Projection-based Model Order Reduction techniques

Olivier Zahm[*]

Lecture notes M2:SIAM, 2019.

Consider the computational chain

$$\underbrace{x \in \mathbb{R}^d}_{\text{parameter}} \xrightarrow{\text{linear system (1)}} \underbrace{u(x) \in \mathbb{R}^N}_{\text{model output}} \xrightarrow{\text{post-processing (2)}} \underbrace{Y(x) \in \mathbb{R}}_{\text{quantity of interest}}$$

where $u(x)$ is the solution to a parametrized algebraic equation

$$A(x)u(x) = b(x), \tag{1}$$

and where $Y(x)$ is a post-processing of $u(x)$ defined by

$$Y(x) = q(x)^T u(x). \tag{2}$$

The matrix $A(x) \in \mathbb{R}^{N\times N}$, the right-hand side $b(x) \in \mathbb{R}^N$ and the extractor $q(x) \in \mathbb{R}^N$ depend on a d-dimensional parameter $x = (x_1, \dots, x_d)$ which belongs to a parameter set $\mathcal{P} \subset \mathbb{R}^d$. For each new parameter value $x \in \mathcal{P}$, the cost for evaluating $x \mapsto Y(x)$ is dominated by the cost for solving the N-by-N algebraic system (1): this usually scales as $\mathcal{O}(N^3)$, which is prohibitive when $N \gg 1$ is large. The goal of this short note is to introduce standard projection-based methods which aims at reducing this cost.

Remark 0.1. Projection-based methods are usually presented and analyzed for parameter dependent *variational equations* of the form of

$$\text{find } u(x) \in V \text{ such that} \quad a(u(x), v\,; x) = \ell(v\,; x) \quad v \in V,$$

see for instance the nice tutorial [3] or the more detailed article [2]. This formulation is convenient to deal with parameterized Partial Differential Equations (PDEs) where V is typically a Sobolev space (generally the Hilbert space $V = H^1(\Omega)$) of functions and where the (bi)linear forms $a(\cdot,\cdot\,; x)$ and $\ell(\cdot,; x)$ are defined by means of integrals and of partial derivatives. In the simplified case where $V = \mathbb{R}^N$, $a(u(x), v\,; x) = v^T A(x)u(x)$ and $\ell(v\,; x) = v^T b(x)$, the variational equation is equivalent to the algebraic system (1). For the sake of simplicity we restrict our analysis to the algebraic setting. However all the techniques presented in this note can be extended to functions spaces (instead of algebraic spaces) using variational formulations.

[*]olivier.zahm@inria.fr

1 Galerkin projections on reduced spaces

The basic idea of *projection-based methods* is to build a low-dimensional subspace

$$V_r = \mathrm{span}\{v_1, \dots, v_r\} \subset \mathbb{R}^N, \qquad r \ll N,$$

which is able to "reproduce well" the solution $u(x)$ for any $x \in \mathcal{P}$. For instance, the basis vectors v_i's could be snapshots of the solution, meaning $v_i = u(x^{(i)})$ for some parameter $x^{(i)} \in \mathcal{P}$. Of course this is not the only possibility, as we will see later in Sections 2 and 3. Once the reduced space V_r is built, we seek an approximation of the form of

$$\widetilde{u}_r(x) = \sum_{i=1}^{r} v_i \lambda_i(x) = \mathbf{V}_r \lambda(x) \ \in V_r \tag{3}$$

for some vector $\lambda(x) \in \mathbb{R}^r$, where $\mathbf{V}_r = [v_1, \dots, v_r] \in \mathbb{R}^{N\times r}$ denotes the matrix containing the basis $\{v_1, \dots, v_r\}$. An approximation as in (3) can be interpreted as a low-rank tensor approximation in the canonical format, see [4]. In practice it is always better to orthogonalize the basis so that $\mathbf{V}_r^T\mathbf{V}_r = I_r$. The classical way to define $\lambda(x)$, and thus $\widetilde{u}_r(x)$, is to use *Galerkin projection*. Galerkin projection means we impose some orthogonality conditions on the residual of Equation (1) to an approximation to $u(x)$. The standard approach it to define $\lambda(x)$ as the (unique) vector such that the residual $A(x)\widetilde{u}_r(x) - b$ is orthogonal to V_r. This is equivalent to $\mathbf{V}_r^T(A(x)\widetilde{u}_r(x) - b) = 0$, or to

$$\Bigg(\underbrace{\mathbf{V}_r^T A(x) \mathbf{V}_r}_{=\widetilde{A}(x)} \Bigg) \lambda(x) = \Bigg(\underbrace{\mathbf{V}_r^T b(x)}_{=\widetilde{b}(x)} \Bigg). \tag{4}$$

The cost for solving the above *reduced system* is $\mathcal{O}(r^3)$, which is computationally much cheaper than solving the original system $A(x)u(x) = b$ when $r \ll N$. Once $\lambda(x)$ is computed, we reassemble $\widetilde{u}_r(x) = \mathbf{V}_r\lambda(x)$ and compute the quantity of interest

$$\widetilde{Y}(x) = q(x)^T \widetilde{u}(x).$$

Algorithm 1: Projection-based approximation

```
Offline phase
  | Build the reduced space $\mathbf{V}_r$
end
Online phase: given a new parameter value $x \in \mathcal{P}$ do:
  | Assemble $\widetilde{A}(x) = \mathbf{V}_r^T A(x)\mathbf{V}_r$ and $\widetilde{b}(x) = \mathbf{V}_r^T b(x)$
  | Solve the reduced system $\lambda(x) = \widetilde{A}(x)^{-1}\widetilde{b}(x)$
  | Rebuild the approximation $\widetilde{u}_r(x) = \mathbf{V}_r\lambda(x)$
  | Return: the quantity of interest $Y(x) = q^T\widetilde{u}_r(x)$
end
```

1.1 *A priori* error bound

The goal of *a priori* error analysis is to give the guaranty that, even before we compute it, the approximation *will be* a good/suitable approximation. In the present context, we want to guarantee that the Galerkin projection (4) is a good way to construct the approximation $\widetilde{u}_r(x)$ as in (3). This is given by the Céa's Lemma.

Lemma 1.1 (Céa's Lemma)**.** Denote by $\|\cdot\| = \sqrt{(\cdot)^T(\cdot)}$ the canonical norm of $\mathbb{R}^N$. Assume the matrix $A(x)$ is coercive, meaning

$$\alpha_c(x) = \inf_{v\in\mathbb{R}^N} \frac{v^T A(x) v}{\|v\|^2} > 0. \tag{5}$$

Then the Galerkin projection $\widetilde{u}_r(x)$ defined by (3) and (4) satisfies

$$\|u(x) - \widetilde{u}_r(x)\| \leq \frac{\beta(x)}{\alpha_c(x)} \min_{v_r\in V_r} \|u(x) - v_r\|, \tag{6}$$

where $\beta(x)$ is the continuity constant of $A(x)$ defined by $\beta(x) = \sup_{v\in\mathbb{R}^N} \|A(x)v\|/\|v\|$.

Inequality (6) shows that the error $\|u(x) - \widetilde{u}_r(x)\|$ is no greater than a constant times $\min_{v_r\in V_r} \|u(x) - v_r\|$, which is the smallest error we can achieve when approximating $u(x)$ by an element in V_r. This ensures the stability of the Galerkin projection.

Proof of Lemma 1.1. For simplicity we omit the dependence in x and we write A, b, u,... instead of $A(x)$, $b(x)$, $u(x)$,... By (5) we have that $\|v\|^2 \leq \alpha_c^{-1} v^T A v$ for any $v \in \mathbb{R}^N$. In particular for $v = u - \widetilde{u}_r$ we have

$$\|u - \widetilde{u}_r\|^2 \leq \frac{1}{\alpha_c}(u - \widetilde{u}_r)^T A (u - \widetilde{u}_r) = \frac{1}{\alpha_c}(u - \widetilde{u}_r)^T (b - A\widetilde{u}_r).$$

Next, for any $v_r \in V_r$ we can write

$$\begin{aligned}
\|u - \widetilde{u}_r\|^2 &\leq \frac{1}{\alpha_c}\left((u - v_r)^T (b - A\widetilde{u}_r) + \underbrace{(\overbrace{v_r - \widetilde{u}_r}^{\in V_r})^T (b - A\widetilde{u}_r)}_{=0 \text{ by } (4)} \right) \\
&\overset{\text{C.S.}}{\leq} \frac{1}{\alpha_c} \|u - v_r\| \, \|b - A\widetilde{u}_r\| \\
&\leq \frac{\beta}{\alpha_c} \|u - v_r\| \, \|u - \widetilde{u}_r\|,
\end{aligned}$$

where, for the last inequality, we used the definition of β. Taking the infimum over $v_r \in V_r$ gives (6) and concludes the proof. □

1.2 Offline-online efficiency

Algorithm 1 can be improved by precomputing some quantities in the *offline* phase and to keep in memory those quantities in order to speedup the *online* phase. The key is to assume that $A(x)$, $b(x)$ and $q(x)$ admit the so-called *affine parametric decomposition*

$$A(x) = \sum_{k=1}^{m_a} \Psi_k^A(x)\, A_k, \qquad b(x) = \sum_{k=1}^{m_b} \Psi_k^b(x)\, b_k, \qquad q(x) = \sum_{k=1}^{m_q} \Psi_k^q(x)\, q_k, \tag{7}$$

with $m_a, m_b, m_q \ll N$. Here, the functions Ψ_k^A, Ψ_k^b, $\Psi_k^q : \mathcal{P} \to \mathbb{R}$ are scalar valued functions and where $A_k \in \mathbb{R}^{N\times N}$, $b_k \in \mathbb{R}^N$ and $q_k \in \mathbb{R}^N$ are independent on x. The reduced system (4) can be written as

$$\left(\sum_{k=1}^{m_A} \Psi_k^A(x)\, \underbrace{\mathbf{V}_r^T A_k \mathbf{V}_r}_{=\widetilde{A}_k} \right) \lambda(x) = \left(\sum_{k=1}^{m_b} \Psi_k^b(x)\, \underbrace{\mathbf{V}_r^T b_k}_{=\widetilde{b}_k} \right), \tag{8}$$

and the quantity of interest $\widetilde{Y}(x)$ as

$$\widetilde{Y}(x) = q(x)^T \widetilde{u}(x) = q(x)^T \mathbf{V}_r \lambda(x) = \left(\sum_{k=1}^{m_q} \Psi_k^q(x)\, \underbrace{\mathbf{V}_r^T q_k}_{=\widetilde{q}_k} \right)^T \lambda(x). \tag{9}$$

By precomputing and storing in memory the reduced quantities $\widetilde{A}_k \in \mathbb{R}^{r\times r}$, $\widetilde{b}_k \in \mathbb{R}^r$ and $\widetilde{q}_k \in \mathbb{R}^r$ during the offline phase, the online phase then consists in assembling and solving the reduced system (8) and in computing $\widetilde{Y}(x)$ according to (9)

Algorithm 2: Projection-based approximation (online-efficient)

```
Requires: Affine decomposition (7) of $A(x), b(x)$ and $q(x)$.
Offline phase
    Build the reduced space $\mathbf{V}_r$
    Pre-assemble all the reduced matrices $\widetilde{A}_k = \mathbf{V}_r^T A_k \mathbf{V}_r$
    Pre-assemble all the right-hand sides $\widetilde{b}_k = \mathbf{V}_r^T b_k$
    Pre-assemble all the extractors $\widetilde{q}_k = \mathbf{V}_r^T q_k$
end
Online phase: given a new parameter value $x \in \mathcal{P}$ do:
    Assemble the reduced matrix $\widetilde{A}(x) = \sum_{k=1}^{m_a} \Psi_k^a(x)\widetilde{A}_k$
    Assemble the reduced vector $\widetilde{b}(x) = \sum_{k=1}^{m_b} \Psi_k^b(x)\widetilde{b}_k$
    Solve the reduced system $\lambda(x) = \widetilde{A}(x)^{-1}\widetilde{b}(x)$
    Assemble the reduced vector $\widetilde{q}(x) = \sum_{k=1}^{m_q} \Psi_k^q(x)\widetilde{q}_k$
    Return: the quantity of interest $Y(x) = \widetilde{q}(x)^T \lambda(x)$
end
```

Remark 1.2. Any parametrized matrix $A(x)$ admits an affine decomposition as in (7) with, for instance, $\{A_k\}_{k\geq 1}$ being the canonical basis of $\mathbb{R}^{N\times N}$. But with $m_a = N^2 \gg 1$ terms, the strategy (8) becomes inefficient. This is why we need (7) with $m_a \ll N$.

2 The Reduced Basis method

The basic idea of the *Reduced Basis method* (RB) is to construct V_r by means of snapshots of the solution

$$V_r = \text{span}\{u(x^{(1)}), \dots, u(x^{(r)})\},$$

where the parameters $x^{(1)}, \dots, x^{(r)}$ are constructed one after the other in a *greedy* way. The algorithm proceeds as follow. Once the first r parameters have been computed, we seek the next evalutation point $x^{(r+1)}$ where the residual norm is the largest:

$$x^{(r+1)} \in \arg\max_{x \in \mathcal{P}} \|A(x)\widetilde{u}_r(x) - b(x)\|. \tag{10}$$

Here $\widetilde{u}_r(x)$ is the Galerkin projection of $u(x)$ on V_r, see Section 1. Then we compute the snapshot $u(x^{(r+1)})$ and we enrich the reduced basis $V_{r+1} = V_r + \text{span}\{u(x^{(r+1)})\}$. In practice the parameter set $\mathcal{P}$ in (10) is replaced by a finite *training set* $\mathcal{P}^{\text{train}} = \{X^{(1)}, \dots, X^{(K)}\}$, where $X^{(k)} \sim \mu$ are independent draws of the parameter. This permits us to solve approximately the max problem (10), which is otherwise hard to solve when $\text{card}\{\mathcal{P}\} = \infty$.

Notice that the residual norm $\|A(x)\widetilde{u}_r(x) - b(x)\|$ is a cheap estimation of the expensive-to-compute error $\|u(x) - \widetilde{u}_r(x)\|$. Thus, at each iteration, the RB method tries to improve the current approximation in the region where the error is the largest. In that sense it is a methods which, contrarily to the POD, targets the L^∞ error (the sup-error).

The RB method is widely used in the literature. It is popular because (i) it requires a few number of solution evaluations to build the reduced space and (ii) it comes with an error estimator (the residual norm) which allows certifying the quality of the approximation $\widetilde{u}_r(x)$ and (iii) convergence rate in r are proven to be *independent on* N for many applications [1].

Algorithm 3: RB construction of V_r

```
Requires: Reduced dimension $r$, number of training points $K$
Draw $K$ random points $X^{(1)}, \dots, X^{(K)}$ in $\mathcal{P}$ and let $\mathcal{P}^{\text{train}} = \{X^{(1)}, \dots, X^{(K)}\}$
Pick a $x^{(1)} \in \mathcal{P}^{\text{train}}$ and initialize $\mathbf{V}_1 = [u(x^{(1)})] \in \mathbb{R}^{N \times 1}$
for $i = 1$ to $r - 1$ do
    forall $x \in \mathcal{P}^{train}$ do
        Compute the reduced solution $\widetilde{u}_i(x)$ using $\mathbf{V}_i$
        Compute the residual $R(x) = \|A(x)\widetilde{u}_i(x) - b(x)\|$
    end
    Find $x^{(i+1)} \in \mathcal{P}^{\text{train}}$ the maximizer of $R(x)$
    Compute the solution $u(x^{(i+1)})$
    Update the matrix $\mathbf{V}_{i+1} = [\mathbf{V}_i \ , \ u(x^{(i+1)})] \in \mathbb{R}^{N \times (i+1)}$
    Orthogonalize the columns of $\mathbf{V}_{i+1}$
end
Return: the columns $v_1, \dots, v_r$ of $\mathbf{V}_r$.
```

Remark 2.1. Algorithm 3 requires to prescribe the reduced dimension r. Another option would be to prescribe a desired tolerance ε and to stop the iterations in i as soon as $R(x) \leq \varepsilon$ holds for all $x \in \mathcal{P}^{\text{train}}$.

Remark 2.2. If $A(x)$ and $b(x)$ admit affine decomposition, the complexity for computing $\|A(x)\widetilde{u}_r(x) - b(x)\|$ can be significantly reduced. The idea is to use similar techniques as in Section (1) to express the residual in terms of matrices/vectors of size r instead of size N.

Exercise 1. Show

$$\alpha(x)\|\widetilde{u}_r(x) - u(x)\| \leq \|A(x)\widetilde{u}_r(x) - b(x)\| \leq \beta(x)\|\widetilde{u}_r(x) - u(x)\|,$$

for any $x \in \mathcal{P}$, where

$$\alpha(x) = \min_{v\in\mathbb{R}^N} \max_{v\in\mathbb{R}^N} \frac{v^T A(x) w}{\|v\|\,\|w\|} \quad \text{and} \quad \beta(x) = \max_{v\in\mathbb{R}^N} \max_{v\in\mathbb{R}^N} \frac{v^T A(x) w}{\|v\|\,\|w\|}. \tag{11}$$

Why is this result important? How $\alpha(x)$ relates to $\alpha_c(x)$ defined in (5)?

Exercise 2. How $\alpha(x)$ and $\beta(x)$ defined in (11) are related to the singular values of $A(x)$?

3 The Proper Orthogonal Decomposition

The *Proper Orthogonal Decomposition* (POD) is a technique to build the reduced space $V_r = \text{span}\{v_1, \ldots, v_r\}$ which targets an L^2-optimal basis. Given a probability measure μ on the parameter set $\mathcal{P}$, we assume that $\mathbb{E}(\|u(X)\|^2) < \infty$ where $X \sim \mu$. The function $u : x \mapsto u(x)$ then admits a *Singular Value Decomposition* (SVD),

$$u(x) = \sum_{i\geq 1} \sigma_i\, u_i\, \phi_i(x), \tag{12}$$

where $\sigma_1 \geq \sigma_2 \geq \ldots \geq 0$ are the singular values, $u_i \in \mathbb{R}^N$ are the left singular vectors and $\phi_i : \mathcal{P} \to \mathbb{R}$ are the right singular vectors. Because of the probabilistic nature of X, the SVD (12) is also called the *Karhunen Loève* decomposition. The vectors u_i and the functions $\phi_i : \mathcal{P} \to \mathbb{R}$ are orthogonal in the following sense: for any $i, j \geq 1$ we have $u_i^T v_j = \delta_{i,j}$ and $\mathbb{E}(\phi_i(X)\phi_j(X)) = \delta_{i,j}$. By the Eckart-Young theorem, the truncated SVD yields L^2-optimal approximation:

$$\mathbb{E}\left(\left\|u(X) - \sum_{i=1}^{r} \sigma_i\, u_i\, \phi_i(X)\right\|^2\right) = \min_{\substack{\lambda_i:\mathcal{P}\to\mathbb{R}\\ v_i\in\mathbb{R}^N}} \mathbb{E}\left(\left\|u(X) - \sum_{i=1}^{r} v_i \lambda_i(X)\right\|^2\right) = \sum_{i\geq r+1} \sigma_i^2. \tag{13}$$

Thus, a good choice for the reduced basis $\{v_1, \dots, v_r\}$ is to take the first r left singular vectors of u, meaning $v_1 = u_1, \dots, v_r = u_r$. In order to have access to the u_i's we introduce the non-centered covariance matrix of $u(X)$ defined by $\mathbb{E}(u(X)u(X)^T) \in \mathbb{R}^{N\times N}$. This matrix satisfies

$$\begin{aligned}
\mathbb{E}\left(u(X)u(X)^T\right) &= \mathbb{E}\left(\Big(\sum_{i\geq 1} \sigma_i\, u_i\, \phi_i(X)\Big)\Big(\sum_{j\geq 1} \sigma_j\, u_j\, \phi_j(X)\Big)^T\right) \\
&= \sum_{i,j\geq 1} \sigma_i \sigma_j\, u_i u_j^T\, \underbrace{\mathbb{E}\left(\phi_i(X)\phi_j(X)\right)}_{=\delta_{i,j}} \\
&= \sum_{i\geq 1} \sigma_i^2 u_i u_i^T.
\end{aligned}$$

The first eigenvectors of $\mathbb{E}(u(X)u(X)^T)$ are then exactly the first left singular vectors of u. To numerically get the u_i's, we consider a *Monte Carlo approximation* of $\mathbb{E}(u(X)u(X)^T)$ in which the expectation is replaced by a K-sample average

$$\mathbb{E}(u(X)u(X)^T) \approx \widehat{\mathrm{Cov}}(u(X)) = \frac{1}{K}\sum_{k=1}^{K} u(X^{(k)})u(X^{(k)})^T, \qquad X^{(k)} \overset{\text{i.i.d.}}{\sim} \mu.$$

The eigendecomposition of $\widehat{\mathrm{Cov}}(u(X)) = \sum_{i\geq 1} \hat{\sigma}_i^2 \hat{u}_i \hat{u}_i^T$ provides estimates $\hat{u}_i$ and $\hat{\sigma}_i^2$ of the eigenvectors and eigenvalues of $\mathbb{E}(u(X)u(X)^T)$.

Algorithm 4: POD construction of V_r

```
**Requires:** Reduced dimension $r$, number of snapshots $K$
Draw $K$ i.i.d samples $X^{(1)}, \dots, X^{(K)} \sim \mu$
Compute the snapshots $u(X^{(1)}), \dots, u(X^{(K)})$
Assemble $\widehat{\mathrm{Cov}}(u(X))$
Compute the eigendecomposition $\widehat{\mathrm{Cov}}(u(X)) = \sum_{i\geq 1} \hat{\sigma}_i^2 \hat{u}_i \hat{u}_i^T$
**Return:** the $r$ vectors $v_1 = \hat{u}_1, \dots, v_r = \hat{u}_r$.
```

Remark 3.1. Algorithm 4 requires to prescribe the reduced dimension r. Another option would be to prescribe a desired tolerance $\varepsilon \geq 0$ and to chose adaptively r via relation (13) to make sure the L^2-error $(\mathbb{E}\|u(X) - \sum_{i=1}^r \sigma_i u_i \phi_i(X)\|^2)^{1/2} \approx (\sum_{i\geq r+1} \hat{\sigma}_i^2)^{1/2}$ is below ε.

Remark 3.2. The matrix of snapshots

$$\mathbf{U} = \frac{1}{\sqrt{K}}[u(X^{(1)}), \dots, u(X^{(K)})] \quad \in \mathbb{R}^{N\times K},$$

is such that $\mathbf{U}\mathbf{U}^T = \widehat{\mathrm{Cov}}(u(X))$. One can easily show that the first left singular vectors of $\mathbf{U}$ are the first eigenvectors of $\widehat{\mathrm{Cov}}(u(X))$. Therefore it is not necessary to assemble the (possibly huge!) matrix $\widehat{\mathrm{Cov}}(u(X))$ to compute its eigendecomposition. Instead, a truncated SVD of $\mathbf{U} \approx \sum_{i=1}^r \hat{\sigma}_i \hat{u}_i \hat{v}_i^T$ provides the first eigenpairs $(\hat{\sigma}_i^2, \hat{u}_i)$ of $\widehat{\mathrm{Cov}}(u(X))$.

References

[1] P. Binev, A. Cohen, W. Dahmen, R. DeVore, G. Petrova, and P. Wojtaszczyk, *Convergence rates for greedy algorithms in reduced basis methods*, SIAM journal on mathematical analysis, 43 (2011), pp. 1457–1472.

[2] N. N. Cuong, K. Veroy, and A. T. Patera, *Certified real-time solution of parametrized partial differential equations*, in Handbook of Materials Modeling, Springer, 2005, pp. 1529–1564.

[3] A. Janon, *Reduced-basis solutions of affinely-parametrized linear partial differential equations*, (2010).

[4] A. Nouy, *Low-rank methods for high-dimensional approximation and model order reduction*, in Model reduction and approximation, P. Benner, A. Cohen, M. Ohlberger, and K. Willcox, eds., SIAM, Philadelphia, PA, 2017, pp. 171–226.