Figu

Extension to Non-Affine Problems

Christophe Prud'homme
prudhomme@unistra.fr

December 11, 2025

CeMosis - http://www.cemosis.fr
IRMA
Université de Strasbourg

Empirical Interpolation Method

Context and Motivation

Figu

We are now interested in the case non affine parametric dependence: The an offline/online decomposition relies on that assumption.

Solution: we recover an approximate affine expansion by means of the empirical interpolation method (EIM).

We shall discuss an alternative called **discrete empirical interpolation method (DEIM)**.

We consider linear problems but it will apply as well to non-linear problems.

EIM i

Figu

Affine decomposition is crucial for the offline/online decomposition for the RB method.

$$a(u,v;\mu) = \sum_{q=1}^{Q} \underbrace{\theta^q(\mu)}_{\text{parameter dependent coefficients}} \underbrace{a^q(u,v)}_{\text{parameter independent matrices}}$$

$$\ell(v;\mu) = \sum_{q=1}^{Q^f} \theta^q(\mu)\ell^q(v)$$

EIM ii

if it is not possible to write the problem in this form, eg. $\ell(v;\mu) = \int_\Omega g(x;\boldsymbol{\mu})v$, we can recover an approximate affine expansion by means of the empirical interpolation method (EIM), e.g.

$$g(x;\boldsymbol{\mu}) = g_M(x;\boldsymbol{\mu}) + e_{EIM}(x;\boldsymbol{\mu}) = \sum_{m=1}^{M} \gamma_m(\boldsymbol{\mu})\rho_m(x) + e_{EIM}(x;\boldsymbol{\mu}) \quad (1)$$

and we require then for a given tolerance

$$\varepsilon_M(\boldsymbol{\mu}) = \|e_{EIM}(\cdot;\boldsymbol{\mu})\|_{L^\infty(\Omega)} \leq \varepsilon \quad \forall \boldsymbol{\mu} \in \mathcal{D} \quad (2)$$

$$\ell_M(v;\boldsymbol{\mu}) = \int_\Omega g_M(x;\boldsymbol{\mu}) v d\Omega = \sum_{m=1}^{M} \gamma_m(\boldsymbol{\mu}) \int_\Omega \rho_m(x) v d\Omega. \quad (3)$$

EIM iii

Figu

We recover the affine expansion provided we set $Q_f = M$ and

$$\Theta_f^m(\boldsymbol{\mu}) = \gamma_m(\boldsymbol{\mu}), \quad \ell^m(v) = \int_\Omega \rho_m(\mathrm{x}) v d\Omega, \quad 1 \leq m \leq M.$$

We now turn to an efficient interpolation technique for $\boldsymbol{\mu}$-dependent functions like $g(\mathrm{x}; \boldsymbol{\mu})$ and analyzing the impact of this further approximation on the RB methods

EIM v.s. Polynomial Interpolation i

Figu

We denote $\mathcal{G} = \{g(\cdot;\boldsymbol{\mu}), \boldsymbol{\mu} \in \mathcal{D}\} \subset C^0(\bar{\Omega})$

The empirical interpolation method (EIM) relies on the use of basis functions built by sampling g at a suitably selected set of points in $\mathcal{D}$ instead of using predefined basis functions (not exploiting the coupling of $\boldsymbol{x}$ and $\boldsymbol{\mu}$) defined on a fixed domain such as Gauss-Lobatto interpolation.

EIM v.s. Polynomial Interpolation ii

- introduced in Barrault, M., Maday, Y., Nguyen, N.C., Patera, A.T.: An empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. C. R. Math. Acad. Sci. Paris 339(9), 667-672 (2004) to treat non-affine problems, eg Grepl, M.A., Maday, Y., Nguyen, N.C., Patera, A.T.: Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations. ESAIM Math. Modelling Numer. Anal. 41(3), 575-605 (2007)
- more general scope see Maday, Y., Nguyen, N.C., Patera, A.T., Pau, G.S.H.: A general multipurpose interpolation procedure: the magic points. Commun. Pure Appl. Anal. 8(1), 383-404 (2009)
- no predefined basis functions
- adaptive and hierachical interpolation, no need to reconstrut the basis functions when adding a new point
- exponential convergence rate

Figu

Empirical Interpolation i

The purpose of EIM is to find approximations to elements of $\mathcal{G}$ through an operator $\mathcal{I}^{\mathrm{x}}_M$ that interpolates the function $g(\cdot;\boldsymbol{\mu})$ at some carefully selected points in Ω.

Figu

Given an interpolatory system defined by a set of basis functions $\{\rho_1,\ldots,\rho_M\}$ (linear combination of particular snapshots $g\left(\cdot;\boldsymbol{\mu}^1_{EIM}\right),\ldots,g\left(\cdot;\boldsymbol{\mu}^M_{EIM}\right)$) and interpolation points $T_M=\left\{\mathrm{t}^1,\ldots,\mathrm{t}^M\right\}\subset\bar{\Omega}-$ commonly referred to as magic points $-$ the interpolant $\mathcal{I}^{\mathrm{x}}_M g(\cdot;\boldsymbol{\mu})$ of $g(\cdot;\boldsymbol{\mu})$ with $\boldsymbol{\mu}\in\mathcal{D}$ admits the separable expansion

$$\mathcal{I}^{\mathrm{x}}_M g(\mathrm{x};\boldsymbol{\mu})=\sum_{j=1}^{M}\gamma_j(\boldsymbol{\mu})\rho_j(\mathrm{x}),\quad \mathrm{x}\in\Omega \tag{4}$$

Empirical Interpolation ii

and satisfies the M interpolation constraints

$$\mathcal{I}^{\times}_M g\left(\mathrm{t}^i;\boldsymbol{\mu}\right) = g\left(\mathrm{t}^i;\boldsymbol{\mu}\right), \quad i = 1,\dots,M \tag{5}$$

Indeed, (**??**) yields the following linear system to solve

$$\sum_{j=1}^{M} \rho_j\left(\mathrm{t}^i\right)\gamma_j(\boldsymbol{\mu}) = g\left(\mathrm{t}^i;\boldsymbol{\mu}\right), \quad i = 1,\dots,M \tag{6}$$

that is, in matrix form

$$\mathbb{B}_M\boldsymbol{\gamma}(\boldsymbol{\mu}) = \mathrm{g}_M(\boldsymbol{\mu}) \quad \forall\boldsymbol{\mu}\in\mathcal{D} \tag{7}$$

Figu

Empirical Interpolation iii

Figu

where

$$(\mathbb{B}_M)_{ij} = \rho_j\left(\mathrm{t}^i\right), \quad (\boldsymbol{\gamma}(\boldsymbol{\mu}))_j = \gamma_j(\boldsymbol{\mu}), \quad (\mathrm{g}_M(\boldsymbol{\mu}))_i = g\left(\mathrm{t}^i; \boldsymbol{\mu}\right), \quad i,j = 1,\dots,M. \tag{8}$$

We need conditions ensuring that $\mathbb{B}_M$ is invertible.

EIM Greedy Algorithm i

The construction of the basis functions yielding the approximation space $X_M = \text{span}\{\rho_1, \dots, \rho_M\}$ and interpolation points $T_M = \left\{\mathrm{t}^1, \dots, \mathrm{t}^M\right\}$ is based on a greedy algorithm.

Figu

This procedure provides also a sample of parameter points $S_M = \left\{\boldsymbol{\mu}^1_{EIM}, \dots, \boldsymbol{\mu}^M_{EIM}\right\}$, needed to construct the basis functions $\rho_i(\mathrm{x}), i = 1, \dots, M$.

To start, let us choose our first sample point as

$$\boldsymbol{\mu}^1_{EIM} = \arg\max_{\boldsymbol{\mu}\in\mathcal{D}} \|g(\cdot;\boldsymbol{\mu})\|_{L^\infty(\Omega)},$$

define $S_1 = \left\{\boldsymbol{\mu}^1_{EIM}\right\}$ and the first generating function as

$$\xi_1(\mathrm{x}) = g\left(\mathrm{x};\boldsymbol{\mu}^1_{EIM}\right).$$

EIM Greedy Algorithm ii

Concerning the interpolation nodes, we first set

$$\mathrm{t}^1 = \arg\max_{x \in \bar{\Omega}} |\xi_1(x)|, \quad T_1 = \left\{\mathrm{t}^1\right\};$$

then, we define the first basis function as

$$\rho_1(x) = \xi_1(x)/\xi_1\left(\mathrm{t}^1\right),$$

and set $X_1 = \operatorname{span}\{\rho_1\}$.

Finally, we set the initial interpolation matrix

$$(\mathbb{B}_M)_{11} = \rho_1\left(\mathrm{t}^1\right) = 1.$$

At this stage, the available information allows to define the interpolant as the only function colinear with ρ_1 that coincides with g at t^1, that is

EIM Greedy Algorithm iii

Figu

$\mathcal{I}_1^{\mathrm{x}} g(\mathrm{x};\boldsymbol{\mu}) = g\left(\mathrm{t}^1;\boldsymbol{\mu}\right)\rho_1(\mathrm{x})$. Note that the first interpolation point t^1 is the point where the first basis function attains its maximum.

EIM Greedy Algorithm iv

At the m-th step, $m = 1, \dots, M-1$, given the (nested) set $T_m = \left\{ \mathrm{t}^1, \dots, \mathrm{t}^m \right\}$ of interpolation points and the set $\{\rho_1, \dots, \rho_m\}$ of basis functions, we select as $(m+1)$ th generating function the snapshot which is the worst approximated by the current interpolant.

In other words, we select the snapshot which maximizes the error between g and $\mathcal{I}_m^{\mathrm{x}} g$

$$\begin{aligned} \boldsymbol{\mu}_{EIM}^{m+1} &= \arg\max_{\boldsymbol{\mu} \in \mathcal{D}} \| g(\cdot; \boldsymbol{\mu}) - \mathcal{I}_m^{\mathrm{x}} g(\cdot; \boldsymbol{\mu}) \|_{L^\infty(\Omega)}, \\ \xi_{m+1}(\mathrm{x}) &= g\left(\mathrm{x}; \boldsymbol{\mu}_{EIM}^{m+1}\right). \end{aligned} \tag{9}$$

We then set $S_{m+1} = S_m \cup \left\{ \boldsymbol{\mu}_{EIM}^{m+1} \right\}$.

EIM Greedy Algorithm v

To choose the $(m+1)$-th interpolation point, we first evaluate the residual

$$r_{m+1}(\mathrm{x}) = \xi_{m+1}(\mathrm{x}) - \mathcal{I}_m^{\mathrm{x}} \xi_{m+1}(\mathrm{x})$$

by solving the linear system

$$\sum_{j=1}^{m} \rho_j\left(\mathrm{t}^i\right)\gamma_j = \xi_{m+1}\left(\mathrm{t}^i\right), \quad i = 1, \ldots, m$$

to characterize the interpolant $\mathcal{I}_m^{\mathrm{x}} \xi_{m+1}$; then, we set

$$\mathrm{t}^{m+1} = \arg\max_{\mathrm{x}\in\bar{\Omega}} |r_{m+1}(\mathrm{x})| \tag{10}$$

Figu

EIM Greedy Algorithm vi

Figu

that is, that point of Ω where ξ_{m+1} is worst approximated. Finally, we define the new basis function as

$$\rho_{m+1}(\mathsf{x}) = \frac{\xi_{m+1}(\mathsf{x}) - \mathcal{I}_m^{\mathsf{x}}\xi_{m+1}(\mathsf{x})}{\xi_{m+1}\left(\mathsf{t}^{m+1}\right) - \mathcal{I}_m^{\mathsf{x}}\xi_{m+1}\left(\mathsf{t}^{m+1}\right)} = \frac{r_{m+1}(\mathsf{x})}{r_{m+1}\left(\mathsf{t}^{m+1}\right)}$$

and we set $X_{m+1} = \operatorname{span}\{\rho_i, i = 1, \dots, m+1\}$.

EIM algorithm i

Figu

The whole procedure is performed until a given tolerance $\varepsilon_{\mathrm{EIM}}$ is reached, or a given number $M_{\max}$ of terms is computed.

```
Input: max number of iterations $M_{\max}$, tolerance $\varepsilon$
Output: basis functions $\{\rho_1(\mathbf{x}),\dots,\rho_M(\mathbf{x})\}$, interpolation points $\{\mathbf{t}^1,\dots,\mathbf{t}^M\}$
1: $M = 0,\ e_0 = \varepsilon + 1,\ \mathscr{I}_0^{\mathbf{x}} g(\mathbf{x};\boldsymbol{\mu}) = 0,$
2: $\boldsymbol{\mu}^1 = \arg\max_{\boldsymbol{\mu}\in\mathscr{P}} \|g(\cdot,\boldsymbol{\mu})\|_{L^\infty(\Omega)}$
3: while $M < M_{\max}$ and $e_M > \varepsilon$
4:     $M \leftarrow M + 1$
5:     $r(\mathbf{x}) = g(\mathbf{x},\boldsymbol{\mu}^M) - \mathscr{I}_{M-1}^{\mathbf{x}} g(\mathbf{x},\boldsymbol{\mu}^M)$
6:     $\mathbf{t}^M = \arg\max_{\mathbf{x}\in\overline{\Omega}} |r(\mathbf{x})|$
7:     $\rho_M(\mathbf{x}) = r(\mathbf{x})/r(\mathbf{t}^M)$
8:     $[e_M, \boldsymbol{\mu}^{M+1}] = \arg\max_{\boldsymbol{\mu}\in\mathscr{P}} \|g(\cdot,\boldsymbol{\mu}) - \mathscr{I}_M^{\mathbf{x}} g(\cdot,\boldsymbol{\mu})\|_{L^\infty(\Omega)}$
9: end while
```

EIM: properties i

Remark EIM yields a sequence of hierarchical spaces $X_1 \subset X_2 \subset \dots X_M$, such that the interpolation is exact for any $v \in X_M$ - that is,

$$\mathcal{I}_M^x v = v \quad \forall v \in X_M$$

provided that $\dim(X_M) = M$ and that the matrix $\mathbb{B}_M \in \mathbb{R}^{M\times M}$ is invertible.

We can show that the construction discussed so far yields indeed a set $\{\rho_1, \dots, \rho_M\}$ of linearly independent basis functions.

EIM: properties ii

Figu

Theorem (EIM Properties)
The construction of the interpolation points is well-defined and, for any $M \leq M_{\max} < \dim(\operatorname{span}\{\mathcal{G}\}), X_M = \operatorname{span}\{\rho_1, \dots, \rho_M\} = \operatorname{span}\{\xi_1, \dots, \xi_M\}$ is of dimension M. In addition, $\mathbb{B}_M$ is lower triangular with $(\mathbb{B}_M)_{ii} = 1, i = 1, \dots, M$.

EIM: properties iii

Figu

Proof.

The property $\mathrm{span}\left\{\rho_1,\ldots,\rho_M\right\} = \mathrm{span}\left\{\xi_1,\ldots,\xi_M\right\} = X_M$ directly follows from the construction of the normalized ρ_i 's with respect to the ξ_j 's. Then, we proceed by induction. Clearly, $X_1 = \mathrm{span}\left\{\rho_1\right\}$ has dimension 1 and $\mathbb{B}_1 = 1$ is invertible. Next, lets us assume that $X_{M-1} = \mathrm{span}\left\{\rho_1,\ldots,\rho_{M-1}\right\}$ is of dimension $M-1$. If

- $\mathbb{B}_{M-1}$ is invertible
- $\left|r_M\left(\mathrm{t}^M\right)\right| > 0$

EIM: properties iv

we may form $X_M = \text{span}\{\rho_1, \dots, \rho_M\}$.

To prove that $\mathbb{B}_M$ is invertible, it is enough to observe that

$$(\mathbb{B}_{M-1})_{ij} = \rho_j\left(\mathrm{t}^i\right) = r_j\left(\mathrm{t}^i\right)/r_j\left(\mathrm{t}^j\right), \quad i,j = 1,\dots,M-1. \qquad (11)$$

Then $(\mathbb{B}_{M-1})_{ij} = 0$ if $i<j$, $(\mathbb{B}_{M-1})_{ij} = 1$ if $i=j$, whereas $\left|(\mathbb{B}_M)_{ij}\right| \leq 1$ if $i>j$ since $\mathrm{t}^j = \arg\max_{\mathrm{x}\in\bar{\Omega}} |r_j(\mathrm{x})|, j = 1,\dots,M$, according to (**??**). The matrix $\mathbb{B}_M$ is lower triangular and it is such that $(\mathbb{B}_M)_{ii} = 1, i = 1,\dots,M$, hence it is invertible.

To prove that $\dim(X_M) = M$ (whence $\mathrm{t}^1,\dots,\mathrm{t}^M$ are distinct) we observe that

$$\|r_M\|_{L^\infty(\Omega)} = \|g\left(\cdot;\boldsymbol{\mu}^M_{EIM}\right) - \mathcal{I}^{\mathrm{x}}_{M-1} g\left(\cdot;\boldsymbol{\mu}^M_{EIM}\right)\|_{L^\infty(\Omega)}$$
$$\geq \max_{\boldsymbol{\mu}\in\mathcal{D}} \left\|g(\cdot;\boldsymbol{\mu}) - \mathcal{I}^{\mathrm{x}}_{M-1} g(\cdot;\boldsymbol{\mu})\right\|_{L^\infty(\Omega)} \geq d_{M_{\max}}(\mathcal{G};X),$$

Figu

EIM: properties v

Figu

with

$$d_{M_{\mathbf{max}}}(\mathcal{G};X) = \inf_{\substack{\hat{X}\subset X\\ \dim(\tilde{X})=M_{\max}}} \sup_{\boldsymbol{\mu}\in\mathcal{D}} \inf_{z\in\hat{X}} \|g(\cdot;\boldsymbol{\mu}) - z\|_{L^\infty(\Omega)} > 0$$

since $M_{\max} < \dim(\mathrm{span}\{\mathcal{G}\})$. If $\dim(X_M) \neq M$, we have that $\xi_M \in X_{M-1}$ and thus $\|r_M\|_{L^\infty(\Omega)} = 0$, which provides the contradiction and yields $\dim(X_M) = M$. $\square$

EIM: properties vi

Figu

Note: if $\dim(\text{span}\{\mathcal{G}\}) = M^*$, the algorithm stops after $M = M^*$ iterations. As long as $M \leq M^*$, the previous theorem ensures that the basis functions $\{\rho_1, \ldots, \rho_M\}$ and the snapshots $\{\xi_1, \ldots, \xi_M\}$ span the same space X_M. In particular, it is better to deal with the former since, as resulting from **??**,

$$\rho_i\left(\mathrm{t}^i\right) = 1, \quad i = 1, \ldots, M, \quad \rho_j\left(\mathrm{t}^i\right) = 0, \quad 1 \leq i < j \leq M$$

EIM: Error Analysis i

Figu

We carry out an error analysis of the EIM based on the results found in previous references.

Let us first introduce a set of characteristic (Lagrangian) functions $\{l_i^M \in X_M\}$ to construct the interpolation operator $\mathcal{I}_M^{x}$ in X_M over the set of magic points T_M. For any given M, we can express

$$\mathcal{I}_M^{x} g(x;\boldsymbol{\mu}) = \sum_{i=1}^{M} g\left(t^i;\boldsymbol{\mu}\right) l_i^M(x), \quad l_i^M(x) = \sum_{j=1}^{M} \rho_j(x)\left(\mathbb{B}_M^{-1}\right)_{ji}$$

by definition, $l_i^M\left(t^j\right) = \delta_{ij}, i, j = 1, \ldots, M$. The existence of characteristic functions directly follows from the nonsingularity of the matrix $\mathbb{B}_M$.

EIM: Error Analysis ii

Figu

The a priori error analysis of the EIM involves the Lebesgue constant

$$\Lambda_M = \sup_{x\in\Omega} \sum_{i=1}^{M} \left| l_i^M(x) \right| ;$$

note that Λ_M depends on X_M and the magic points T_M, but is μ-independent.

An upper bound (indeed quite pessimistic) for the Lebesgue constant is $\Lambda_M \leq 2^M - 1$.

EIM: Error Analysis iii

Proposition
For any $g \in \mathcal{G}$, the interpolation error satisfies

$$\varepsilon_M(\boldsymbol{\mu}) := \|g(\cdot;\boldsymbol{\mu}) - \mathcal{I}^{\times}_M g(\cdot;\boldsymbol{\mu})\|_{L^\infty(\Omega)} \leq (1+\Lambda_M) \inf_{g_M \in X_M} \|g(\cdot;\boldsymbol{\mu}) - g_M\|_{L^\infty(\Omega)} \tag{12}$$

The estimate provides a theoretical basis for the stability of the empirical interpolation method. The last term in the right-hand side of (**??**) is referred to as the best approximation of g by elements in X_M in the L^∞-norm.

EIM: Error Analysis iv

Similarly to the convergence result for the greedy RB algorithm, also in the case of the empirical interpolation method it is possible to link the convergence rate of EIM approximations to the Kolmogorov M-width of the manifold $\mathcal{G}$.

EIM: Error Analysis v

Theorem
Assume that $\mathcal{G} \subset X = C^0(\Omega)$*, and that there exists a sequence of nested finite-dimensional spaces* $\mathcal{Z}_1 \subset \mathcal{Z}_2 \dots, \dim(\mathcal{Z}_M) = M$*, and* $\mathcal{Z}_M \subset \operatorname{span}\{\mathcal{G}\}$ *such that there exists* $c > 0$ *and* $\alpha > \log 4$ *with*

$$d(\mathcal{G}, \mathcal{Z}_M) = \sup_{\boldsymbol{\mu}\in\mathcal{D}} \inf_{v_M \in \mathcal{Z}_M} \|g(\cdot;\boldsymbol{\mu}) - v_M\|_X \le c e^{-\alpha M}.$$

Then

$$\sup_{\boldsymbol{\mu}\in\mathcal{D}} \|g(\cdot;\boldsymbol{\mu}) - \mathcal{I}_M^{\mathrm{x}} g(\cdot;\boldsymbol{\mu})\|_{L^\infty(\Omega)} \le c e^{-(\alpha - \log 4)M}.$$

Remark the condition implies in particular that $d_M(\mathcal{G}; X) \le c e^{-\alpha M}$, i.e. $\mathcal{G}$ has an exponentially small Kolmogorov M-width.

A Posteriori Error Estimate i

Figu

We now derive an a posteriori error estimate for the EIM error we proceed as follows.

Given an approximation $g_M(\cdot;\boldsymbol{\mu})$, for $M \leq M_{\max} - 1$, let us define

$$E_M(\mathrm{x};\boldsymbol{\mu}) = \hat{\varepsilon}_M(\boldsymbol{\mu})\rho_{M+1}(\mathrm{x}), \quad \hat{\varepsilon}_M(\boldsymbol{\mu}) = \left|g\left(\mathrm{t}^{M+1};\boldsymbol{\mu}\right) - \mathcal{I}^{\mathrm{x}}_M g\left(\mathrm{t}^{M+1};\boldsymbol{\mu}\right)\right|.$$

A Posteriori Error Estimate ii

Figu

In general, $\varepsilon_M(\boldsymbol{\mu}) \geq \hat{\varepsilon}_M(\boldsymbol{\mu})$.

However, it is possible to show that

Proposition
If $g(\cdot;\boldsymbol{\mu}) \in X_{M+1}$, *then*

1. $g(\mathrm{x};\boldsymbol{\mu}) - \mathcal{I}_M^{\mathrm{x}} g(\mathrm{x};\boldsymbol{\mu}) = \pm E_M(\mathrm{x};\boldsymbol{\mu})$;
2. $\varepsilon_M(\boldsymbol{\mu}) = \hat{\varepsilon}_M(\boldsymbol{\mu})$.

A Posteriori Error Estimate iii

Proof.

If $g(\cdot;\boldsymbol{\mu}) \in X_{M+1}$ there exists $\kappa(\boldsymbol{\mu}) \in \mathbb{R}^{M+1}$ such that $g(\mathrm{x};\boldsymbol{\mu}) - \mathcal{I}_M^{\mathrm{x}} g(\mathrm{x};\boldsymbol{\mu}) = \sum_{i=1}^{M+1} \kappa_i(\boldsymbol{\mu})\rho_i(\mathrm{x})$. Taking $\mathrm{x} = \mathrm{t}^j, j = 1,\dots,M+1$, we get

$$\sum_{i=1}^{M+1} \kappa_i(\boldsymbol{\mu})\rho_i\left(\mathrm{t}^i\right) = g\left(\mathrm{t}^j;\boldsymbol{\mu}\right) - \mathcal{I}_M^{\mathrm{x}} g\left(\mathrm{t}^j;\boldsymbol{\mu}\right),$$

from which we obtain that $(i)\kappa_i(\boldsymbol{\mu}) = 0, i = 1,\dots,M$ since $g\left(\mathrm{t}^i;\boldsymbol{\mu}\right) - \mathcal{I}_M^{\mathrm{x}} g\left(\mathrm{t}^i;\boldsymbol{\mu}\right) = 0, i = 1,\dots,M$ and $\mathbb{B}_M$ is lower triangular, i.e. $\rho_i\left(\mathrm{x}^j\right) = 0$ if $i < j$, and that (ii) $\kappa_{M+1}(\boldsymbol{\mu}) = g\left(\mathrm{t}^{M+1};\boldsymbol{\mu}\right) - \mathcal{I}_M^{\mathrm{x}} g\left(\mathrm{t}^{M+1};\boldsymbol{\mu}\right)$ since $\rho_{M+1}\left(\mathrm{t}^{M+1}\right) = 1$. This concludes the proof of point 1 . Point 2 directly follows since $\|\rho_{M+1}\|_{L^\infty(\Omega)} = 1$. □

A Posteriori Error Estimate iv

Figu

Note that in general $g(\cdot;\boldsymbol{\mu}) \notin X_{M+1}$, so we only have that $\varepsilon_M(\boldsymbol{\mu}) \geq \hat{\varepsilon}_M(\boldsymbol{\mu})$ for any $\boldsymbol{\mu} \in \mathcal{D}$, i.e., $\hat{\varepsilon}_M(\boldsymbol{\mu})$ is a lower bound of the interpolation error in L^∞-norm.

Nevertheless, if $\varepsilon_M(\boldsymbol{\mu}) \to 0$ very fast, we expect the effectivity

$$\eta_M(\boldsymbol{\mu}) = \frac{\hat{\varepsilon}(\boldsymbol{\mu})}{\|g(\cdot;\boldsymbol{\mu}) - \mathcal{I}_M^\times g(\cdot;\boldsymbol{\mu})\|_{L^\infty(\Omega)}}$$

to be close to 1 .

A Posteriori Error Estimate v

Figu

In any case, evaluating the estimator only requires an additional evaluation of $g(\cdot;\boldsymbol{\mu})$ at a single point in Ω. For this reason, we define the one point error estimator as

$$\Delta_M(\boldsymbol{\mu}) = \hat{\varepsilon}_M(\boldsymbol{\mu})$$

corresponding to the interpolation error at the $(M+1)$-th magic point, the one where the residual $r_M(\mathrm{x})$ attains its maximum.

How to Compute the Lebesgue Constant i

Figu

Definition:

$$\Lambda_M = \sup_{x\in\Omega} \sum_{i=1}^{M} |l_i^M(x)|$$

where l_i^M are the Lagrange characteristic functions satisfying $l_i^M(t^j) = \delta_{ij}$.

How to Compute the Lebesgue Constant ii

Step 1: Compute characteristic functions

The characteristic functions are defined by:

$$l_i^M(\mathrm{x}) = \sum_{j=1}^{M} \rho_j(\mathrm{x})(\mathbb{B}_M^{-1})_{ji}$$

In matrix form: $\mathsf{L} = \mathbb{Q}\mathbb{B}_M^{-1}$ where

- $\mathbb{Q} = [\boldsymbol{\rho}_1|\ldots|\boldsymbol{\rho}_M] \in \mathbb{R}^{N_q \times M}$ (basis matrix)
- $\mathbb{B}_M = \mathbb{Q}_{\mathfrak{I}} \in \mathbb{R}^{M \times M}$ (interpolation matrix at magic points)
- $\mathsf{L} \in \mathbb{R}^{N_q \times M}$ with $(\mathsf{L})_{ki} = l_i^M(\mathrm{x}^k)$

Figu

How to Compute the Lebesgue Constant iii

Step 2: Compute Lebesgue constant on discrete grid

Figu

$$\Lambda_M = \max_{k=1,\dots,N_q} \sum_{i=1}^{M} |(\mathsf{L})_{ki}| = \max_{k=1,\dots,N_q} \|\mathsf{L}_{k,:}\|_1$$

Algorithm:

1. Compute $\mathbb{B}_M^{-1}$ (cost: $O(M^3)$, done once)
2. Compute $\mathsf{L} = \mathbb{Q}\mathbb{B}_M^{-1}$ (cost: $O(N_q M^2)$)
3. For each grid point k: compute $s_k = \sum_{i=1}^{M} |L_{ki}|$
4. Return $\Lambda_M = \max_k s_k$

Properties:

- $\Lambda_M \geq 1$ (always)

How to Compute the Lebesgue Constant iv

Figu

- Theoretical bound: $\Lambda_M \leq 2^M - 1$ (pessimistic)
- In practice: often grows like $\log(M)$ or $\sqrt{M}$ for good point selection

Practical Implementation i

Similarly to the (weak) greedy RB algorithm , finding the supremum in (**??**) and (**??**) is not computationally feasible unless an approximation of both Ω and $\mathcal{D}$ is considered.

Figu

For this reason, we introduce:

1. a fine sample $\Xi_{\text{train}}^{EIM} \subset \mathcal{D}$ of cardinality $\left|\Xi_{\text{train}}^{EIM}\right| = n_{\text{train}}^{EIM}$ to train the EIM algorithm ;
2. a discrete approximation $\Omega_h = \left\{\mathrm{x}^k\right\}_{k=1}^{N_q}$ of Ω of dimension N_q. In the finite element context, the points x^i can be for instance the vertices of the computational mesh, or the quadrature points.

Problems (**??**) and (**??**) are now turned into simpler enumeration problems.

Practical Implementation ii

In this setting, we can also provide an algebraic version of the EIM.

```
function $[\mathbb{Q}, \mathfrak{I}] = \text{EIM\_OFFLINE}(\Xi^{EIM}_{\text{train}}, \Omega_h, M_{\max}, \varepsilon_{\text{EIM}})$
    $M = 0,\ e_0 = \varepsilon_{\text{EIM}} + 1$
    $\boldsymbol{\mu}^1 = \arg\max_{\boldsymbol{\mu}\in\Xi^{EIM}_{\text{train}}} \|\mathbf{g}(\boldsymbol{\mu})\|_\infty$
    $\mathbf{r} = \mathbf{g}(\boldsymbol{\mu}^1),\ \mathbb{Q} = [\,]$
    while $M < M_{\max}$ and $e_M > \varepsilon_{\text{EIM}}$
        $M \leftarrow M + 1$
        $i_M = \arg\max_{i=1,\dots,N_q} |\mathbf{r}_i|$
        $\boldsymbol{\rho}_M = \mathbf{r}/\mathbf{r}_{i_M}$
        $\mathbb{Q} \leftarrow \mathbb{Q} \cup \boldsymbol{\rho}_M,\ \mathfrak{I} \leftarrow \mathfrak{I} \cup i_M,$
        $[e_M, \boldsymbol{\mu}^{M+1}] = \arg\max_{\boldsymbol{\mu}\in\Xi^{EIM}_{\text{train}}} \|\mathbf{g}(\boldsymbol{\mu}) - \mathbb{Q}\mathbb{Q}_{\mathfrak{I}}^{-1}\mathbf{g}_{\mathfrak{I}}(\boldsymbol{\mu})\|_\infty$
        $\mathbf{r} = \mathbf{g}(\boldsymbol{\mu}^{M+1}) - \mathbb{Q}\mathbb{Q}_{\mathfrak{I}}^{-1}\mathbf{g}_{\mathfrak{I}}(\boldsymbol{\mu}^{M+1})$
    end while
end function
```

```
function $\boldsymbol{\gamma}(\boldsymbol{\mu}) = \text{EIM\_ONLINE}(\mathbb{Q}_{\mathfrak{I}}, \boldsymbol{\mu}, \{\mathbf{t}^1, \dots, \mathbf{t}^M\})$
    form $\mathbf{g}_{\mathfrak{I}}(\boldsymbol{\mu})$ by evaluating $g(\cdot, \boldsymbol{\mu})$ in the interpolation points $\{\mathbf{t}^1, \dots, \mathbf{t}^M\}$
    solve $\mathbb{Q}_{\mathfrak{I}}\boldsymbol{\gamma}(\boldsymbol{\mu}) = \mathbf{g}_{\mathfrak{I}}(\boldsymbol{\mu})$
end function
```

Figu

Practical Implementation iii

We first introduce the vector representation $\mathrm{g} : \mathcal{D} \to \mathbb{R}^{N_q}$ of $g : \Omega_h \times \mathcal{D} \to \mathbb{R}$, defined as

$$(\mathrm{g}(\boldsymbol{\mu}))_k = g\left(\mathrm{x}^k ; \boldsymbol{\mu}\right), \quad k = 1, \ldots, N_q$$

obtained by evaluating the function g in Ω_h, for any $\boldsymbol{\mu} \in \mathcal{D}$. Then, we denote by $\mathbb{Q} \in \mathbb{R}^{N_q \times M}$ the matrix

$$\mathbb{Q} = [\boldsymbol{\rho}_1 | \ldots | \boldsymbol{\rho}_M]$$

whose columns are the discrete representation of the basis functions $\{\rho_1, \ldots, \rho_M\}$, i.e. $(\mathbb{Q})_{kj} = \rho_j\left(\mathrm{x}^k\right)$. Moreover we denote by $\mathfrak{I} = \{i_1, \ldots, i_M\}$ a set of interpolation indices such that

Figu

Practical Implementation iv

$\{t^1, \dots, t^M\} = \{x^{i_1}, \dots, x^{i_M}\}$. The discrete representation $g_M : \mathcal{D} \to \mathbb{R}^{N_q}$ of the interpolation operator $\mathcal{I}_M^x$ is given by

$$g_M(\boldsymbol{\mu}) = \mathbb{Q}\boldsymbol{\gamma}(\boldsymbol{\mu}) \in \mathbb{R}^{N_q},$$

where $\boldsymbol{\gamma}(\boldsymbol{\mu}) \in \mathbb{R}^M$ is the solution of the following linear system

$$(g_M(\boldsymbol{\mu}))_{i_m} = \sum_{j=1}^{M} \gamma_j(\boldsymbol{\mu}) \left(\boldsymbol{\rho}_j\right)_{i_m} = (g(\boldsymbol{\mu}))_{i_m}, \quad m = 1, \dots, M. \qquad (13)$$

Denoting by $g_{\mathfrak{I}}(\boldsymbol{\mu}) \in \mathbb{R}^M$ the vector whose components are $(g_{\mathfrak{I}}(\boldsymbol{\mu}))_m = (g(\boldsymbol{\mu}))_{i_m}$ for $m = 1, \dots, M$, and noting that the $M \times M$ matrix $\mathbb{B}_M$ can be easily formed by restricting the $N_q \times M$ matrix $\mathbb{Q}$ to the rows $\mathfrak{I}$, i.e. $\mathbb{B}_M = \mathbb{Q}_{\mathfrak{I}}$,

Practical Implementation v

Figu

(**??**) can be written in compact form as

$$\mathbb{Q}_{\mathfrak{I}}\gamma(\boldsymbol{\mu}) = \mathrm{g}_{\mathfrak{I}}(\boldsymbol{\mu}). \tag{14}$$

We finally obtain the following expression for the EIM approximation

$$\mathrm{g}_M(\boldsymbol{\mu}) = \mathbb{Q}_{\mathfrak{I}}^{-1}\mathrm{g}_{\mathfrak{I}}(\boldsymbol{\mu}) \quad \forall \boldsymbol{\mu} \in \mathcal{D}.$$

Practical Implementation vi

Note: that the solution of the dense linear system (**??**) has complexity $O\left(M^2\right)$, since the matrix $\mathbb{Q}_{\mathfrak{I}}$ is lower triangular.

Figu

Remark At each iteration, the algorithm requires to evaluate $\mathrm{g}(\boldsymbol{\mu})$ for $\boldsymbol{\mu} \in \Xi_{\text{train}}^{EIM}$. Should this operation be expensive, one may form and store the (possibly dense) matrix

$$\mathbb{S} = \left[\mathrm{g}\left(\boldsymbol{\mu}^1\right) | \ldots | \mathrm{g}\left(\boldsymbol{\mu}^{n_{\textbf{train}}^{EIM}}\right)\right] \in \mathbb{R}^{N_q \times n_{\textbf{train}}^{EIM}}$$

once and for all before entering the while loop. However, already for moderately large N_q and n_{train}^{EIM}, storing the matrix $\mathbb{S}$ can be quite challenging. For instance, approximating a function defined over a set of $N_q = 4 \cdot 10^5$ points (corresponding to 4 quadrature points for each tetrahedron) using a training set of dimension $n_{\text{train}}^{EIM} = 10^3$, requires to store as much as about 7 GB of data.

DEIM: an alternative

Discrete Empirical Interpolation Method i

An alternative to the EIM for approximating a nonaffinely parametrized function is the so-called discrete empirical interpolation method (DEIM), originally introduced in Chaturantabut, S., Sorensen, D.C.: Nonlinear model reduction via discrete empirical interpolation. SIAM J. Sci. Comput. 32(5), 2737-2764 (2010) .

Similarly to EIM, DEIM approximates a nonlinear function $\mathrm{g} : \boldsymbol{\mu} \in \mathcal{D} \subset \mathbb{R}^P \to \mathrm{g}(\boldsymbol{\mu}) \in \mathbb{R}^{N_q}$ by projection onto a low-dimensional subspace spanned by a basis $\mathbb{Q}$,

$$\mathrm{g}(\boldsymbol{\mu}) \approx \mathrm{g}_M(\boldsymbol{\mu}) = \mathbb{Q}\gamma(\boldsymbol{\mu}), \tag{15}$$

where $\mathbb{Q} = [\boldsymbol{\rho}_1, \dots, \boldsymbol{\rho}_M] \in \mathbb{R}^{N_q \times M}$ and $\gamma(\boldsymbol{\mu}) \in \mathbb{R}^M$ is the corresponding vector of coefficients, with $M \ll N_q$.

Figu

Discrete Empirical Interpolation Method ii

Figu

The difference is on the construction of the basis $\mathbb{Q}$, that is obtained operating a POD on a set of snapshots

$$\mathbb{S} = \left[\mathrm{g}\left(\boldsymbol{\mu}^{1}_{DEIM}\right) | \dots | \mathrm{g}\left(\boldsymbol{\mu}^{n_s}_{DEIM}\right) \right], \quad n_s^{DEIM} > M$$

instead than being embedded in the EIM greedy algorithm.

Note that for both EIM and DEIM the interpolation points are iteratively selected with the same greedy algorithm.

Discrete Empirical Interpolation Method iii

DEIM thus requires to:

1. construct a set of snapshots obtained by sampling $g(\boldsymbol{\mu})$ at values $\boldsymbol{\mu}^i_{DEIM}, i = 1, \dots, n_s$ and apply POD to extract the basis

$$[\boldsymbol{\rho}_1, \dots, \boldsymbol{\rho}_M] = \text{POD}\left(\left[\mathrm{g}\left(\boldsymbol{\mu}^1_{DEIM}\right), \dots, \mathrm{g}\left(\boldsymbol{\mu}^{n_s}_{DEIM}\right)\right], \varepsilon_{\mathrm{POD}}\right),$$

   where $\varepsilon_{\text{DEIM}}$ is a prescribed tolerance;

2. select iteratively M indices $\mathfrak{I} \subset \{1, \cdots, N_q\}, |\mathfrak{I}| = M$ from the basis $\mathbb{Q}$ using a greedy procedure, which minimizes at each step the interpolation error over the snapshots set measured in the maximum norm. This operation is indeed the same as for the selection of the EIM magic points;

Discrete Empirical Interpolation Method iv

Figu

3 given a new $\boldsymbol{\mu}$, in order to compute the coefficients vector $\gamma(\boldsymbol{\mu})$, interpolation constraints are imposed at the M points corresponding to the selected indices, thus requiring the solution of the following linear system

$$\mathbb{Q}_{\mathfrak{I}}\gamma(\boldsymbol{\mu}) = \mathrm{g}_{\mathfrak{I}}(\boldsymbol{\mu}), \tag{16}$$

where $\mathbb{Q}_{\mathfrak{I}} \in \mathbb{R}^{M\times M}$ is the matrix formed by the $\mathfrak{I}$ rows of $\mathbb{Q}$. As a result,

$$\mathrm{g}_M(\boldsymbol{\mu}) = \mathbb{Q}_{\mathfrak{I}}^{-1}\mathrm{g}_{\mathfrak{I}}(\boldsymbol{\mu})$$

Discrete Empirical Interpolation Method v

Here we only point out that the error between g and its DEIM approximation g_M can be bounded as

$$\|g(\boldsymbol{\mu}) - g_M(\boldsymbol{\mu})\|_2 \leq \left\|\mathbb{Q}_{\mathfrak{J}}^{-1}\right\|_2 \left\|\left(\mathbb{I} - \mathbb{Q}\mathbb{Q}^T\right) g(\boldsymbol{\mu})\right\|_2, \qquad (17)$$

with

$$\left\|\left(\mathbb{I} - \mathbb{Q}\mathbb{Q}^T\right) g(\boldsymbol{\mu})\right\|_2 \approx \sigma_{M+1}, \qquad (18)$$

being σ_{M+1} the first discarded singular value of the matrix $\mathbb{S}$ when selecting M basis through the POD procedure.

Discrete Empirical Interpolation Method vi

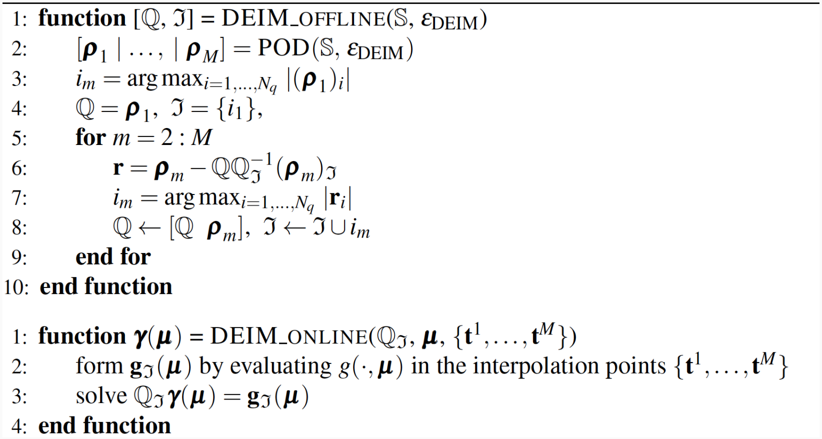
Figu

This approximation holds for any $\boldsymbol{\mu} \in \mathcal{D}$ provided a suitable sampling in the parameter space has been carried out to build the snapshot matrix $\mathbb{S}$.

- the predictive projection error (**??**) is comparable to the training projection error σ_{M+1}
- Similar to the one point error estimator of EIM, (**??**) exploits the information related to the first discarded term and can be seen as a heuristic measure of the DEIM error.

Discrete Empirical Interpolation Method vii

THE DEIM procedure reads



```
1:  function $[\mathbb{Q}, \mathfrak{I}] = \text{DEIM\_OFFLINE}(\mathbb{S}, \varepsilon_{\text{DEIM}})$
2:      $[\boldsymbol{\rho}_1 \mid \dots, \mid \boldsymbol{\rho}_M] = \text{POD}(\mathbb{S}, \varepsilon_{\text{DEIM}})$
3:      $i_m = \arg\max_{i=1,\dots,N_q} |(\boldsymbol{\rho}_1)_i|$
4:      $\mathbb{Q} = \boldsymbol{\rho}_1,\ \mathfrak{I} = \{i_1\},$
5:      for $m = 2 : M$
6:          $\mathbf{r} = \boldsymbol{\rho}_m - \mathbb{Q}\mathbb{Q}_{\mathfrak{I}}^{-1}(\boldsymbol{\rho}_m)_{\mathfrak{I}}$
7:          $i_m = \arg\max_{i=1,\dots,N_q} |\mathbf{r}_i|$
8:          $\mathbb{Q} \leftarrow [\mathbb{Q}\ \ \boldsymbol{\rho}_m],\ \mathfrak{I} \leftarrow \mathfrak{I} \cup i_m$
9:      end for
10: end function

1:  function $\boldsymbol{\gamma}(\boldsymbol{\mu}) = \text{DEIM\_ONLINE}(\mathbb{Q}_{\mathfrak{I}}, \boldsymbol{\mu}, \{\mathbf{t}^1, \dots, \mathbf{t}^M\})$
2:      form $\mathbf{g}_{\mathfrak{I}}(\boldsymbol{\mu})$ by evaluating $g(\cdot, \boldsymbol{\mu})$ in the interpolation points $\{\mathbf{t}^1, \dots, \mathbf{t}^M\}$
3:      solve $\mathbb{Q}_{\mathfrak{I}}\boldsymbol{\gamma}(\boldsymbol{\mu}) = \mathbf{g}_{\mathfrak{I}}(\boldsymbol{\mu})$
4:  end function
```

Figu

DEIM vs EIM: Key Differences

EIM:

- Greedy basis construction
- Basis = normalized snapshots
- Magic points from greedy
- Online: $O(M^2)$ triangular solve
- Sharp error estimator

DEIM:

- POD basis construction
- Basis = left singular vectors
- Interpolation points from greedy
- Online: $O(M^2)$ solve
- Heuristic error bound

Figu

Which to use?

- DEIM: when snapshots easily obtained, want optimal basis
- EIM: when function evaluations expensive, want adaptive construction
- Both: similar accuracy in practice, DEIM slightly better conditioning

Method Comparison: EIM, DEIM, Gappy POD

Feature	EIM	DEIM	Gappy POD/DEIM
Basis construction	Greedy	POD	POD
Sample type	Points (M)	Points (M)	Points ($K > M$)
System type	Square	Square	Overdetermined
Noise robustness	Moderate	Moderate	High
Flexibility	Low	Low	Medium
Cost (offline)	$O(M)$ solves	1 POD	1 POD
Cost (online)	$O(M^2)$	$O(M^2)$	$O(M^3)$ or $O(KM^2)$
Error estimate	Sharp	Heuristic	—

Note: All three methods achieve exponential convergence for smooth parametric manifolds.

Gappy POD and Gappy DEIM

Gappy POD: Motivation and Formulation i

Limitation of standard EIM/DEIM: Use exactly M interpolation points for M basis functions.

Gappy POD idea: Use $K > M$ sample points (overdetermined system) for more robust reconstruction, especially with noisy data.

Figu

Original references:

- Bui-Thanh, T., Damodaran, M., Willcox, K.: Proper orthogonal decomposition extensions for parametric applications in transonic aerodynamics (AIAA Paper 2003-4213). In: Proceedings of the 15th AIAA Computational Fluid Dynamics Conference (2003)

Gappy POD: Motivation and Formulation ii

Figu

- Carlberg, K., Farhat, C., Cortial, J., Amsallem, D.: The GNAT method for nonlinear model reduction: Effective implementation and application to computational fluid dynamics and turbulent flows. J. Comput. Phys. 242, 623–647 (2013)

Gappy POD: Motivation and Formulation iii

Problem setup:

- Basis: $\mathbb{Q} = [\boldsymbol{\rho}_1 | \cdots | \boldsymbol{\rho}_M] \in \mathbb{R}^{N_q \times M}$ (from POD or EIM greedy)
- Sample indices: $\mathfrak{I} = \{i_1, \ldots, i_K\}$ with $K \geq M$ (can be $K > M$)
- Point values: $\mathrm{y} = \mathrm{g}_{\mathfrak{I}}(\boldsymbol{\mu}) = [g(x_{i_1}; \boldsymbol{\mu}), \ldots, g(x_{i_K}; \boldsymbol{\mu})]^T \in \mathbb{R}^K$ (possibly noisy)

Standard DEIM ($K = M$): Square system

$$\mathbb{Q}_{\mathfrak{I}} \boldsymbol{\gamma} = \mathrm{y}, \quad \mathbb{Q}_{\mathfrak{I}} \in \mathbb{R}^{M \times M}$$

Gappy POD ($K > M$): Overdetermined least squares

$$\boldsymbol{\gamma}^*(\boldsymbol{\mu}) = \arg\min_{\boldsymbol{\gamma} \in \mathbb{R}^M} \|\mathbb{Q}_{\mathfrak{I}} \boldsymbol{\gamma} - \mathrm{y}\|_2^2 \tag{19}$$

where $\mathbb{Q}_{\mathfrak{I}} \in \mathbb{R}^{K \times M}$ with $K > M$.

Gappy POD: Motivation and Formulation iv

Figu

Real-world scenarios requiring gappy POD:

1. **Sensor failure:** Deploy $K = 30$ sensors; if 5 fail, still have $25 > M = 20$ for reconstruction
2. **Data assimilation:** Satellite measurements ($K = 1000$) into weather model ($M = 50$ modes)
3. **MRI reconstruction:** Partial k-space data (K measurements) $\rightarrow$ full image (M basis functions)
4. **Distributed sensing:** IoT networks with redundant, noisy sensors

Gappy POD: Motivation and Formulation v

Advantages of overdetermination:

Figu

1. **Noise robustness:** Extra sample points average out noise
2. **Stability:** Better conditioning, less sensitive to missing data
3. **Accuracy:** Can improve reconstruction when $K \approx 1.5M$ to $2M$

Reconstruction:

$$\mathrm{g}_M(\boldsymbol{\mu}) = \mathbb{Q}\boldsymbol{\gamma}^* \in \mathbb{R}^{N_q}$$

With regularization: For very noisy data, add Tikhonov term

$$\boldsymbol{\gamma}^*(\boldsymbol{\mu}) = \arg\min_{\boldsymbol{\gamma}\in\mathbb{R}^M} \left\{ \|\mathbb{Q}_{\mathfrak{I}}\boldsymbol{\gamma} - \mathrm{y}\|_2^2 + \tau\|\boldsymbol{\gamma}\|_2^2 \right\} \tag{20}$$

where $\tau > 0$ is the regularization parameter.

Gappy POD: Motivation and Formulation vi

Error analysis for Gappy POD:

Figu

For overdetermined system with $K > M$ samples and regularization τ:

$$\|g(\mu) - \mathbb{Q}\gamma^*\|_2 \leq \underbrace{\sigma_{M+1}}_{\text{truncation}} + \underbrace{\|(\mathbb{Q}_{\mathfrak{J}}^T \mathbb{Q}_{\mathfrak{J}} + \tau I)^{-1}\|_2 \epsilon}_{\text{noise amplification}}$$

Key observations:

- As $K \to M$: error dominated by truncation σ_{M+1}
- As $K \gg M$: overdetermination reduces noise sensitivity
- Optimal $K \approx 1.5M$ to $2M$ balances cost and robustness
- Regularization τ trades bias for variance: $\tau \uparrow \to$ smoother but less accurate

Methods for Solving Gappy POD Least Squares i

Figu

Given the overdetermined system $\mathbb{Q}_{\mathfrak{I}}\gamma = \mathrm{y}$ with $\mathbb{Q}_{\mathfrak{I}} \in \mathbb{R}^{K\times M}$, $K > M$, we need to solve

$$\min_{\gamma\in\mathbb{R}^M} \|\mathbb{Q}_{\mathfrak{I}}\gamma - \mathrm{y}\|_2^2$$

Three main approaches:

Methods for Solving Gappy POD Least Squares ii

Figu

Method 1: Normal Equations

The minimizer satisfies the **normal equations**:

$$\mathbb{Q}_{\mathfrak{J}}^T \mathbb{Q}_{\mathfrak{J}} \boldsymbol{\gamma}^* = \mathbb{Q}_{\mathfrak{J}}^T \mathrm{y}$$

Algorithm:

1. Form $\mathbb{A} = \mathbb{Q}_{\mathfrak{J}}^T \mathbb{Q}_{\mathfrak{J}} \in \mathbb{R}^{M \times M}$ (symmetric positive definite)
2. Form $\mathrm{b} = \mathbb{Q}_{\mathfrak{J}}^T \mathrm{y} \in \mathbb{R}^M$
3. Solve $\mathbb{A}\boldsymbol{\gamma}^* = \mathrm{b}$ using Cholesky factorization

Methods for Solving Gappy POD Least Squares iii

Figu

Cost: $O(KM^2)$ to form $\mathbb{A}$, then $O(M^3)$ for Cholesky solve.

Pros: Simple, efficient for well-conditioned problems.

Cons: Squares the condition number: $\kappa(\mathbb{A}) = \kappa(\mathbb{Q}_{\mathfrak{I}})^2$, numerically unstable for ill-conditioned $\mathbb{Q}_{\mathfrak{I}}$.

Methods for Solving Gappy POD Least Squares iv

Method 2: QR Factorization

Compute the QR decomposition of $\mathbb{Q}_{\mathfrak{I}}$:

$$\mathbb{Q}_{\mathfrak{I}} = \mathsf{Q}_{\mathsf{QR}}\mathsf{R}$$

where $\mathsf{Q}_{\mathsf{QR}} \in \mathbb{R}^{K\times M}$ has orthonormal columns and $\mathsf{R} \in \mathbb{R}^{M\times M}$ is upper triangular.

Algorithm:

1. Compute QR: $\mathbb{Q}_{\mathfrak{I}} = \mathsf{Q}_{\mathsf{QR}}\mathsf{R}$
2. The LS problem becomes: $\min_{\gamma} \|\mathsf{R}\gamma - \mathsf{Q}_{\mathsf{QR}}^T \mathsf{y}\|_2^2$
3. Solve $\mathsf{R}\gamma^* = \mathsf{Q}_{\mathsf{QR}}^T \mathsf{y}$ by back substitution

Figu

Methods for Solving Gappy POD Least Squares v

Figu

Cost: $O(KM^2)$ for QR, then $O(M^2)$ for back substitution.

Pros: More stable than normal equations; $\kappa(\mathrm{R}) = \kappa(\mathbb{Q}_{\mathfrak{I}})$.

Cons: More expensive than normal equations.

Methods for Solving Gappy POD Least Squares vi

Method 3: Singular Value Decomposition (SVD)

Figu

Compute the thin SVD of $\mathbb{Q}_{\mathfrak{I}}$:

$$\mathbb{Q}_{\mathfrak{I}} = \mathrm{U}\boldsymbol{\Sigma}\mathrm{V}^T$$

where $\mathrm{U} \in \mathbb{R}^{K\times M}$ (orthogonal columns), $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \ldots, \sigma_M)$ with $\sigma_1 \geq \cdots \geq \sigma_M > 0$, and $\mathrm{V} \in \mathbb{R}^{M\times M}$ (orthogonal).

Solution:

$$\gamma^* = \mathrm{V}\boldsymbol{\Sigma}^{-1}\mathrm{U}^T\mathrm{y} = \sum_{i=1}^{M} \frac{\mathrm{u}_i^T\mathrm{y}}{\sigma_i}\mathrm{v}_i$$

where $\mathrm{u}_i, \mathrm{v}_i$ are columns of U, V.

Cost: $O(KM^2)$ for thin SVD.

Methods for Solving Gappy POD Least Squares vii

Figu

Pros: Most stable; reveals ill-conditioning; can truncate small σ_i (regularization).

Cons: More expensive than QR; overkill if $\mathbb{Q}_{\mathfrak{J}}$ is well-conditioned.

Methods for Solving Gappy POD Least Squares viii

Figu

Method 4: Moore-Penrose Pseudoinverse

The solution can be written as:

$$\gamma^* = \mathbb{Q}_{\mathfrak{J}}^{+} \mathrm{y}$$

where $\mathbb{Q}_{\mathfrak{J}}^{+} = (\mathbb{Q}_{\mathfrak{J}}^{T}\mathbb{Q}_{\mathfrak{J}})^{-1}\mathbb{Q}_{\mathfrak{J}}^{T}$ is the Moore-Penrose pseudoinverse.

Note: In practice, compute via SVD: $\mathbb{Q}_{\mathfrak{J}}^{+} = \mathrm{V}\mathbf{\Sigma}^{-1}\mathrm{U}^{T}$.

Methods for Solving Gappy POD Least Squares ix

Regularized Least Squares (Tikhonov):

For noisy data, solve:

$$\gamma^* = \arg\min_{\gamma} \left\{ \|\mathbb{Q}_{\mathfrak{J}}\gamma - \mathrm{y}\|_2^2 + \tau\|\gamma\|_2^2 \right\}$$

Normal equations:

$$(\mathbb{Q}_{\mathfrak{J}}^T\mathbb{Q}_{\mathfrak{J}} + \tau\mathbb{I}_M)\gamma^* = \mathbb{Q}_{\mathfrak{J}}^T\mathrm{y}$$

SVD solution:

$$\gamma^* = \sum_{i=1}^{M} \frac{\sigma_i}{\sigma_i^2 + \tau}(\mathrm{u}_i^T\mathrm{y})\mathrm{v}_i$$

Note: Tikhonov acts as a filter, damping contributions from small singular values.

Figu

Methods for Solving Gappy POD Least Squares x

Practical recommendations:

Scenario	Method	Reason
Well-conditioned, $M < 50$	Normal eqs	Fast, simple
Moderate conditioning	QR	Good stability/cost
Ill-conditioned or noisy	SVD + Tikhonov	Most robust
Need error estimates	SVD	Access to σ_i

Figu

Conditioning check:

- Compute $\kappa(\mathbb{Q}_{\mathfrak{I}}) = \sigma_1/\sigma_M$ from SVD
- If $\kappa > 10^{10}$: use regularization
- If $\kappa > 10^{15}$: problem is severely ill-posed

Computational Complexity Summary

Operation	Offline	Online
EIM greedy (per iteration)	$O(N_q n_{\text{train}})$	—
EIM online solve	—	$O(M^2)$
DEIM: POD	$O(N_q^2 n_s)$	—
DEIM greedy (points)	$O(MN_q n_s)$	—
DEIM online solve	—	$O(M^2)$
Gappy LS (normal eq)	$O(KM^2)$	$O(M^3)$
Gappy LS (QR)	$O(KM^2)$	$O(M^2)$
Gappy LS (SVD)	$O(KM^2)$	$O(MK)$
GEIM greedy (per iteration)	$O(N_q N_s n_{\text{train}})$	—
GEIM online solve	—	$O(M^2)$

Typical values: $N_q \sim 10^4$-10^6, $M \sim 10$-50, $K \sim 1.5M$-$2M$, $n_{\text{train}} \sim 10^3$

Figu

Gappy POD/DEIM: Implementation Tips

Choosing K (overdetermination factor):

- Typical: $K = \alpha M$ with $\alpha \in [1.5, 2.5]$
- More sample points (α larger) $\rightarrow$ better noise robustness, higher cost
- Diminishing returns for $\alpha > 3$

Point selection for gappy POD:

- Use EIM/DEIM greedy to select first M interpolation points
- Continue greedy to add next $K - M$ points
- Alternative: use uniform sampling or importance sampling based on residual magnitude

Regularization parameter τ:

- Estimate noise level: $\|\boldsymbol{\epsilon}\|_2 \approx \epsilon$
- **Discrepancy principle:** Choose τ such that $\|\mathbb{Q}_{\mathfrak{I}}\gamma^*_\tau - \mathrm{y}\|_2 \approx \epsilon$ (match residual to noise level)

Figu

Non affine problems

EIM for Non Affine Problems i

We now exploit the empirical interpolation method to recover an affine parametric dependence in the originally nonaffine operators.

Although both EIM and DEIM lead to the same computational procedure from a practical standpoint, we exploit the former to develop our analysis in a more straightforward way.

Consider our usual model problem where for the sake of interpolation we assume that both $s(\cdot;\boldsymbol{\mu})$ and $k(\cdot;\boldsymbol{\mu}) \in C^0(\Omega)$ for any $\boldsymbol{\mu} \in \mathcal{D}$.

Note: We assume to deal with a strongly coercive problem, although all the results can be extended to the more general case of weakly coercive problems.

EIM for Non Affine Problems ii

The high-fidelity approximation reads: find $u_h(\boldsymbol{\mu}) \in V_h$ such that

$$a\left(u_h, v_h; \boldsymbol{\mu}\right) = f\left(v_h; \boldsymbol{\mu}\right) \quad \forall v_h \in V_h,$$

with

$$a\left(u_h, v_h; \boldsymbol{\mu}\right) = \int_\Omega k(\mathrm{x}; \boldsymbol{\mu}) \nabla u_h \cdot \nabla v_h d\Omega, \quad f\left(v_h; \boldsymbol{\mu}\right) = \int_\Omega s(\mathrm{x}; \boldsymbol{\mu}) v_h d\Omega$$

We assume that the diffusion coefficient $k(\mathrm{x}; \boldsymbol{\mu})$ and the source term $s(\mathrm{x}; \boldsymbol{\mu})$ are nonaffine functions of $\boldsymbol{\mu}$, yielding a nonaffine parametric

EIM for Non Affine Problems iii

Figu

dependence of the linear and bilinear forms. Using the EIM, we replace them by the corresponding interpolants

$$k_M(\mathrm{x};\boldsymbol{\mu}) = \mathcal{I}_M^{\mathrm{x}} k(\mathrm{x};\boldsymbol{\mu}) = \sum_{j=1}^{M_k} \gamma_j^k(\boldsymbol{\mu})\rho_j^k(\mathrm{x})$$

$$s_M(\mathrm{x};\boldsymbol{\mu}) = \mathcal{I}_M^{\mathrm{x}} s(\mathrm{x};\boldsymbol{\mu}) = \sum_{j=1}^{M_s} \gamma_j^s(\boldsymbol{\mu})\rho_j^s(\mathrm{x})$$

EIM for Non Affine Problems iv

We now write the corresponding high-fidelity problem with EIM approximation (to which we refer to as EIM high-fidelity approximation) becomes:

find $u_h^M(\boldsymbol{\mu}) \in V_h$ such that

$$a_M\left(u_h^M(\boldsymbol{\mu}), v_h; \boldsymbol{\mu}\right) = f_M\left(v_h; \boldsymbol{\mu}\right) \quad \forall v_h \in V_h,$$

where we have set

$$a_M\left(u_h, v_h; \boldsymbol{\mu}\right) = \int_\Omega k_M(\mathrm{x}; \boldsymbol{\mu}) \nabla u_h \cdot \nabla v_h d\Omega, \quad f_M\left(v_h; \boldsymbol{\mu}\right) = \int_\Omega s_M(\mathrm{x}; \boldsymbol{\mu}) v_h d\Omega.$$

The linear and the bilinear forms now depend on the number M_k, M_s of terms appearing in the interpolants; we denote the EIM dependence by the subscript M.

Figu

EIM for Non Affine Problems v

Figu

The associated Galerkin RB problem, thus reads:

find $u_N^M(\boldsymbol{\mu}) \in V_N$ such that

$$a_M\left(u_N^M(\boldsymbol{\mu}), v_N; \boldsymbol{\mu}\right) = f_M\left(v_N; \boldsymbol{\mu}\right) \quad \forall v_N \in V_N \tag{21}$$

EIM for Non Affine Problems vi

Remark: We remark that the bilinear form $a_M(\cdot, \cdot; \boldsymbol{\mu})$ does not necessarily preserve the properties of $a(\cdot, \cdot; \boldsymbol{\mu})$. While the symmetry of $a(\cdot, \cdot; \boldsymbol{\mu})$ is automatically inherited by its approximation, this is not the case for the (strong) coercivity. However, by requiring that the high-fidelity problem is coercive, the EIM RB problem is well-posed too, being a Galerkin projection.

Note: The EIM RB problem can be considered as a generalized Galerkin method.

EIM for Non Affine Problems vii

Theorem (Convergence)
Let us suppose that $a_M(\cdot,\cdot;\boldsymbol{\mu})$ is continuous on $V_h \times V_h$ and coercive on V_h for any $\boldsymbol{\mu} \in \mathcal{D}$, that is

$$\exists \alpha_h^M(\boldsymbol{\mu}) : a_M(v,v;\boldsymbol{\mu}) \geq \alpha_h^M(\boldsymbol{\mu}) \|v\|_V^2 \quad \forall v \in V_h, \forall \boldsymbol{\mu} \in \mathcal{D}.$$

Moreover, let us suppose that $f_M(\cdot;\boldsymbol{\mu})$ is continuous on V_h.

*Then problem (**??**) admits a unique solution $u_N^M(\boldsymbol{\mu}) \in V_N$ which satisfies*

$$\left\| u_N^M(\boldsymbol{\mu}) \right\|_V \leq \frac{1}{\alpha_N^M(\boldsymbol{\mu})} \left\| f_M(\cdot;\boldsymbol{\mu}) \right\|_{V_h'},$$

being

$$\alpha_N^M(\boldsymbol{\mu}) = \inf_{v \in V_N} \frac{a_M(v,v;\boldsymbol{\mu})}{\|v\|_V^2} \geq \alpha_h^M(\boldsymbol{\mu})$$

the stability factor of the EIM RB problem, and fulfills the following a priori error estimate

EIM RB method A posteriori estimate i

It is also possible to derive an a posteriori error estimate by combining the error estimator obtained in the case of a generic linear elliptic PDE, and a suitable indicator of the empirical interpolation error.

Let us denote by $e_h^M(\boldsymbol{\mu}) = u_h(\boldsymbol{\mu}) - u_N^M(\boldsymbol{\mu})$ the error between the high-fidelity and the EIM RB solutions and by

$$r_M(v;\boldsymbol{\mu}) = f_M(v;\boldsymbol{\mu}) - a_M\left(u_N^M(\boldsymbol{\mu}), v; \boldsymbol{\mu}\right) \quad \forall v \in V$$

the residual of the EIM high-fidelity problem computed on the RB solution. We assume that $a(\cdot,\cdot;\boldsymbol{\mu})$ is strongly coercive over $V_h \times V_h$ for any $\boldsymbol{\mu} \in \mathcal{D}$ and denote by $\alpha_h(\boldsymbol{\mu})$ its stability factor

EIM RB method A posteriori estimate ii

Proposition
The following a posteriori estimates hold

$$\left\|u_h(\boldsymbol{\mu}) - u_N^M(\boldsymbol{\mu})\right\|_V \leq \frac{1}{\alpha_h(\boldsymbol{\mu})}\left(\left\|r_M(\cdot;\boldsymbol{\mu})\right\|_{V_h'} + C_f\delta_s(\boldsymbol{\mu}) + C_a\delta_k(\boldsymbol{\mu})\left\|u_N^M(\boldsymbol{\mu})\right\|_V\right)$$

where

$$C_f = \sup_{v\in V_h}\frac{\int_\Omega v d\Omega}{\|v\|_V}, \quad C_a = \sup_{v\in V_h w\in V_h}\sup_{\Omega}\frac{\int_\Omega \nabla v\cdot\nabla w d\Omega}{\|v\|_V\|w\|_V}$$

and

$$\delta_k(\boldsymbol{\mu}) = \|k(\cdot;\boldsymbol{\mu}) - k_M(\cdot;\boldsymbol{\mu})\|_{L^\infty(\Omega)}, \quad \delta_s(\boldsymbol{\mu}) = \|s(\cdot;\boldsymbol{\mu}) - s_M(\cdot;\boldsymbol{\mu})\|_{L^\infty(\Omega)}$$

Figu

EIM RB method A posteriori estimate iii

Proof: we have that, for all $v_h \in V_h$,

$$a\left(u_h(\boldsymbol{\mu}), v_h; \boldsymbol{\mu}\right) - a\left(u_N^m(\boldsymbol{\mu}), v_h; \boldsymbol{\mu}\right) = f\left(v_h; \boldsymbol{\mu}\right) - a\left(u_N^m(\boldsymbol{\mu}), v_h; \boldsymbol{\mu}\right)$$
$$= f\left(v_h; \boldsymbol{\mu}\right) - f_M\left(v_h; \boldsymbol{\mu}\right) + f_M\left(v_h; \boldsymbol{\mu}\right)$$
$$- a_M\left(u_N^m(\boldsymbol{\mu}), v_h; \boldsymbol{\mu}\right) + a_M\left(u_N^m(\boldsymbol{\mu}), v_h; \boldsymbol{\mu}\right) - a\left(u_N^m(\boldsymbol{\mu}), v_h; \boldsymbol{\mu}\right)$$

By taking $v_h = e_h^M(\boldsymbol{\mu})$, exploiting the continuity of the linear and bilinear forms and the coercivity of $a(\cdot, \cdot; \boldsymbol{\mu})$, and dividing by $\left\|e_h^M(\boldsymbol{\mu})\right\|_V$, we find

$$\alpha_h(\boldsymbol{\mu}) \left\|e_h^M(\boldsymbol{\mu})\right\|_V \leq \|f(\cdot; \boldsymbol{\mu}) - f_M(\cdot; \boldsymbol{\mu})\|_{V_h'}$$
$$+ \|a(\cdot, ; ; \boldsymbol{\mu}) - a_M(\cdot, \cdot; \boldsymbol{\mu})\|_{\mathcal{L}\left(V_h, V_h'\right)} \left\|u_N^M(\boldsymbol{\mu})\right\|_V + \|r_M(\cdot; \boldsymbol{\mu})\|_{V_h'} .$$

EIM RB method A posteriori estimate iv

The first term can be bounded as

$$\|f(\cdot;\boldsymbol{\mu}) - f_M(\cdot;\boldsymbol{\mu})\|_{V_h'} \leq \sup_{v\in V_h} \frac{\left|\int_\Omega (s(\cdot;\boldsymbol{\mu}) - s_M(\cdot;\boldsymbol{\mu}))\, v d\Omega\right|}{\|v\|_V} \leq C_f \|s(\cdot;\boldsymbol{\mu}) - s_M(\cdot;\boldsymbol{\mu})\|_{L^\infty(\Omega)}.$$

Figu

Similarly for the second term we find

$$\|a(\cdot,\cdot;\boldsymbol{\mu}) - a_M(\cdot,\cdot;\boldsymbol{\mu})\|_{\mathcal{L}(V_h,V_h')} \leq \sup_{v\in V_h w\in V_h} \frac{\left|\int_\Omega (k(\cdot;\boldsymbol{\mu}) - k_M(\cdot;\boldsymbol{\mu}))\, \nabla v \cdot \nabla w d\Omega\right|}{\|v\|_V \|w\|_V} \leq C_a \|k(\cdot;\boldsymbol{\mu}) - k_M(\cdot;\boldsymbol{\mu})\|_{L^\infty(\Omega)}.$$

□

Remark: Surrogates for $\delta_k(\boldsymbol{\mu})$ and $\delta_s(\boldsymbol{\mu})$ could be either the tolerance of the EIM algorithm or the estimator $\hat{\varepsilon}_M(\boldsymbol{\mu})$.

Extension: Gappy Methods in EIM-RB

In practice: Gappy POD/GEIM can improve robustness in EIM-RB when:

- Function evaluations $g(\mathrm{x}^k;\mu)$ are noisy (e.g., experimental data)
- Quadrature points N_q are unreliable (adaptive mesh, sensor placement)
- Want to use fewer evaluations than basis functions for efficiency

Figu

Modified EIM-RB procedure:

1. Use GEIM to select $K > M$ optimal quadrature points
2. Solve overdetermined $\mathbb{Q}_{\mathfrak{I}}\boldsymbol{\gamma} = \mathrm{g}_{\mathfrak{I}}$ with Tikhonov regularization
3. Affine expansion: $f_M(v;\mu) = \sum_{m=1}^{M} \gamma_m(\mu) \int_\Omega \rho_m v \, d\Omega$
4. Use regularized coefficients in RB assembly

Benefits: More robust to quadrature errors, measurement noise, mesh artifacts

Numerical Example: Parametrized Gaussian Source

Problem: Mass transfer with Gaussian source term

$$s(\mathrm{x};\boldsymbol{\mu}) = \exp\left(-\frac{(x_1-\mu_3)^2+(x_2-\mu_4)^2}{\mu_5^2}\right)$$

where μ_3, μ_4 control location and μ_5 controls width.

Setup:

- Domain: $\Omega = [0,1]^2$
- High-fidelity: $\mathbb{P}_1$ FE, $N_h = 5305$ vertices, 10368 triangles
- Quadrature: 4th-order Dunavant rule, $N_q = 62208$ points
- Parameters: $\mu_3 \in [0.2, 0.8]$, $\mu_4 \in [0.15, 0.35]$, $\mu_5 = 0.25$ (fixed)

Figu

Numerical Results

Test case 1: $\mu_3 \in [0.2, 0.8]$, $\mu_4 \in [0.15, 0.35]$, $\mu_5 = 0.25$ (fixed)

Figu

DEIM:

- Training: $n_s^{\text{DEIM}} = 100$ LHS samples, $\varepsilon_{\text{DEIM}} = 10^{-6}$
- Result: POD extracts $M = 40$ basis functions

EIM:

- Training: $|\Xi_{\text{train}}^{EIM}| = 1000$, $M_{\max} = 40$
- Result: 40 magic points selected

Observations:

- Both achieve similar accuracy ($\sim 10^{-6}$)
- EIM error estimator is sharper than DEIM heuristic bound
- Test error computed on 200 ran

More Challenging Case

Test case 2: Varying width $\mu_5 \in [0.1, 0.35]$ (in addition to location)

Figu

DEIM:

- Training: $n_s^{\text{DEIM}} = 200$, $\varepsilon_{\text{DEIM}} = 10^{-5}$
- Result: $M = 83$ basis functions needed ($4\times$ more than fixed width case)

EIM:

- Training: $|\Xi_{\text{train}}^{EIM}| = 2000$, $M_{\max} = 83$
- Result: 83 magic points selected

Conclusion: Varying width parameter significantly increases manifold complexity, requiring $\approx 4\times$ more basis functions to achieve same accuracy.

GEIM: Generalized Empirical Interpolation

From EIM/DEIM to GEIM

- **EIM**: enforce pointwise interpolation at "magic points" t_i.
- **DEIM**: same interpolation strategy; basis from POD (SVD).
- **GEIM**: replace point samples by **linear functionals** $L_i(\cdot)$ (sensors, measurements, integrals).

Interpolant: $I_M g(x;\mu) = \sum_{j=1}^{M} \gamma_j(\mu)\,\rho_j(x)$ with measurement matching

$$L_i(I_M g) = L_i(g), \quad i = 1, \dots, M.$$

Discrete algebra: on a grid Ω_h, let $Q = [\rho_1|\cdots|\rho_M] \in \mathbb{R}^{N_q \times M}$ and $A \in \mathbb{R}^{N_s \times N_q}$ stack candidate sensors (rows are $\ell^\top$ so $L(v) = \ell^\top v$). Pick sensors $J = \{j_1, \dots, j_M\}$; with $B_M = A_J Q$ and $y(\mu) = A_J g(\mu)$:

$$B_M\,\gamma(\mu) = y(\mu), \qquad I_M g = Q\,\gamma(\mu).$$

Figu

What are sensors? Linear functionals i

Figu

Sensors are linear functionals $L : V \to \mathbb{R}$ that extract information from functions.

Continuous form: $L(g) = \int_\Omega \ell(x)g(x)dx$ or $L(g) = \int_\Gamma \ell(s)g(s)ds$

Discrete form: On grid Ω_h with N_q points, $g \in \mathbb{R}^{N_q}$, sensor is a row vector $\ell \in \mathbb{R}^{N_q}$:

$$L(g) = \ell^T g = \sum_{i=1}^{N_q} \ell_i g_i$$

What are sensors? Linear functionals ii

Examples of linear functionals:

Figu

1. **Pointwise evaluation** (as in EIM/DEIM):

$$L_i(g) = g(x_i) \quad \Longrightarrow \quad \ell = e_i = [0, \dots, 0, 1, 0, \dots, 0]^T$$

2. **Local average over region** ω:

$$L(g) = \frac{1}{|\omega|} \int_\omega g(x)dx \quad \Longrightarrow \quad \ell_i = \begin{cases} 1/n_\omega & \text{if } x_i \in \omega \\ 0 & \text{otherwise} \end{cases}$$

where n_ω is the number of grid points in ω

What are sensors? Linear functionals iii

Figu

3. **Weighted average (Gaussian sensor)**:

$$L(g) = \int_\Omega w(x) g(x) dx, \quad w(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\|x-x_0\|^2/(2\sigma^2)}$$

Discrete: $\ell_i = w(x_i)\Delta x_i$ (quadrature weights)

4. **Line integral along Γ**:

$$L(g) = \int_\Gamma g(s) ds \quad \Longrightarrow \quad \ell_i = \begin{cases} \Delta s_i & \text{if } x_i \in \Gamma \\ 0 & \text{otherwise} \end{cases}$$

What are sensors? Linear functionals iv

5 **Directional derivative in direction** n:

$$L(g) = \int_\Omega \nabla g \cdot \mathrm{n}\, dx \quad \text{or} \quad L(g) = \left.\frac{\partial g}{\partial n}\right|_{x_0}$$

Figu

6 **PDE-informed functional** (flux through boundary):

$$L(u) = \int_{\Gamma_{\mathrm{out}}} \mathrm{q} \cdot \mathrm{n}\, ds, \quad \mathrm{q} = -\kappa \nabla u$$

Key advantage: GEIM allows sensors matching physical measurement devices (thermocouples, pressure gauges, flow meters) rather than requiring point values.

GEIM greedy algorithm (offline)

Figu

1. Training set Ξ_{train}; snapshots $S = [g(\mu_1)|\cdots|g(\mu_{n_s})]$ on Ω_h; sensor dictionary $\mathcal{L}$ (rows of A).
2. **Init** ($m = 1$): pick $\mu_1 = \arg\max_\mu \|g(\cdot;\mu)\|_\infty$, set $\xi_1 = g(\cdot;\mu_1)$; choose $L_1 \in \mathcal{L}$ maximizing $|L_1(\xi_1)|$; define $\rho_1 = \xi_1/L_1(\xi_1)$.
3. **Loop** $m = 1, \ldots, M-1$:
   1. $\mu_{m+1} = \arg\max_{\mu\in\Xi_{\mathrm{train}}} \|g(\cdot;\mu) - I_m g(\cdot;\mu)\|_\infty$; set $\xi_{m+1} = g(\cdot;\mu_{m+1})$.
   2. Residual $r_{m+1} = \xi_{m+1} - I_m\xi_{m+1}$.
   3. Choose $L_{m+1} \in \mathcal{L} \setminus \{L_1, \ldots, L_m\}$ maximizing $|L_{m+1}(r_{m+1})|$.
   4. Normalize $\rho_{m+1} = r_{m+1}/L_{m+1}(r_{m+1})$.

Online: given $y_i(\mu) = L_i(g(\cdot;\mu))$, solve $(A_J Q)\gamma = y$ and return $I_M g = Q\gamma$.

Key properties (EIM-like)

Figu

- **Triangular & unit diagonal:** $B_M = [L_i(\rho_j)]$ is lower triangular with $(B_M)_{ii} = 1$ by normalization.
- **Exactness:** $I_M g = g$ for $g \in X_M = \mathrm{span}\{\rho_j\}_{j=1}^M$.
- **Lebesgue bound:** $\|g - I_M g\|_\infty \leq (1 + \Lambda_M) \inf_{z \in X_M} \|g - z\|_\infty$, $\Lambda_M = \max_x \sum_{i=1}^M |(QB_M^{-1})_{x,i}|$.
- **One-functional estimator:** if $g(\cdot;\mu) \in X_{M+1}$ then $\|g - I_M g\|_\infty = \big|L_{M+1}(g - I_M g)\big|$.
- **Stability:** noise amplification governed by $\|B_M^{-1}\|$ and the discrete Λ_M.

GEIM Exponential Convergence

Theorem (GEIM Convergence Rate)
Assume $\mathcal{G} \subset C^0(\Omega)$, and there exists a sequence of nested spaces $\mathcal{Z}_1 \subset \mathcal{Z}_2 \subset \ldots$ with $\dim(\mathcal{Z}_M) = M$ and $\mathcal{Z}_M \subset \mathrm{span}\{\mathcal{G}\}$ such that for some $c > 0$ and $\alpha > \log 4$:

$$d(\mathcal{G}, \mathcal{Z}_M) = \sup_{\mu \in \mathcal{D}} \inf_{v_M \in \mathcal{Z}_M} \|g(\cdot;\mu) - v_M\|_{C^0(\Omega)} \leq c e^{-\alpha M}$$

Then, provided the sensor dictionary $\mathcal{L}$ is sufficiently rich:

$$\sup_{\mu \in \mathcal{D}} \|g(\cdot;\mu) - I_M g(\cdot;\mu)\|_{C^0(\Omega)} \leq c e^{-(\alpha - \log 4) M}$$

Figu

Interpretation: GEIM inherits exponential convergence from EIM/DEIM, but sensor richness matters!

Gappy-GEIM and noise i

Notation:

Figu

- $g \in \mathbb{R}^{N_q}$: function to approximate (discretized on grid Ω_h)
- $A_J \in \mathbb{R}^{|J|\times N_q}$: sensor matrix with rows $\{\ell_i^T\}_{i\in J}$ corresponding to selected sensors $\{L_i\}_{i\in J}$
- $y = A_J g \in \mathbb{R}^{|J|}$: exact measurements from selected sensors

Noisy measurements: $y^\delta = y + \delta$, where $y = A_J g$ and $\|\delta\| \le \epsilon$.

Overdetermination: Use $|J| > M$ sensors (more measurements than unknowns).

- *In practice:* If $M = 20$ basis functions, deploy $|J| = 30$ or 40 sensors

Gappy-GEIM and noise ii

Figu

- *Benefit:* Extra measurements average out noise, stabilize reconstruction
- *Trade-off:* More sensors $\rightarrow$ higher cost, but better noise robustness

Gappy-GEIM and noise iii

Tikhonov regularization: Solve overdetermined least-squares problem

$$\min_{\gamma} \|A_J Q\,\gamma - y^{\delta}\|_2^2 + \tau\|\gamma\|_2^2,$$

where $\tau > 0$ is the regularization parameter that prevents overfitting to noise.

Regularization parameter: Choose τ via:

- L-curve method
- Discrepancy principle: τ such that $\|A_J Q\,\gamma_\tau - y^{\delta}\| \approx \epsilon$
- Cross-validation

Diagnostics: track $\kappa(A_J Q)$ and Λ_m vs m; examine error vs noise level and overdetermination.

Figu

Gappy GEIM algorithm i

Offline phase:

Figu

1. **Input:**
   - Basis $Q \in \mathbb{R}^{N_q \times M}$, sensor dictionary $\mathcal{L}$ (matrix $A \in \mathbb{R}^{N_s \times N_q}$)
   - Overdetermination factor $\alpha \in [1.5, 2.5]$ (typical: $\alpha = 2$)
2. **Run GEIM greedy:** Select M sensors $\{L_{j_1}, \ldots, L_{j_M}\}$ forming index set J_M
3. **Add extra sensors:** Continue greedy to select $K = \lceil \alpha M \rceil$ total sensors, index set J_K with $|J_K| > M$
4. **Form matrices:** $A_{J_K} \in \mathbb{R}^{K \times N_q}$ (overdetermined sensor matrix), $B_K = A_{J_K} Q \in \mathbb{R}^{K \times M}$
5. **Precompute:** $B_K^T B_K$ and factorizations (QR or SVD) for fast online solve

Gappy GEIM algorithm ii

Online phase:

Figu

1. **Measure:** Noisy data $y^\delta \in \mathbb{R}^K$ from K sensors
2. **Choose:** Regularization parameter $\tau \in [10^{-8}, 10^{-4}]$ based on estimated noise level ϵ
   - Low noise ($\epsilon < 10^{-6}$): $\tau \sim 10^{-8}$
   - Moderate noise ($\epsilon \sim 10^{-4}$): $\tau \sim 10^{-6}$
   - High noise ($\epsilon > 10^{-3}$): $\tau \sim 10^{-4}$
3. **Solve regularized LS:** $\gamma^* = \arg\min_\gamma \|B_K\gamma - y^\delta\|_2^2 + \tau\|\gamma\|_2^2$
4. **Reconstruct:** $\hat{g} = Q\gamma^* \in \mathbb{R}^{N_q}$

Remark: Choosing τ at online stage allows adaptation to actual noise conditions without recomputing offline data.

Gappy GEIM algorithm iii

Solution methods for regularized LS:

The minimization problem

$$\gamma^* = \arg\min_{\gamma} \|B_K\gamma - y^\delta\|_2^2 + \tau\|\gamma\|_2^2$$

admits the closed-form solution (normal equations):

$$(B_K^T B_K + \tau I_M)\gamma^* = B_K^T y^\delta$$

Method 1: Direct solve (with online τ choice)

$$\gamma^* = (B_K^T B_K + \tau I_M)^{-1} B_K^T y^\delta$$

Offline: precompute $B_K^T B_K$ and B_K^T; Online: solve $(M \times M)$ system for chosen τ

Cost: $O(M^2K)$ (offline), $O(MK + M^3)$ (online with Cholesky)

Figu

Gappy GEIM algorithm iv

Method 2: QR factorization

1. Offline: Compute QR decomposition $B_K = \tilde{Q}R$ with $\tilde{Q} \in \mathbb{R}^{K\times M}$, $R \in \mathbb{R}^{M\times M}$
2. Online: Solve $(R^T R + \tau I_M)\gamma^* = R^T \tilde{Q}^T y^\delta$

Cost: $O(M^2K)$ (offline), $O(MK + M^3)$ (online)

Method 3: SVD (most stable, expensive)

1. Offline: SVD $B_K = U\Sigma V^T$
2. Online: $\gamma^* = V(\Sigma^T\Sigma + \tau I_M)^{-1}\Sigma^T U^T y^\delta = \sum_{i=1}^{M} \frac{\sigma_i}{\sigma_i^2+\tau}(u_i^T y^\delta)v_i$

Cost: $O(M^2K)$ (offline), $O(MK)$ (online), best conditioning

Practical choice: Method 1 (direct) for $M < 100$, Method 3 (SVD) for ill-conditioned problems.

GEIM as optimal sensor selection i

Figu

Key insight: GEIM greedy algorithm automatically selects the "best" sensors from a large pool!

Sensor placement problem:

- **Available:** Large dictionary $\mathcal{L}$ with $N_s \gg M$ candidate sensors
- **Goal:** Select M sensors $\{L_{j_1}, \ldots, L_{j_M}\}$ that maximize information content
- **GEIM solution:** Greedy maximization of $|L_i(r_m)|$ at each iteration m

GEIM as optimal sensor selection ii

Figu

In practice:

Scenario: Temperature monitoring in a building with potential for $N_s = 500$ sensor locations.

1. Build candidate dictionary: $\mathcal{L} = \{L_1, \dots, L_{500}\}$
2. Generate snapshots: thermal fields for various scenarios (day/night, seasons, occupancy)
3. Run GEIM greedy: algorithm selects $M = 15$ sensors with maximum discriminating power
4. Result: indices $J = \{j_1, \dots, j_{15}\} \subset \{1, \dots, 500\}$

GEIM as optimal sensor selection iii

Figu

GEIM criterion: At step m, select sensor L_{j_m} that captures the most "energy" (largest residual):

$$j_m = \arg \max_{i \in \mathcal{L} \setminus J_{m-1}} |L_i(r_m)|$$

GEIM as optimal sensor selection iv

Why is this optimal?

- Each selected sensor adds maximum **new information** not captured by previous sensors
- Sensors are chosen to be **complementary**, not redundant
- Greedy selection ensures **well-conditioned** B_M matrix (small $\kappa(B_M)$)
- Selected sensors capture the **most energetic modes** of the phenomenon

Figu

Practical benefits:

- Deploy fewer sensors while maintaining accuracy
- Sensors automatically placed at "informative" locations
- Robust to variations in the parameter space
- Can incorporate sensor costs: restrict $\mathcal{L}$ to affordable locations

GEIM as optimal sensor selection v

Example: Comparing random vs GEIM sensor placement

Method	Sensors deployed	Reconstruction error
Random selection	30	8.2×10^{-3}
Uniform grid	25	5.4×10^{-3}
GEIM greedy	**15**	2.1×10^{-3}

Figu

Conclusion: GEIM achieves better accuracy with 50% fewer sensors!

Sensor dictionary design:

- Include all feasible sensor locations/types in $\mathcal{L}$
- GEIM will automatically select the most informative subset
- Can add constraints: priority regions, accessibility, cost

GEIM for data assimilation

Aim. Recover a hidden field (source, coefficient, or state) from indirect/noisy measurements.

1. Build reduced space X_M (basis Q) from model snapshots.
2. Choose a diverse sensor dictionary $\mathcal{L}$ (point probes, averages, Gaussians, PDE–informed functionals).
3. Greedy: select sensors L_i to stabilize inversion; precompute $B_M = A_J Q$.
4. Online: given y, solve $B_M \gamma = y$ (or gappy/regularized LS), then $\hat{g} = Q\gamma$.

With RB solvers: insert $\hat{g}$ into affine forms to enable fast forward/adjoint solves.

Figu

Example: Temperature field reconstruction i

Problem. Heat conduction with unknown source term $s(x;\mu)$:

$$-\nabla \cdot (\kappa \nabla u) = s(x;\mu) \quad \text{in } \Omega = [0,1]^2$$

Figu

Offline phase:

1. Generate $n_s = 100$ snapshots of source $s(x;\mu_i)$ with varying parameters $\mu_i \in \mathcal{D}$.
2. Build snapshot matrix $S = [s(\cdot;\mu_1)|\cdots|s(\cdot;\mu_{n_s})]$ on grid with $N_q = 10000$ points.
3. Apply GEIM greedy to select $M = 20$ sensors from dictionary:
   - 1000 candidate pointwise sensors: $L_i(g) = g(x_i)$
   - 100 box average sensors: $L_j(g) = \frac{1}{|\omega_j|}\int_{\omega_j} g\,dx$

Example: Temperature field reconstruction ii

Figu

- 50 Gaussian sensors: $L_k(g) = \int w_k(x)g(x)dx$

4. Result: $Q \in \mathbb{R}^{10000\times 20}$ (basis), $B_{20} = A_J Q$ (sensor-basis matrix)

Example: Temperature field reconstruction iii

Online phase:

1. Deploy $M = 20$ physical sensors at selected locations/regions.
2. Measure: $y_i = L_i(s(\cdot;\mu)), \quad i = 1, \ldots, 20$
3. Solve: $B_{20}\gamma = y \quad \Rightarrow \quad \gamma \in \mathbb{R}^{20}$
4. Reconstruct: $\hat{s}(x) = Q\gamma = \sum_{j=1}^{20} \gamma_j \rho_j(x)$
5. Insert into RB solver for thermal field $u_N(\mu) \approx u(\mu)$

Figu

Results:

- Reduction: $N_q = 10000 \to M = 20$ sensors (99.8% compression)
- Accuracy: $\|s - \hat{s}\|_{L^2} / \|s\|_{L^2} < 10^{-4}$
- Speedup: Full evaluation $\sim$ 5ms, GEIM reconstruction $\sim$ 0.1ms
- Robustness: Works with 5% measurement noise using Tikhonov regularization

Example: Temperature field reconstruction iv

Comparison with EIM/DEIM:

Figu

Method	Sensor type	M needed	Robustness
EIM	Point values	25	Poor (noise)
DEIM	Point values	22	Poor (noise)
GEIM	Mixed (avg+point)	20	Good

Why GEIM wins:

- Averaging sensors smooth out noise
- Flexibility to match available instrumentation
- Can incorporate physical constraints via sensor design

Example: Coefficient identification in heat equation i

Inverse problem. Identify spatially-varying conductivity $\kappa(x;\mu)$ from temperature measurements.

Figu

Forward model:

$$-\nabla \cdot (\kappa(x;\mu)\nabla u) = f \quad \text{in } \Omega, \qquad u|_{\partial\Omega} = g$$

Setup:

- Unknown: $\kappa(x;\mu) = \kappa_0(1 + 0.5\sum_{i=1}^{5} \mu_i \phi_i(x))$, $\mu_i \in [-1, 1]$
- Available: $m = 30$ temperature measurements $\{u(x_j)\}_{j=1}^{30}$ on boundary Γ_{obs}

Example: Coefficient identification in heat equation ii

GEIM approach:

Figu

1. Generate snapshots $\{\kappa(\cdot;\mu_k)\}_{k=1}^{200}$ with LHS sampling
2. Build basis $Q \in \mathbb{R}^{N_q \times M}$ with $M = 15$ via greedy GEIM
3. Sensors: Choose L_i to maximize sensitivity $\left|\frac{\partial u(x_j)}{\partial \kappa}\right|$
4. Solve inverse problem:

$$\min_{\gamma} \sum_{j=1}^{m} |u_{\text{meas}}(x_j) - u(x_j;\kappa = Q\gamma)|^2 + \tau\|\gamma\|^2$$

with PDE-constrained optimization (adjoint method)

Example: Coefficient identification in heat equation iii

Numerical results:

Figu

- True conductivity: 3 localized inclusions with $\kappa \in [0.5, 2.0]$
- GEIM reconstruction with $M = 15$: relative error $\|\kappa - \hat{\kappa}\|_{L^2} / \|\kappa\|_{L^2} = 2.3\%$
- Classical finite element discretization would need $N_q = 5000$ unknowns
- GEIM reduced optimization from $5000 \to 15$ parameters ($300\times$ speedup)

Key insight: GEIM provides low-dimensional parametrization enabling efficient solution of inverse/data assimilation problems.

Practical tips (sensor design)

Figu

- Start with a heterogeneous dictionary (subsampled Dirac, box averages, Gaussians at several widths).
- Avoid near-collinear sensors; monitor $\kappa(A_J Q)$ as m grows.
- Use overdetermined solves with mild Tikhonov when noise is non-negligible.
- For inverse problems: choose sensors to maximize observability (sensitivity analysis).
- Match sensor types to available physical instruments (thermocouples, flow meters, etc.).

Limitations and Open Challenges

EIM/DEIM limitations:

- Requires smooth parametric dependence (fails for discontinuous g)
- Lebesgue constant Λ_M can grow exponentially (up to $2^M - 1$)
- Magic points may cluster, leading to poor spatial coverage
- Offline cost grows with N_q and n_{train}

Figu

Gappy POD challenges:

- Choosing K and τ is problem-dependent (no universal rule)
- Overdetermination increases measurement cost
- SVD scales poorly for very large K ($K > 10^4$)

GEIM challenges:

- Sensor dictionary design is application-specific
- Greedy may select impractical sensor locations (accessibility, cost)

Summary and Recommendations

Method selection guide:

Use case	Recommended method
Function evaluations expensive	EIM (adaptive greedy)
Snapshots readily available	DEIM (optimal POD basis)
Noisy measurements	Gappy POD + Tikhonov
Physical sensors (non-pointwise)	GEIM
Inverse problems	GEIM (sensor selection)
Robustness critical	Gappy GEIM ($K > M$)

Figu

Key takeaways:

- All methods achieve exponential convergence for smooth manifolds
- Overdetermination ($K > M$) trades cost for robustness
- GEIM generalizes EIM/DEIM to arbitrary linear functionals

PBDW: measure the distance in a natural norm

Parametrized Background Data Weak (PBDW)

Figu

The PBDW method aims to achieve the most precise real-time approximation of the physical state given a parametrized model $\mathcal{P}^{bk}$ and a number of measurements.

Advantages

- Correction of uncertainties in $\mathcal{P}^{bk}$ or unmodeled physics thanks to an update of the best-knowledge approximation from M observations
- Non intrusive and non iterative method
- Efficient online computation for real-time state estimation
- No need to identify the model parameters to approximate the true physical state

PBDW: Model Setting

Figu

Consider a parametrized, linear elliptic PDE:

$$a(u(\mu), v; \mu) = f(v), \quad \forall v \in V$$

Key assumption: The true physical state u^{true} satisfies:

$$u^{\text{true}} = u(\mu^{\text{true}}) + \epsilon_{\text{model}}$$

where ϵ_{model} represents model error/unmodeled physics.

Manifold:

$$\mathcal{M} = \{u(\mu) : \mu \in \mathcal{D}\} \subset V$$

PBDW: Model + Data i

Figu

Given (perfect) observations

$$\mathrm{y}_{\mathrm{obs}} = \boldsymbol{\ell}\left(u^{\mathsf{true}}\right) \quad \in \mathbb{R}^M$$

find $u^* \in \mathcal{M}$ such that

$$u^* = \arg\min_{u \in \mathcal{M}} \text{“the distance between } u \text{ and } u^{\mathsf{true}}\text{”}$$

PBDW: Model + Data ii

Distance:

Figu

- Let $\mathrm{l}(u) = [\ell_1(u), \ell_2(u), \dots, \ell_M(u)]^T$ consist of linear functionals $\ell_i \in V'$
- Define the measurement space of observable states as

$$\mathcal{U}_M := \text{span}\{q_m, m = 1, \dots, M\}$$

where $q_m \in V$ are the **Riesz representers**:

$$(q_m, v)_V = \ell_m(v), \quad \forall v \in V$$

References: Bennett ('85), Maday, Patera, Penn & Yano ('14), Moore, Arango, Edwards ('17)

PBDW: Main Idea i

Entries:

1. a best-knowledge (bk) model (parametrized PDE)
2. M observations of the physical system

Figu

Goal: estimate the physical solution u^{true} at a reduced cost

Approach: Parametrized-Background Data-Weak (PBDW) for steady problems

- Maday, Patera, Penn, Yano ('14)
- Binev, Cohen, Dahmen, DeVore, Petrova, Wojtaszczyk ('16)
- Binev, Cohen, Mula, Nichols ('18)

PBDW: Main Idea ii

Figu

Idea: add a data-based correction term

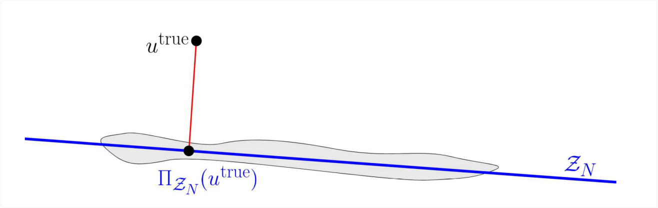
Ingredients:

- model-based background space $\mathcal{Z}_N$
- data-based observable space $\mathcal{U}_M$

PBDW Spaces i

Characterization of the bk model $u^{\text{true}} \notin \mathcal{Z}_N$

Figu



Interpretation

- $\mathcal{Z}_N = \text{Span}\{\zeta_1, \dots, \zeta_N\}$ renders anticipated uncertainty
- $\mathcal{U}_M = \text{Span}\{q_1, \dots, q_M\}$ captures non-anticipated uncertainty

PBDW Spaces ii

Figu

Background space $\mathcal{Z}_N$:

- POD, Greedy algorithm... etc

Observable space $\mathcal{U}_M$

- Collection of Riesz representations of linear observation functionals

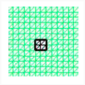
$$\ell_m^{\text{obs}}\left(u^{\text{true}}\right) = \left(q_m, u^{\text{true}}\right)$$
$$\mathcal{U}_M = \text{Span}\left\{q_1, \dots, q_M\right\}$$

- Example: Local uniform integration

$$\ell_m^{\text{obs}}\left(u^{\text{true}}\right) = \frac{1}{|\mathcal{R}_m|}\int_{\mathcal{R}_m} u^{\text{true}}(x)dx$$

PBDW Spaces iii

Figu



Parametrized Background Data Weak (PBDW) i

The approximation of the true physical state u^{true} is given by

$$u_{N,M} = z_N + \eta_M$$

where $z_N \in \mathcal{Z}_N$ and $\eta_M \in \mathcal{U}_M$.

To find z_N and η_M we solve the mixed problem:

$$\begin{aligned} (\eta_M, q)_V + (z_N, q)_V = (u^{\text{true}}, q)_V = y_m^{\text{obs}} \qquad & \forall q \in \mathcal{U}_M \\ (\eta_M, p)_V = 0 \qquad & \forall p \in \mathcal{Z}_N \end{aligned}$$

The corresponding algebraic problem is to find $\vec{\eta}_M$ and $\vec{z}_N$ such that

$$\begin{pmatrix} \mathrm{A} & \mathrm{B} \\ \mathrm{B}^T & 0 \end{pmatrix} \begin{pmatrix} \vec{\eta}_M \\ \vec{z}_N \end{pmatrix} = \begin{pmatrix} \vec{y}^{\text{obs}} \\ 0 \end{pmatrix}$$

where $(\vec{y}^{\text{obs}})_m = (q_m, u^{\text{true}})_V$, $\mathrm{A}_{m,m'} = (q_m, q_{m'})_V$ and $\mathrm{B}_{m,n} = (q_m, \zeta_n)_V$, for $1 \leq m, m' \leq M$ and $1 \leq n \leq N$.

Parametrized Background Data Weak (PBDW) ii

Figu

So that we can write

$$u_{N,M} = \underbrace{\sum_{n=1}^{N} (\vec{z}_N)_n \zeta_n}_{z_N \in \mathcal{Z}_N} + \underbrace{\sum_{m=1}^{M} (\vec{\eta}_M)_m q_m}_{\eta_M \in \mathcal{U}_M}$$

Computational cost:

- **Offline**: $\mathcal{O}(N_{\text{train}})$ PDE solves + matrix assembly
- **Online**: $\mathcal{O}((N+M)^3)$ for linear system solve

PBDW: Well-Posedness

Proposition (Maday et al., 2014)
The problem is well-posed with inf-sup constant $\beta_{N,M} > 0$ if and only if:

$$\mathcal{Z}_N \cap \mathcal{U}_M^\perp = \{0\}$$

Inf-sup constant:

$$\beta_{N,M} = \inf_{z \in \mathcal{Z}_N} \sup_{q \in \mathcal{U}_M} \frac{(z, q)_V}{\|z\|_V \|q\|_V}$$

Condition
We need $M \geq N$ and sensors must be “informative” w.r.t. the background space.

Computing the Inf-Sup Constant i

Derivation: Starting from the inf-sup definition

$$\beta_{N,M} = \inf_{z \in \mathcal{Z}_N} \sup_{q \in \mathcal{U}_M} \frac{(z, q)_V}{\|z\|_V \|q\|_V}$$

For $z = \sum_{n=1}^{N} (\vec{z})_n \zeta_n \in \mathcal{Z}_N$, the supremum over $q \in \mathcal{U}_M$ equals:

$$\sup_{q \in \mathcal{U}_M} \frac{(z, q)_V}{\|q\|_V} = \|\Pi_{\mathcal{U}_M} z\|_V$$

where $\Pi_{\mathcal{U}_M}$ is the orthogonal projection onto $\mathcal{U}_M$.

Computing the Inf-Sup Constant ii

Why does $\sup_{q \in \mathcal{U}_M} \dfrac{(z, q)_V}{\|q\|_V} = \|\Pi_{\mathcal{U}_M} z\|_V$?

Figu

- Decompose $z = \Pi_{\mathcal{U}_M} z + z^\perp$ where $z^\perp \perp \mathcal{U}_M$
- For any $q \in \mathcal{U}_M$: $(z, q)_V = (\Pi_{\mathcal{U}_M} z, q)_V$ since $(z^\perp, q)_V = 0$
- By Cauchy-Schwarz: $\dfrac{(z, q)_V}{\|q\|_V} = \dfrac{(\Pi_{\mathcal{U}_M} z, q)_V}{\|q\|_V} \leq \|\Pi_{\mathcal{U}_M} z\|_V$
- Equality when $q = \Pi_{\mathcal{U}_M} z$ (if non-zero), giving:

$$\sup_{q \in \mathcal{U}_M} \frac{(z, q)_V}{\|q\|_V} = \frac{(\Pi_{\mathcal{U}_M} z, \Pi_{\mathcal{U}_M} z)_V}{\|\Pi_{\mathcal{U}_M} z\|_V} = \|\Pi_{\mathcal{U}_M} z\|_V$$

Computing the Inf-Sup Constant iii

Why $\|\Pi_{\mathcal{U}_M} z\|_V^2 = \vec{z}^T \mathsf{B}^T \mathsf{A}^{-1} \mathsf{B}\, \vec{z}$?

The projection $\Pi_{\mathcal{U}_M} z = \sum_{m=1}^M \alpha_m q_m$ satisfies: find $\vec{\alpha}$ such that

$$(\Pi_{\mathcal{U}_M} z, q_{m'})_V = (z, q_{m'})_V \quad \forall m' = 1, \ldots, M$$

In matrix form: $\mathsf{A}\vec{\alpha} = \mathsf{B}\vec{z}$, hence $\vec{\alpha} = \mathsf{A}^{-1}\mathsf{B}\vec{z}$.

The projection norm is:

$$\begin{aligned}\|\Pi_{\mathcal{U}_M} z\|_V^2 &= (\Pi_{\mathcal{U}_M} z, \Pi_{\mathcal{U}_M} z)_V = \vec{\alpha}^T \mathsf{A}\vec{\alpha}\\ &= (\mathsf{A}^{-1}\mathsf{B}\vec{z})^T \mathsf{A}(\mathsf{A}^{-1}\mathsf{B}\vec{z})\\ &= \vec{z}^T \mathsf{B}^T \mathsf{A}^{-1} \mathsf{B}\, \vec{z}\end{aligned}$$

Computing the Inf-Sup Constant iv

Figu

Assuming $\{\zeta_n\}_{n=1}^N$ is **orthonormal** in V, we have $\|z\|_V^2 = \vec{z}^T\vec{z}$, thus:

$$\beta_{N,M}^2 = \inf_{\vec{z}\neq 0} \frac{\vec{z}^T \mathrm{B}^T \mathrm{A}^{-1} \mathrm{B}\,\vec{z}}{\vec{z}^T\vec{z}} = \lambda_{\min}(\mathrm{B}^T\mathrm{A}^{-1}\mathrm{B})$$

Computing the Inf-Sup Constant v

Computation of $\beta_{N,M}$:

- Let $\mathrm{C} = \mathrm{B}^T \mathrm{A}^{-1} \mathrm{B} \in \mathbb{R}^{N \times N}$
- Then, $\beta_{N,M} = \sqrt{\lambda_{\min}(\mathrm{C})}$ where $\lambda_{\min}(\mathrm{C})$ is the smallest eigenvalue of C

Non-orthonormal basis
If $\{\zeta_n\}$ is not orthonormal, let $\mathrm{G}_{n,n'} = (\zeta_n, \zeta_{n'})_V$. Then:

$$\beta_{N,M} = \sqrt{\lambda_{\min}(\mathrm{G}^{-1} \mathrm{B}^T \mathrm{A}^{-1} \mathrm{B})}$$

Interpretation:

- $\beta_{N,M}$ measures the alignment between $\mathcal{Z}_N$ and $\mathcal{U}_M$

Figu

Computing the Inf-Sup Constant vi

Figu

- Larger $\beta_{N,M}$ implies better stability and accuracy
- Sensor placement strategies can be designed to maximize $\beta_{N,M}$

Computing the Inf-Sup Constant: Python Implementation

```python
import numpy as np
from scipy.linalg import eigh, solve

def compute_infsup_constant(A, B, G=None):
    """
    Compute the inf-sup constant beta_{N,M} for PBDW.

    Parameters:
        A : (M, M) array - Gramian of U_M: A[m,m'] = (q_m, q_m')_V
        B : (M, N) array - Cross-Gramian: B[m,n] = (q_m, zeta_n)_V
        G : (N, N) array - Gramian of Z_N (None if orthonormal)

    Returns:
        beta : inf-sup constant
    """
    # Compute C = B^T A^{-1} B
    A_inv_B = solve(A, B, assume_a='pos')
    C = B.T @ A_inv_B

    if G is None:
        # Orthonormal basis: standard eigenvalue problem
        eigenvalues = eigh(C, eigvals_only=True)
    else:
        # Non-orthonormal: generalized eigenvalue problem
        eigenvalues = eigh(C, G, eigvals_only=True)

    lambda_min = np.min(eigenvalues)
    beta = np.sqrt(np.maximum(lambda_min, 0))
    return beta
```

PBDW: Error Bound

Figu

Theorem (A priori error estimate)
For noise-free observations, the PBDW approximation satisfies:

$$\|u^{true} - u_{N,M}\|_V \leq \left(1 + \frac{1}{\beta_{N,M}}\right) \inf_{z \in \mathcal{Z}_N} \|u^{true} - z\|_V$$

Interpretation:

- Error controlled by best approximation in $\mathcal{Z}_N$
- Stability factor $(1 + 1/\beta_{N,M})$ depends on sensor placement
- If $u^{\text{true}} \in \mathcal{Z}_N$, then $u_{N,M} = u^{\text{true}}$ (exact recovery)

PBDW with Noisy Observations i

Realistic setting: Observations are corrupted by noise

$$y_m^{\text{obs}} = \ell_m(u^{\text{true}}) + \epsilon_m, \quad |\epsilon_m| \leq \delta$$

Regularized PBDW (Maday, Mula, Patera, Yano, 2015)
Minimize the Tikhonov-regularized functional:

$$J_\lambda(z, \eta) = \|\mathrm{y}^{\text{obs}} - \mathrm{l}(z + \eta)\|_2^2 + \lambda \|\eta\|_V^2$$

subject to $z \in \mathcal{Z}_N$, $\eta \in \mathcal{U}_M$, $(\eta, p)_V = 0 \; \forall p \in \mathcal{Z}_N$

PBDW with Noisy Observations ii

Theorem (Noisy error bound)
With noise level δ and regularization λ:

$$\|u^{true} - u^{\lambda}_{N,M}\|_V \leq C_1 \inf_{z \in \mathcal{Z}_N} \|u^{true} - z\|_V + C_2(\lambda)\delta$$

Figu

Trade-off:

- Small λ: sensitive to noise but accurate for clean data
- Large λ: robust to noise but may over-regularize
- Optimal $\lambda \sim \delta$ (Morozov discrepancy principle)

PBDW: Overdetermined Case

Practical scenario: $M > N$ (more sensors than basis functions)

Figu

Least-squares formulation
Find $z_N \in \mathcal{Z}_N$ minimizing:

$$\|P_{\mathcal{U}_M}(u^{\text{true}} - z_N)\|_V^2 = \sum_{m=1}^{M} |\ell_m(u^{\text{true}}) - \ell_m(z_N)|^2 \|q_m\|_V^{-2}$$

Advantages:

- Better noise robustness through averaging
- Improved stability (larger effective $\beta_{N,M}$)
- Natural connection to Tikhonov regularization

Reference: Binev, Cohen, Dahmen, DeVore, Petrova, Wojtaszczyk (2017)

Practical Noise Handling in PBDW i

Sources of noise in practice:

- **Measurement noise**: sensor precision, electronic noise
- **Model error**: unmodeled physics, parameter uncertainty
- **Discretization error**: FE approximation

Figu

Regularized saddle-point system:

$$\begin{pmatrix} A + \lambda I & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} \vec{\eta} \\ \vec{z} \end{pmatrix} = \begin{pmatrix} \vec{y}^{\text{obs}} \\ 0 \end{pmatrix}$$

where $\lambda > 0$ is the regularization parameter.

Practical Noise Handling in PBDW ii

Choosing λ in practice:

1. **A priori**: $\lambda = \delta^2$ where δ is estimated noise level
2. **Morozov discrepancy**: choose λ such that residual $\approx$ noise level

$$\|\mathrm{y}^{\mathrm{obs}} - \mathrm{l}(u_{N,M}^{\lambda})\|_2 \approx \delta\sqrt{M}$$

3. **L-curve**: plot $\|\eta\|_V$ vs residual, choose corner (optimal trade-off)
4. **Cross-validation**: leave-one-out on sensors (data-driven, no δ needed)

Rule of thumb
Start with $\lambda \approx 10^{-2} \cdot \|\mathrm{A}\|$ and adjust based on residual.

Noise Estimation from Data

Problem: Often noise level δ is unknown!

Practical strategies:

1. Repeated measurements
If multiple measurements $y_m^{(k)}$ available:

$$\hat{\delta}_m^2 = \frac{1}{K-1}\sum_{k=1}^{K}(y_m^{(k)} - \bar{y}_m)^2, \quad \bar{y}_m = \frac{1}{K}\sum_{k=1}^{K} y_m^{(k)}$$

2. Residual-based estimation (overdetermined case)
With $M > N$, estimate from fit residual:

$$\hat{\delta}^2 \approx \frac{\|\mathrm{y}^{\mathrm{obs}} - \mathrm{I}(u_{N,M})\|_2^2}{M-N}$$

Robust PBDW: Practical Recommendations

For robust state estimation:

Figu

1. **Use** $M \gg N$: Overdetermined systems are more robust
   - Rule of thumb: $M \geq 2N$ to $3N$
2. **Sensor placement**: Maximize $\beta_{N,M}$
   - Avoid clustering sensors
   - Cover regions where $\mathcal{Z}_N$ has large variations
3. **Regularization**: Always use $\lambda > 0$ in practice
   - Even "clean" data has numerical noise
   - $\lambda \sim 10^{-6}$ to 10^{-2} depending on noise level
4. **Validation**:
   - Check $\|\mathrm{y}^{\mathrm{obs}} - \mathrm{l}(u_{N,M})\|_2 \lesssim \delta\sqrt{M}$
   - Monitor $\|\eta_M\|_V$ — large values indicate model deficiency

PBDW: Practical Implementation i

Figu

Offline phase (done once):

1. **Build background space $\mathcal{Z}_N$**:
   - Generate snapshots $\{u(\mu_i)\}_{i=1}^{n_{\text{train}}}$ by solving PDE
   - Apply POD/SVD: $\mathcal{Z}_N = \text{span}\{\zeta_1, \ldots, \zeta_N\}$
2. **Build observable space $\mathcal{U}_M$**:
   - Define sensor functionals ℓ_m (e.g., local averages, point values)
   - Compute Riesz representers: solve $(q_m, v)_V = \ell_m(v)$ for all $v \in V$
3. **Assemble matrices**:
   - $\mathrm{A}_{m,m'} = (q_m, q_{m'})_V$ (Gramian of $\mathcal{U}_M$)
   - $\mathrm{B}_{m,n} = (q_m, \zeta_n)_V$ (cross-Gramian)

PBDW: Practical Implementation ii

Online phase (real-time, for each new observation):

1. **Collect measurements**: $y_m^{\mathrm{obs}} = \ell_m(u^{\mathrm{true}}),\ m = 1, \dots, M$
2. **Solve saddle-point system**:

$$\begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} \vec{\eta} \\ \vec{z} \end{pmatrix} = \begin{pmatrix} \vec{y}^{\,\mathrm{obs}} \\ 0 \end{pmatrix}$$

3. **Reconstruct state**: $u_{N,M} = \sum_{n=1}^{N} z_n \zeta_n + \sum_{m=1}^{M} \eta_m q_m$

Complexity: $\mathcal{O}((N+M)^3)$ for dense solve, or $\mathcal{O}((N+M)^2)$ with Schur complement

Figu

PBDW: Schur Complement Approach

More efficient solution strategy:

From the saddle-point system, eliminate $\overrightarrow{\eta}$:

$$\overrightarrow{\eta} = \mathsf{A}^{-1}(\overrightarrow{y}^{\mathrm{obs}} - \mathsf{B}\overrightarrow{z})$$

Substitute into second equation to get the **Schur complement system**:

$$\underbrace{(\mathsf{B}^T\mathsf{A}^{-1}\mathsf{B})}_{\mathsf{S}\in\mathbb{R}^{N\times N}}\overrightarrow{z} = \mathsf{B}^T\mathsf{A}^{-1}\overrightarrow{y}^{\mathrm{obs}}$$

Algorithm:

1. **Offline**: Precompute A^{-1} (or Cholesky factor) and $\mathsf{S} = \mathsf{B}^T\mathsf{A}^{-1}\mathsf{B}$
2. **Online**:
   - Compute $\tilde{y} = \mathsf{A}^{-1}\overrightarrow{y}^{\mathrm{obs}}$ $\quad(\mathcal{O}(M^2))$
   - Solve $\mathsf{S}\overrightarrow{z} = \mathsf{B}^T\tilde{y}$ $\quad(\mathcal{O}(N^3)$ or $\mathcal{O}(N^2)$ with prefactorization)
   - Compute $\overrightarrow{\eta} = \tilde{y} - \mathsf{A}^{-1}\mathsf{B}\overrightarrow{z}$ $\quad(\mathcal{O}(NM))$

GEIM and PBDW: Synergy

Key insight: GEIM can be used to construct the observable space $\mathcal{U}_M$ for PBDW!

Figu

GEIM $\rightarrow$ PBDW Pipeline

1. **GEIM offline**: Select optimal sensors $\{\sigma_m\}$ and basis $\{q_m\}$ from dictionary
2. **Build $\mathcal{U}_M$**: Use Riesz representers of selected GEIM functionals
3. **Build $\mathcal{Z}_N$**: POD/Greedy on parametrized model solutions
4. **PBDW online**: Combine model + data for state estimation

Benefits:

- GEIM provides *optimal* sensor selection (greedy maximization)
- PBDW provides *rigorous* error bounds and noise handling
- Lebesgue constant Λ_M from GEIM relates to $\beta_{N,M}$ in PBDW

Comparison: GEIM vs PBDW

	GEIM	**PBDW**
Approach	Interpolation	Least-squares/Saddle-point
Background space	From snapshots	From parametrized model
Sensor selection	Greedy (offline)	Given or designed
Noise handling	Via regularization	Built-in (λ parameter)
Error bound	$(1 + \Lambda_M)\epsilon_{\text{best}}$	$(1 + 1/\beta_{N,M})\epsilon_{\text{best}}$
Model error	Implicit	Explicit correction η_M

Recommendation:

- Use GEIM for sensor selection when dictionary is available
- Use PBDW for rigorous state estimation with model correction
- Combine both: GEIM sensors + PBDW formulation

PBDW/GEIM: Key References

Foundational papers:

- Maday, Patera, Penn, Yano (2014): *PBDW approach to variational data assimilation*
- Maday, Mula, Patera, Yano (2015): *The PBDW method with noise*
- Binev et al. (2017): *Data assimilation in reduced modeling*

GEIM:

- Maday, Mula, Patera, Yano (2015): *The GEIM*
- Argaud, Bouriquet, Gong, Maday, Mula (2017): *Stabilization of GEIM*

Optimal sensor placement:

- Cohen, DeVore, Petrova, Wojtaszczyk (2013): *Optimal estimation*
- Binev, Cohen, Mula, Nichols (2018): *Greedy algorithms for measurements*