random initial condition drawn from the periodic "truth" solution for this problem and stepped forward over N_t time steps. This unsteady case demonstrates the use of the constraint reduction methods for an unsteady problem with a large "parameter" training set (i.e., time steps and ensemble trajectories). We use 160 cores for all computations.

8.3.2 | Single trajectory

We first compare NNLS and NNLS-CR with a single unsteady solution trajectory as the training set (i.e., without an ensemble of trajectories). We also construct an unsteady DWR error estimator for the time-averaged drag. The setup for this case is replicated from [36], but we use a prescribed RB size N and training snapshots Ξ^{train}, instead of the greedy algorithm in [36]. We use a time step of $\delta t = 0.25$ over the time interval of $[0, 12]$ to generate $\Xi^{\text{train}}_{N_t} = \{t_i\}_{i=1}^{N_t}$ and solution snapshots $\{u_h(t_i)\}_{i=1}^{N_t}$ for $N_t = 49$. We use adaptive mesh refinement to drive the FOM error estimate to 0.5%; i.e., $\eta_h \leq 0.002$. The resulting FE approximation space has $N_h = 51{,}912$ degrees of freedom and $K^h = 105{,}198$ quadrature points. We then apply POD to the snapshots to obtain an RB of size $N = 22$; the RB size is chosen such that the RB error for is less than 0.5%, i.e., $\left|s_N(t_0) - s_h(t_0)\right|_{t_0 \in \Xi^{\text{test}}} \leq 0.002$. We then construct RQ for the primal residual, output functional, and DWR using $\texttt{EQP}^r(\Xi^{\text{train}}_{N_t}, \delta^r)$, $\texttt{EQP}^q(\Xi^{\text{train}}_{N_t}, \delta^r/10)$, and $\texttt{EQP}^\eta(\Xi^{\text{train}}_{N_t}, \delta^r)$, respectively, for a prescribed output error tolerance of 0.5%, i.e. $\delta^r = 0.002$.

Tables 8 and Table 9 summarize the resulting ROMs and the training times, respectively. We first note that both ROMs yield output errors that are within the prescribed tolerance for both NNLS and NNLS-CR, and the error estimates bound the true errors. The NNLS-CR result exhibits a "lucky cancellation" of the RB error $|s_N - s_h|$ and the RQ error $|\tilde{s}_N - s_N|$, resulting in a total RB-RQ error $|\tilde{s}_N - s_h|$ that is lower than either of these error sources individually; this lucky cancellation contributes to the relatively conservative error estimate. We next note that the RQ rules associated with NNLS-CR are approximately twice the size of those associated with NNLS due to the conservative estimate of the transformed tolerance $\delta_{\mathbf{Q}}$; however, there is only $\approx 25\%$ increase in the online evaluation time, due to the fixed parallel overhead associated with using 160 cores on a small

TABLE 8 Comparison of ROMs obtained by NNLS and NNLS-CR for the single-trajectory unsteady flow case. Online solve times are normalized by the single FOM evaluation time.

	RQ rule size			output errors				online
	K^r	K^q	K^η	$\lvert\tilde{s}_N - s_N\rvert$	$\lvert s_N - s_h\rvert$	$\lvert\tilde{s}_N - s_h\rvert$	$\tilde{\eta}_N$	time
NNLS	361	9	464	3.80×10^{-4}	1.65×10^{-3}	1.27×10^{-3}	5.63×10^{-3}	0.0221
NNLS-CR	838	13	836	8.52×10^{-4}	1.65×10^{-3}	7.98×10^{-4}	4.08×10^{-3}	0.0276

TABLE 9 Comparison of constraint set size and hyperreduction time for the NNLS and NNLS-CR methods applied to the single-trajectory unsteady flow case. All times are normalized by the single FOM evaluation time. "Constraint reduction" refers to the row-wise QR factorization performed for constraint reduction.

(a) primal residual RQ rule

	m or $\tilde{m}$	constraint construction	constraint reduction	NNLS solve	total
NNLS	4226	0.121	—	1.31	1.43
NNLS-CR	839	0.130	0.744	0.627	1.53

(b) output functional RQ rule

	m or $\tilde{m}$	constraint construction	constraint reduction	NNLS solve	total
NNLS	192	0.0078	—	0.0011	0.0089
NNLS-CR	13	0.0076	0.00033	0.0013	0.0092

(c) DWR RQ rule

	m or $\tilde{m}$	constraint construction	constraint reduction	NNLS solve	total
NNLS	8450	0.465	—	2.57	3.04
NNLS-CR	839	0.462	1.44	0.392	2.33

ROM. We next observe in Table 9 that the offline training time is slightly smaller for NNLS-CR. For NNLS-CR, the constraint reduction time dominates the hyperreduction time. The total hyperreduction time is not reduced by much when NNLS-CR is used for this case; due to the increased quadrature rules sizes K^r, K^q, and K^η, the NNLS method may be preferred.

8.3.3 | Multiple trajectories

We now construct ROMs from an ensemble of four solution trajectories for MF-EnKF. We define a parameter set $\Xi^{\text{train}}_{4N_t} = \{\{t_i^j\}_{i=1}^{N_t}\}_{j=1}^4$ and solution snapshots $\{\{u_h(t_i^j)\}_{i=1}^{N_t}\}_{j=1}^4$, where N_t varies with the time step size $\Delta t \in [0.15, 0.4]$. The adaptive DG method yields an approximation space with $N_h = 55{,}296$ degrees of freedom and $K^h = 108{,}050$ quadrature points to meet the output error tolerance of $\eta_{l_h} \leq 0.002$. We apply POD to the snapshots to construct an RB of size $N = 22$, such that the RB error is controlled to within 0.5%. We then invoke $\texttt{EQP}^r_{\text{glob}}(\Xi^{\text{train}}_{4N_t}, \delta_r)$ and $\texttt{EQP}^q(\Xi^{\text{train}}_{4N_t}, \delta_r/10)$ to find the primal residual and output function RQ rules, for a prescribed error tolerance of 0.5%, i.e. $\delta_r = 0.002$. The $\texttt{EQP}^r_{\text{glob}}$ [16] is a version of EQP that uses the entropy variables and different forms of the manifold accuracy constraints (21) to control the global solution error. For this case, we modify the NNLS-CR procedure (Algorithm 2) in the following manner: instead of starting with $\tilde{m} = 0.05m$ and increasing $\tilde{m}$ by $0.05m$ each iteration, we start with $\tilde{m} = 600$ and increase $\tilde{m}$ by 100 each iteration. This is simply to ensure that the NNLS-CR method tries the same values of $\tilde{m}$ for all cases with different Δt and thus different constraint set sizes m.

Table 10 summarizes the constraint reduction details for various time-step sizes Δt. We focus on the construction of the primal residual RQ rule, since the construction of output functional RQ rules are a few orders of magnitude faster. We observe that for all the time-step sizes the NNLS-CR method is able to determine a set of approximately $\tilde{m} \approx 700$ reduced constraints that represent the original $m = 10{,}000$–$30{,}000$ constraints. For the largest step size of $\Delta t = 0.40$, we see the smallest number of reduced constraints $\tilde{m}$, which suggests that Δt may be too large to sufficiently sample the parametric constraint manifold $\mathcal{C}$. As the step size is reduced, the number of reduced constraints converges to 700. This convergence suggests that we have sufficiently sampled our constraint space and that $\mathcal{C}$ is amenable to approximation by a set of ≈ 700 constraints. As discussed in Remark 12, the true minimum value of required $\tilde{m}$ is likely slightly lower than this. We finally note that the hyperreduction time is dominated by the QR factorization used in constraint reduction, which scales with the number of original constraints.

Remark 18. The NNLS method is too computationally expensive to use in the multi-trajectory case without constraint reduction of some kind. Results in [16] use a heuristic constraint reduction algorithm, that generates a reduced set of constraints but does

TABLE 10 Comparison of size of reduced set of constraints, RQ rule size and hyperreduction time for various time-step values for the EnKF case. All times are normalized by the single FOM evaluation time for $\Delta t = 0.25$. "Constraint reduction" refers to the row-wise QR factorization performed for constraint reduction.

				hyperreduction time			
Δt	m	$\tilde{m}$	K^r	constraint construction	constraint reduction	NNLS solve	Total
0.40	10082	600	589	0.344	0.863	0.0970	1.30
0.30	14082	800	778	0.280	1.60	0.344	2.22
0.25	16898	800	785	0.342	1.92	0.334	2.60
0.20	21122	700	685	0.416	2.07	0.208	2.70
0.15	29442	700	690	0.583	2.86	0.217	3.66

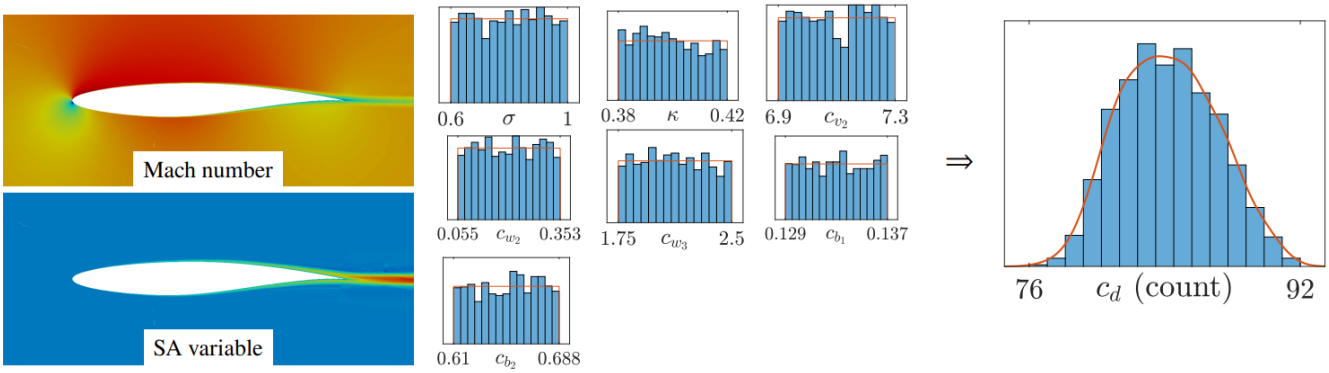


FIGURE 5 Illustration of the RANS UQ problem for the RAE2822 airfoil [19].

not ensure that all original constraints are satisfied. In comparison, the proposed NNLS-CR method ensures that all original constraints are satisfied, and is able to find a smaller set of reduced constraints that represents $\mathcal{C}$ for a given tolerance.

8.4 | UQ of RANS flow

8.4.1 | Case description

As the final case, we consider UQ of the RANS flow over an RAE2822 airfoil. Specifically, we wish to quantify the impact of the uncertainties in the SA turbulence model [37] on the drag. The SA turbulence model involves many coefficients whose values are empirically calibrated, leading to uncertainty in the output. We wish to rapidly and reliably quantify this uncertainty using a ROM to calculate the drag value for many different realizations of the empirical parameters. Following [34], we treat seven empirical parameters of the SA model as independent uniform random variables: $\sigma \in [0.6, 1.0]$, $\kappa \in [0.38, 0.42]$, $c_{v_1} \in [6.9, 7.3]$, $c_{w_2} \in [0.055, 0.3525]$, $c_{w_3} \in [1.75, 2.5]$, $c_{b_1} \in [0.12893, 0.137]$, and $c_{b_2} \in [0.60983, 0.6875]$. The flow condition is fixed with the freestream Mach number $M_\infty = 0.3$, the angle of attack $\alpha = 2°$, and the Reynolds number $Re_c = 6.5 \times 10^6$. We hence have a seven-dimensional parameter domain $\mathcal{D}$. For these parameter values and flow conditions, our output of interest lies in the range $c_d \in [0.0076, 0.0092]$; we hence set the error tolerances to 9×10^{-5} for 1% error or 4.5×10^{-5} for 0.5% error. Figure 5 illustrates the uncertainty propagation process.

In order to efficiently sample this high-dimensional parameter space such that we can construct a ROM that exhibits δ^{ROM}-robustness $\forall \mu \in \mathcal{D}$, we employ the greedy training procedures outlined in Section 7. Specifically, we use Algorithms 4 and 5 to adaptively determine our training sets Ξ^{RB} and Ξ^{EQP}, while simultaneously constructing our ROM. We select the following algorithmic parameters. We set the training and test set sizes to $J = 300$ and $K = 1000$. We set the FOM tolerance of $\delta^{\text{FOM}} = 4.5 \times 10^{-5}$, so that each adaptive $\mathbb{P}^2$ DG snapshot yields the DWR error estimate of less than 0.5%; i.e., $\eta_h(\mu) \leq 4.5 \times 10^{-5}$, $\forall \mu \in \Xi^{\text{RB}}$; at the end of the greedy algorithm, this yields in an adapted mesh with $N_h = 41{,}490$ degrees of freedom and $K^h = 67{,}848$ quadrature points. We similarly set the ROM tolerance to $\delta^{\text{ROM}} = 4.5 \times 10^{-5}$: i.e., 0.5% error. We finally set the EQP tolerances to $\delta^r = 2.25 \times 10^{-5}$, $\delta^q = 2.25 \times 10^{-6}$ and $\delta^\eta = 2.25 \times 10^{-5}$, so that EQP tolerances are small relative to the

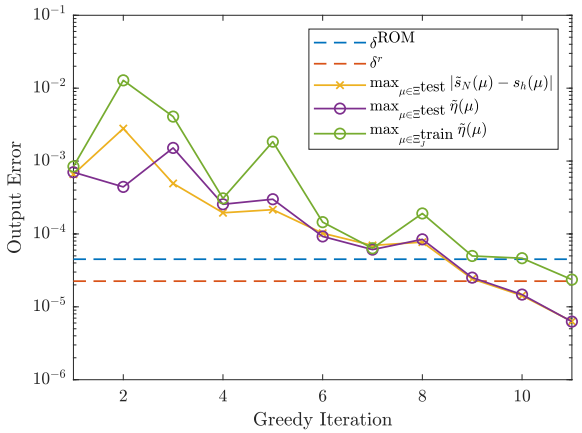


FIGURE 6 Output errors and error estimates for the RAE2822 RANS UQ case.

TABLE 11 Constraint reduction and ROM details for each greedy iteration for the RAE2822 RANS UQ case.

			primal residual			output functional			DWR		
iteration	N	$\lvert\Xi^{\text{EQP}}\rvert$	m	$\tilde{m}$	K^r	m	$\tilde{m}$	K^q	m	$\tilde{m}$	K^η
1	2	6	28	14	13	6	2	1	26	18	17
2	4	32	206	71	67	32	4	3	258	169	168
3	6	48	392	122	119	48	6	5	578	309	309
4	8	104	1050	224	223	104	10	9	1666	664	664
5	10	55	666	237	237	56	7	6	1102	495	494
6	12	56	790	308	307	56	7	6	1346	469	469
7	14	57	918	353	352	57	9	8	1598	712	711
8	16	58	1050	418	417	58	9	8	1858	829	829
9	18	59	1186	490	489	59	11	10	2126	848	846
10	20	60	1326	567	566	60	11	10	2402	1080	1075
11	22	61	1470	630	629	61	12	12	2686	804	804

FOM and ROM errors. In each iteration, we collect two RB functions, one for RANS mean-flow equation and the other for the SA equation; hence, the RB size N is twice the number of greedy iterations. We use 40 cores for all computations.

8.4.2 | Results

We now present the results of applying the greedy procedures to the seven-dimensional RANS UQ problem for the RAE2822 airfoil. Figure 6 shows the convergence of the output error with greedy iterations. The test set Ξ^{test} contains 10 randomly chosen parameter values over $\mathcal{D}$. The greedy algorithm obtains a ROM that meets the target output error tolerance δ^{ROM} on the 11th iteration. Throughout the iterations, the DWR error estimate $\tilde{\eta}(\mu)$ is quite accurate, even on the test set Ξ^{test} which is different from the training set; i.e., the error estimate is accurate in predictive setting.

Table 11 shows details of constraint reduction and ROM in each greedy iteration. We observe that the size of the EQP training set constructed by the adaptive EQP (Algorithm 5) saturates to ~ 60, and the constraint reduction method consistently reduce the number of constraints by another factor of 2 to 3. At the last iteration, the primal residual EQP have $|\Xi^{\text{train}}| = 61$ training points, which yields $m = 1470$ constraints, which is reduced to $\tilde{m} = 630$ constraints. However, due to the saturation check performed by the algorithm, this reduced set of 630 constraints approximately represent all of the approximately 7,200 constraints that would be constructed for all parameter values in Ξ_J^{train} and in $\mathcal{C}$. Thus, the combination of the *a priori* reduction by the adaptive Ξ^{EQP} selection and the *a posteriori* reduction by the constraint reduction yields significant reduction in the effective number of constraints.

Table 12 shows the timing details of the greedy algorithm. The error estimate sampling for the last two iterations is much more computationally expensive than for previous iterations, since the "termination check" using the richer parameter set Ξ_K^{test} with $K = 1000$ parameter values is used check δ^{ROM}-robustness (lines 9–13 of Algorithm 4). Table 13 summarizes the computation

TABLE 12 Breakdown of computation time for each iteration of the greedy algorithm as applied to the RAE2822 RANS UQ case. Online solve times are normalized by the single FOM evaluation time *without error estimate*.

iteration	err. est. sampling	FOM solve	EQP primal residual	EQP output functional	DWR	total
1	–	6.74	0.187	0.0190	0.13	7.30
2	0.432	1.38	0.703	0.0297	0.76	3.89
3	0.916	1.36	1.02	0.0360	1.80	5.67
4	0.605	1.41	2.83	0.0611	7.40	13.5
5	0.783	1.43	1.90	0.0420	3.93	8.76
6	1.03	1.92	2.28	0.0427	4.08	9.96
7	1.16	1.52	3.18	0.0455	7.25	13.8
8	1.26	1.52	3.46	0.0466	9.48	16.5
9	1.28	1.39	5.25	0.0510	10.5	19.2
10	3.62	1.61	5.71	0.0499	16.9	26.6
11	4.72	1.75	7.10	0.0538	11.3	28.0
12	7.57	–	–	–	–	–
Total	23.4	22.0	33.6	0.477	73.5	153

TABLE 13 Summary of computation time for the entire greedy algorithm as a percentage of the total greedy training time for the RAE2822 RANS UQ case.

err. est. sampling	FOM solves	constraint construction	constraint reduction	NNLS solves	others
14.5%	13.7%	19.5%	26.7%	19.4%	6.18%

TABLE 14 Summary of the ROM constructed for the RAE2822 RANS UQ case. Online solve times are normalized by the single FOM evaluation time *without error estimate*.

	RQ rule size			maximum output errors for $\mu \in \Xi^{\text{test}}$				online time
N	K^r	K^q	K^η	$\lvert\bar{s}_N - s_N\rvert$	$\lvert s_N - s_h\rvert$	$\lvert\bar{s}_N - s_h\rvert$	η	
22	629	12	804	6.20×10^{-7}	5.76×10^{-6}	6.30×10^{-6}	6.26×10^{-6}	0.0070

time for the entire greedy algorithm by the key operations. The QR factorization performed for constraint reduction is the largest contributor to the total greedy algorithm time, followed by the constraint construction and NNLS solve operations.

Finally, Table 14 shows details for the ROM obtained at the end of the greedy algorithm. By the design of the greedy algorithm, we hope that the ROM exhibits δ^{ROM}-robustness for all of $\mathcal{D}$ for $\delta^{\text{ROM}} = 4.5 \times 10^{-5}$ (i.e., 0.5%); indeed, the ROM achieves output errors of less than 0.5% for all parameters in the test set Ξ^{test}. The DWR error estimate is also effectively estimates the error in the RB–RQ approximation. The online computational speedup is approximately 150×.

9 | SUMMARY

In this work, we proposed improvements to EQP for cases that (i) demand tight hyperreduction tolerances, (ii) involve a large number of residual-matching conditions, and (iii) involve a high-dimensional parameter space. To address (i), we developed constraints for the EQP that provides second-order error control (Section 3.3) and a rounding-error stable NNLS (Section 5). To address (ii), we developed NNLS-CR, which incorporates a constraint reduction strategy that identifies a reduced set of orthogonal constraints whose solution satisfies all of the original constraints (Section 6). To address (iii), we developed a greedy algorithm that incorporates an adaptive EQP (Section 7.2) and NNLS-CRi which enables efficient solution of the EQP problem (Section 7.1). We then demonstrated the algorithms developed using four different classes of numerical examples (Section 8). We also emphasize that, while we demonstrated the algorithmic improvements using EQP, all modifications to the NNLS are equally applicable to other hyperreduction methods based on direct integration (cf. Remarks 4 and 14).

While numerical examples show that the techniques developed improves the performance of EQP hyperreduction, there are still a number of limitations that warrant further studies. For instance, as discussed and observed, NNLS-CR can yield constraints that are too conservative; it may be possible to improve the form of constraints $\{\mathbf{Q}, \mathbf{b}_\mathbf{Q}, \delta_\mathbf{Q}\}$ as well as the method to determine

$\tilde{m}$ to mitigate this conservativeness. Similarly, current constraints for second-order error control in $\texttt{EQP}^{r2}$ do not exploit the quantitative connection between δ^J and the error. Finally, while it was not a focus of the present work, some of the ROMs constructed in Sections 8.1 suffered from numerical instabilities (cf. Remark 17); the development of more stable ROMs, and the associated hyperreduction methods, remains an important topic.

ACKNOWLEDGMENTS

The financial support for this work was provided by the Natural Sciences and Engineering Research Council of Canada. Some of the 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.

REFERENCES

1. R. Alexander, *Diagonally implicit Runge-Kutta methods for stiff O.D.E.'s*, SIAM Journal on Numerical Analysis **14** (1977), no. 6, 1006–1021.
2. S. S. An, T. Kim, and D. L. James, *Optimizing cubature for efficient integration of subspace deformations*, ACM Trans. Graph. **27** (2008), no. 5, 165:1–165:10. URL http://doi.acm.org/10.1145/1409060.1409118.
3. 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** (2004), 667—672.
4. R. Becker and R. Rannacher, *An optimal control approach to a posteriori error estimation in finite element methods*, Acta Numerica **10** (2001), 1–102.
5. P. Benner, S. Gugercin, and K. Willcox, *A survey of projection-based model reduction methods for parametric dynamical systems*, SIAM Review **57** (2015), no. 4, 483–531.
6. G. Bombara, M. Cococcioni, B. Lazzerini, and F. Marcelloni, *S-nnls: An efficient non-negative least squares algorithm for sequential data*, International Journal for Numerical Methods in Biomedical Engineering **27** (2011), 770 – 773.
7. R. Bro and S. Jong, *A fast non-negativity-constrained least squares algorithm*, Journal of Chemometrics **11** (1997), 393–401.
8. S. L. Brunton, J. L. Proctor, and J. N. Kutz, *Discovering governing equations from data by sparse identification of nonlinear dynamical systems*, Proceedings of the National Academy of Sciences (2016).
9. T. Bui-Thanh, M. Damodaran, and K. Willcox, *Proper orthogonal decomposition extensions for parametric applications in compressible aerodynamics*, *21st AIAA Applied Aerodynamics Conference*, 2003-4213, American Institute of Aeronautics and Astronautics, 2003, .
10. T. Bui-Thanh, K. Willcox, and O. Ghattas, *Model reduction for large-scale systems with high-dimensional parametric input space*, SIAM Journal on Scientific Computing **30** (2008), no. 6, 3270–3288. URL https://doi.org/10.1137/070694855.
11. 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** (2011), no. 2, 155–181.
12. 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** (2017), no. 12, 1623–1654.
13. S. Chaturantabut and D. C. Sorensen, *Nonlinear model reduction via Discrete Empirical Interpolation*, SIAM Journal on Scientific Computing **32** (2010), no. 5, 2737–2764.
14. P. Chen and O. Ghattas, *Hessian-based sampling for high-dimensional model reduction*, International Journal for Uncertainty Quantification **9** (2019), no. 2, 103–121.
15. 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** (2015), 1105–1109.
16. G. Donoghue and M. Yano, *A multi-fidelity ensemble kalman filter with hyperreduced reduced-order models*, Computer Methods in Applied Mechanics and Engineering **398** (2022), 115282. URL https://www.sciencedirect.com/science/article/pii/S0045782522004005.
17. Z. Drmac and S. Gugercin, *A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions*, SIAM Journal on Scientific Computing **38** (2016).
18. M. Drohmann, B. Haasdonk, and M. Ohlberger, *Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation*, SIAM Journal on Scientific Computing **34** (2012), no. 2, 33.
19. E. Du, M. Sleeman, and M. Yano, *Adaptive discontinuous-Galerkin reduced-basis reduced-quadrature method for many-query CFD problems*, *AIAA AVIATION 2021 FORUM*, American Institute of Aeronautics and Astronautics, 2021, .
20. E. Du and M. Yano, *Efficient hyperreduction of high-order discontinuous Galerkin methods: Element-wise and point-wise reduced quadrature formulations*, Journal of Computational Physics **466** (2022), 111399.
21. R. Everson and L. Sirovich, *Karhunen-Loève procedure for gappy data*, Journal of the Optical Society of America A, Optics and Image Science **12** (1995), no. 8, 1657–1664.
22. 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** (2015), no. 5, 1077–1110. URL http://dx.doi.org/10.1002/nme.4820, nme.4820.
23. 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** (2014), no. 9, 625–662.
24. S. Foucart and H. Rauhut, *A mathematical introduction to compressive sensing*, Birkhäuser Basel, 2013, .
25. L. Grasedyck, D. Kressner, and C. Tobler, *A literature survey of low-rank tensor approximation techniques*, GAMM-Mitteilungen **36** (2013), no. 1, 53–78. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/gamm.201310004.

26. M. A. Grepl, Y. Maday, N. C. Nguyen, and A. T. Patera, *Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations*, ESAIM: M2AN **41** (2007), no. 3, 575–605. URL https://doi.org/10.1051/m2an:2007031.
27. J. A. Hernández, M. A. Caicedo, and A. Ferrer, *Dimensional hyper-reduction of nonlinear finite element models via empirical cubature*, Comput. Methods Appl. Mesh. Eng. **313** (2017), 687–722.
28. J. S. Hesthaven, G. Rozza, and B. Stamm, *Certified reduced basis methods for parametrized partial differential equations*, Springer, 2016, .
29. J. S. Hesthaven, B. Stamm, and S. Zhang, *Efficient greedy algorithms for high-dimensional parameter spaces with applications to empirical interpolation and reduced basis methods*, ESAIM: M2AN **48** (2014), no. 1, 259–283. URL https://doi.org/10.1051/m2an/2013100.
30. C. L. Lawson and R. J. Hanson, *Solving least squares problems*, vol. 18, 518–520 , Society for Industrial & Applied Mathematics (SIAM), 1976, .
31. A. Nouy, *Low-rank methods for high-dimensional approximation and model order reduction*, chap. 4, 2017. 171–226.
32. A. T. Patera and M. Yano, *An LP empirical quadrature procedure for parametrized functions*, Comptes Rendus Mathematique **355** (2017), no. 11, 1161–1167.
33. 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** (2008), no. 3, 229–275.
34. J. Schaefer, A. Cary, M. Mani, and P. Spalart, *Uncertainty quantification and sensitivity analysis of SA turbulence model coefficients in two and three dimensions*, AIAA 2017-1710, AIAA, 2017, .
35. T. W. Sederberg and S. R. Parry, *Free-form deformation of solid geometric models*, *Proceedings of the 13th annual conference on Computer graphics and interactive techniques - SIGGRAPH 86*, ACM Press, 1986, .
36. M. K. Sleeman and M. Yano, *Goal-oriented model reduction for parametrized time-dependent nonlinear partial differential equations*, Computer Methods in Applied Mechanics and Engineering **388** (2022), 114206.
37. P. R. Spalart and S. R. Allmaras, *A one-equation turbulence model for aerodynamics flows*, La Recherche Aérospatiale **1** (1994), 5–21.
38. P. Tiso and D. J. Rixen, *Discrete empirical interpolation method for finite element structural dynamics*, *Topics in Nonlinear Dynamics, Volume 1*, Springer New York, 2013. 203–212, .
39. 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*, *16th AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, 2003, .
40. M. Yano, *Discontinuous Galerkin reduced basis empirical quadrature procedure for model reduction of parametrized nonlinear conservation laws*, Advances in Computational Mathematics **45** (2019), no. 5-6, 2287–2320.
41. M. Yano, *Goal-oriented model reduction of parametrized nonlinear partial differential equations: Application to aerodynamics*, International Journal for Numerical Methods in Engineering **121** (2020), no. 23, 5200–5226.
42. 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** (2019), 1104–1123.