Clustering Reduced Order Models for Computational Fluid Dynamics

Gabriele Boncoraglio, Forest Fraser

Abstract

We introduce a novel approach to solving PDE-constrained optimization problems, specifically related to aircraft design. These optimization problems require running expensive computational fluid dynamics (CFD) simulations which have previously been approximated with a reduced order model (ROM) to lower the computational cost. Instead of using a single global ROM as is traditionally done, we propose using multiple piecewise ROMs, constructed and used with the aid of machine learning techniques. Our approach consists of clustering a set of precomputed non linear partial differential equations (PDE) solutions from which we build our piecewise ROMs. Then during the optimization problem, when we need to run a simulation for a given optimization parameter, we select the optimal piecewise ROM to use. Initial results on our test dataset are promising. We were able to achieve the same or better accuracy by using piecewise ROMs rather than a global ROM, while further reducing the computational cost associated with running a simulation.

1 Introduction

Improving the design of aircrafts often requires solving PDE-constrained optimization problems such as maximizing the $\frac{\text{lift}}{\text{drag}}$ with respect to some parameters, $\boldsymbol{\mu}$.

$$\begin{aligned} \max_{\boldsymbol{\mu}\in\mathcal{D}} \quad & \frac{\text{Lift}(\boldsymbol{\mu})}{\text{Drag}(\boldsymbol{\mu})} \\ \text{s.t.} \quad & \text{Lift}(\boldsymbol{\mu}) \geqslant \text{Lift}_0 \\ & \boldsymbol{\mu}_{\text{lb}} \leqslant \boldsymbol{\mu} \leqslant \boldsymbol{\mu}_{\text{ub}} \end{aligned} \tag{1}$$

Here $\boldsymbol{\mu}$ is an optimization vector containing parameters that we want to optimize. It is also common practice to have a lower bound $\boldsymbol{\mu}_{\text{lb}}$ and upper bound $\boldsymbol{\mu}_{\text{ub}}$ on this vector $\boldsymbol{\mu}$.

To find the optimal $\boldsymbol{\mu}_*$ we must update it iteratively, running a computational fluid dynamics (CFD) simulation at each optimization step. In figure (1) we can see the multiple queried points to find the optimal solution.

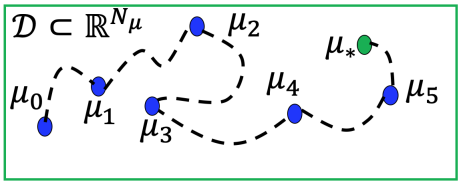

Figure 1: Optimization process

In our case, we want optimize the $\frac{\text{lift}}{\text{drag}}$ for the mAEWing2 glider with parameters $\boldsymbol{\mu} = [\mu_1, \mu_2, \mu_3] \in \mathcal{D} \subset \mathbb{R}^3$ where μ_1 modifies the dihedral angle of the wing and $\{\mu_2, \mu_3\}$ modify the sweep angle of the wing. Figure (2) shows how the different parameters modify the shape of the aircraft.

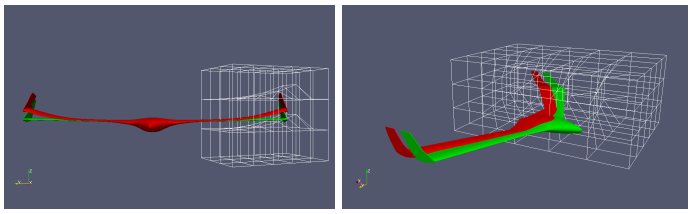

Figure 2: μ_1 changes the dihedral angle (left) and $\{\mu_2, \mu_3\}$ changes the sweep angle (right) of the mAEWing2

In figure (3) we can see the result of running a CFD simulation on this aircraft, where we are calculating how the pressure varies along the surface of the aircraft for a specific choice of $\boldsymbol{\mu}$.

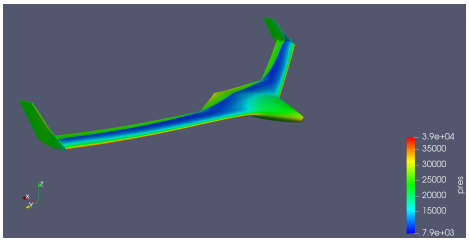

Figure 3: CFD simulation of the mAEWing2

CFD simulations are very computationally expensive. One technique in order to speed up CFD simulations is to use a reduced order model (ROM) [1]. The objective of this project is to accelerate the optimization process further by using clustering/classification techniques to generate and use multiple piecewise ROMs for less expensive, yet still accurate simulations.

1.1 High Dimensional Model (HDM)

Fluid flow problems are governed by nonlinear partial differential equations (PDE). Solving these equations, using CFD techniques such as finite volume method, is equivalent to solving a set of nonlinear equations:

$$\mathbf{r}(\mathbf{w}(\boldsymbol{\mu}), \boldsymbol{\mu}) = 0 \tag{2}$$

where $\boldsymbol{\mu}$ is the set of parameters for our simulation and $\mathbf{w}$ is the unknown vector of dimension N, $\mathbf{w} \in \mathbb{R}^N$, called the "state" vector. Specifically, a row of the state, $\mathbf{w}[i]$, represents a property of the fluid flow, such as pressure, at point

i of the CFD mesh. Thus, the CFD mesh has N points. In figure (4) we can see some of the points of the mesh of the mAEWing2.

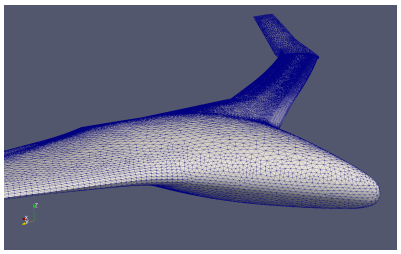

Figure 4: Mesh for the mAEWing2

Computing $\mathbf{w}$ allows us to compute the lift and drag generated by the mAEWing2 during the flight. This high dimensional model (HDM), Eq. (2), can be solved in a least squares sense by considering the following problem

$$\min_{\mathbf{w}\in\mathbb{R}^N} \|\mathbf{r}(\mathbf{w}(\boldsymbol{\mu}), \boldsymbol{\mu})\|_2^2 \tag{3}$$

Unfortunately, this problem is very expensive to solve when N is large, as it is in the case of solving CFD problems where N is in the order of thousands or millions.

1.2 Reduced Order Model (ROM)

In order to solve CFD problems faster, a reduced order model (ROM) can be used in order to approximate the HDM, (2), reducing the number of unknowns in Eq. (3) and hence reducing the cost of solving the least squares problem. The fundamental assumption made in reduced order modelling is that the state $\mathbf{w}$ belongs to an affine subspace of $\mathbb{R}^N$, where the dimension n of the affine subspace is typically orders of magnitude smaller than N. Therefore, we search for an approximated solution for $\mathbf{w}$ in the form

$$\mathbf{w}(\boldsymbol{\mu}) = \mathbf{V}_{gl}\mathbf{w}_r(\boldsymbol{\mu}) \tag{4}$$

where $\mathbf{V}_{gl} \in \mathbb{R}^{N\times n}$ denotes the global reduce order basis (ROB), and $\mathbf{w}_r \in \mathbb{R}^n$ denotes the new vector of unknowns, called the "reduced state". Substituting Eq. (4) into Eq. (2) results in the following system of N nonlinear equations in terms of n variables $\mathbf{w}_r$

$$\mathbf{r}(\mathbf{V}_{gl}\mathbf{w}_r(\boldsymbol{\mu}), \boldsymbol{\mu}) = 0 \tag{5}$$

Now the least squares problem to solve is

$$\min_{\mathbf{w}_r\in\mathbb{R}^n} \|\mathbf{r}(\mathbf{V}_{gl}\mathbf{w}_r(\boldsymbol{\mu}), \boldsymbol{\mu})\|_2^2 \tag{6}$$

1.2.1 Global Reduce Order Basis (ROB) $\mathbf{V}_{gl}$

To build a ROM we first need to find the global reduced order basis (ROB), $\mathbf{V}_{gl}$. This is done by solving the non linear equation (2) for many optimization vectors $\boldsymbol{\mu}$. Thus, given a specific vector $\boldsymbol{\mu}_i$ we can define the solution state vector:

$$\mathbf{r}(\mathbf{w}(\boldsymbol{\mu}_i), \boldsymbol{\mu}_i) = 0 \leftarrow \mathbf{w}(\boldsymbol{\mu}_i) \tag{7}$$

Therefore, for a set of k optimization vectors, $\{\boldsymbol{\mu}\}_1^k$, we solve (3) and we get a set of state vectors, $\{\mathbf{w}(\boldsymbol{\mu}_i)\}_1^k$. Once we have all the state vectors, we create the "solution matrix" M

$$M = \begin{bmatrix} \vdots & \vdots & & \vdots \\ \mathbf{w}(\boldsymbol{\mu}_1) & \mathbf{w}(\boldsymbol{\mu}_2) & \cdots & \mathbf{w}(\boldsymbol{\mu}_k) \\ \vdots & \vdots & & \vdots \end{bmatrix} \tag{8}$$

Finally, we perform a singular value decomposition (SVD) on the matrix M to compute the global ROB $\mathbf{V}_{gl}$:

$$\mathbf{M} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{D} = \begin{bmatrix}\mathbf{V}_{gl} & \mathbf{V}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{\Sigma}_{gl} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_2\end{bmatrix}\begin{bmatrix}\mathbf{D}_{gl} \\ \mathbf{D}_2\end{bmatrix} \tag{9}$$

Here, $\mathbf{V}_{gl}$ is computed by only selecting the first n columns of the matrix $\mathbf{U}$ and therefore $\mathbf{V}_{gl} \in \mathbb{R}^{N\times n}$. Thus, the global ROM has dimension n and can be used in the entire domain $\mathcal{D}$:

$$\mathbf{w}(\boldsymbol{\mu}) = \mathbf{V}_{gl}\mathbf{w}_r(\boldsymbol{\mu}), \quad \boldsymbol{\mu} \in \mathcal{D} \tag{10}$$

1.3 Piecewise ROMs in the Design Space

In this project we propose creating multiple piecewise ROMs in the domain $\mathcal{D}$, each having smaller dimensions than a global ROM would. These piecewise ROMs do not need to be accurate in the entire domain $\mathcal{D}$ but only in limited region of the design space $\mathcal{D}$. By using machine learning techniques we hypothesize that we can improve quality of the reduced order basis (ROB) within a given design space $\mathcal{D}$. Then, using these piecewise ROMs, will allow us to solve even cheaper least squares problems than (6), whilst maintaining a similar or better level of accuracy relative to the HDM.

To do this we must group the precomputed solutions $\{\mathbf{w}(\boldsymbol{\mu}_i)\}_1^k$ into multiple clusters and then create multiple piecewise reduced order basis $\{\mathbf{V}_i\}_1^c$ where c is the number of clusters. For instance, choosing two clusters, in figure (5) we can see a schematic comparison of a global ROM versus 2 piecewise ROMs built by clustering $\{\mathbf{w}(\boldsymbol{\mu}_i\}_1^k$ into 2 clusters. On the left, all the training solutions $\{\mathbf{w}_i\}_1^{10}$ computed solving (2) using $\{\boldsymbol{\mu}_i\}_1^{10}$ are used to create $\mathbf{V}_{gl}$ and therefore a global ROM. On the right, we first cluster the training solutions $\{\mathbf{w}_i\}_1^{10}$ into 2 clusters and then we construct 2 reduced order basis $\mathbf{V}_1$ and $\mathbf{V}_2$. $\mathbf{V}_1$ is built using the solutions computed using the parameters $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_5, \boldsymbol{\mu}_6, \boldsymbol{\mu}_7, \boldsymbol{\mu}_{10}\}$ and $\mathbf{V}_2$ using $\{\boldsymbol{\mu}_2, \boldsymbol{\mu}_3, \boldsymbol{\mu}_4, \boldsymbol{\mu}_8, \boldsymbol{\mu}_9\}$.

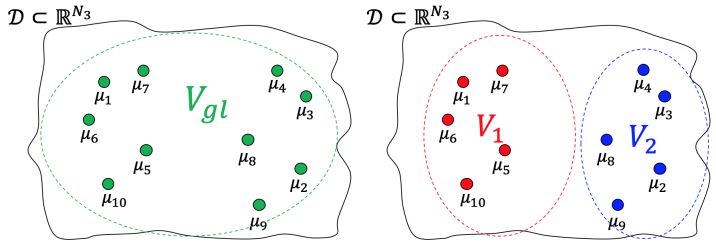

Figure 5: On the left global ROM approach. On the right our proposed piecewise ROM approach

As such, the global ROM uses $\mathbf{V}_{gl} \in \mathbb{R}^{N\times n}$. The two piecewise ROMs, instead use $\mathbf{V}_1 \in \mathbb{R}^{N\times n_1}$ and $\mathbf{V}_2 \in \mathbb{R}^{N\times n_2}$

respectively, where, by construction $n_1 < n$ and $n_2 < n$. Therefore, the first piecewise ROM makes the following approximation using $\mathbf{V}_1$

$$\mathbf{w}(\boldsymbol{\mu}) = \mathbf{V}_1 \mathbf{w}_r(\boldsymbol{\mu}) \tag{11}$$

and the second piecewise ROM makes another approximation using $\mathbf{V}_2$:

$$\mathbf{w}(\boldsymbol{\mu}) = \mathbf{V}_2 \mathbf{w}_r(\boldsymbol{\mu}) \tag{12}$$

Therefore by using either (11) or (12) we can solve

$$\min_{\mathbf{w}_r \in \mathbb{R}^n} \|\mathbf{r}(\mathbf{V}_i \mathbf{w}_r(\boldsymbol{\mu}), \boldsymbol{\mu})\|_2^2 \tag{13}$$

where i indicates which piecewise ROM i is used.

Using this method with piecewise ROMs gives rise to two machine learning problems that we must solve.

1. Given multiple precomputed solutions $\{\mathbf{w}_i\}_1^k$, how do we cluster them most effectively into $\{\mathbf{V}_i\}_1^c$?
2. Given an arbitrary $\boldsymbol{\mu}$, which piecewise ROM $\{\mathbf{V}_i\}_1^c$ should we use to best represent the HDM?

In the next section we describe the methods we have implemented for addressing the above problems.

2 Methods

2.1 Overview

Our proposed methodology to solve a PDE-constrained optimization problem operates in two phases, an offline phase and an online phase. In the offline phase, we cluster precomputed training solutions, from which we build our piecewise ROMs that are used in the online phase. Figure (6) shows the outline of this phase.

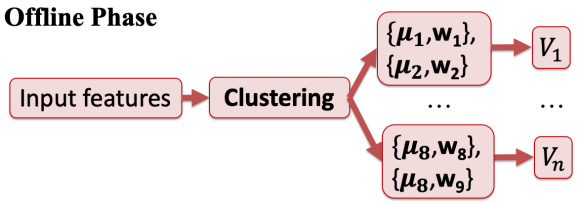

Figure 6: Offline phase scheme

In the online phase, we query multiple $\boldsymbol{\mu}$, during the optimization process. For each queried $\boldsymbol{\mu}_i$, we need select which piecewise ROM ($\mathbf{V}_i$) to use. Then we run the simulation to compute $\mathbf{w}(\boldsymbol{\mu}_i)$ and $\frac{\text{lift}}{\text{drag}}(\boldsymbol{\mu}_i)$. Figure (7) shows the outline of this phase.

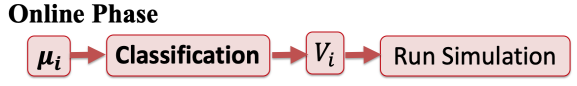

Figure 7: Online phase scheme

2.2 Clustering

Since our goal is to break our domain $\mathcal{D}$ into smaller sub domains, we believe clustering the training points based on the euclidean distance between features will be most effective. We have applied three algorithms in order to implement this; **K-Means**, **Expectation-Maximization** (to fit a gaussian mixture model) and **Agglomerative Clustering**.

In terms of clustering features, we have considered using $\boldsymbol{\mu}$ in addition to the $\frac{\text{lift}}{\text{drag}}$ and $\frac{\partial \frac{\text{lift}}{\text{drag}}}{\partial \boldsymbol{\mu}}$ from our precomputed solutions. Intuitively if two vectors $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ are close together in terms of euclidean distance, then fluid flow properties should also be similar. Therefore, they should be clustered together as the resulting ROB can efficiently represent both $\mathbf{w}(\boldsymbol{\mu}_1)$ and $\mathbf{w}(\boldsymbol{\mu}_2)$. We considered $\frac{\text{lift}}{\text{drag}}$ and $\frac{\partial \frac{\text{lift}}{\text{drag}}}{\partial \boldsymbol{\mu}}$ a features as they provide more information on the physics problem we are trying to solve and obviously lift and drag are both related to the fluid state.

2.3 Classification

The number of training points used when constructing ROMs are relatively low when compared with other machine learning problems. Additionally, we do not have ground truth values for our classifications, and only are able to determine how well our algorithms performs after doing a ROM simulation. Therefore we have chosen to evaluate two simple methods, **Nearest Centroid** and **Multinomial Logistic Regression** as our algorithms for performing classifications. For Nearest Centroid, we simply select the piecewise ROM whose clustered points have the closest centroid to the queried point, while Multinomial Logistic regression is trained using a cross entropy loss and the labels output from clustering during the offline phase. Since during the online phase we will not have access to the lift or drag for a given query point, we are only able to use $\boldsymbol{\mu}$ as a feature for classification.

3 Design of Experiment

In order to to create a train/validation/test set we sampled the parameter domain $\mathcal{D} \subset \mathbb{R}^3$ to compute 90 solutions $\{\mathbf{w}(\boldsymbol{\mu}_i)\}_1^{90}$. The sampling strategy adopted here is to use latin hypercube sampling in order to generate controlled random samples. Using this sampling strategy we created 90 vectors $\{\boldsymbol{\mu}_i\}_1^{90} \in \mathcal{D} \subset \mathbb{R}^3$. Once we created this set of vectors, we also compute the solutions of the HDM for each $\boldsymbol{\mu}_i$, thus $\{\mathbf{w}(\boldsymbol{\mu}_i)\}_1^{90}$. We then randomly split our data into training/validation/test sets of size 50, 30 and 10 respectively. In our case, the validation set is used for finding the optimal clustering and classification algorithms and parameters, while the test set is used to do one final comparison of our piecewise ROM versus a global ROM.

4 Experiments Results

For all of the following experiments we define our error to be the difference in $\frac{\text{lift}}{\text{drag}}$ calculated with a ROM and $\frac{\text{lift}}{\text{drag}}$ calculated with the HDM. We refer to MSE as the mean squared error across our test points and Max Error % as the highest percentage deviation from the $\frac{\text{lift}}{\text{drag}}$ calculated with the HDM.

$$\text{Error} = \left| \frac{\left(\frac{\text{lift}}{\text{drag}}_{\text{ROM}} - \frac{\text{lift}}{\text{drag}}_{\text{HDM}} \right)}{\frac{\text{lift}}{\text{drag}}_{\text{HDM}}} \right| \tag{14}$$

4.1 Model Parameter Experiments

For our first set of experiments we determine the best parameters for our methodology, given our design space. Specifically, we use the validation set to determine the:

- clustering algorithm
- classification algorithm
- clustering features
- number of clusters

For each experiment we ran three tests, each with 20 training points folded from our total 50 training points and test the error on our validation set. In subsection (4.3) we investigate using a predictor to automatically determine our parameters without having to run any simulations on a validation set.

First we tested for the best clustering algorithms; fixing our classification algorithm to Nearest Centroid, the number of clusters to 4, and clustering features to $\{\boldsymbol{\mu}, \frac{\text{lift}}{\text{drag}}\}$.

Clustering analysis	MSE	Max Error %
K-Means	0.327	23.117
Gaussian mixtures	0.421	28.898
Agglomerative clus.	0.340	23.917

Figure 8: Error of different clustering methods

Second, we tested various different classification algorithms; fixing our clustering algorithm to K-Means, the number of clusters to 4, and clusters features to $\{\boldsymbol{\mu}, \frac{\text{lift}}{\text{drag}}\}$.

Classification algorithm	MSE	Max Error %
Logistic regression	0.341	29.490
Nearest centroid	0.318	29.490

Figure 9: Error of different classification methods

Next, we determined the best cluster features, fixing the clustering algorithm to K-Means, the classification algorithm to Nearest Centroid, and number of clusters to 4.

Cluster features	MSE	Max Error %
$\left(\boldsymbol{\mu}, \frac{\text{lift}}{\text{drag}}\right)$	0.3543	30.955
$\left(\boldsymbol{\mu}, \frac{\partial \frac{\text{lift}}{\text{drag}}}{\partial \boldsymbol{\mu}}\right)$	0.3145	30.955
$\left(\boldsymbol{\mu}, \frac{\text{lift}}{\text{drag}}, \frac{\partial \frac{\text{lift}}{\text{drag}}}{\partial \boldsymbol{\mu}}\right)$	0.3409	30.955

Figure 10: Error with different clustering features

Finally, we tested the effect of using different numbers of clusters. We used K-Means for clustering, Nearest Centroid for classification, and $\{\boldsymbol{\mu}, \frac{\text{lift}}{\text{drag}}\}$ as the clustering features.

Number of clusters	MSE	Max Error %
2 clusters	0.255	23.072
3 clusters	0.288	23.520
4 clusters	0.354	26.133
5 clusters	0.417	29.826

Figure 11: Error with different numbers of clusters

From the above experiments we see that K-Means with features $\{\boldsymbol{\mu}, \frac{\partial \frac{\text{lift}}{\text{drag}}}{\partial \boldsymbol{\mu}}\}$, Multinomial Logistic Regression performs better than Nearest Centroid and lower numbers of clusters reduce the error.

4.2 Global ROM Comparison

With our optimal clustering/classification algorithms and features derived from subsection (4.1), and clusters of size 2 and 4, we tested the accuracy of our methodology versus a global ROM approach for calculating the $\frac{\text{lift}}{\text{drag}}$, the objective function of the optimization problem. For the test with two clusters, we show the offline clustering phase in figure (12).

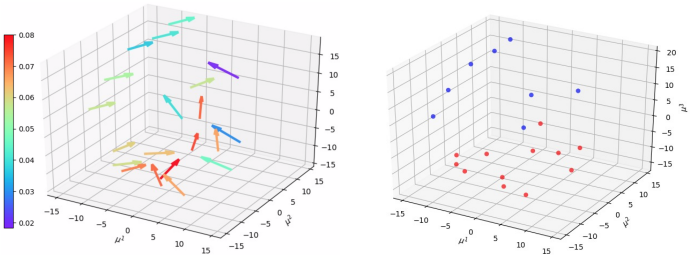

Figure 12: On the left, gradient $\frac{\partial \frac{\text{lift}}{\text{drag}}}{\partial \boldsymbol{\mu}}$ of the training points. On the right, training points clustered into two clusters.

In figure (13) we show how the test points are labeled after classification, in order to chose which ROMs to use for the simulation.

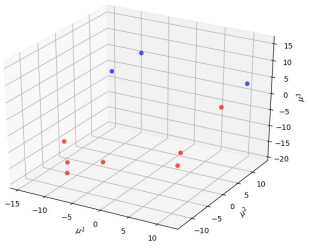

Figure 13: Testing point labeled after classification.

Method	# Clusters	ROM Sizes	MSE	Max Error %
Cluster	2	(12, 8)	0.051	7.726
Cluster	4	(5, 7, 5, 3)	0.200	17.439
Global	-	14	0.194	21.156

Figure 14: Relative error with different methods

The results in table (14) show that the cluster method either with two clusters or four clusters is able to have the same or better accuracy of the global ROM using a smaller ROM size. Here the ROM size indicates the number of columns of the reduced order basis (ROB), $\mathbf{V}_i$, used for the ROM approximation $\mathbf{w}(\boldsymbol{\mu}) = \mathbf{V}_i \mathbf{w}_r(\boldsymbol{\mu})$.

4.3 Predictor for ROM Accuracy

In practice, users would not want to have to use a validation set to determine the best clustering parameters, as the time required to do this may outweigh any efficiency savings from using piecewise ROMs. Therefore a predictor for a reliable set of clustering parameters is necessary for real world applications. We tested different cluster scoring methods, including Silhouette Score [2], Calinski-Harabaz Index [3] and Davies-Bouldin Index [4], on all combinations of the clustering parameters described in subsection (4.3). Each combination was trained using 20 points from our training set, and the error calculated our validation set. Extreme outliers in terms of relative error were also removed. We then calculated the Pearson correlation coefficient for each scoring method, relative to the average error of the approximated $\frac{\text{lift}}{\text{drag}}$ for our validation points.

Cluster Scoring Method	Correlation
Silhouette Score	-0.68548199
Calinski-Harabaz Index	-0.68738437
Davies-Bouldin Index	-0.50457497

Figure 15: Clustering score correlation with relative error

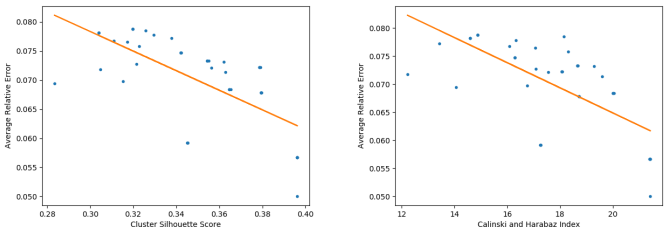

Figure 16: On the left, Silhouette Score versus average relative error. On the right, Calinski-Harabaz Index versus average relative error.

5 Discussion

From our results we see that the K-Means and Agglomerative clustering perform somewhat similarly, compared to the Gaussian Mixture Model. This makes sense as the points used in the offline phase are not necessarily Gaussian distributed, while K-Means and Agglomerative clustering less strong of an assumption. As for clustering features, the difference between feature sets is relatively small. This makes sense as for many points in the design space our optimization vector, $\boldsymbol{\mu}$ will be highly correlated with the lift and drag.

We are also able to see an interesting trade off when it comes to the number of clusters used. From the results we can clearly see that the error decreases with the number of cluster sizes. This is sensible because as we increase the number of clusters, the number of points are assigned points used to create each ROB decreases, decreasing the accuracy of the ROM approximation. However as the number of points used to build the ROB decreases, so does the computation cost of running a simulation with the corresponding ROM. Therefore the number of clusters used should be chosen on a per application basis, where the user would select the number of clusters corresponding to the acceptable error.

Overall, we can see that the our proposed methodology is superior when compared with using a global ROM. We can see that we are either able to get a much higher accuracy than the global ROM with a similar computational cost (related to the ROM size), or we are able to achieve a similar accuracy with half the computation cost of the global ROM.

With regards to predictors for parameter selection, we can see that all three cluster scoring methods show some indication that they could be used as a predictor for cluster ROM accuracy, at least for our design space. Silhouette Score and the Calinski-Harabaz Index may be slightly more correlated than the Davies-Bouldin as the distance between points on the edges of clusters are reflected in their scores, rather then only accounting for the distances between cluster centroids. However more rigorous testing is needed, especially we do not know if it will generalize to other PDE-constrained optimization problems.

6 Conclusion & Future Work

In conclusion, we present a novel approach to solving PDE-constrained optimization problems by utilizing multiple piecewise ROMs. This approach has proven to be both more accurate and more computationally efficient than using a single global ROM. It shows particularly strong promise for time constrained applications with high dimensional design spaces. In these scenarios, the global ROM would need to be very large in order to be accurate across the whole design space and thus it might not be able to meet real-time deadlines. Piecewise ROMs on the other hand, can be more efficient and thus able to meet the timing constraints.

We would like to continue testing the performance of our approach in more realistic, higher dimensional design spaces (50-60 parameters). For this project we chose a limited design space due to time constraints, as running tests in higher dimensional design spaces is naturally more computationally expensive and takes more time. We would also like to continue research on predictors for clustering effectiveness, as this is a key component for this approach to be practical in real world problems.

References

[1] Kyle M. Washabaugh, Matthew J. Zahr, and Charbel Farhat. (2016). "On the Use of Discrete Nonlinear Reduced-Order Models for the Prediction of Steady-State Flows Past Parametrically Deformed Complex Geometries", 54th AIAA Aerospace Sciences Meeting, AIAA SciTech Forum, (AIAA 2016-1814).

[2] Peter J. Rousseeuw (1987). "Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis". Computational and Applied Mathematics 20: 53–65.

[3] Caliński, T., Harabasz, J. (1974). "A dendrite method for cluster analysis". Communications in Statistics-theory and Methods 3: 1-27.

[4] Davies, David L.; Bouldin, Donald W. (1979). "A Cluster Separation Measure" IEEE Transactions on Pattern Analysis and Machine Intelligence. PAMI-1 (2): 224-227.

7 Contributions

The first part of this project was discussing and creating a new methodology for solving PDE-constrained optimization problem. This was a significant part of the project where both Forest and Gabriele discussed on the optimal approach to take. To implement this methodology, Gabriele wrote code to build and run ROMs from a set of training points in addition to writing code to generate the data to start the experiments. Forest was responsible for implementing the machine learning algorithms from external libraries as well as automating testing. Both Gabriele and Forest contributed towards research and decision making for the use of machine learning techniques in this project in addition writing routines to output and post-process results for analysis.

8 Code

Unfortunately the code must be run on a super computer with many external libraries from the Stanford Aeronautics & Astronautics department. We have included a zip file containing the only the code written for this project available at:

https://drive.google.com/file/d/1BP4iW6RIR_Cn3hxWL58cF-Pi5XppSI4W/view?usp=sharing