Efficient hyperreduction of high-order discontinuous Galerkin methods: element-wise and point-wise reduced quadrature formulations

Eugene Du[a,1], Masayuki Yano[a,2,*]

[a]*Institute for Aerospace Studies, University of Toronto, 4925 Duffein Street, Toronto, M3H 5T6, Ontario, Canada*

Abstract

We develop and assess projection-based model reduction methods for high-order discontinuous Galerkin (DG) discretizations of parametrized nonlinear partial differential equations with applications in aerodynamics. Our emphasis is on the choice of hyperreduction methods. We analyze computational complexity and use numerical examples to show that typical hyperreduction methods based on element-wise or index-wise sampling of the residual vector cannot effectively reduce high-order DG discretizations, with a large number of degrees of freedom and quadrature points per element and conversely a small number of elements. To overcome this limitation, we devise a hyperreduction method for high-order DG methods that samples individual quadrature points, instead of elements, to provide a finer decomposition of the residual. We compare the formulation, implementation, and computational complexity of the element-wise and point-wise formulations. We then numerically assess the two formulations using Euler flow over a shape-parametrized airfoil and laminar and turbulent Navier-Stokes flow over a three-dimensional aerodynamic body. In each case, we observe that the point-wise hyperreduced model reduces the computational time by several orders of magnitude relative to the high-order DG method and a few orders of magnitude relative to the element-wise hyperreduced model. In addition, the computational performance of the point-wise formulation does not deteriorate with the DG polynomial degree, unlike for the element-wise formulation.

Keywords: discontinuous Galerkin method, adaptive high-order method, model reduction, hyperreduction, empirical quadrature procedure, aerodynamics

1. Introduction

We consider rapid and reliable solution of parametrized nonlinear partial differential equations (PDEs), with applications in conservation laws in aerodynamics. Our goal is to evaluate quantities of interest (i.e., output), such as lift and drag, for various input parameters, which describe configurations such as the flight condition and geometry. We consider many-query scenarios, where we wish to rapidly evaluate the output for many different input parameter values. Our approach to the problem is projection-based model reduction based on offline-online computational decomposition: in the offline stage, we invoke the full-order model (FOM) several times to explore the parameter space and to train a reduced-order model (ROM); in the online stage, we evaluate the ROM for many different parameter values. The focus of this work is the development and assessment of an efficient model reduction technique for *high-order discontinuous Galerkin (DG) discretizations* of parametrized *nonlinear* PDEs.

To clearly describe the objective of this work, we first introduce an abstract parametrized PDE problem. To begin, we introduce a P-dimensional parameter space $\mathcal{D} \subset \mathbb{R}^P$ and a N_h-dimensional approximation

*Corresponding author

Email addresses: eugene.du@alumni.utoronto.ca (Eugene Du), masa.yano@utoronto.ca (Masayuki Yano)

[1]Former Graduate Student, Institute for Aerospace Studies, University of Toronto

[2]Assistant Professor, Institute for Aerospace Studies, University of Toronto

space $\mathcal{V}_h$. The FOM problem is as follows: given $\mu \in \mathcal{D}$, find $u_h(\mu) \in \mathcal{V}_h$ such that

$$R(u_h(\mu), v_h; \mu) = 0 \quad \forall v_h \in \mathcal{V}_h, \tag{1}$$

where $R : \mathcal{V} \times \mathcal{V} \times \mathcal{D} \to \mathbb{R}$ is a semi-linear form. To accelerate the solution of the FOM problem (1) using projection-based model reduction, we introduce two ingredients. The first is a N-dimensional *reduced basis (RB) space* $\mathcal{V}_N \subset \mathcal{V}_h$, in which we seek the ROM solution, where $N \ll N_h$. We can construct an RB space using, for instance, proper orthogonal decomposition (POD) or the greedy algorithm [39]. Both methods require the FOM solution snapshots $\{u_h(\mu)\}_{\mu \in \Xi_{\text{train}}}$ associated with a training parameter set $\Xi_{\text{train}} \subset \mathcal{D}$, and their computation often constitutes the majority of the offline training cost. The approximation of the parametrized PDE in the RB space yields the following (non-hyperreduced) Galerkin ROM problem: given $\mu \in \mathcal{D}$, find $u_N(\mu) \in \mathcal{V}_N$ such that

$$R(u_N(\mu), v_N; \mu) = 0 \quad \forall v_N \in \mathcal{V}_N.$$

This (non-hyperreduced) ROM cannot be evaluated efficiently, as it requires the evaluation of the FOM residual form $R(\cdot, \cdot; \cdot)$. To address the issue, we introduce the second ingredient of projection-based model reduction of nonlinear PDEs: *hyperreduction*. The goal of hyperreduction is to approximate the residual form $R(\cdot, \cdot; \cdot)$ by a hyperreduced residual form $\tilde{R}(\cdot, \cdot; \cdot)$, which can be evaluated in $\mathcal{O}(N)$ cost. We then evaluate the hyperreduced ROM solution $\tilde{u}_N(\mu) \in \mathcal{V}_N$ such that

$$\tilde{R}(u_N(\mu), v_N; \mu) = 0 \quad \forall v_N \in \mathcal{V}_N$$

in $\mathcal{O}(N)$ operations. The goal of model reduction for parametrized nonlinear PDEs is to construct accurate and efficient hyperreduced ROMs.

While the computational reduction in the online stage is often the primary focus of model reduction, controlling the offline training cost is also important particularly (i) for complex and large-scale computational fluid dynamics (CFD) problems and (ii) in practical engineering settings with limited computing resources and stringent turnaround time requirements. As the computation of the solution snapshots often dominates the offline training cost, we may reduce the offline cost by using an efficient FOM. In this work, we use an adaptive high-order DG method. The efficacy of adaptive high-order DG methods to solve complex aerodynamics problems has been demonstrated in various studies in the past decade [40, 23, 19, 17]. Compared to second-order methods on meshes constructed following best-practices, adaptive high-order methods often reduce the computational time to achieve a given solution accuracy by a few orders of magnitude; we will also observe this in all of our examples in Section 5. In the context of model reduction, the method has the potential to provide a commensurate reduction in the offline cost and hence is well-suited as the FOM from which to build a ROM. Previous ROM works that use (non-adaptive) DG methods include [21, 14, 41, 29].

Given a FOM, we construct a ROM using the aforementioned two ingredients: RB and hyperreduction. In this work, we construct RB using the standard POD procedure and focus on the development and assessment of efficient hyperreduction methods *for high-order DG methods*. There are many hyperreduction methods, including the empirical interpolation method (EIM) [4]; its discrete variant, discrete EIM (DEIM) [11]; its variant for (continuous) finite-element method, unassembled DEIM (UDEIM) [38]; *the* hyperreduction method [32]; missing point estimation [3]; the Gauss-Newton approximate tensor (GNAT) method [7, 8]; the energy-conservative sampling and weighting (ECSW) method [15, 10]; the empirical quadrature procedure (EQP) [43]; and its variant for DG methods [41]. With the exception of the (non-discrete) EIM and the (non-DG variant of) EQP, these methods consider sampling subsets of elements or indices of the discrete residual vector associated with the (semi-)discrete FOM. In this work, using the DG variant of EQP [41] as a representative example, we will show that these discrete approaches cannot effectively reduce high-order DG discretizations, both in theory and in practice. As we will study in Section 3, this is because high-order DG methods have (i) a large number of degrees of freedom and quadrature points per element and (ii) conversely a small number of elements. The former means that the computation of any entry of the DG residual vector is expensive, and the latter means that, at least for methods based on element-wise sampling, there are fewer opportunities for hyperreduction.

The contributions of this work are the development and assessment of an efficient hyperreduction method for high-order DG methods. To overcome the aforementioned limitations of discrete hyperreduction methods, we first develop a hyperreduction method for high-order DG methods that samples individual quadrature points, instead of elements or discrete indices, to provide a finer decomposition of the DG residual. This is an extension of the (point-wise) EQP for continuous Galerkin methods developed in [26, 41] to high-order DG methods. More important contribution of the work is the comparison of the efficacy of the point-wise and element-wise hyperreduction methods, both in theory and in practice. The theoretical assessment follows from the analysis of computational operations for the point-wise and element-wise methods; the analysis shows the limitation of *any* discrete hyperreduction methods applied to high-order DG methods. The practical assessment is based on large-scale parametrized aerodynamics problems; the assessment demonstrates (a) the efficacy of adaptive high-order DG methods applied to aerodynamics problems and (b) the ability of the point-wise formulation to effectively hyperreduce adaptive high-order DG methods.

We now clarify the scope and limitation of this work. The focus of this work is the assessment of two hyperreduction methods for aerodynamics problems; however, we limit the scope to subsonic flows. Transonic and supersonic flows, with parameter-dependent shocks, are not amenable to model reduction using a standard (linear) RB. While several nonlinear approximation techniques have been recently developed and applied to transonic flows (e.g., [25, 16]), in this work we consider only subsonic flows to provide a concrete assessment of hyperreduction methods and to not introduce additional complications associated with nonlinear approximation spaces.

The remainder of this paper is organized as follows. In Section 2, we present the DG formulation and the associated point-wise and element-wise hyperreduction formulations. In Section 3, we discuss implementation of the point-wise and element-wise hyperreduction formulations and analyze the computational cost and storage requirement. In Section 4, we discuss the extension of the formulations to treat shape-parametrized problems. In Section 5, we assess the point-wise and element-wise hyperreduction formulations using three aerodynamics problems: Euler flow over a shape-parameterized airfiol; laminar Navier-Stokes flow over a three-dimensional aerodynamic body parametrized by the flight condition; and Reynolds-averaged Navier-Stokes (RANS) flow over a three-dimensional aerodynamic body parametrized by the flight condition.

2. Formulation

2.1. Problem statement

We first introduce the problem that we considered throughout this work. We introduce a P-dimensional parameter domain $\mathcal{D} \subset \mathbb{R}^P$, and a d-dimensional spatial domain $\Omega \subset \mathbb{R}^d$. The parametrized system of m PDEs that we consider is of the form

$$\nabla \cdot F(u(\mu);\mu) - \nabla \cdot (K(u(\mu);\mu)\nabla u(\mu)) = S(u(\mu), \nabla u(\mu);\mu) \quad \text{in } \Omega,$$

where $F : \mathbb{R}^m \times \mathcal{D} \to \mathbb{R}^{m\times d}$ is the convective flux, $K : \mathbb{R}^m \times \mathcal{D} \to \mathbb{R}^{m\times d\times m\times d}$ is the diffusion tensor, and $S : \mathbb{R}^m \times \mathbb{R}^{m\times d} \times \mathcal{D} \to \mathbb{R}^m$ is the source function. We complement the PDE with appropriate boundary conditions. Given the solution, we evaluate the functional output

$$s(\mu) \coloneqq J(u(\mu);\mu),$$

where J is the output functional. We will consider a variant of this problem with parametrized domains $\Omega(\mu) \subset \mathbb{R}^d$ in Section 4; however, in this section, we consider the non-parametrized domain Ω to simplify the presentation.

2.2. Discontinuous Galerkin method

We now review the DG formulation. For brevity, we limit our presentation to the aspects that are necessary to describe our hyperreduction procedure; we refer to [2, 18, 19] for detail. We first introduce a triangulation $\mathcal{T}_h$ of the domain Ω, which may contain hanging nodes. We next introduce the set of interior

facets $\Sigma \coloneqq \cup_{\kappa\in\mathcal{T}_h}\partial\kappa \setminus \partial\Omega$ and boundary facets $\Gamma \coloneqq \cup_{\kappa\in\mathcal{T}_h}\partial\kappa \cap \partial\Omega$. We then introduce a DG approximation space

$$\mathcal{V}_h \coloneqq \{v \in L^2(\Omega)^m \mid v|_\kappa \in \mathbb{P}^p(\kappa)^m,\ \forall\kappa \in \mathcal{T}_h\},$$

where $\mathbb{P}^p(\kappa) : \kappa \to \mathbb{R}$ is the space of polynomials of degree at most p. To simplify the presentation, we assume that the polynomial degree for all elements is the same; this assumption can be readily relaxed. The dimension of the DG space is $N_h \coloneqq \dim(\mathcal{V}_h) = mN_pN_e = N_\kappa N_e$, where $N_e \coloneqq |\mathcal{T}_h|$ is the number of elements, N_p is the number of degrees of freedom per element for each component, and $N_\kappa = mN_p$ is the number of degrees of freedom per element. Note that $N_p = \prod_{i=1}^d (p+i)/i$ for complete polynomials and $N_p = (p+1)^d$ for tensor-product polynomials.

We next introduce the (parametrized) residual form that defines the DG discretization. The symmetric interior penalty (IP) DG discretization of the system of conservation laws [18, 19] yields the residual semilinear form $R : \mathcal{V}_h \times \mathcal{V}_h \times \mathcal{D} \to \mathbb{R}$ such that

$$R(w, v; \mu) = \int_\Omega r_\Omega(w, v; \mu)dx + \int_\Sigma r_\Sigma(w, v; \mu)ds + \int_\Gamma r_\Gamma(w, v; \mu)ds, \tag{2}$$

where the volume integrand $r_\Omega : \mathcal{V}_h\times\mathcal{V}_h\times\mathcal{D} \to L^\infty(\Omega)$, the interior-facet integrand $r_\Sigma : \mathcal{V}_h\times\mathcal{V}_h\times\mathcal{D} \to L^\infty(\Sigma)$, and the boundary-facet integrand $r_\Gamma : \mathcal{V}_h \times \mathcal{V}_h \times \mathcal{D} \to L^\infty(\Gamma)$ are given by

$$r_\Omega(w, v; \mu) \coloneqq -\nabla v : F(w; \mu) + \nabla v : K(w; \mu)\nabla w - v \cdot S(w, \nabla w; \mu), \tag{3}$$

$$r_\Sigma(w, v; \mu) \coloneqq [v]^+_- \cdot \hat{F}(w^+, w^-; \hat{n}, \mu) - \{K(w; \mu)\nabla v\} : [\![w]\!] - [\![v]\!] : \{K(w; \mu)(\nabla w - \sigma[\![v]\!])\}, \tag{4}$$

$$\begin{aligned} r_\Gamma(w, v; \mu) \coloneqq\ & v \cdot \hat{F}_\Gamma(w; \hat{n}, \mu) - K(w; \mu)\nabla v : ((w - u_\Gamma(w; \mu)) \otimes \hat{n}) \\ & - (v \otimes \hat{n}) : G(K(w; \mu)(\nabla w - \sigma(w - u_\Gamma(w; \mu)) \otimes \hat{n}); \hat{n}, \mu); \end{aligned} \tag{5}$$

here, $\hat{F} : \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^d \times \mathcal{D} \to \mathbb{R}^m$ is the interior facet numerical flux, $\hat{F}_\Gamma : \mathbb{R}^m \times \mathbb{R}^d \times \mathcal{D} \to \mathbb{R}^m$ is the boundary facet numerical flux, $G : \mathbb{R}^{m\times d} \times \mathbb{R}^d \times \mathcal{D} \to \mathbb{R}^{m\times d}$ is the boundary diffusion flux, $[\![\cdot]\!]$ is the so-called jump operator given by $[\![w]\!] = w^+ \otimes \hat{n}^+ + w^- \otimes \hat{n}^-$, $\{\cdot\}$ is the so-called averaging operator given by $\{w\} = (w^+ + w^-)/2$, $\sigma \in \mathbb{R}_{>0}$ is the penalty constant for the IP DG method, and $\otimes$ denotes the tensor product of two order-1 tensors to yield a order-2 tensor. Throughout this work, we use Roe's approximate Riemann solver [30] to compute the numerical fluxes. Similarly, the output functionals that we consider in this work may be written in terms of its integrands as

$$J(w; \mu) \coloneqq \int_\Omega j_\Omega(w; \mu)dx + \int_\Gamma j_\Gamma(w; \mu)ds, \tag{6}$$

where $j_\Omega : \mathcal{V}_h \times \mathcal{D} \to L^\infty(\Omega)$ is the volume integrand, and $j_\Gamma : \mathcal{V}_h \times \mathcal{D} \to L^\infty(\Gamma)$ is the boundary integrand.

For PDEs with non-polynomial nonlinearities, we cannot exactly evaluate the integrals in (2) or (6). We hence approximate the terms using element-/facet-wise Gauss-like quadrature rules. We denote the pairs of quadrature points and weights on Ω, Σ, and Γ by $\{x_{\Omega,q}, \rho_{\Omega,q}\}_{q=1}^{Q_\Omega}$, $\{x_{\Sigma,q}, \rho_{\Sigma,q}\}_{q=1}^{Q_\Sigma}$, and $\{x_{\Gamma,q}, \rho_{\Gamma,q}\}_{q=1}^{Q_\Gamma}$, respectively. The quadrature approximation of the semilinear form (2) is given by

$$R_h(w, v; \mu) \coloneqq \sum_{q=1}^{Q_\Omega} \rho_{\Omega,q} r_\Omega(w, v; \mu)(x_{\Omega,q}) + \sum_{q=1}^{Q_\Sigma} \rho_{\Sigma,q} r_\Sigma(w, v; \mu)(x_{\Sigma,q}) + \sum_{q=1}^{Q_\Gamma} \rho_{\Gamma,q} r_\Gamma(w, v; \mu)(x_{\Gamma,q}), \tag{7}$$

and the quadrature approximation of the output functional (6) is given by

$$J_h(w; \mu) \coloneqq \sum_{q=1}^{Q_\Omega} \rho_{\Omega,q} j_\Omega(w; \mu)(x_{\Omega,q}) + \sum_{q=1}^{Q_\Gamma} \rho_{\Gamma,q} j_\Gamma(w; \mu)(x_{\Gamma,q}). \tag{8}$$

The subscript h on $R_h(\cdot, \cdot; \cdot)$ and $J_h(\cdot; \cdot)$ signifies the dependence of the forms on h due to the use of the mesh-dependent quadrature rules. The total number of quadrature points is $Q_h \coloneqq Q_\Omega + Q_\Sigma + Q_\Gamma$, which

is $\mathcal{O}(N_h)$. Throughout this work, we use Gauss-like quadrature rules that exactly integrate polynomials of degree $3p + p_{\text{geom}}$, where p_{geom} is the polynomial degree of parametric mapping for curved geometries. While it is impossible to exactly integrate the nonlinear flux and source terms associated with the Euler, Navier-Stokes, and RANS equations considered in Section 5, we have found the $3p + p_{\text{geom}}$ rule balances the accuracy and computational cost in practice.

We may now state the DG problem: given $\mu \in \mathcal{D}$, find $u_h(\mu) \in \mathcal{V}_h$ such that

$$R_h(u_h(\mu), v_h; \mu) = 0 \quad \forall v_h \in \mathcal{V}_h, \tag{9}$$

and then evaluate the output

$$s_h(\mu) \coloneqq J_h(u_h(\mu); \mu).$$

In practice, the (nonlinear) DG problem (9) is solved using a Newton-like method with pseudo-transient continuation (PTC) [22]. Newton-like methods require the solution of a sequence of linearized problems: given $\mu \in \mathcal{D}$ and a linearization point $y_h \in \mathcal{V}_h$, find $w_h \in \mathcal{V}_h$ such that

$$R_h'(y_h; w_h, v_h; \mu) = -R_h(y_h(\mu), v_h; \mu) \quad \forall v_h \in \mathcal{V}_h,$$

where $R_h'(y_h; w_h, v_h; \mu)$ is the Gateaux derivative of $R_h(\cdot, v_h; \mu)$ about $y_h \in \mathcal{V}_h$ in the direction $w_h \in \mathcal{V}_h$. More explicitly,

$$R_h'(y; w, v; \mu) = \sum_{q=1}^{Q_\Omega} \rho_{\Omega,q} r_\Omega'(y; w, v; \mu)(x_{\Omega,q}) + \sum_{q=1}^{Q_\Sigma} \rho_{\Sigma,q} r_\Sigma'(y; w, v; \mu)(x_{\Sigma,q}) + \sum_{q=1}^{Q_\Gamma} \rho_{\Gamma,q} r_\Gamma'(y; w, v; \mu)(x_{\Gamma,q}),$$

where, for instance, $r_\Omega'(y; w, v; \mu)$ is the Gateaux derivative of $r_\Omega(\cdot, v; \mu)$ about $y \in \mathcal{V}_h$ in the direction $w \in \mathcal{V}_h$ and is given by

$$\begin{aligned} r_\Omega'(y; w, v; \mu) \coloneqq &- \nabla v : DF(y; \mu) w + \nabla v : K(y; \mu) \nabla w + \nabla v : DK(y; \mu)(w \otimes \nabla y) \\ &- v \cdot D_1 S(y, \nabla y; \mu) w - v \cdot D_2 S(y, \nabla y; \mu) \nabla w, \end{aligned} \tag{10}$$

where $DF(y; \mu) \in \mathcal{L}(\mathbb{R}^m, \mathbb{R}^{m \times d})$ is the Jacobian of $F(\cdot; \mu) \in \mathbb{R}^{m \times d}$ evaluated about $y \in \mathbb{R}^m$, $DK(y; \mu) \in \mathcal{L}(\mathbb{R}^{m \times m \times d}, \mathbb{R}^{m \times d})$ is the Jacobian of $K(\cdot; \mu) \in \mathbb{R}^{m \times d \times m \times d}$ evaluated about $y \in \mathbb{R}^m$, $D_1 S(y, \nabla y; \mu) \in \mathcal{L}(\mathbb{R}^m, \mathbb{R}^m)$ is the Jacobian of $S(\cdot, \nabla y; \mu)$ evaluated about $y \in \mathbb{R}^m$, and $D_2 S(y, \nabla y; \mu) \in \mathcal{L}(\mathbb{R}^{m \times d}, \mathbb{R}^m)$ is the Jacobian of $S(y, \cdot; \mu)$ evaluated about $\nabla y \in \mathbb{R}^{m \times d}$. The Gateaux derivatives $r_\Sigma'(\cdot; \cdot, \cdot; \cdot)$ and $r_\Gamma'(\cdot; \cdot, \cdot; \cdot)$ are defined analogously; we omit the explicit presentation for brevity. The solution of the linearized problem requires the solution of a $N_h \times N_h$ sparse linear system. We solve the linear system using GMRES [33] with a block ILU preconditioner with the minimum discarded fill reordering [28].

We conclude the section with a remark on the choice of the DG method.

Remark 1. Instead of the IP method, we may consider other DG methods to discretize the diffusion term. However, not all DG methods admit a quadrature-point-wise decomposition of the form (7) and (8). In particular, any DG method that uses a lifting operator do not admit the decomposition because the evaluation of the lifting operators requires the solution of element-wise subproblems that depend on the solution everywhere in the element. Hence, methods such as the Bassi and Rebay's second method (BR2) [6] and the compact DG method (CDG) [27] do not provide a quadrature-point-wise decomposition. Of the nine DG methods reviewed in [2], the IP and nonsymmetric interior penalty Galerkin (NIPG) methods are the only methods that are consistent, are stable, and admit quadrature-point-wise decomposition. Between IP and NIPG, only the IP method is dual consistent. We hence use the IP method in this work.

2.3. Reduced-basis approximation

We now introduce a reduced basis (RB) approximation of the DG problem (9). To this end, we first introduce a reduced basis space $\mathcal{V}_N \subset \mathcal{V}_h$ and the associated basis $\{\phi_i\}_{i=1}^N$ for $N \ll N_h$. While we may construct the basis using various techniques (see, e.g., [20]), in this work we use the POD. Namely, we

introduce a parameter training set $\Xi_{\text{POD}} \subset \mathcal{D}$ of size $|\Xi_{\text{POD}}| \geq N$, solve the DG problem (9) to obtain snapshots $\{u_h(\mu)\}_{\mu\in\Xi_{\text{POD}}}$, and then apply POD to obtain the RB $\{\phi_i\}_{i=1}^N = \text{POD}_N(\{u_h(\mu)\}_{\mu\in\Xi_{\text{POD}}})$. Given the RB, we introduce the RB approximation: given $\mu \in \mathcal{D}$, find $u_N(\mu) \in \mathcal{V}_N$ such that

$$R_h(u_N(\mu), v_N; \mu) = 0 \quad \forall v_N \in \mathcal{V}_N, \tag{11}$$

and then evaluate the output

$$s_N(\mu) \coloneqq J_h(u_N(\mu); \mu).$$

The RB approximation yields a nonlinear algebraic system of dimension $N \ll N_h$, which is solved using a Newton-like method with pseudo-transient continuation (PTC) [22]. However, the evaluation of the residual form $R_h(\cdot,\cdot;\cdot)$ given by (7) requires $\mathcal{O}(Q_h) \approx \mathcal{O}(N_h)$ operations, which renders the approximation not online efficient; i.e., the online computational cost depends on Q_h, and hence the dimension N_h, of the underlying DG discretization.

2.4. Point-wise reduced-basis reduced-quadrature formulation

To enable online-efficient (i.e., N_h- and Q_h-independent) evaluation of the DG residual form (7) and the output form (8), we introduce reduced quadrature (RQ) rules. One approach to construct RQ rules is to replace the Gauss-like quadrature rules on Ω, Σ, and Γ in (7) by RQ rules given by $\{\tilde{x}^r_{\Omega,q}, \tilde{\rho}^r_{\Omega,q}\}_{q=1}^{\tilde{Q}^r_\Omega}$, $\{\tilde{x}^r_{\Sigma,q}, \tilde{\rho}^r_{\Sigma,q}\}_{q=1}^{\tilde{Q}^r_\Sigma}$, and $\{\tilde{x}^r_{\Gamma,q}, \tilde{\rho}^r_{\Gamma,q}\}_{q=1}^{\tilde{Q}^r_\Gamma}$. Here the superscript "$r$" signifies that the RQ rules are associated with the residual form (7). The RQ points are subsets of the FE quadrature points in the sense that $\{\tilde{x}^r_{\Omega,q}\}_{q=1}^{\tilde{Q}^r_\Omega} \subset \{x_{\Omega,q}\}_{q=1}^{Q_\Omega}$, $\{\tilde{x}^r_{\Sigma,q}\}_{q=1}^{\tilde{Q}^r_\Sigma} \subset \{x_{\Sigma,q}\}_{q=1}^{Q_\Sigma}$, and $\{\tilde{x}^r_{\Gamma,q}\}_{q=1}^{\tilde{Q}^r_\Gamma} \subset \{x_{\Gamma,q}\}_{q=1}^{Q_\Gamma}$. We discuss the construction of the RQ rules using the empirical quadrature procedure in Section 2.6; for now we assume the RQ rules are given. We refer to the RQ rules as "point-wise" RQ rules because the reduction is performed at individual quadrature point level, as opposed to at the element level.

The point-wise RQ approximation of (7) is given by

$$\tilde{R}(w, v; \mu) \coloneqq \sum_{q=1}^{\tilde{Q}^r_\Omega} \tilde{\rho}^r_{\Omega,q} r_\Omega(w, v; \mu)(\tilde{x}^r_{\Omega,q}) + \sum_{q=1}^{\tilde{Q}^r_\Sigma} \tilde{\rho}^r_{\Sigma,q} r_\Sigma(w, v; \mu)(\tilde{x}^r_{\Sigma,q}) + \sum_{q=1}^{\tilde{Q}^r_\Gamma} \tilde{\rho}^r_{\Gamma,q} r_\Gamma(w, v; \mu)(\tilde{x}^r_{\Gamma,q}). \tag{12}$$

Similarly, in (8), we approximate the FE quadrature on Ω and Γ by RQ rules given by $\{\tilde{x}^j_{\Omega,q}, \tilde{\rho}^j_{\Omega,q}\}_{q=1}^{\tilde{Q}^j_\Omega}$ and $\{\tilde{x}^j_{\Gamma,q}, \tilde{\rho}^j_{\Gamma,q}\}_{q=1}^{\tilde{Q}^j_\Gamma}$, so that its RQ approximation is given by

$$\tilde{J}(w; \mu) \coloneqq \sum_{q=1}^{\tilde{Q}^j_\Omega} \tilde{\rho}^j_{\Omega,q} j_\Omega(w; \mu)(\tilde{x}^j_{\Omega,q}) + \sum_{q=1}^{\tilde{Q}^j_\Gamma} \tilde{\rho}^j_{\Gamma,q} j_\Gamma(w; \mu)(\tilde{x}^j_{\Gamma,q}); \tag{13}$$

the superscript "j" signifies that the RQ rules are associated with the output functional (8). The RQ rules for $\tilde{R}(\cdot,\cdot;\cdot)$ and $\tilde{J}(\cdot;\cdot)$ are in general different. We now state the RB-RQ approximation problem: given $\mu \in \mathcal{D}$, find $u_N(\mu) \in \mathcal{V}_N$ such that

$$\tilde{R}(\tilde{u}_N(\mu), v_N; \mu) = 0 \quad \forall v_N \in \mathcal{V}_N, \tag{14}$$

and then evaluate the output

$$\tilde{s}_N(\mu) \equiv \tilde{J}(\tilde{u}_N(\mu); \mu).$$

The point-wise RB-RQ problem (14) is solved using a Newton-like method, which requires the solution of linearized problems: given $\mu \in \mathcal{D}$ and a linearization point $y_N \in \mathcal{V}_N$, find $w_N \in \mathcal{V}_N$ such that

$$\tilde{R}'(y_N; w_N, v_N; \mu) = -\tilde{R}(y_N, v_N; \mu) \quad \forall v_N \in \mathcal{V}_N,$$

where $\tilde{R}'(y_N; w_N, v_N; \mu)$ is the Gateaux derivative of $R(\cdot, v_N; \mu)$ about $y_N \in \mathcal{V}_N$ in the direction of $w_N \in \mathcal{V}_N$ and is given by

$$\tilde{R}'(y; w, v; \mu) = \sum_{q=1}^{\tilde{Q}^r_\Omega} \tilde{\rho}^r_{\Omega,q} r'_\Omega(y; w, v; \mu)(\tilde{x}^r_{\Omega,q}) + \sum_{q=1}^{\tilde{Q}^r_\Sigma} \tilde{\rho}^r_{\Sigma,q} r'_\Sigma(y; w, v; \mu)(\tilde{x}^r_{\Sigma,q}) + \sum_{q=1}^{\tilde{Q}^r_\Gamma} \tilde{\rho}^r_{\Gamma,q} r'_\Gamma(y; w, v; \mu)(\tilde{x}^r_{\Gamma,q}). \quad (15)$$

We note that the Gateaux derivative of (14) can be expressed as the RQ-wise sum of the point-wise Gateaux derivatives $r'_\Omega(\cdot;\cdot,\cdot;\cdot)$, $r'_\Sigma(\cdot;\cdot,\cdot;\cdot)$, and $r'_\Gamma(\cdot;\cdot,\cdot;\cdot)$. The decomposition plays a crucial role in the implementation of the point-wise RB-RQ and its computational cost analysis, which we will present in Section 3.

Remark 2. The point-wise RB-RQ forms (12) and (13) rely on the quadrature-point-wise decomposition of the DG forms (7) and (8), respectively. As noted in Remark 1, some DG methods use a lifting operator and do not admit a quadrature-point-wise decomposition. Specifically, DG methods such as BR2 and CDG are not compatible with the point-wise RQ formulation.

2.5. Element-wise reduced-basis reduced-quadrature formulation

An alternative approach to introduce RQ rules is to decompose the DG residual form (7) at the element level instead of at the quadrature-point level. We now review a element-wise RQ approach to the hyperreduction of DG methods introduced in [41]. To this end, we first introduce an element-wise decomposition of the DG residual (7):

$$R_h(w, v; \mu) = \sum_{\kappa \in \mathcal{T}_h} r_\kappa(w, v; \mu), \quad (16)$$

where the element-wise residual form $r_\kappa : \mathcal{V}_h \times \mathcal{V}_h \times \mathcal{D} \to \mathbb{R}$ is given by

$$r_\kappa(w, v; \mu) \coloneqq \sum_{q \in \mathcal{Q}(\kappa)} \rho_{\Omega,q} r_\Omega(w, v; \mu)(x_{\Omega,q}) + \sum_{q \in \mathcal{Q}(\partial\kappa \cap \Sigma)} \rho_{\Sigma,q} r^{\mathrm{mod}}_\Sigma(w, v; \mu)(x_{\Sigma,q}) + \sum_{q \in \mathcal{Q}(\partial\kappa \cap \Gamma)} \rho_{\Gamma,q} r_\Gamma(w, v; \mu)(x_{\Gamma,q}); \quad (17)$$

here, $\mathcal{Q}(\kappa)$, $\mathcal{Q}(\partial\kappa \cap \Sigma)$, and $\mathcal{Q}(\partial\kappa \cap \Gamma)$ are the sets of quadrature points that belong to the element κ, interior facets $\partial\kappa \cap \Sigma$, and boundary facets $\partial\kappa \cap \Gamma$, respectively, and $r^{\mathrm{mod}}_\Sigma(\cdot,\cdot;\cdot)$ is a modified version of $r_\Sigma(\cdot,\cdot;\cdot)$ in (4), which at the bare minimum accounts for the fact that the interior facets are shared by two elements and ideally also enables additional properties such as the energy stability; see, e.g., [41]. We similarly introduce an element-wise decomposition of the DG output functional (8):

$$J_h(w; \mu) = \sum_{\kappa \in \mathcal{T}_h} j_\kappa(w; \mu), \quad (18)$$

where $j_\kappa : \mathcal{V}_h \times \mathcal{D} \to \mathbb{R}$ is given by

$$j_\kappa(w; \mu) \coloneqq \sum_{q \in \mathcal{Q}(\kappa)} \rho_{\Omega,q} j_\Omega(w; \mu)(x_{\Omega,q}) + \sum_{q \in \mathcal{Q}(\partial\kappa \cap \Gamma)} \rho_{\Gamma,q} j_\Gamma(w; \mu)(x_{\Gamma,q}). \quad (19)$$

To construct an element-wise RQ approximation of the residual, we introduce a set of element-wise quadrature weights $\{\omega^r_\kappa\}_{\kappa \in \tilde{\mathcal{T}}^r_h}$, where $\tilde{\mathcal{T}}^r_h \subset \mathcal{T}_h$ is a sparse set of elements used in RQ. Given the element-wise RQ weights, the RQ approximation of the residual form (7) is given by

$$\tilde{R}^{\mathrm{el}}(w, v; \mu) \coloneqq \sum_{\kappa \in \tilde{\mathcal{T}}^r_h} \tilde{\omega}^r_\kappa r_\kappa(w, v; \mu). \quad (20)$$

Similarly, given a set of element-wise RQ weights $\{\omega^j_\kappa\}_{\kappa \in \tilde{\mathcal{T}}^j_h \subset \mathcal{T}_h}$ for the functional output (8), the element-wise RQ approximation of (8) is given by

$$\tilde{J}^{\mathrm{el}}(w; \mu) \coloneqq \sum_{\kappa \in \tilde{\mathcal{T}}^j_h} \tilde{\omega}^j_\kappa j_\kappa(w; \mu). \quad (21)$$

We now state the element-wise RB-RQ approximation problem: given $\mu \in \mathcal{D}$, find $\tilde{u}_N(\mu) \in \mathcal{V}_N$ such that

$$\tilde{R}^{\mathrm{el}}(\tilde{u}_N(\mu), v_N) = 0 \quad \forall v_N \in \mathcal{V}_N, \tag{22}$$

and then evaluate the output

$$\tilde{s}_N(\mu) \equiv \tilde{J}^{\mathrm{el}}(\tilde{u}_N(\mu); \mu).$$

Similarly to the point-wise RB-RQ problem (14), the element-wise RB-RQ problem (22) is solved using a Newton-like method. The Gateaux derivative of $\tilde{R}^{\mathrm{el}}(\cdot, v; \mu)$ about $y \in \mathcal{V}_N$ in the direction $w \in \mathcal{V}_N$ is given by

$$\tilde{R}^{\mathrm{el}\prime}(y; w, v; \mu) \coloneqq \sum_{\kappa \in \tilde{\mathcal{T}}_h^r} \tilde{\omega}_\kappa^r r'_\kappa(y; w, v; \mu), \tag{23}$$

where $r'_\kappa(y; w, v; \mu)$ is the Gateaux derivative of $r_\kappa(\cdot, v; \mu)$ evaluated about $y \in \mathcal{V}_N$ in the direction $w \in \mathcal{V}_N$. We will compare the computational cost of the element-wise and point-wise RB-RQ approximations in Section 3.

Remark 3. Unlike the point-wise RB-RQ formulation which requires a point-wise decomposition of the DG form (as discussed in Remark 2), the element-wise RB-RQ formulation requires only element-wise decomposition of the forms (16) and (18). Hence, the element-wise RB-RQ formulation can accommodate all nine DG methods reviewed in [2]. The element-wise RB-RQ formulation in [41] is based on the BR2 formulation.

2.6. Empirical quadrature procedure (EQP)

We find the quadrature rules for the point-wise reduced quadrature forms (12) and (13) and the element-wise reduced quadrature forms (20) and (21) using the EQP [26, 43]. The goal of the EQP is to find quadrature rules that are (i) sparse and (ii) accurate. The accuracy requirement is enforced by imposing "residual matching conditions" at a set of training points in the parameter space $\mathcal{D}$. An appropriate choice of "residual matching conditions" depends on the residual (or output) form that we wish to reduce. To this end, we first introduce the general form of the EQP for (a) a form $\bullet \in \{$"r" $\equiv$ residual, "j" $\equiv$ output$\}$ and (b) a domain $\Lambda \in \{\Omega, \Sigma, \Gamma\}$.

Definition 4 (point-wise EQP for RQ for $\bullet \in \{r, j\}$ on $\Lambda \in \{\Omega, \Sigma, \Gamma\}$)**.** Given a user-prescribed manifold-accuracy tolerance $\delta^\bullet \in \mathbb{R}_{>0}$ and a constant-accuracy tolerance $\delta^\Lambda \in \mathbb{R}_{>0}$, find a set of non-negative quadrature weights

$$\hat{\rho}_\Lambda^{\bullet,\star} = \underset{\hat{\rho}_\Lambda^\bullet \in \mathbb{R}^{Q_\Lambda}_{\geq 0}}{\arg\min} \|\hat{\rho}_\Lambda^\bullet\|_0$$

that satisfy the constant accuracy constraint

$$||\Lambda| - \sum_{q=1}^{Q_\Lambda} \hat{\rho}_{\Lambda,q}^\bullet| \leq \delta^\Lambda,$$

and $K|\Xi_{\mathrm{EQP}}|$ manifold accuracy constraints (or "residual matching conditions")

$$c_\Lambda^\bullet(\hat{\rho}_\Lambda^\bullet; \mu)_i \leq \delta^\bullet, \quad i = 1, \ldots, K, \ \forall \mu \in \Xi_{\mathrm{EQP}}, \tag{24}$$

where $c_\Lambda^\bullet : \mathbb{R}^{Q_\Lambda} \times \mathcal{D} \to \mathbb{R}^K$ defines the constraints, and $\Xi_{\mathrm{EQP}} \subset \mathcal{D}$ is a training parameter set. Once a set of quadrature weights $\{\hat{\rho}_{\Lambda,q}^{\bullet,\star}\}_{q=1}^{Q_\Lambda}$ is found, extract the nonzero weights to construct a RQ rule $\{\tilde{x}_{\Lambda,q}^\bullet, \tilde{\rho}_{\Lambda,q}^\bullet\}_{q=1}^{\tilde{Q}_\Lambda^\bullet}$.

In this work, we set $\Xi_{\mathrm{EQP}} = \Xi_{\mathrm{POD}}$ for simplicity, though the two sets need not be the same.

Having defined the general form of the EQP, we now define specific manifold accuracy constraints. We use the EQP introduced in [42], which is designed to control the output error $|s_N(\mu) - \tilde{s}_N(\mu)|$ due to hyperreduction (instead of some norm of the error $\|u_N(\mu) - \tilde{u}_N(\mu)\|$). By way of preliminaries, we introduce a

training state set $\{\hat{u}_N(\mu)\}_{\mu\in\Xi_{\mathrm{EQP}}}$ associated with the training parameter set Ξ_{EQP}. We in addition introduce the dual solutions $\{\hat{z}_N(\mu)\}_{\mu\in\Xi_{\mathrm{EQP}}}$, where $\hat{z}(\mu)\in\mathcal{V}_N$ satisfies

$$R'_h(\hat{u}_N(\mu); w_N, \hat{z}_N(\mu); \mu) = J'_h(\hat{u}_N(\mu); w_N(\mu); \mu) \quad \forall w_N \in \mathcal{V}_N(\mu).$$

We then express $\hat{z}_N(\mu)\in\mathcal{V}_N$ in terms of the RB $\{\phi_i\}_{i=1}^N$ and the associated coefficients $\hat{\mathbf{z}}_N\in\mathbb{R}^N$ as

$$\hat{z}_N(\mu) = \sum_{i=1}^N \hat{\mathbf{z}}_{N,i}(\mu)\phi_i. \tag{25}$$

We now introduce manifold accuracy constraints for the residual form (12) and the output form (13).

Definition 5 (point-wise EQP constraints for residual on $\Lambda\in\{\Omega,\Sigma,\Gamma\}$)**.** Given a training parameter set $\Xi_{\mathrm{EQP}}\subset\mathcal{D}$, the associated training state set $\{\hat{u}_N(\mu)\}_{\mu\in\Xi_{\mathrm{EQP}}}$, and the associated training dual state set $\{\hat{z}_N(\mu)\}_{\mu\in\Xi_{\mathrm{EQP}}}$, the point-wise RQ rule $\{\tilde{\rho}^r_{\Lambda,q}\}_{q=1}^{\tilde{Q}^r_\Lambda}$ for the residual on $\Lambda\in\{\Omega,\Sigma,\Gamma\}$ is the solution to the EQP (Definition 4) with the accuracy constraints (24) given by

$$c^r_\Lambda(\hat{\rho}^\bullet_\Lambda;\mu)_i \coloneqq |R_\Lambda(\hat{u}_N(\mu), \phi_i\hat{\mathbf{z}}_{N,i}(\mu);\mu) - \sum_{q=1}^{Q_\Lambda}\hat{\rho}^r_{\Lambda,q} r_\Lambda(\hat{u}_N(\mu),\phi_i\hat{\mathbf{z}}_{N,i}(\mu);\mu)(x_{\Lambda,q})| \le \delta^r,$$

for $i=1,\dots,N$, $\forall\mu\in\Xi_{\mathrm{EQP}}$, where $R_\Lambda(w,v;\mu)\coloneqq\sum_{q=1}^{Q_\Lambda}\rho_{\Lambda,q}r_\Lambda(w,v;\mu)$ is the part of the DG residual (7) associated with the integral over $\Lambda\in\{\Omega,\Sigma,\Gamma\}$, and $\hat{\mathbf{z}}_N(\mu)\in\mathbb{R}^N$ satisfies (25).

Definition 6 (point-wise EQP constraints for output functional on $\Lambda\in\{\Omega,\Gamma\}$)**.** Given a training parameter set $\Xi_{\mathrm{EQP}}\subset\mathcal{D}$ and the associated training state set $\{\hat{u}_N(\mu)\}_{\mu\in\Xi_{\mathrm{EQP}}}$, the point-wise RQ rule $\{\rho^j_{\Lambda,q}\}_{q=1}^{\tilde{Q}^j_\Lambda}$ for the output on $\Lambda\in\{\Omega,\Gamma\}$ is the solution to the EQP (Definition 4) with the accuracy constraints (24) given by

$$c^j_\Lambda(\hat{\rho}^\bullet_\Lambda;\mu) \coloneqq |J_\Lambda(\hat{u}_N(\mu);\mu) - \sum_{q=1}^{Q_\Lambda}\hat{\rho}^j_{\Lambda,q} j_\Lambda(\hat{u}_N(\mu);\mu)(x_{\Lambda,q})| \le \delta^j \quad \forall\mu\in\Xi_{\mathrm{EQP}},$$

where $J_\Lambda(w;\mu)\coloneqq\sum_{q=1}^{Q_\Lambda}\rho_{\Lambda,q}j_\Lambda(w;\mu)$ is the part of the DG output functional (8) associated with the integral over $\Lambda\in\{\Omega,\Gamma\}$.

Remark 7. The EQP for the element-wise residual form (20) and (21) are defined analogously; see [42] for a detailed presentation.

Remark 8. The sparse quadrature optimization problem associated with the EQP (Definition 4) can be solved using many different methods. The original work on the EQP [43] recast the problem as a linear programming problem and solve it using a simplex method. In the present work, we recast the problem as a non-negative least square (NNLS) problem. We solve the problem using a QR-based method [24] and in particular its parallel version developed for ECSW [10]. In Section 5, we will observe that the parallel NNLS solver can efficiently solve the EQP problem with tens of millions of quadrature points.

Remark 9. Both the point-wise and element-wise RB-RQ formulations readily apply to adaptive p-refined DG approximation spaces, where elements use different polynomial degrees. This is a direct consequence of the fact that the DG residual associated with adaptive p-refined spaces admits point-wise decomposition (7) and element-wise decomposition (16).

Remark 10. While in this work we use the point-wise RB-RQ formulation to hyperreduce the residual and output forms for parametrized steady nonlinear PDEs, the formulation can be extended to hyperreduce other forms. For instance, we may follow the formulation in [42] to equip the ROM with an a posteriori error estimate but use point-wise instead of element-wise decomposition of the residual to further improve online efficiency, as demonstrated in [13]. Similarly, the method can be readily extended to time-dependent problems [35]. As noted in the introduction, the primary contribution of the present work is the detailed theoretical and practical comparison of the point-wise and element-wise formulations, rather than its application to various problems.

	data	Λ in	$\bullet$ in	memory requirement
RQ rules	$\{\tilde{x}^\bullet_{\Lambda,q} \in \mathbb{R}^d, \tilde{\rho}^\bullet_{\Lambda,q} \in \mathbb{R}\}_{q=1}^{\tilde{Q}^\bullet_\Lambda}$	$\{\Omega, \Sigma, \Gamma\}$	$\{r, j\}$	$\sum_\bullet \sum_\Lambda (d+1)\tilde{Q}^\bullet_\Lambda$
Normals	$\{\hat{n}^\bullet_{\Lambda,q} \in \mathbb{R}^d\}_{q=1}^{\tilde{Q}^\bullet_\Lambda}$	$\{\Sigma, \Gamma\}$	$\{r, j\}$	$\sum_\bullet \sum_\Lambda d\tilde{Q}^\bullet_\Lambda$
IP penalties	$\{\sigma^\bullet_{\Lambda,q} \in \mathbb{R}\}_{q=1}^{\tilde{Q}^\bullet_\Lambda}$	$\{\Sigma, \Gamma\}$	$\{r, j\}$	$\sum_\bullet \sum_\Lambda \tilde{Q}^\bullet_\Lambda$
RB	$\{\{\phi_i(\tilde{x}^\bullet_{\Lambda,q}) \in \mathbb{R}^m\}_{i=1}^N\}_{q=1}^{\tilde{Q}^\bullet_\Lambda}$ $\{\{\nabla\phi_i(\tilde{x}^\bullet_{\Lambda,q}) \in \mathbb{R}^{m\times d}\}_{i=1}^N\}_{q=1}^{\tilde{Q}^\bullet_\Lambda}$	$\{\Omega, \Sigma, \Gamma\}$	$\{r, j\}$	$\sum_\bullet \sum_\Lambda m(d+1)N\tilde{Q}^\bullet_\Lambda$

Table 1: Summary of the `OnlineDataset` for the point-wise RB-RQ formulation.

3. Computational cost analysis

3.1. Point-wise RB-RQ formulation

We first analyze the online complexity and memory requirement of the point-wise RB-RQ formulation discussed in Section 2.4. Throughout this section, we denote by C^r_Λ the cost to evaluate the flux and source functions in $r_\Lambda(w, v; \mu)$ at a quadrature point, for $\Lambda \in \{\Omega, \Sigma, \Gamma\}$; for instance, C^r_Ω corresponds to the cost to evaluate $F(w(\tilde{x}_{\Omega,q}); \mu)$, $K(w(\tilde{x}^r_{\Omega,q}); \mu)\nabla w(\tilde{x}^r_{\Omega,q})$, and $S(w(\tilde{x}^r_{\Omega,q}), \nabla w(\tilde{x}^r_{\Omega,q}); \mu)$ in the volume residual form $r_\Omega(w, v; \mu)$ given by (3) for a given quadrature point index q. Similarly, we denote by $C^{r'}_\Lambda$ the cost to evaluate the flux and source Jacobians in $r'_\Lambda(u; w, v; \mu)$ given by, for instance, (10). We evaluate the point-wise RB-RQ residual form (12) and the Jacobian form (15) using the following point-wise assembly procedure:

- Residual evaluation. We evaluate the residual $\sum_\Lambda \sum_{q=1}^{\tilde{Q}^r_\Lambda} \tilde{\rho}^r_\Lambda r_\Lambda(\tilde{u}_N, \phi_i; \mu)$, $i = 1, \ldots, N$, given by (12) in two steps. We first evaluate the flux and source functions at the quadrature points, which requires $\sum_\Lambda C^r_\Lambda \tilde{Q}^r_\Lambda$ operations. We next multiply the test functions and the flux and source values, which requires approximately $\sum_\Lambda 2N\tilde{Q}^r_\Lambda$ operations. The total cost is hence $\sum_\Lambda (C^r_\Lambda \tilde{Q}^r_\Lambda + 2N\tilde{Q}^r_\Lambda)$.

- Jacobian evaluation. We evaluate the Jacobian $\sum_\Lambda \sum_{q=1}^{\tilde{Q}^r_\Lambda} \tilde{\rho}^r_\Lambda r_\Lambda(\tilde{u}_N; \phi_j, \phi_i; \mu)$, $i, j = 1, \ldots, N$, given by (15) in two steps. We first evaluate the flux and source Jacobians at the quadrature points, which requires $\sum_\Lambda C^{r'}_\Lambda \tilde{Q}^r_\Lambda$ operations. We next multiply the test and trial functions and the flux and source Jacobian values, which requires approximately $\sum_\Lambda 2N^2\tilde{Q}^r_\Lambda$ operations. The total cost is hence $\sum_\Lambda (C^{r'}_\Lambda \tilde{Q}^r_\Lambda + 2N^2\tilde{Q}^r_\Lambda)$.

The `OnlineDataset` required to evaluate the residual form (12) (and the output functional (13)) for the point-wise RB-RQ formulation using the above procedure is summarized in Table 1. For a practical value of N, the storage of the RB evaluated at quadrature points dominate the memory requirement.

We make three remarks.

Remark 11. The evaluation of the point-wise RB-RQ residual and Jacobian using the above procedure is identical to the typical procedure to evaluate the elemental residual and Jacobian in (high-order) DG methods. In DG methods, we apply the above procedure to a polynomial basis and a Gauss-like quadrature rule. In the RB-RQ method, we apply the above procedure to an RB and an RQ rule.

Remark 12. The interpretation of the point-wise RB-RQ as a (very) high-order DG method based on empirical RB and RQ in Remark 11 means that the method is amenable to efficient computational implementation based on BLAS, just like high-order DG methods. In this context, the `OnlineDataset` serves the same role as pre-computed bases and their gradients in the DG formulation.

Remark 13. We can readily partition the `OnlineDataset` by quadrature points to enable parallel computation of the RB-RQ approximation.

3.2. Element-wise RB-RQ formulation

We now analyze the online computational complexity and memory requirement of the element-wise RB-RQ formulation discussed in Section 2.5. The element-wise RB-RQ formulation is most readily implemented in a "non-intrusive" manner. Here, "non-intrusive" refers to the use of the existing elemental and facet residual vector (and Jacobian matrix) evaluation routines of the (high-order) DG code. To describe the procedure and analyze the associated computational cost, we introduce a few variables. We denote by $\mathcal{N}(\kappa)$ the set of elements that neighbors κ; analogously, $\mathcal{N}(\tilde{\mathcal{T}}_h)$ denotes the set of elements that neighbors elements in $\tilde{\mathcal{T}}_h$. In addition, we denote the number of volume, interior-facet, and boundary-facet quadrature points in/on an element κ by $Q_{\kappa,\Omega} = |\mathcal{Q}(\kappa)|$, $Q_{\kappa,\Sigma} = |\mathcal{Q}(\partial\kappa \cap \Sigma)|$, and $Q_{\kappa,\Gamma} = |\mathcal{Q}(\partial\kappa \cap \Gamma)|$, respectively. We finally recall that C_Λ^r and $C_\Lambda^{r'}$ denote the cost to evaluate the residual $r_\Lambda(w, v; \mu)$ and the Jacobian $r'_\Lambda(u; w, v; \mu)$ at a quadrature point on $\Lambda \in \{\Omega, \Sigma, \Gamma\}$, as introduced in Section 3.1.

We evaluate the element-wise RB-RQ residual form (20) and the Jacobian form (23) using the following "non-intrusive" assembly procedure:

- Residual evaluation. By way of preliminaries, we express the elemental residual associated with the element κ as

$$r_\kappa(u_N, \phi_i; \mu) = \left[\boldsymbol{\Phi}_\kappa^T \mathbf{r}_\kappa(\mathbf{u}_\kappa) + \sum_{\kappa' \in \mathcal{N}(\kappa)} \boldsymbol{\Phi}_\kappa^T \mathbf{r}_{\kappa,\kappa'}(\mathbf{u}_\kappa, \mathbf{u}_{\kappa'})\right]_i \tag{26}$$

  for $i = 1, \ldots, N$, where $\boldsymbol{\Phi}_\kappa \in \mathbb{R}^{N_\kappa \times N}$ is the elemental RB coefficient matrix, $\mathbf{u}_\kappa = \boldsymbol{\Phi}_\kappa \mathbf{u}_N \in \mathbb{R}^{N_\kappa}$ is the elemental DG basis coefficients associated with the RB coefficients $\mathbf{u}_N \in \mathbb{R}^N$, $\mathbf{r}_\kappa : \mathbb{R}^{N_\kappa} \to \mathbb{R}^{N_\kappa}$ is the DG volume residual operator associated with the element κ, $\mathbf{r}_{\kappa,\kappa'} : \mathbb{R}^{N_\kappa} \times \mathbb{R}^{N_\kappa} \to \mathbb{R}^{N_\kappa}$ is the DG facet residual operator associated with the facet shared by κ and κ'. We first evaluate $\mathbf{r}_\kappa(\mathbf{u}_\kappa)$ and $\mathbf{r}_{\kappa,\kappa'}(\mathbf{u}_\kappa, \mathbf{u}_{\kappa'})$, $\kappa' \in \mathcal{N}(\kappa)$; this step requires the evaluation of the flux and source functions at all of the element and facet quadrature points, which requires $\sum_\Lambda C_\Lambda^r Q_{\kappa,\Lambda}$ operation, and the multiplication of the element DG test functions and the flux and source values, which requires $\sum_\Lambda 2N_\kappa Q_{\kappa,\Lambda}$ operations. We then multiply the residual vectors with the elemental coefficient RB matrix $\boldsymbol{\Phi}_\kappa$, which requires $2NN_\kappa$ operations. The cost per element is hence $\sum_\Lambda (C_\Lambda^r Q_{\kappa,\Lambda} + 2N_\kappa Q_{\kappa,\Lambda}) + 2NN_\kappa$. The total cost is hence $(\sum_\Lambda (C_\Lambda^r Q_{\kappa,\Lambda} + 2N_\kappa Q_{\kappa,\Lambda}) + 2NN_\kappa)\tilde{N}_e^r$, where $\tilde{N}_e^r = |\tilde{\mathcal{T}}_h|$.

- Jacobian evaluation. By way of preliminaries, we express the elemental Jacobian associated with the element κ as

$$r'_\kappa(u_N; \phi_j, \phi_i; \mu) = \left[\boldsymbol{\Phi}_\kappa^T \left.\frac{\partial \mathbf{r}_\kappa}{\partial \mathbf{u}_\kappa}\right|_{\mathbf{u}_\kappa} \boldsymbol{\Phi}_\kappa + \sum_{\kappa' \in \mathcal{N}(\kappa)} \boldsymbol{\Phi}_\kappa \left.\frac{\partial \mathbf{r}_{\kappa,\kappa'}}{\partial \mathbf{u}_\kappa}\right|_{\mathbf{u}_\kappa, \mathbf{u}_{\kappa'}} \boldsymbol{\Phi}_\kappa + \sum_{\kappa' \in \mathcal{N}(\kappa)} \boldsymbol{\Phi}_\kappa \left.\frac{\partial \mathbf{r}_{\kappa,\kappa'}}{\partial \mathbf{u}_{\kappa'}}\right|_{\mathbf{u}_\kappa, \mathbf{u}_{\kappa'}} \boldsymbol{\Phi}_{\kappa'}\right]_{ij}, \tag{27}$$

  where $\frac{\partial \mathbf{r}_\kappa}{\partial \mathbf{u}_\kappa} : \mathbb{R}^{N_\kappa} \to \mathbb{R}^{N_\kappa \times N_\kappa}$ is the DG volume Jacobian operator associated with the element κ, and $\frac{\partial \mathbf{r}_{\kappa,\kappa'}}{\partial \mathbf{u}_\kappa} : \mathbb{R}^{N_\kappa} \times \mathbb{R}^{N_\kappa} \to \mathbb{R}^{N_\kappa \times N_\kappa}$ and $\frac{\partial \mathbf{r}_{\kappa,\kappa'}}{\partial \mathbf{u}_{\kappa'}} : \mathbb{R}^{N_\kappa} \times \mathbb{R}^{N_\kappa} \to \mathbb{R}^{N_\kappa \times N_\kappa}$ are the DG facet Jacobian operator associated with the facet shared by κ and κ'. We first evaluate $\left.\frac{\partial \mathbf{r}_\kappa}{\partial \mathbf{u}_\kappa}\right|_{\mathbf{u}_\kappa}$, $\left.\frac{\partial \mathbf{r}_{\kappa,\kappa'}}{\partial \mathbf{u}_\kappa}\right|_{\mathbf{u}_\kappa, \mathbf{u}_{\kappa'}}$, and $\left.\frac{\partial \mathbf{r}_{\kappa,\kappa'}}{\partial \mathbf{u}_{\kappa'}}\right|_{\mathbf{u}_\kappa, \mathbf{u}_{\kappa'}}$, $\kappa' \in \mathcal{N}(\kappa)$; this step requires the evaluation of the flux and source Jacobians at all of the element and facet quadrature points, which requires $\sum_\Lambda C_\Lambda^{r'} Q_{\kappa,\Lambda}$ operation, and the multiplication of the element DG test functions and the flux and source values, which requires $\sum_\Lambda 2N_\kappa^2 Q_{\kappa,\Lambda}$ operations. We then multiply the Jacobian matrices with the elemental coefficient RB matrices $\boldsymbol{\Phi}_\kappa$ and $\boldsymbol{\Phi}_{\kappa'}$, $\kappa' \in \mathcal{N}(\kappa)$, which requires $|\mathcal{N}(\kappa)|(2N^2 N_\kappa + 2N^2 N_\kappa)$ operations. The cost per element is hence $\sum_\Lambda (C_\Lambda^{r'} Q_{\kappa,\Lambda} + 2N_\kappa^2 Q_{\kappa,\Lambda}) + |\mathcal{N}(\kappa)|(2N^2 N_\kappa + 2N^2 N_\kappa)$. The total cost is hence $(\sum_\Lambda (C_\Lambda^{r'} Q_{\kappa,\Lambda} + 2N_\kappa^2 Q_{\kappa,\Lambda}) + |\mathcal{N}(\kappa)|(2N^2 N_\kappa + 2N^2 N_\kappa))\tilde{N}_e^r$.

The `OnlineDataset` required to evaluate the residual form (20) (and the output functional (21)) for the element-wise RB-RQ formulation using the above non-intrusive proecedure is summarized in Table 2. We also provide in Table 3 the typical number of degrees of freedom and quadrature points per element in two

	data	$\bullet$ in	memory requirement
RQ rules	$\{\tilde{\kappa}_q^\bullet, \tilde{\omega}_q^\bullet \in \mathbb{R}\}_{q=1}^{\tilde{N}_e^\bullet}$	$\{r, j\}$	$\sum_\bullet 2\tilde{N}_e^\bullet$
RB coefficients	$\{\Phi_\kappa \in \mathbb{R}^{N_\kappa \times N}\}_{\kappa \in \mathcal{N}(\tilde{\mathcal{T}}_h^\bullet)}$	$\{r, j\}$	$\sum_\bullet N_\kappa N \lvert\mathcal{N}(\tilde{\mathcal{T}}_h^\bullet)\rvert$

Table 2: Summary of the `OnlineDataset` for the element-wise RB-RQ formulation.

	dimension	$p=1$	$p=2$	$p=3$	$p=4$
#dof (N_κ)	2	12	24	40	60
	3	20	50	100	175
#quad points ($\sum_\Lambda Q_{\kappa,\Lambda}$)	2	21	45	77	117
	3	81	275	637	1215

Table 3: The number of degrees of freedom and quadrature points per element under the following assumption: (i) complete polynomial basis; (ii) quadrilateral ($d = 2$) and hexahedral ($d = 3$) element; (iii) volume and facet quadrature using the (tensor-product) Gauss quadrature that integrates the degree $3p + p_{\text{geom}}$ polynomials exactly; (iv) isoparametric mapping so that $p_{\text{geom}} = p$; and (iv) the Euler and Navier-Stokes equations so that $m = d + 2$.

and three dimensions using complete polynomials. We use complete (instead of tensor-product) polynomials for quadrilaterals and hexahedrals, since they require much fewer degrees of freedom for a given order of accuracy; for instance, the $p = 4$ approximation in three dimensions require 625 degrees of freedom for tensor-product polynomials but (only) 175 degrees of freedom for complete polynomials. The storage of the RB coefficients dominate the memory requirement.

3.3. Comparisons of computational costs of point-wise and element-wise formulations

We now compare the computational complexity and memory requirement of the point-wise and element-wise RB-RQ formulations. Tables 4a and 4b compare the cost to evaluate the RB-RQ residuals (12) and (20) and the associated Jacobians. We first note that the number of function evaluations at quadrature points for the point-wise and element-wise RB-RQ formulations are $\sum_\Lambda \tilde{Q}_\Lambda^r$ and $\sum_\Lambda Q_{\kappa,\Lambda}\tilde{N}_e^r$, respectively. In practice, as we will see in Section 5, $\sum_\Lambda \tilde{Q}_\Lambda^r$ is greater than $\tilde{N}_e^r$ but are of the same order (i.e., within a factor of 5). In addition, as shown in Table 3, the quantity $\sum_\Lambda Q_{\kappa,\Lambda}$ can be large in three dimensions for a high p. As a result, $\sum_\Lambda Q_{\kappa,\Lambda}\tilde{N}_e^r$ is typically a few orders of magnitude greater than $\sum_\Lambda \tilde{Q}_\Lambda^r$, and the difference increases with p.

We next compare the cost to multiply the flux and source functions evaluated at quadrature points with the basis functions to form the residual vector. The cost for this operation for the point-wise and element-wise RB-RQ formulations are $\sum_\Lambda 2N\tilde{Q}_\Lambda^r$ and $\sum_\Lambda 2N_\kappa Q_{\kappa,\Lambda}\tilde{N}_e^r + 2NN_\kappa\tilde{N}_e^r$, respectively. Again, in practice,

	flux and source evaluation	basis multiplication
point-wise	$\sum_\Lambda C_\Lambda^r \tilde{Q}_\Lambda^r$	$\sum_\Lambda 2N\tilde{Q}_\Lambda^r$
element-wise	$\sum_\Lambda C_\Lambda^r Q_{\kappa,\Lambda}\tilde{N}_e^r$	$\sum_\Lambda 2N_\kappa Q_{\kappa,\Lambda}\tilde{N}_e^r + 2NN_\kappa\tilde{N}_e^r$

(a) residual evaluation cost

	flux and source evaluation	basis multiplication
point-wise	$\sum_\Lambda C_\Lambda^{r'} \tilde{Q}_\Lambda^r$	$\sum_\Lambda 2N^2\tilde{Q}_\Lambda^r$
element-wise	$\sum_\Lambda C_\Lambda^{r'} Q_{\kappa,\Lambda}\tilde{N}_e^r$	$\sum_\Lambda 2N_\kappa^2 Q_{\kappa,\Lambda}\tilde{N}_e^r + 2(N^2N_\kappa + NN_\kappa^2)\lvert\mathcal{N}(\kappa)\rvert\tilde{N}_e^r$

(b) Jacobian evaluation cost

	memory requirement
point-wise	$\sum_\Lambda m(d+1)\tilde{Q}_\Lambda^r N$
element-wise	$N_\kappa \lvert\mathcal{N}(\tilde{\mathcal{T}}_h^r)\rvert N$

(c) approximate storage requirement

Table 4: Summary of computational cost for the point-wise and element-wise RB-RQ formulations.

$\sum_\Lambda \tilde{Q}^r_\Lambda$ is greater than $\tilde{N}^r_e$ but within a factor of 5. On the other hand, given typical values of N_κ and $\sum_\Lambda Q_{\kappa,\Lambda}$ in Table 3, the quantity $(\sum_\Lambda Q_{\kappa,\Lambda} + N)N_\kappa$ is a few orders of magnitude greater than N. As a result, the cost of basis multiplication to form the residual vector for the element-wise RB-RQ formulation, $\sum_\Lambda 2N_\kappa Q_{\kappa,\Lambda}\tilde{N}^r_e + 2NN_\kappa\tilde{N}^r_e$, is typically a few orders of magnitude greater than that for the point-wise RB-RQ formulation, $\sum_\Lambda 2N\tilde{Q}^r_\Lambda$, and the difference again increases with p. The difference in the cost of basis multiplication to form the Jacobian matrix is even greater due to the appearance of the squared terms.

We finally compare the memory requirement for the point-wise and element-wise RB-RQ formulations, summarized in Table 4c. In both formulations, the storage of the RB dominates the overall memory requirement, which requires $\sum_\bullet \sum_\Lambda m(d+1)\tilde{Q}^\bullet_\Lambda N$ and $\sum_\bullet N_\kappa|\mathcal{N}(\tilde{\mathcal{T}}^\bullet_h)|N$ for the point-wise and element-wise RB-RQ formulations, respectively. We first note that $|\mathcal{N}(\tilde{\mathcal{T}}^\bullet_h)|$ is greater than but of the same order as $\tilde{N}^r_e$, which in turn is comparable to $\sum_\Lambda \tilde{Q}^r_\Lambda$. On the other hand, given the typical value of N_κ in Table 3, $m(d+1)$ is substantially smaller than N_κ for a typical p, and the difference increases with p. As a result, the memory requirement is also a few orders of magnitude smaller for the point-wise RB-RQ formulation than for the element-wise RB-RQ formulation.

In summary, we conclude that both the computational complexity and memory requirement of the point-wise RB-RQ formulation is a few orders of magnitude smaller than the element-wise RB-RQ formulation. In addition, the difference in the efficiencies of the two formulations increases as p increases. In Section 5, we will observe that the computational time for the point-wise RB-RQ formulation is 70–800$\times$ less than that for the element-wise RB-RQ formulation, and the difference increases with p. We conclude this section with a few remarks.

Remark 14. This section analyzes and compares the computational cost of point-wise and element-wise RB-RQ formulations in terms of the number of floating-point operations and memory requirement. The analysis does not account for differences in computational efficiency that may be achieved on a modern computer with a deep memory hierarchy. However, both element-wise and point-wise formulations are amenable to an efficient BLAS-based implementation. The element-wise RB-RQ formulation requires evaluation of the DG residual and Jacobian on selected RQ elements, where each DG residual can be efficiently evaluated using BLAS especially for a large p. The point-wise formulation can be interpreted as a (very) high-order DG method with empirical RB and RQ, which enables efficient BLAS-based implementation; see Remarks 11 and 12.

Remark 15. While the detail of the cost analysis for the element-wise RB-RQ formulation is specific to the formulation, other "discrete" or "non-intrusive" hyperreduction methods that work with the discrete DG residual $\mathbf{r}_\kappa$ in (26) suffer from the same efficiency loss when they are applied to high-order DG methods. This is because the evaluation of even just *one* entry of the discrete DG residual $\mathbf{r}_\kappa$ requires the evaluation of the flux and source functions at *all* of the quadrature points in the element κ. As high-order DG methods use a large number of quadrature points per element (as summarized in Table 3), the cost $\sum_\Lambda C^{r'}_\Lambda Q_{\kappa,\Lambda}\tilde{N}^r_e$ is significant. The efficiency loss observed for the element-wise RB-RQ is a fundamental limitation shared with all "discrete" or "non-intrusive" hyperreduction methods that work with the discrete DG residual $\mathbf{r}_\kappa$, including DEIM [11], UDEIM [38], GNAT [7], and ECSW [15]. Moreover, the limitation applies not just to high-order DG methods but more generally to high-order finite element methods. The finer decomposition of the residual into quadrature points (instead of elements) is necessary to overcome this fundamental limitation, as done in the point-wise RB-RQ formulation.

Remark 16. While the element-wise RB-RQ formulation is more computationally expensive than the point-wise RB-RQ formulation, the former may provide stability advantages. Namely, with an appropriate choice of the interface residual $r^{\text{mod}}_\Sigma(\cdot,\cdot;\cdot)$ in (17), the ROM can inherit the energy stability of the underlying DG method for *any* choice of (non-negative) RQ quadrature weights [41]. This, to our knowledge, cannot be achieved using the point-wise RB-RQ formulation. However, in practice, we did not observe stability issues with the point-wise RB-RQ formulation, at least for the problems considered in Section 5. (We also refer to [9] for the development of a nonlinearly stable (i.e., entropy stable) ROMs for conservation laws.)

4. Treatment of shape-parameterized problems

In this section we discuss the treatment of problems with parametrized shapes (i.e., geometries). Our treatment follows the standard "reference domain" or "map-then-discretize" formulation to treat parametrized geometries in model reduction; see, e.g., [31, 37, 12]. To begin, we introduce a mapping $T(\cdot;\mu) : \Omega_0 \to \Omega(\mu)$ from the reference domain Ω_0 to the transformed domain $\Omega(\mu)$ parametrized by $\mu \in \mathcal{D}$. In practice, this mapping may be defined using, for instance, free-form deformation [34] or, as we consider in Section 5.1, cubic splines, where the parameter μ is associated with the coordinates of the control points. We then introduce an approximation space $\mathcal{V}_{\Omega(\mu)}$ on $\Omega(\mu)$ and the associated semi-linear form $R : \mathcal{V}_{\Omega(\mu)} \times \mathcal{V}_{\Omega(\mu)} \times \mathcal{D} \to \mathbb{R}$ such that

$$R(w_\mu, v_\mu; \mu) = \int_{\Omega(\mu)} r_\Omega(w_\mu, v_\mu; \mu)dx + \int_{\Sigma(\mu)} r_\Sigma(w_\mu, v_\mu; \mu)ds + \int_{\Gamma(\mu)} r_\Gamma(w_\mu, v_\mu; \mu)ds \quad \forall w_\mu, v_\mu \in \mathcal{V}_{\Omega(\mu)},$$

where $r_\Omega(\cdot,\cdot;\cdot)$, $r_\Sigma(\cdot,\cdot;\cdot)$, and $r_\Gamma(\cdot,\cdot;\cdot)$ are given by (3), (4), and (5), respectively. The solution is $u_\mu(\mu) \in \mathcal{V}_{\Omega(\mu)}$ such that

$$R(u_\mu(\mu), v_\mu; \mu) = 0 \quad \forall v_\mu \in \mathcal{V}_{\Omega(\mu)}. \tag{28}$$

This problem can be recast in the reference domain. Namely, we introduce an approximation space $\mathcal{V}$ on Ω_0 and the associated semi-linear form

$$R_0(w, v; \mu) = \int_{\Omega_0} r_{\Omega_0}(w, v; \mu)dx + \int_{\Sigma_0} r_{\Sigma_0}(w, v; \mu)ds + \int_{\Gamma_0} r_{\Gamma_0}(w, v; \mu)ds \quad \forall w, v, \in \mathcal{V}, \tag{29}$$

where

$$r_{\Omega_0}(w, v; \mu) \coloneqq \Big[-\nabla_\mu v : F(w;\mu) + \nabla_\mu v : K(w;\mu)\nabla_\mu w - v \cdot S(w, \nabla_\mu w; \mu)\Big]\det(J(\mu)), \tag{30}$$

$$\begin{aligned} r_{\Sigma_0}(w, v; \mu) \coloneqq \Big[&[v]^+_- \cdot \hat{F}(w^+, w^-; \hat{n}_\mu, \mu) - \{K(w;\mu)\nabla_\mu v\} : [\![w]\!]_\mu \\ &- [\![v]\!]_\mu : \{K(w;\mu)(\nabla_\mu w - \sigma[\![v]\!]_\mu)\}\Big]\det(J(\mu))\|J^{-T}(\mu)\hat{n}_0\|_2, \end{aligned} \tag{31}$$

$$\begin{aligned} r_{\Gamma_0}(w, v; \mu) \coloneqq \Big[&v \cdot \hat{F}_\Gamma(w; \hat{n}_\mu, \mu) - K(w;\mu)\nabla_\mu v : ((w - u_\Gamma(w;\mu)) \otimes \hat{n}_\mu) \\ &- (v \otimes \hat{n}_\mu) : G(K(w;\mu)(\nabla_\mu w - \sigma(w - u_\Gamma(w;\mu)) \otimes \hat{n}_\mu); \hat{n}_\mu, \mu)\Big]\det(J(\mu))\|J^{-T}(\mu)\hat{n}_0\|_2, \end{aligned} \tag{32}$$

where $\nabla_\mu v \coloneqq \nabla_0 v J^{-1}(\mu)$ is the transformed gradient, $\hat{n}_\mu \coloneqq J^{-T}\hat{n}_0/\|J^{-T}\hat{n}_0\|_2$ is the transformed unit normal vector, and $J(\mu) : \Omega_0 \to \mathbb{R}^{d\times d}$ is the Jacobian of the transformation associated with $T(\mu) : \Omega_0 \to \Omega(\mu)$. The solution is $u(\mu) \in \mathcal{V}$ such that

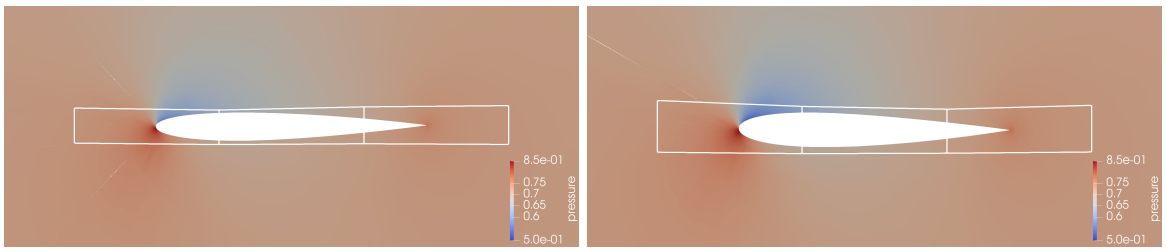
$$R(u(\mu), v; \mu) = 0 \quad \forall v \in \mathcal{V}. \tag{33}$$

The two solutions, $u_\mu(\mu) \in \mathcal{V}_{\Omega(\mu)}$ given by (28) and $u(\mu) \in \mathcal{V}$ given by (33), are related by $u(\mu) = u_\mu(\mu) \circ T(\cdot;\mu)$ over Ω_0.

To solve problems with parametrized shapes, we apply the DG and RB-RQ formulations developed in Section 2 to the transformed semilinear form (29) instead of (2). In other words, we simply replace the original integrands (3), (4), and (5) with the transformed integrands (30), (31), and (32), respectively. For the point-wise RB-RQ formulation, we can accomplish the transformation efficiently by transforming the `OnlineDataset` described in Table 1. We summarize the required transformation in Table 5. We apply the transformation to the `OnlineDataset`, once, before we solve the nonlinear problem (14) so that the transformation need not be performed on-the-fly during the Newton iteration.

data	transformation	Λ in
RQ points	$\tilde{x}^\bullet_{\Lambda,q} \leftarrow T(\tilde{x}^\bullet_{\Lambda,q};\mu)$	$\{\Omega,\Sigma,\Gamma\}$
RQ weights	$\tilde{\rho}^\bullet_{\Omega,q} \leftarrow \det(J(\tilde{x}^\bullet_{\Lambda,q};\mu))\tilde{\rho}^\bullet_{\Omega,q}$	
	$\tilde{\rho}^\bullet_{\Lambda,q} \leftarrow \det(J(\tilde{x}^\bullet_{\Lambda,q};\mu))\|J^{-T}(\tilde{x}^\bullet_{\Lambda,q};\mu)\hat{n}^\bullet_{\Lambda,q}\|_2\tilde{\rho}^\bullet_{\Lambda,q}$	$\{\Sigma,\Gamma\}$
Normals	$\hat{n}^\bullet_{\Lambda,q} \leftarrow J^{-T}(\tilde{x}^\bullet_{\Lambda,q};\mu)\hat{n}^\bullet_{\Lambda,q}/\|J^{-T}(\tilde{x}^\bullet_{\Lambda,q};\mu)\hat{n}^\bullet_{\Lambda,q}\|_2$	$\{\Sigma,\Gamma\}$
RB gradients	$\nabla\phi_i(\tilde{x}^\bullet_{\Lambda,q}) \leftarrow \nabla\phi_i(\tilde{x}^\bullet_{\Lambda,q})J^{-1}(\tilde{x}^\bullet_{\Lambda,q};\mu)$	$\{\Omega,\Sigma,\Gamma\}$

Table 5: Transformation of point-wise RB-RQ `OnlineDataset`.



(a) snapshot: low lift

(b) snapshot: high lift

Figure 1: Example snapshots for the shape-parametrized airfoils. The 4×2 cage represent the geometry control points.

5. Examples

5.1. Shape-parametrized airfoils: Euler

We first consider inviscid flow over a family of shape-parametrized airfoils based on the NACA0012 airfoil. The governing equation is the two-dimensional compressible Euler equations in entropy variables [5]. We fix the Mach number and the angle of attack to $M_\infty = 0.3$ and $\alpha = 3°$, respectively. The airfoil is transformed using cubic spline interpolation with 4×2 control points over the $1.6c \times 0.16c$ rectangular region centered about the airfoil, where c is the chord. We allow each of the eight control points to move in the x_2 direction by a distance of $\Delta x_2 \in [-0.02c, 0.02c]$, which yields an eight-dimensional parameter space $\mathcal{D} \subset \mathbb{R}^8$. The quantity of interest is the lift coefficient, which takes on a value in $[2.67 \times 10^{-1}, 5.90 \times 10^{-1}]$ over $\mathcal{D}$. Example solutions and geometry transformations are shown in Figure 1. We consider this two-dimensional problem (i) because the element-wise and point-wise RQ rules are easier to visualize in two dimensions than in three dimensions and (ii) to demonstrate the formulations for a problem with a relatively large number of parameters.

We construct the element-wise and point-wise RB-RQ models as follows. For each $p \in \{1, 2, 3, 4\}$, we start with the initial mesh that contains 150 elements and obtain the solution snapshots for 50 randomly chosen parameter values in $\Xi_{\text{POD}} \subset \mathcal{D} \subset \mathbb{R}^8$ using anisotropic-h adaptive DG method, such that that lift error estimate for each of the snapshots is less than 1.5×10^{-3} ($\sim 0.3\%$). We then apply POD to identify $N = 18$ most dominant modes; as we will see shortly, this results in RB approximations (without hyperreduction) with the maximum error of $\sim 3 \times 10^{-3}$ ($\sim 0.5\%$). We will choose the EQP tolerance so that the combination of the DG, RB, and RQ errors is less than 1%. We then apply the element-wise and point-wise EQP to construct the element-wise and point-wise RB-RQ models, respectively. All computations are performed on a 10-core workstation. Throughout Section 5, all CPU times are reported in core-seconds, which we compute as the product of the wall-clock time in seconds and the number of cores used.

Table 6a summarizes the error in the RB, element-wise RB-RQ, and point-wise RB-RQ approximations over a randomly chosen test parameter set $\Xi_{\text{test}} \neq \Xi_{\text{POD}}$ of size $|\Xi_{\text{test}}| = 10$. Columns 2–4 report the mean, standard deviation, and maximum error in the (non-hyperreduced) RB approximation with respect to the FE approximation (i.e., $|s_h(\mu) - s_N(\mu)|$) over Ξ_{test}. We observe that $N = 18$ POD modes is sufficient to achieve the maximum lift error of $\sim 3 \times 10^{-4}$ ($\sim 0.5\%$). Columns 5–7 report the statistics of the output error due to hyperreduction (i.e., $|s_N(\mu) - \tilde{s}_N(\mu)|$) for the element-wise RB-RQ approximations; we observe that

	non-hyperreduced RB $\lvert s_h - s_N\rvert \times 10^4$			element-wise RB-RQ $\lvert s_N - \tilde{s}_N\rvert \times 10^4$			point-wise RB-RQ $\lvert s_N - \tilde{s}_N\rvert \times 10^4$		
p	mean	std	max	mean	std	max	mean	std	max
1	18.14	12.21	34.07	8.37	4.10	17.95	1.36	0.90	2.79
2	9.78	8.04	27.93	1.14	1.42	4.76	1.07	1.00	2.64
3	13.00	7.17	23.88	0.79	0.53	1.96	1.91	2.10	7.38
4	14.96	11.37	35.85	1.45	0.82	2.68	3.00	2.70	9.52

(a) reduced-order model error summary

	adaptive DG				element-wise RB-RQ			point-wise RB-RQ		
p	N_h	N_e	Q_h	time	N	$\tilde{N}_e$	time	N	$\tilde{Q}$	time
1	53,508	4,459	160,200	196	18	503	10.70	18	1,313	0.0883
2	8,808	367	18,198	26	18	258	8.55	18	1,030	0.0820
3	11,640	291	23,960	41	18	213	9.33	18	998	0.0793
4	19,980	333	33,804	72	18	232	14.30	18	951	0.0729

(b) full- and reduced-order model cost summary

p	constraint evaluation	optimization solve	EQP/FOM solve
1	118.7	325.4	2.27
2	15.0	76.5	3.52
3	15.5	60.7	1.87
4	31.0	85.1	1.62

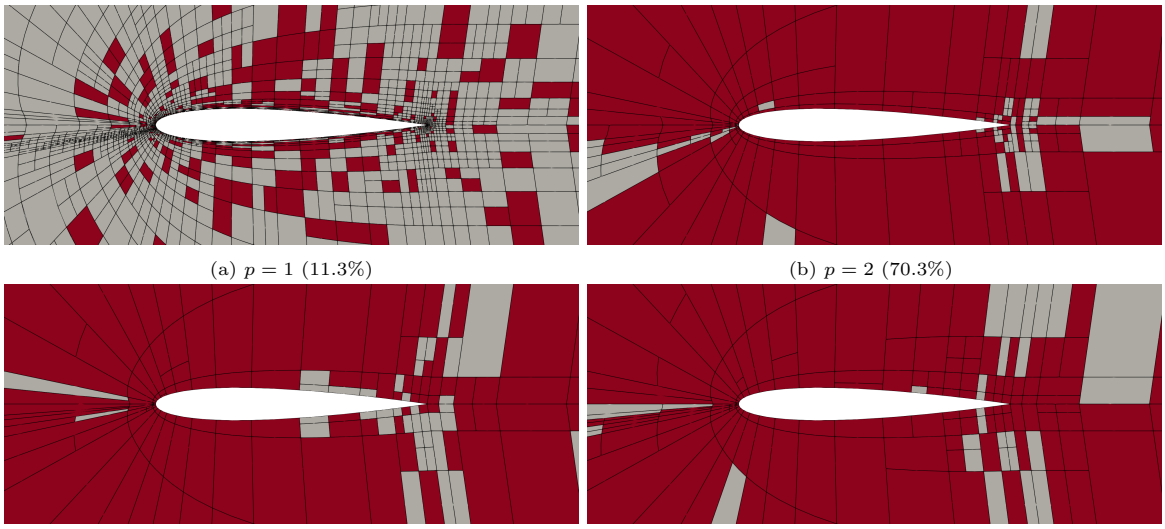
(c) point-wise EQP training CPU time

Table 6: Summary of the model reduction for the Euler flow over shape-parametrized airfoils. All CPU times are in core-seconds, which we define as the product of the wall-clock time in seconds and the number of cores used. The error summary in (a) shows the statistics (i.e., mean, standard deviation, and maximum) of the DG vs (non-hyperreduced) RB output errors (columns 2–4), the RB vs element-wise RB-RQ output errors (columns 5–7), and the RB vs point-wise RB-RQ errors (columns 8–10) over Ξ_{test}. Note that the errors are scales by 10^4. The first five columns of (b) summarize the adaptive DG cost and show the polynomial degree p, the number of degrees of freedom N_h, the number of elements N_e, the number of quadrature points Q_h, and the average CPU time to solve a single DG problem on the final adapted mesh (and not the time for the entire adaptation procedure) over Ξ_{test}. The next three columns summarize the element-wise RB-RQ cost and show the size of the RB N, the number of reduced elements $\tilde{N}_e$, and the average CPU time. The last three columns summarize the point-wise RB-RQ cost and show the size of the RB N, the number of RQ points $\tilde{Q}$, and the average CPU time. The point-wise EQP training summary in (c) shows the polynomial degree p, the CPU time to compute the constraints and solve the optimization problem, and the ratio of the total EQP time to a single DG solve on the final adapted mesh.

the element-wise EQP provides a tight control of hyperreduction error, with the maximum error of $\sim 0.3\%$ over Ξ_{test}. Similarly, Columns 8-10 show that the point-wise EQP the maximum error of $\sim 0.15\%$ over Ξ_{test}.

Having verified that the error due to the RB approximation and hyperreduction are well controlled, we now assess the computational cost. Table 6b summarizes the computational cost of the $p = 1, \ldots, 4$ adaptive DG, element-wise RB-RQ, and point-wise RB-RQ approximations. Columns 2–5 show that the $p > 1$ DG approximations are significantly more efficient than the $p = 1$ DG approximation both in terms of the number of degrees of freedom N_h and the CPU time required to meet the target drag tolerance of 2×10^{-4}. (The CPU time is the time required to solve a single DG problem on the final adapted mesh; it is not the time for the entire adaptation procedure.) We also observe that the $p > 1$ DG approximations require far fewer elements than the $p = 1$ DG approximation; for instance, the $p = 1$ DG approximation requires $N_e = 4459$ elements to meet the target lift error tolerance, whereas the $p = 3$ DG approximation requires only $N_e = 291$ elements. The final adapted meshes are shown in Figure 2. We note that the $p = 4$ DG approximation happens to use a larger number of elements than the $p = 3$ DG approximation due to the differences in the sequence of meshes generated in the mesh adaptation process.

Columns 6–8 of Table 6b summarize the cost of the element-wise RB-RQ approximations. We first observe that the fraction of elements used by RB-RQ varies significantly, from 11.3% (503 elements) for $p = 1$ to $\sim 70\%$ for $p = 2, 3, 4$. As also shown in Figure 2, the fraction increases with p because the number



(a) $p = 1$ (11.3%) (b) $p = 2$ (70.3%)

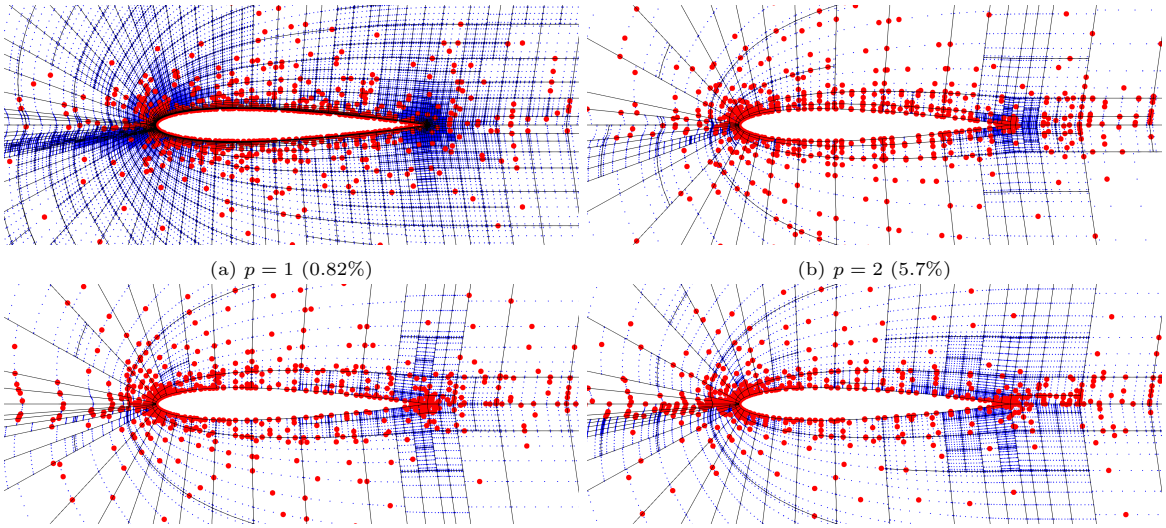
(c) $p = 3$ (73.2%) (d) $p = 4$ (69.7%)

Figure 2: Adaptive DG meshes and the RQ elements (in red) used in the element-wise RB-RQ approximation in the vicinity of the airfoil. The percentage indicates the fraction of the elements used in the element-wise RB-RQ approximation; i.e., $\tilde{N}_e/N_e$.

of DG elements required to meet the target error tolerance is much smaller for the $p > 1$ approximations than for the $p = 1$ approximation (~ 300 vs 4459), while the number of element-wise RQ elements does not decrease significantly (~ 230 vs 503). In other words, as the adaptive high-order methods are efficient and use a small number of elements, there is less opportunity for element-wise hyperreduction. This results in a limited reduction in CPU time; the CPU time speedup is 18.3× for the $p = 1$ approximation, but is only 3.0–5.0× for $p > 1$ approximations. In fact, as the majority of the DG elements are used by the element-wise RB-RQ, the residual evaluation time per PTC iteration for DG and RB-RQ are similar; however, we achieve the moderate speedup as the linear system solved in each PTC iteration is smaller, and the number of PTC iterations required to converge the nonlinear problem is fewer.

Columns 9–11 of Table 6b summarize the cost of the point-wise RB-RQ approximations. We first observe that the number of RQ points used by the $p = 1, \ldots, 4$ approximations are all ~ 1000; the variation in the number of RQ points is small. Figure 3 shows that the RQ points are clustered in the vicinity of the airfoil. As the number of RQ points are similar for all p and the cost to evaluate the residual at a point is the same for all p, the CPU times are also similar for all p. The CPU time speedup varies from 2220× for $p = 1$ to 317× for $p = 2$; the difference is primarily due to the difference in the efficiency of the baseline DG approximations. With respect to the most efficient DG method ($p = 2$), the speed up is 294–357× for $p = 1, \ldots, 4$. We also observe that the point-wise RB-RQ approximation is 104–196× more efficient than the element-wise RB-RQ approximation.

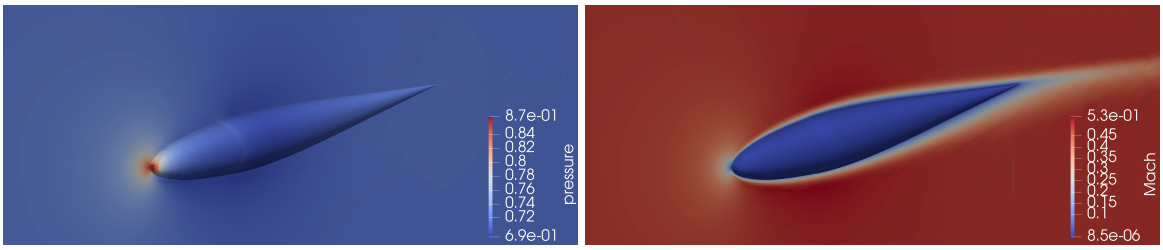
We finally discuss the offline training cost for point-wise EQP, which is summarized in Table 6c. As discussed, the EQP comprises two steps: the evaluation of the EQP constraints (24); and the solution of the optimization problem given by Definition 4. Both steps are computed in parallel, and hence the reported CPU time is the product of the wall-clock time and the number of cores (i.e., 10); see Remark 8 for a discussion of the parallel NNLS solver used to solve the EQP problem. This problem with $|\Xi_{\text{EQP}}| = |\Xi_{\text{POD}}| = 50$ and $N = 18$ yields $K = 900$ manifold accuracy constraints, which is relatively large. As a result, the cost of the EQP relative to a *single* FOM solve on the final adapted mesh varies from 1.62 to 3.52, which is higher than other cases considered in this work. Nevertheless, the EQP time is less than 7% of the time required to compute $|\Xi_{\text{POD}}| = 50$ FOM snapshots, and hence the computation of the snapshots dominates the overall offline training time.



(a) $p = 1$ (0.82%) (b) $p = 2$ (5.7%)

(c) $p = 3$ (4.2%) (d) $p = 4$ (2.8%)

Figure 3: Adaptive DG meshes and the DG quadrature points (in blue) and the RQ points (in red) used in the point-wise RB-RQ approximation in the vicinity of the airfoil. The percentage indicates the fraction of quadrature points used in the point-wise RB-RQ approximation; i.e., $\tilde{Q}/Q_h$.



(a) pressure (b) Mach

Figure 4: Laminar Navier-Stokes flow over the BTC0 body at $\alpha = 5°$ and $M_\infty = 0.5$.

5.2. ADIGMA BTC0: laminar Navier-Stokes

We next consider a parametrized variant of the laminar BTC0 test case introduced in the ADIGMA project, a European initiative to develop adaptive higher-order methods for aerospace applications [23]. The case was also a test case in the First International Workshop on High-order CFD Methods [40]. The governing equation is the three-dimensional compressible Navier-Stokes equations in entropy variables [5]. The parameters are the angle of attack $\alpha \in [0°, 5°]$ and the freestream Mach number $M_\infty \in [0.3, 0.5]$; the Reynolds number is fixed at $Re = 5{,}000$. The quantity of interest is drag, which takes on a value in $[6.40 \times 10^{-2}, 6.92 \times 10^{-2}]$ over $\mathcal{D}$. An example solution is shown in Figure 4.

We construct the element-wise and point-wise RB-RQ models as follows. For each $p \in \{1, 2, 3, 4\}$, we start with a coarse initial mesh that contains 768 elements and obtain the solution snapshots for 4×4 parameter values in $\Xi_{\text{POD}} \subset \mathcal{D} \subset \mathbb{R}^2$ using an anisotropic-h adaptive DG method, so that that drag error estimate is less than 2×10^{-4} ($\sim 0.3\%$). We then apply POD to identify $N = 10$ most dominant modes; as we will see shortly, this results in the RB approximation with the maximum error of $\sim 2 \times 10^{-4}$ ($\sim 0.3\%$). We will choose the EQP tolerance so that the combination of the DG, RB, and RQ errors is less than 1%. We then apply the element-wise and point-wise EQP procedures to construct the element-wise and point-wise RB-RQ models. All computations are performed on a 80-core cluster.

Table 7a summarizes the error in the RB, element-wise RB-RQ, and point-wise RB-RQ approximations over a randomly chosen test parameter set $\Xi_{\text{test}} \neq \Xi_{\text{POD}}$ of size $|\Xi_{\text{test}}| = 10$. Columns 2–4 report the drag

	non-hyperreduced RB $\lvert s_h - s_N \rvert \times 10^5$			element-wise RB-RQ $\lvert s_N - \tilde{s}_N \rvert \times 10^5$			point-wise RB-RQ $\lvert s_N - \tilde{s}_N \rvert \times 10^5$		
p	mean	std	max	mean	std	max	mean	std	max
1	8.22	4.23	14.73	5.84	2.64	9.74	2.84	1.99	7.68
2	7.88	4.43	14.54	1.92	0.94	3.46	0.56	0.43	1.41
3	11.02	6.06	21.42	1.65	1.27	3.93	2.78	3.39	8.94
4	13.35	7.77	25.30	1.11	0.65	2.38	3.39	3.08	11.31

(a) reduced-order model error summary

	adaptive DG				element-wise RB-RQ			point-wise RB-RQ		
p	N_h	N_e	Q_h	time	N	$\tilde{N}_e$	time	N	$\tilde{Q}$	time
1	6,477,280	323,864	37,984,704	104,800	10	42	3.59	10	158	0.0507
2	198,700	3,974	1,356,804	1,808	10	49	11.00	10	149	0.0454
3	142,800	1,428	733,873	1,624	10	40	13.50	10	155	0.0443
4	181,650	1,038	1,046,358	3,224	10	45	36.40	10	160	0.0484

(b) full- and reduced-order model cost summary

p	constraint evaluation	optimization solve	EQP/FOM solve
1	46,128	1,643.3	0.456
2	1,634	87.0	0.952
3	709	55.0	0.470
4	1,143	72.0	0.377

(c) point-wise EQP training time

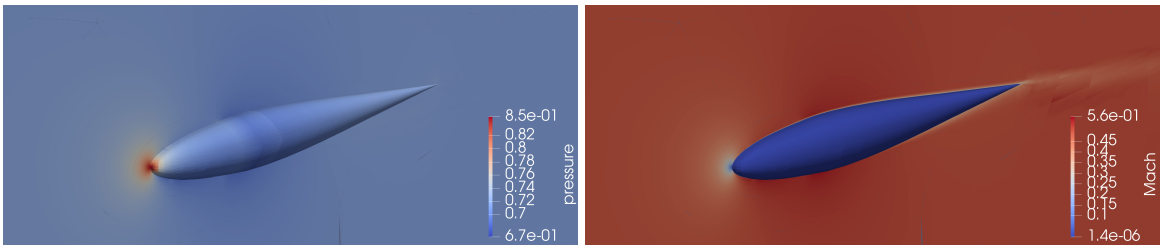
Table 7: Summary of the BTC0 laminar Navier-Stokes flow case. See the caption of Table 6 for the description of the entries.

error for the (non-hyperreduced) RB approximations. We observe that the $N = 10$ RB is sufficient to achieve the maximum drag error of $\sim 2 \times 10^{-4}$ ($\sim 0.3\%$). Columns 5–7 and 8–10 report the statistics of the output error due to hyperreduction for the element-wise and point-wise RB-RQ approximations, respectively. We observe that the maximum hyperreduction error is $\sim 0.15\%$.

Having verified that the errors due to the RB and RQ approximations are well controlled, we now assess the computational cost. Table 7b summarizes the cost of the $p = 1, \ldots, 4$ adaptive DG, element-wise RB-RQ, and point-wise RB-RQ approximations. Columns 2–5 show that the $p > 1$ approximations are significantly more efficient than the $p = 1$ approximation. For this problem, the $p = 3$ approximation is most efficient; compared to the $p = 1$ approximation, the $p = 3$ approximation requires 45× fewer degrees of freedom (6,477,280 vs 142,800) and 65× less CPU time (104,800 vs 1,624). In the context of model reduction, adaptive higher-order methods hence significantly reduces the time required to compute solution snapshots in the offline stage.

Columns 4–6 of Table 7b summarize the cost of the element-wise RB-RQ approximations. We observe that the element-wise EQP selects $\tilde{N}_e \approx 45$ RQ elements for all p. This is sufficient to control the output error due to hyperreduction to 1×10^{-4} ($\sim$0.15%) over Ξ_{test}. We also observe that the CPU time increases with p, as the number of degrees of freedom, quadrature points, and hence operations per element increases with p; this effect is more pronounced in three-dimensions than in two-dimension, as expected from Table 3 in Section 3. Due to the higher RB-RQ computational cost and the lower adaptive DG computational cost, the CPU time speedup decreases from 29,000× for $p = 1$ to merely 89× for $p = 4$.

Columns 9–11 of Table 7b summarize the cost of the point-wise RB-RQ approximations. We observe that the point-wise EQP selects $\tilde{Q} \approx 155$ RQ points for all p, which is sufficient to control the output error due to hyperreduction to $\sim$0.15% over Ξ_{test}. As the number of RQ points are similar for all p and the cost to evaluate the residual at a point is the same for all p, the CPU time are also similar for all p. The CPU speedup with respect to the most efficient adaptive DG method ($p = 3$) is $\sim$37,000× and the least efficient adaptive DG method ($p = 1$) is $\sim$2,400,000×. We also observe that, compared to the element-wise RB-RQ



(a) pressure (b) Mach

Figure 5: RANS flow over the BTC0 body at $\alpha = 5^\circ$ and $M_\infty = 0.5$.

approximations, the point-wise RB-RQ approximations are 71× ($p = 1$) to 750× ($p = 4$) more efficient; the improvement increases with p as the element-wise RB-RQ becomes increasingly less efficient, as anticipated from the analyses in Section 3.

We finally discuss the offline training cost for point-wise EQP, which is summarized in Table 7c. As before, the reported CPU time is the product of the wall-clock time and the number of cores (i.e., 80). This problem with $|\Xi_{\text{EQP}}| = |\Xi_{\text{POD}}| = 16$ and $N = 10$ yields $K = 160$ manifold accuracy constraints, which is smaller than that for the shape-parametrized airfoil problem considered in Section 5.1. The time for the EQP relative to a single FOM solve on the final adapted mesh is hence lower than the earlier case, varying from 0.37 to 0.95. The EQP time is less than 6% of the time required to compute all $|\Xi_{\text{POD}}| = 16$ FOM snapshots. It is worth noting that the parallel NNLS solver, discussed in Remark 8, is effective even for the EQP problem with up to 38 million quadrature points.

5.3. ADIGMA BTC0: Reynolds-averaged Navier-Stokes

We finally consider a parametrized variant of the turbulent ADIGMA BTC0 test case [23], which was also a test case in the First International Workshop on High-order CFD Methods [40]. The governing equation is the three-dimensional compressible RANS equations with the Spalart-Allmaras turbulence model with the so-called SA-neg fix [36, 1]. The parameters are the angle of attack $\alpha \in [0^\circ, 5^\circ]$, and the freestream Mach number $M_\infty \in [0.3, 0.5]$; the Reynolds number is fixed at $Re = 10^6$. The quantity of interest is drag, which takes on a value in $[1.05 \times 10^{-2}, 1.26 \times 10^{-2}]$ over $\mathcal{D}$. An example solution is shown in Figure 5; while the pressure distribution is similar to the laminar case considered in Section 5.2, the boundary layer is noticeably thinner in this higher Reynolds-number flow.

We construct the element-wise and point-wise RB-RQ models as follows. For each $p \in \{2, 3, 4\}$, we start with a coarse initial mesh that contains 832 elements and obtain the solution snapshots for 4×4 parameter values in $\Xi_{\text{POD}} \subset \mathcal{D} \subset \mathbb{R}^2$ using an anisotropic-h adaptive DG method, so that that drag error estimate is less than 3×10^{-5} ($\sim 0.3\%$); for the $p = 1$ approximation, we use the drag error tolerance of 6×10^{-5} as we are unable to meet the target drag error tolerance of 3×10^{-5} on the 80-core cluster in a reasonable time. We then apply POD to identify $N = 7$ most dominant modes; as we will see shortly, this results in the RB approximation with the maximum error of $\sim 3 \times 10^{-5}$ ($\sim 0.3\%$). We will choose the EQP tolerance so that the combination of DG, RB, and RQ errors is less than 1%. We then apply the element-wise and point-wise EQP procedures to construct the element-wise and point-wise RB-RQ models.

Table 8 summarizes the behavior of the adaptive DG approximations and the element-wise and point-wise RB-RQ approximations. As the observations for this RANS case are similar to the laminar case considered in Section 5.2, we summarize the key takeaways. Table 8a confirms that the $N = 7$ RB approximations achieve the maximum drag error of $\sim 0.3\%$ and the element-wise and point-wise RB-RQ method control the error due to hyperreduction to $\sim 0.15\%$ over the random test set of size $|\Xi_{\text{test}}| = 10$. Columns 2–5 of Table 8b show that the $p > 1$ discretizations are significantly more efficient than the $p = 1$ discretization. In fact, as discussed, we were unable to meet the target drag error tolerance of 3×10^{-5} using the $p = 1$ discretization on the 80-core cluster in a reasonable time and hence resorted to using a looser tolerance of 6×10^{-5}. Despite achieving the tighter tolerance, the $p = 3$ discretization compared to the $p = 1$ discretization requires 21×

	non-hyperreduced RB $\lvert s_h - s_N \rvert \times 10^5$			element-wise RB-RQ $\lvert s_N - \tilde{s}_N \rvert \times 10^5$			point-wise RB-RQ $\lvert s_N - \tilde{s}_N \rvert \times 10^5$		
p	mean	std	max	mean	std	max	mean	std	max
1*	0.66*	0.43*	1.32*	0.28*	0.22*	0.72*	0.58*	0.35*	1.23*
2	0.91	0.64	2.10	0.36	0.22	0.82	0.44	0.30	0.82
3	1.32	0.94	2.49	0.34	0.35	1.19	0.70	0.61	1.64
4	1.33	0.71	2.57	0.50	0.34	1.23	0.61	0.79	2.03

(a) reduced-order model error summary

	adaptive DG				element-wise RB-RQ			point-wise RB-RQ		
p	N_h	N_e	Q_h	time	N	$\tilde{N}_e$	time	N	$\tilde{Q}$	time
1*	5,660,208*	235,842*	27,783,392*	170,400*	7*	34*	7.43*	7*	78*	0.0737*
2	389,100	6,485	2,203,776	8,640	7	32	17.20	7	113	0.1020
3	273,000	2,275	1,168,993	6,360	7	34	31.10	7	113	0.1070
4	239,190	1,139	1,147,284	13,200	7	29	65.10	7	126	0.0910

(b) full- and reduced-order model cost summary

p	constraint evaluation	optimization solve	EQP/FOM solve
1*	44,493*	259.2*	0.263*
2	4,103	71.8	0.483
3	1,820	51.4	0.294
4	1,541	56.2	0.121

(c) point-wise EQP training time

Table 8: Summary of the BTC0 RANS flow case. See the caption of Table 6 for the description of the entries. *The adaptive DG tolerance is 6×10^{-5} for $p = 1$ instead of 3×10^{-5} used for $p > 1$.

fewer degrees of freedom (5,660,208 vs 273,000) and 27× lower CPU time (170,400 vs 6,360). The element-wise RB-RQ becomes increasingly less efficient with p, and the CPU time speedup with respect to the most efficient DG method ($p = 3$) varies from 860× for $p = 1$ to merely $\sim$ 98× for $p = 4$. On the other hand, the point-wise RB-RQ achieves CPU time that only weakly varies with p, and its speedup with respect to the most efficient DG method ($p = 3$) is $\sim$ 63,000× and the least efficient DG method ($p = 1$) is $\sim$ 1,600,000×. Finally, Table 8c shows that, despite involving tens of millions of quadrature points, the offline training time for point-wise EQP is a fraction of a single FOM solve (12%–48%), and is much smaller than the CPU time required to compute all 16 FOM snapshots (1%–3%).

6. Summary and perspectives

This work develops and assesses hyperreduction methods for high-order DG discretization of nonlinear PDEs. The two methods considered are based on element-wise and point-wise RQ. Both the theoretical analysis in Section 3 and the numerical assessment in Section 5 show that the element-wise RB-RQ formulation becomes increasingly less efficient as the underlying DG polynomial degree increases, as the number of degrees of freedom and quadrature points per element increases. In the view of Remark 15, this limitation of the element-wise RB-RQ formulation is shared with other discrete hyperreduction methods that works with discrete high-order DG residual, and more generally high-order finite element residual. On the other hand, the point-wise RB-RQ formulation achieves p-independent online computational cost; in addition, the method is a few orders of magnitude more efficient than the element-wise counterpart for any p. We also demonstrate that, even for point-wise EQP which involves tens of millions of DG quadrature points, the offline cost for EQP is a small fraction of the FOM snapshot computation cost, thanks to an efficient parallel EQP solver. The results demonstrate that, while the less intrusive element-wise (or more generally discrete) hyperreduction methods may be attractive from the implementation point of view, a more intrusive

point-wise formulation which provides a finer decomposition of the residual may be necessary for an efficient reduction of high-order DG — and more generally high-order finite element — methods.

Acknowledgments

The financial support for this work was provided by the Natural Sciences and Engineering Research Council of Canada and the Ontario Graduate Scholarship. 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] S. R. Allmaras, F. T. Johnson, and P. R. Spalart. Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model. 7th International Conference on Computational Fluid Dynamics ICCFD7-1902, ICCFD, 2012.

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

[3] P. Astrid, S. Weiland, K. Willcox, and T. Backx. Missing point estimation in models described by proper orthogonal decomposition. *IEEE Transactions on Automatic Control*, 53(10):2237–2251, 2008.

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

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

[6] F. Bassi, S. Rebay, G. Mariotti, S. Pedinotti, and M. Savini. A high-order accurate discontinuous finite element method for inviscid and viscous turbomachinery flows. In R. Decuypere and G. Dibelius, editors, *Turbomachinery - Fluid Dynamics and Thermodynamics, European Conference, 2*, pages 99–108, 1997.

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

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

[9] J. Chan. Entropy stable reduced order modeling of nonlinear conservation laws. *Journal of Computational Physics*, 423:109789, dec 2020.

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

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

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

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

[14] F. Fang, C. C. Pain, I. M. Navon, A. H. Elsheikh, J. Du, and D. Xiao. Non-linear Petrov-Galerkin methods for reduced order hyperbolic equations and discontinuous finite element methods. *Journal of Computational Physics*, 234:540–559, feb 2013.

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

[16] A. Ferrero, T. Taddei, and L. Zhang. Registration-based model reduction of parameterized two-dimensional conservation laws. *Journal of Computational Physics*, 457:111068, may 2022.

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

[18] R. Hartmann and P. Houston. Symmetric interior penalty DG methods for the compressible Navier-Stokes equations I: method formulation. *International Journal of Numerical Analysis & Modeling*, 3(1):1–20, 2006.

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

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

[21] A. Iollo, A. Dervieux, J.-A. Désidéri, and S. Lanteri. Two stable POD-based approximations to the Navier-Stokes equations. *Computing and Visualization in Science*, 3(1-2):61–66, may 2000.

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

[23] N. Kroll, H. Bieler, H. Deconinck, V. Couaillier, H. Ven, and K. Sørensen, editors. *ADIGMA - A European iitiative on the dvelopment of adaptive hgher-order variational methods for aerospace applications*. Springer Berlin Heidelberg, Feb. 2010.

[24] C. L. Lawson and R. J. Hanson. *Solving least squares problems*, volume 18. Society for Industrial & Applied Mathematics (SIAM), jul 1976.

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

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

[27] J. Peraire and P.-O. Persson. The compact discontinuous Galerkin (CDG) method for elliptic problems. *SIAM Journal on Scientific Computing*, 30(4):1806–1824, 2008.

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

[29] S. Riffaud, M. Bergmann, C. Farhat, S. Grimberg, and A. Iollo. The DGDD method for reduced-order modeling of conservation laws. *Journal of Computational Physics*, 437:110336, jul 2021.

[30] P. L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43(2):357–372, 1981.

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

[32] D. Ryckelynck. A priori hyperreduction method: an adaptive approach. *Journal of Computational Physics*, 202(1):346–366, jan 2005.

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

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

[35] 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*, 388:114206, jan 2022.

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

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

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

[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. In *16th AIAA Computational Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics, jun 2003.

[40] Z. J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. T. Huynh, N. Kroll, G. May, P.-O. Persson, B. van Leer, and M. Visbal. High-order CFD methods: current status and perspective. *International Journal for Numerical Methods in Fluids*, 72(8):811–845, 2013.

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

[42] M. Yano. Goal-oriented model reduction of parametrized nonlinear partial differential equations: Application to aerodynamics. *International Journal for Numerical Methods in Engineering*, 121(23):5200–5226, jun 2020.

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