Scientific Computing Group
Department of Computer Science
University of Kaiserslautern

Bachelorthesis

Implementation of a POD Based Surrogate Model for Aerodynamic Shape Optimization

Patrick Mischke

March 24, 2019

Bachelorthesis

Implementation of a POD Based Surrogate Model for Aerodynamic Shape Optimization

Scientific Computing Group
Department of Computer Science
University of Kaiserslautern

Patrick Mischke

Day of issue : 02.10.2018
Day of release : 25.03.2019

First Reviewer : Prof. Dr. Nicolas R. Gauger
Second Reviewer : Dr. Emre Özkaya
Supervisor : Dr. Emre Özkaya

Hereby I declare that I have self-dependently composed the Bachelorthesis at hand. The sources and additives used have been marked in the text and are exhaustively given in the bibliography.

March 24, 2019 – Kaiserslautern

(Patrick Mischke)

Preface

> I don't do a lot of public speaking, so I looked up a memorable quote to start my speech, and this is what I found. OK, you're staring at me blankly, but this whole thing is a quote. I know that sounds confusing, but... You know what, never mind. ([xkcd, 2018])

As student of the computer science and physics department, who is also interested in mathematics, the lecture on "Optimization in Fluid Mechanics" by Prof. Gauger, combining all three fields, naturally caught my interest. I got the permission to contribute hislectures from that field, that are usually part of the master studies, to my computer science bachelor. Finally I decided to ask for a project suitable for a bachelor thesis in his Scientific Computing Group.

I'd like to thank Emre for coming up with the interesting topic and his profound help whenever I got stuck or something didn't work the way it should. Working with the state of art fluid simulation library SU^2 gave me valuable insights in the field of scientific computing.

Abstract

Computational Fluid Dynamics (CFD) simulations are often used with aerodynamic shape optimization in mind. There exist several approaches to perform optimization, for example employing one shot methods or using adjoint computations by utilizing algorithmic differentiation techniques while running the flow solver. However, it may sometimes be preferred to inspect the flow fields across the whole design space, for instance if those strategies seem to operate too local for the problem on hand. This issue can be targeted by the use of surrogate models. A surrogate model predicts the flow field for a given design input in a computationally cheaper way, but also with less accuracy than the full fluid dynamics simulation. The surrogate model uses a set of training data to find reasonable model parameters for future design inputs. The goal of this bachelor thesis is the implementation of an surrogate model for CFD using Proper Orthogonal Decomposition (POD) and Kriging regression. Within this thesis the challenges encountered are discussed, and the NACA 0012 airfoil at transonic flow conditions is presented as test case. While the characteristic shock front and the high number of design parameters describing the airfoil geometry of the given test case lead to a rather expensive training process for the surrogate model, it may still be valuable for other geometries or as foundation for surrogate models using other techniques.

Zusammenfassung

Das Thema dieser Bachelorarbeit ist die Implementierung eines Ersatzmodells unter Verwendung von „Proper Orthogonal Decomposition“ (POD) im Anwendungsfeld der computergestützten aerodynamischen Optimierung (engl Computational Fluid Dynamics (CFD)). Fluiddynamiksimulationen werden regelmäßig zur Optimierung von Geometrien im Bereich der Aerodynamik eingesetzt, weshalb hierauf eines der Hauptaugenmerke von CFD Bibliotheken liegt. Hierfür existieren verschiedenste Tools, beispielsweise unter Verwendung von One-Shot Methoden oder algorithmischem Differenzieren. Abhängig vom betrachteten Problem ist es teilweise wünschenswert, das Strömungsverhalten über den gesamten Raum der Designparameter hinweg zu untersuchen. Dies kann insbesondere der Fall sein, falls übliche Optimierungsstrategien in lokalen Optima konvergieren. In solchen Fällen können Ersatzmodelle gewinnbringend eingesetzt werden. Ersatzmodelle versuchen das Strömungsverhalten für gegebene Designparameter unter erheblich geringerem Einsatz von Rechenleistung, dafür aber auch mit niedrigerer Genauigkeit, vorherzusagen, als dies mit Hilfe einer vollständigen Fluiddynamiksimulation möglich ist. Typischerweise erstellen sie hierzu eine Regression zwischen vorhandenen Trainingsdaten. Das Ziel dieser Bachelorarbeit war, ein solches Ersatzmodell für die Fluiddynamiksimulationen von SU^2 unter Verwendung von POD und Kriging Regression zu entwickeln. Diese Ausarbeitung diskutiert die Herausforderungen die sich hierbei ergeben haben anhand des Beispiels eines NACA 0012 Tragflächenprofils im Überschallbereich. Aufgrund der charakteristischen Überschallschockfront im Strömungsfeld, sowie der relativ hohen Anzahl an Designparametern, die die Form der Tragfläche beschreiben, ist der Trainingsaufwand für diesen Testfall verhältnismäßig groß. Dies führt dazu, dass der Gesamtnutzen für diesen Beispielfall eher als gering zu bewerten ist. Nichtdestotrotz könnte die Methodik für andere Probleme gewinnbringend eingesetzt, oder als Grundlage für die Entwicklung von Ersatzmodellen, die weiterführende Ansätze verfolgen, verwendet werden.

Contents

1. Introduction

As CFD simulations are computationally expensive, it is desired to reduce the number of function evaluations to a minimum while performing shape optimization. One approach is the use of surrogate models, which aim to predict the outcome for a given design in a less expensive way. Surrogate models can be divided into two kinds: Physical based surrogate models still utilize properties of fluid physics to some degree, for instance by using a coarser mesh or simplified equations. General surrogate models treat the process of flow simulation as black box, and do not use any information on the physics involved. They use set of training data to predict the outcome at new design sites. They are, from a fundamental perspective, a regression between those training points.

The goal of this bachelor thesis is the implementation of such a general surrogate model and testing it on the NACA 0012 airfoil at inviscid supersonic conditions. The design parameters for this test case are the amplitudes of 38 Hicks-Henne bump functions at fixed positions, used to deform the airfoil geometry. The flow field contains flow parameters at 5,233 mesh point locations. The implementation involves three steps:

The training data needs to be gathered using CFD. For that purpose the SU^2 software package has been used. SU^2 describes itself on its website as follows: "SU2 is an open-source collection of software tools written in C++ and Python for the analysis of partial differential equations (PDEs) and PDE-constrained optimization problems on unstructured meshes with state-of-the-art numerical methods. SU2 is a leading technology for adjoint-based optimization. Through the initiative of users and developers around the world, SU2 is now a well established tool in the computational sciences with wide applicability to aeronautical, automotive, naval, and renewable energy industries, to name a few." More details about the package can be found in [Economon et al., 2015].

Then Proper Orthogonal Decomposition (POD) is used as data compression method to express the flow field within a reduced number of variables. Finally a Kriging model is used to perform the actual regression.

All data, including source code files, model and simulation data, is available at `https://github.com/Patschke/POD-Surrogate`.

The topic of this thesis was inspired by the work of [Iuliano and Quagliarella, 2013], where a POD based surrogate model has been presented for optimization of the RAE 2822

airfoil. Therefore, the overall approach follows the implementation described therein. [Iuliano and Quagliarella, 2013] also discusses other papers which attempt to utilize POD based approaches for surrogate modeling in the field of aerodynamic shape optimization. However, surrogate models using POD seem to be a more common strategy on related topics like the time dependent description of unsteady flow fields or using other input parameters like Mach number or angle of attack.

[Tesfahunegn et al., 2015] presents a comparison on shape optimization for the NACA 0012 airfoil using different techniques, including surrogate models and adjoint optimization using SU^2. They find, that the general surrogate model approach is by a factor of 20 more expensive than using adjoint techniques, but may have the benefit of allowing for a global search. Therefore, the goal of this thesis is not to speed up the optimization for this test case, but rather the pure implementation and testing of the surrogate model using SU^2, POD and Kriging Regression.

2. Sample Data Acquisition

2.1 Sample Distribution

The choice of sample data used to create a surrogate model is of fundamental importance, and depends heavily on the purpose of the model. If the set of sample data does not already exist, one may choose the points in the design parameter space where samples should be created. A full simulation is computed for that sample points, and used as the input data for the training of the surrogate model. As those simulations usually require high computational effort, the proper choice of sample points should be carefully considered. It may be sufficient to sample a small, specific region of interest, or distribution of sampling points over the whole design space might be needed to create a global surrogate model. Once the region has been chosen, the distribution of samples across that region may still vary between uniformly spread sample points, or importance weighted sampling around points of interest. In general, the model will be more accurate in regions which contain more sample points, and less accurate in predictions far away from the nearest sample. [Iuliano and Quagliarella, 2013, p. 35] suggests to use at least 10 times the design space dimension as sample size.

Without further information on the intention of the surrogate model and possible regions of interest, it is reasonable to assume that a uniform distribution of sample points across the whole input domain should be achieved. The naive approach of choosing random sample points is not the best option for that scenario. While the underlying randomness might be uniformly distributed in general, the finite number of sample points created is not necessarily so. Some parameters may not be sampled uniformly at all. There exists a solution for that problem called Latin Hypercube Sampling (LHS) that was proposed by [McKay et al., 1979]. LHS guarantees that each parameter is represented evenly across its whole domain. To do so, the design space is split into k^n equally sized k-dimensional hypercubes (cells), where k is the number of design parameters and n the number of samples to be taken. Sample points are then randomly chosen from that cells without repeating a cell index of any parameter. However LHS has the drawback of possibly causing correlations between the input parameters. The aerodynamics may heavily depend on the interaction between parameters, so this correlations can be an issue, as they will influence

the surrogate model. Using smarter sampling techniques may improve the resulting surrogate model further.

2.2 Sample Properties

There exist several physical properties, that are computed by aerodynamic flow solvers as part of the flow field. These are typically pressure, density, temperature and momentum. However, not all of them are always necessary for the surrogate model. Thus, one has to carefully evaluate, which properties should be included for the surrogate model. This choice will have great effect on the complexity of the model order reduction, as described in Section 3.2. For each sample point all included data (usually one float per property and mesh point) is concatenated to a single vector y, also called snapshot vector. In a more formal notation, the state vectorfor one sample point reads as follows:

$$y = (\rho_1, ..., \rho_m, p_1, ..., p_m, \rho v_{x1}, ..., \rho v_{xm}, \rho v_{y1}, ..., \rho v_{ym}) \tag{2.1}$$

Only those vectors and the input setting vectors for the samples are used to construct the surrogate model, resulting in high modularity of the process. Of course, as a result, the output of the surrogate model will only be such a data vector, and a template containing common data (e.g. the mesh structure) is needed to convert it back to a flow file.

[Iuliano and Quagliarella, 2013] claims, that including the coordinates of the mesh grid points in the same way as the physical data is beneficial for the resulting surrogate model:

> The snapshot structure is conceived as a combination of mesh coordinates (i.e. spatial locations) and flow field variables (i.e. state vectors). The idea is to let the POD basis catch the coupling effects between space location and state field. Hence, once the surrogate model is built, not only a flow field can be computed, but also an approximation of the volume mesh. ([Iuliano and Quagliarella, 2013, p. 19])

However, no comparison between surrogate models with or without adding mesh coordinates to the sample vectors is presented. The deformation of the mesh is, unlike running the flow solver, computational inexpensive. Therefore there is no direct benefit from computing an approximation of the mesh using the surrogate model. It is unclear, how the surrogate model benefits from the POD modes capturing correlations between the mesh deformation and the flow field. While including them might be likely to result in self consistent flow field and mesh outputs, verification would be required, that the predicted mesh actually matches the real mesh described by the deformation parameters used as input. The main reason given for including the coordinates seem to be worries, that the fact that the data value given in the same row of the state vector is associated with different places in the physical space for different samples, may later on cause issues for the creation of the POD basis. However, as the whole process of model order reduction using POD described in Chapter 3 does not make any assumptions on the provided data structure, this should not be expected to have any negative effect. As long as one does not try to explicitly mimic the physical relation given by the Navier Stokes equations within the surrogate model, there should be no difference whether the snapshot vectors are describing physical properties of points in Cartesian space or mesh points.

2.3 Implementation

The sample data acquisition has been implemented in `acquire_samples.py`. A SU^2 config file and associated mesh is required, specifying the number of design values and parameter ranges. The number of samples to be created and physical properties to be included can be specified. It is also possible to include the mesh coordinates to the snapshot vectors. The LHS implementation from [Lee, 2019] is used to create an LHS distribution matching the dimension and number of sample points provided. It is then projected to the actual design space. The meshes are deformed according to these design parameters and simulations are performed using SU^2. Finally the specified properties are extracted and the snapshot vectors stored in a snapshot matrix.

A set of 400 samples for the NACA 0012 airfoil, provided as test case in [SU2 Test Cases, 2018], has been created, and is used in the present work. The snapshot vectors include four physical properties, namely pressure, density and momentum in x and y direction.

3. Model Order Reduction

One fundamental problem when creating surrogate models for aerodynamic flow problems is the high dimension of the data. While there is a small number of design parameters controlling the object geometry on the input (38 in case of the studied example of the NACA 0012 airfoil), there are ten thousands to millions of data points describing the flow field on the output side (5,233 per physical property for the NACA 0012 test case, yielding in total a 20,932 dimensional output vector). It is therefore desired to reduce the dimension the model is trained on, using called model order reduction techniques. The advantage of using model order reduction over feature selection (e.g. creating a surrogate model that directly predicts lift and drag coefficients) is obtaining a complete flow field as the output of the surrogate model.

One approach commonly used for lossy data compression is POD, also known as Principal Component Analysis (PCA). The basic idea is to find a basis, describing the high dimensional output space, that captures the most information within the coefficients of its first basis vectors. This allows using only these coefficients as the input for model training, while using the basis vectors to restore the full flow field from the model output later on. More formal, the goal is to replace the high dimensional state vector y as defined in Eq. 2.1with the vector $a = (a_1, ..., a_l)$, where $y \approx \sum_j a_j \phi_j$. The set of basis vectors ϕ_j is fix for all sample points, so only $l << n$ coefficients need to be used for the model training.

To allow a significant dimension reduction without introducing large errors, there need to be some simple (e.g. linear or linearly approximable) correlation between the elements of the state vector. This is indeed the case for flow fields: Elements that are associated with mesh points spatial close to each other, are expected to show some correlation, even if describing different properties. At shock fronts these correlations are being broken. Therefore it should be expected, that a surrogate model based on POD modes will provide less accurate results in the region of shocks.

3.1 Mathematical Foundation

In this section a brief introduction to POD and its connection to Singular Value Decomposition (SVD) is given.

Given a set of vectors $Y = [y_1, y_2, ..., y_n] \in \mathbb{R}^{m \times n}$ the goal of POD is finding a basis of l normalized, orthogonal vectors $\phi_1, \phi_2, ..., \phi_l \in \mathbb{R}^m$, that give the best approximation $y_i \approx \tilde{y}_i = \sum_j a_{ij}\phi_j$ in a least squares manner. These basis vectors ϕ_j are also called POD modes. As this will lead to l POD coefficients instead of m data points per sample, and $l << m$, this greatly reduces the problem dimension.

As the ϕ_i are orthogonal, the coefficients are $a_{ij} = \langle y_i, \phi_j \rangle$. If the total squared error should be minimized, we want to find:

$$\min_{\phi_1,\phi_2,...,\phi_l} \sum_{i=1}^{n} \|\tilde{y}_i - y_i\|^2 \tag{3.1}$$

Inserting $\tilde{y}_i$ yields:

$$\min_{\phi_1,\phi_2,...,\phi_l} \sum_{i=1}^{n} \left\| \left(\sum_{j=1}^{l} \langle y_i, \phi_j \rangle \phi_j \right) - y_i \right\|^2 \tag{3.2}$$

Which can be rewritten as follows:

$$\min_{\phi_1,\phi_2,...,\phi_l} \sum_{i=1}^{n} \left\langle \left(\sum_{j=1}^{l} \langle y_i, \phi_j \rangle \phi_j \right) - y_i, \left(\sum_{k=1}^{l} \langle y_i, \phi_k \rangle \phi_k \right) - y_i \right\rangle \tag{3.3}$$

$$\min_{\phi_1,\phi_2,...,\phi_l} \sum_{i=1}^{n} \left(\left(\sum_{j=1}^{l} \sum_{k=1}^{l} \langle y_i, \phi_j \rangle \langle y_i, \phi_k \rangle \langle \phi_j, \phi_k \rangle \right) - \left(\sum_{j=1}^{l} \langle y_i, \phi_j \rangle \langle y_i, \phi_j \rangle \right) - \left(\sum_{k=1}^{l} \langle y_i, \phi_k \rangle \langle y_i, \phi_k \rangle \right) + \langle y_i, y_i \rangle \right) \tag{3.4}$$

As the ϕ_j are orthogonal and normalized, we can use $\langle \phi_j, \phi_k \rangle = \delta_{jk}$ to simplify that expression to:

$$\min_{\phi_1,\phi_2,...,\phi_l} \sum_{i=1}^{n} \left(\|y_i\|^2 - \sum_{j=1}^{l} \langle y_i, \phi_j \rangle^2 \right) \tag{3.5}$$

Since the y_i are fixed, this is equivalent to finding a orthogonal and normalized set of ϕ_j, that maximize:

$$\max_{\phi_1,\phi_2,...,\phi_l} \sum_{i=1}^{n} \sum_{j=1}^{l} \| \langle y_i, \phi_j \rangle \|^2 \tag{3.6}$$

The constrain of orthogonal and normalized ϕ_j can be written as $\delta_{ij} - \langle \phi_i, \phi_j \rangle = 0$, and it can be shown using Lagrangian multipliers and induction over the number l of vectors ϕ_j, that the optimal solution can be obtained using SVD as follows: Given a SVD

$$Y = USV = U \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 & 0 \\ 0 & \sigma_2 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \cdots & 0 & \sigma_d & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 \end{pmatrix} V \tag{3.7}$$

with $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$, $S \in \mathbb{R}^{m \times n}$ and $\sigma_1 \geq \sigma_2 \cdots \geq \sigma_d$, the optimal basis vectors ϕ_j are the first j column vectors of U. A proof for that relation may be found in [Volkwein, 2013, p. 9-11].

Using a larger number l of POD modes will increase the accuracy of the reduced order model, but will on the other hand increase the computational cost for the training and evaluation of the surrogate model. To find a useful trade off, one may inspect the ratio ϵ of the so called energy of the system Y to the energy contained in $\tilde{Y} = [\tilde{y_1}, \tilde{y_2}, ..., \tilde{y_n}]$:

$$\epsilon(l) = \frac{\sum_{i=1}^{l} \sigma_i^2}{\sum_{i=1}^{d} \sigma_i^2} \tag{3.8}$$

Energy does here not refer to any physical energies that might be part of the model data from the fluid dynamics simulations, but is the mathematical energy $trace(YY^\top)$. The closer $\epsilon(l)$ is to 1, the better is the reduced order model. It is however important to note that the absolute value of ϵ does not necessarily provide a direct comparison between POD from different data sources, but it may still be to choose l.

3.2 Implementation

For the purpose of linear algebra computations, the Armadillo C++ library has been used. Armadillo is an open source package, developed to be easy to use by providing high-level syntax, while still being reasonable fast. For more information please refer to [Sanderson and Curtin, 2016].

The implementation of the model order reduction is relatively straightforward and follows [Iuliano and Quagliarella, 2013]. It consists of three steps steps as described below and has been implemented in `model_order_reduction.cpp`:

In the first step, the sample data is read in as matrix, and the mean value of the samples is computed and subtracted from each sample. That mean value needs to be added on reconstruction of flow files from the trained model.

During the second step the actual POD modes are being computed using the singular value decomposition method `svd()` from Armadillo. This yields, as described in Section 3.1, the optimal modes for any given number of l. It is however important to note that the computation of the full singular value decomposition requires storage of a matrix of size $m \times m$, where m is the number of grid points times the number of physical quantities (e.g. pressure, temperature, density) to be included in the surrogate model. As that matrix requires several gigabytes for the NACA 0012 test case, and grows quadratically in size for larger problems, it may be desired to use specialized algebra packages for larger scale problems. One package that could be used as proper replacement is `irlba`, which is described in [Baglama et al., 2019]. The first l POD modes are then stored together with the mean value computed earlier, as they will be needed to reconstruct flow files from the reduced order model.

In the last step, the coefficients of the sample data for the calculated POD modes are computed and stored. Those coefficients are being used together with the input settings for each sample to train the Kriging model as described in Chapter 3. As the POD modes are orthogonal by construction, a coefficient for a mode is simply obtained by projecting the sample data vectors onto the mode vector.

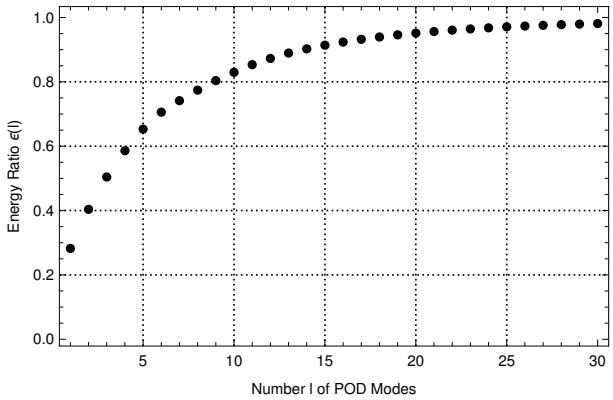


Figure 3.1: Energy Ratio $\epsilon(l)$ of POD Modes

In addition a vector containing the σ_i defined in Eq. 3.7 is stored, to allow a qualified decision on the number of POD modes used for the following steps by inspecting the energy ratio $\epsilon(l)$.

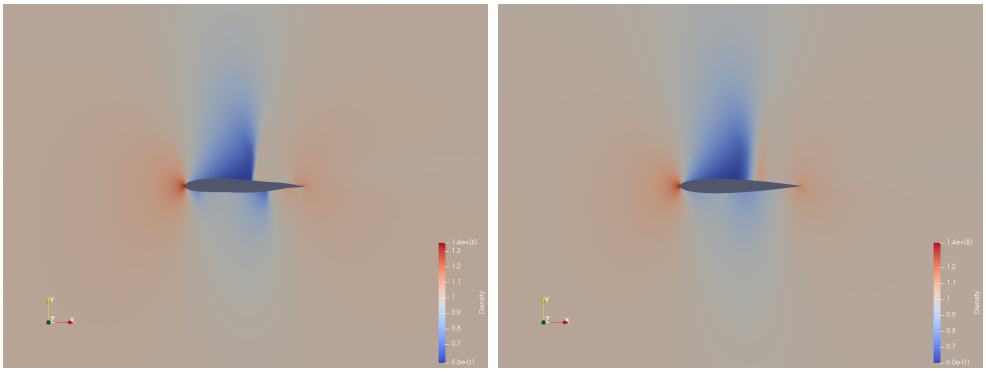
The acquisition of the final training data for the regression method is handled by the separate script `prepare_training_data.py`, which does group the design parameters of the samples with their POD mode coefficients in a convenient way.

3.3 Test Results

The implementation of the model order reduction has been tested on the sample data generated earlier. Figure 3.1 shows the energy ratio as defined in Eq. 3.8. As expected, a relatively small number of POD modes is sufficient to provide a good approximation of the flow field. For the given test case 90% of the energy content is captured using 14 modes, 95% using 20 modes and 97% using 25 modes. Compared to the initial 20,932 dimensional snapshot vector, this means a large reduction in model dimension.

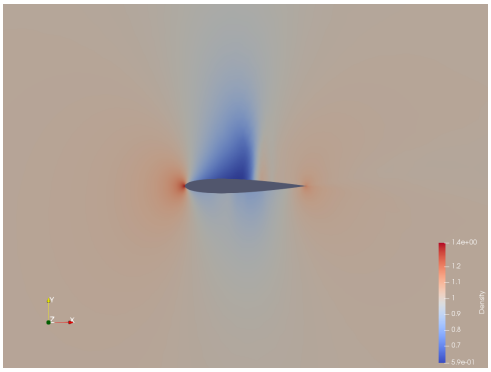
Figure 3.2 and 3.3 show two examples of reconstructions of the density distribution near the airfoil using different numbers of POD modes and how they compare against the flow field directly computed by SU^2. Both samples have been part of the data set used to create the POD basis. The coefficients for the POD modes have been obtained by projecting the snapshot vector onto the basis vectors. These are the coefficients that will serve as the training data for the regression model. As mentioned earlier, the computation of the correct mesh can be done at a low computational cost if needed.

The figures give a good visualization of the strengths and weaknesses of POD for this application: While few modes are sufficient to generate a good reproduction of the overall flow field, difficulties arise in the area of shock fronts. The reason is, that the shock front position varies somewhat continuously for the given test case. As all POD modes are fixed in position, description of features alternating their spatial location, or more precisely their location on the mesh, requires a high number of modes. A high number of modes leads to higher computational costs during the training of the regression model, as discussed in Chapter 4. If very precise description of such features is of great importance for a given application, it may be desired to use specialized algorithms to capture those before doing the model order reduction using POD.

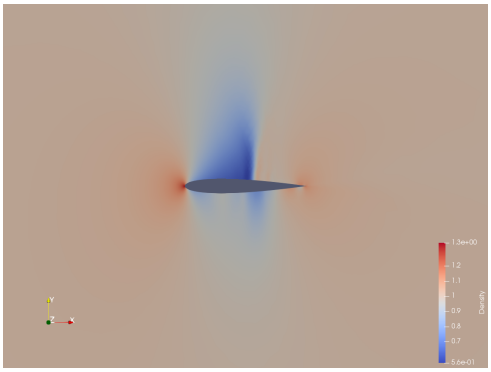


(a) Full Model Solution

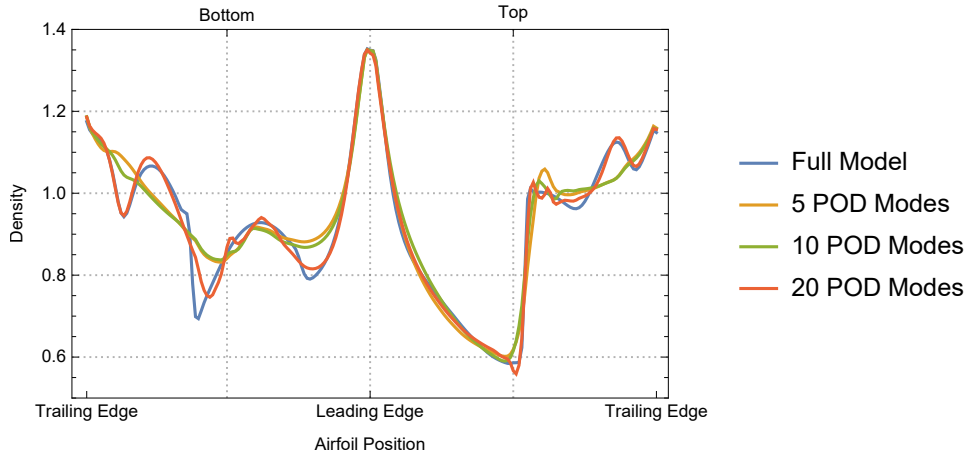
(b) 5 POD Modes



(c) 10 POD Modes

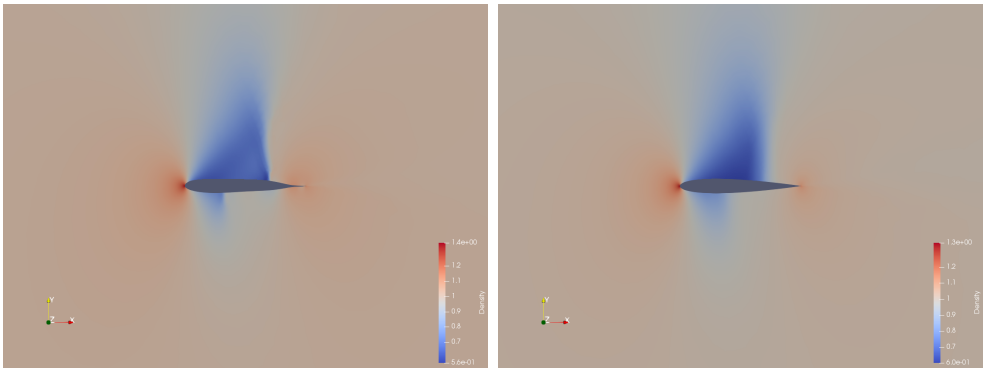


(d) 20 POD Modes



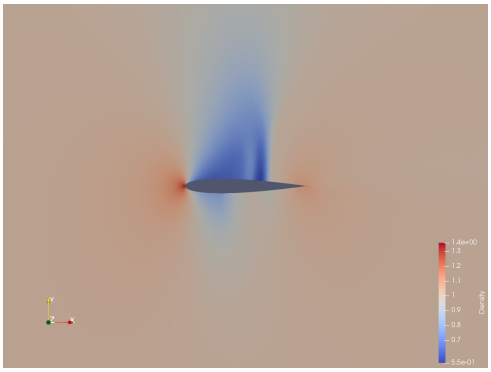
(e) Density distribution along the airfoil surface

Figure 3.2: Reconstructions of the density distribution from different numbers of POD modes. The first sample out of the 400 LHS distributed training data samples is shown.

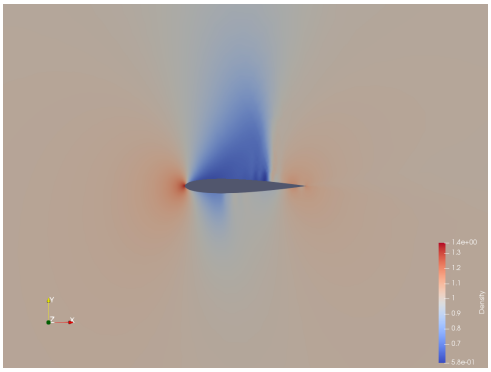


(a) Full Model Solution

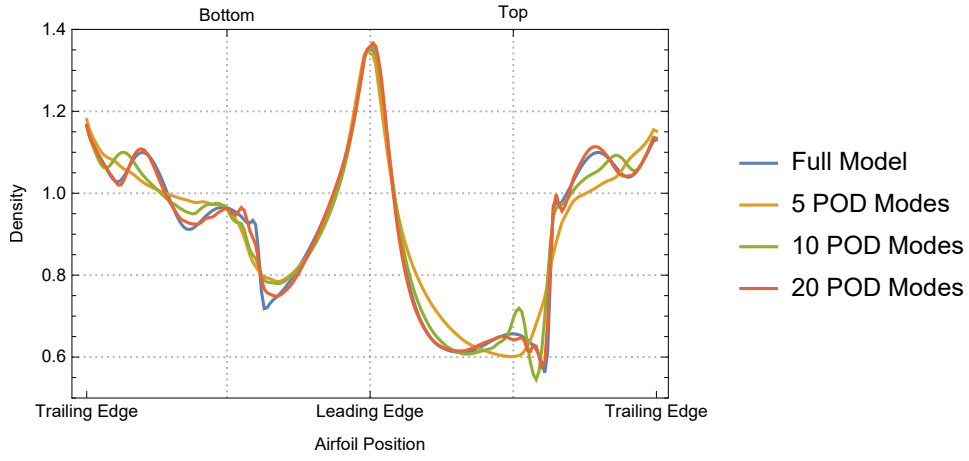
(b) 5 POD Modes



(c) 10 POD Modes



(d) 20 POD Modes



(e) Density distribution along the airfoil surface

Figure 3.3: Reconstructions of the density distribution from different numbers of POD modes. The second sample out of the 400 LHS distributed training data samples is shown.

4. Regression Model

As regression method to predict POD mode coefficients for any given input design parameters, a set of Kriging models is used. Kriging models feature high adaptability to the training data provided, while making only weak assumptions regarding the structure of the input. Kriging models are known to produce good results for complex, multidimensional data input.

Each Kriging model is providing a mapping $\mathbb{R}^k \to \mathbb{R}$. As the output is only scalar valued, training of l different models is required to use all l POD modes computed earlier.

4.1 Kriging Interpolation

This section gives a very short introduction on the basic concepts behind Kriging interpolation. It follows [Forrester et al., 2008, Chapter 2.4], where a more in depth description can be found.

A set of training data containing sample points $x_1, ..., x_n \in \mathbb{R}^k$ and their POD coefficients $c = (c_1, ..., c_n) \in \mathbb{R}^n$ of the mode that Kriging model is to be trained on is given. The basic idea is to treat these coefficients as possible results of some stochastic process and study the correlation $cor[\chi_i, \chi_l]$ between the random vectors $\chi_i = \chi(x_i)$ and $\chi_l = \chi_(x_l)$ defined by a generalized gaussian as follows:

$$cor[\chi_i, \chi_l] = \exp\left(-\sum_{j=1}^{k} \theta_j \|x_{i,j} - x_{l,j}\|^{p_j}\right) \tag{4.1}$$

where $x_{i,j}$ denotes the jth design parameter of the ith training point and $\theta \in \mathbb{R}^k$ and $p \in \mathbb{R}^k$ are the model parameters of the Kriging model. The correlations form a correlation matrix Ψ:

$$\Psi = \begin{pmatrix} cor[\chi_1, \chi_1] & \cdots & cor[\chi_1, \chi_n] \\ \vdots & \ddots & \vdots \\ cor[\chi_n, \chi_1] & \cdots & cor[\chi_n, \chi_n] \end{pmatrix} \tag{4.2}$$

The model parameters θ and p need to be determined in a way, that the likelihood L of obtaining the given POD coefficients c_i from the assumed stochastic process is maximized. This likelihood can be approximated as:

$$L = \exp\left(-\frac{n}{2}\ln(\sigma^2) - \frac{1}{2}\ln(\Psi) + const.\right) \tag{4.3}$$

where $\sigma^2 = \frac{(c-\vec{1}\mu)^\top \Psi^{-1}(c-\vec{1}\mu)}{n}$ and $\mu = \frac{trace(diag(\Psi^{-1}c))}{trace(\Psi^{-1})}$ and the constant does not depend on θ or p and can therefore be ignored for the purpose of maximization. Unfortunately there is no explicit formula or algorithm known to compute optimal values for θ and p, so some global search algorithm on the $2k$ dimensional parameter space needs to be executed. The process involved in finding these optimal or nearly optimal values is also called model training.

To evaluate the Kriging interpolation model at a arbitrary location X, the idea is now to find the most likely result ξ according to the correlations described by the previously obtained values for θ and p. That value ξ is then the predicted POD coefficient for the point X. Using the correlation vector

$$\psi = \begin{pmatrix} cor[\chi_1, \chi(X)] \\ \vdots \\ cor[\chi_n, \chi(X)] \end{pmatrix} \tag{4.4}$$

it is possible to obtain a direct formular for ξ:

$$\xi(X) = \mu + \psi^\top \Psi^{-1}(c - \vec{1}\mu) \tag{4.5}$$

Note that, if the input X is equal to some x_ν that has been part of the training data set, $\psi^\top\Psi^{-1}$ is the νth unit vector, so ξ evaluates as $\xi = \mu + c_\nu - \mu = c_\nu$. This guarantees, that the model will always hit all input points, and therefore is an interpolation of the provided data.

4.2 Kriging Regression

As a pure interpolation as described in Section 4.1 is likely to suffer from overfitting, it is useful to extend the model in a way that a regression instead of a interpolation is computed. Again, a short overview how this can be achieved is given following [Forrester et al., 2008, Chapter 6.1].

To allow derivations from the provided input data, the correlation function needs to be slightly modified. Adding a regression parameter λ to the diagonal elements of the correlation will allow doing so:

$$cor[\chi_i, \chi_l] = \exp\left(-\sum_{j=1}^{k} \theta_j \|x_{i,j} - x_{l,j}\|^{p_j}\right) + \lambda\delta_{il} \tag{4.6}$$

where δ is the Kronecker symbol. This will change the correlation matrix Ψ defined in Eq. 4.2 by adding λ to the diagonal and therefore also change the optimal model parameters θ and p. Using the modified correlation matrix Ψ, the Eq. 4.5 remains valid. However, the

prediction is no longer exactly passing through all sample data, as *lambda* is not added to any entries of the correlation vector ψ as defined in Eq. 4.4 and therefore $\psi^\top \Psi^{-1}$ is no longer a unit vector if $X = x_\nu$. The added regularization parameter allowed us to turn the interpolation into a regression model.

The regularization parameter λ describes the uncertainty of the provided input values. Those might be known in case of physical experiments, but are usually unknown in case of the numerical distortions generated by solving flow problems numerically. It could be added to the set of training variables θ and p, or kept at some fixed value.

As both, Kriging interpolation and Kriging Regression, are using probabilities to compute the predicted outcome, they are able to provide an error estimate as well, based on the probability distribution around the predicted value. However, such an error estimate is not being used for this surrogate model.

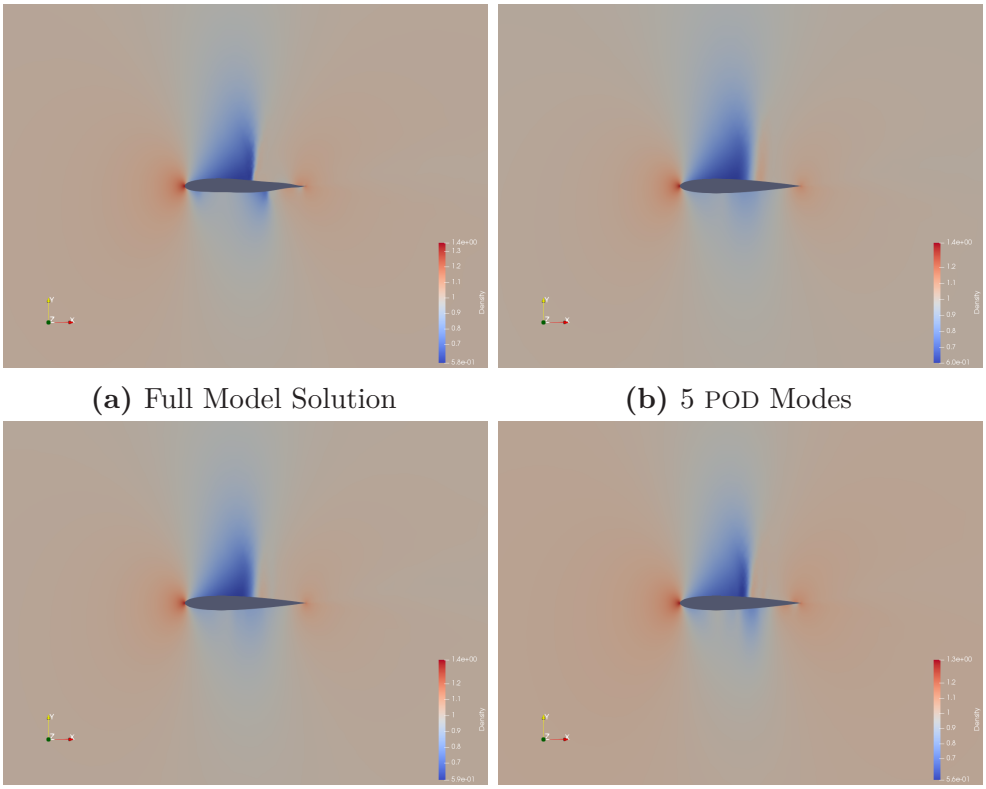
4.3 Model Training

The implementation `model_training.cpp` of the model training is a wrapper of the Kriging model implementation that is part of the Robust Design Optimization (RoDeO) package from [Özkava, 2018]. RoDeO uses a genetic search algorithm to estimate optimal model parameters θ and p. It supports a regression parameter in form of an external argument.

To achieve proper regressions for the 38 dimensional input parameter space of the test case, the genetic algorithm has to run over many iterations. Using 5,000 iterations with the regression parameter set to 2 seem to give good results, but makes this step computational more expensive. Combined with the relatively high number of POD coefficients needed due to the moving shock waves in the flow fields, the training process took about the same time as several thousand flow field computations using SU^2. Thus, creating a surrogate model for the NACA 0012 is unlikely to result in reduced computational costs. However, the computational effort for training one Kriging model does only depend on the number of design parameters and provided samples (where the latter should be chosen according to the number of design parameters) and does not depend on the size of the problem (e.g. the number of mesh points). The creation of a surrogate model is therefore, in comparison to the full model solution, less expensive in cases of smaller design spaces or problems with more costly flow field computations.

4.4 Model Evaluation

The evaluation of the model evaluation is implemented in two different files. The evaluation of the Kriging model and the conversion from the resulting pod coefficients back to a state vector is done in `model_evaluation.cpp`. The evaluation of the Kriging model uses RoDeO ([Özkava, 2018], the preparation necessary to evaluate the model follows the example provided within the source. The computation of the state vector from the POD coefficients is straightforward, as they are just usual coefficients of the representation in a orthonormal basis. In a separate step, the state vector is then converted back into a paraview flow file, which is done in `vector_to_flow_file.py`. If mesh coordinates were not part of the training states, the undeformed mesh is included in that flow file. The

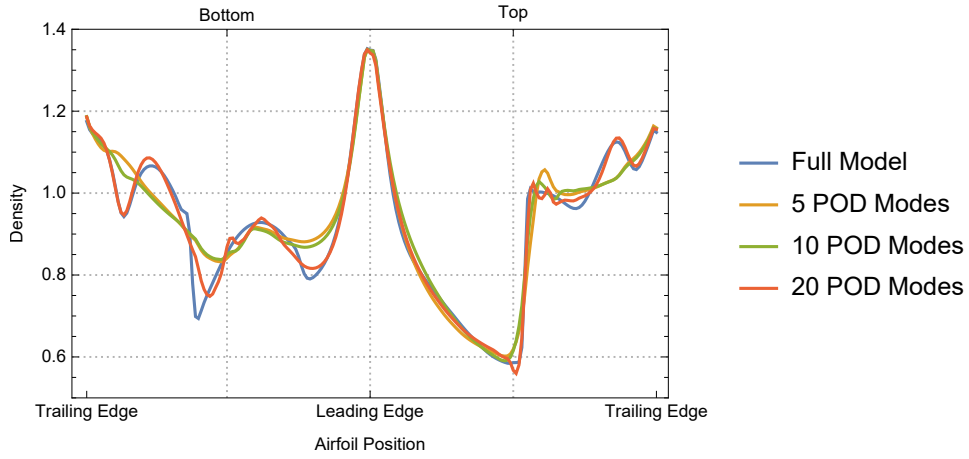


(a) Full Model Solution

(b) 5 POD Modes

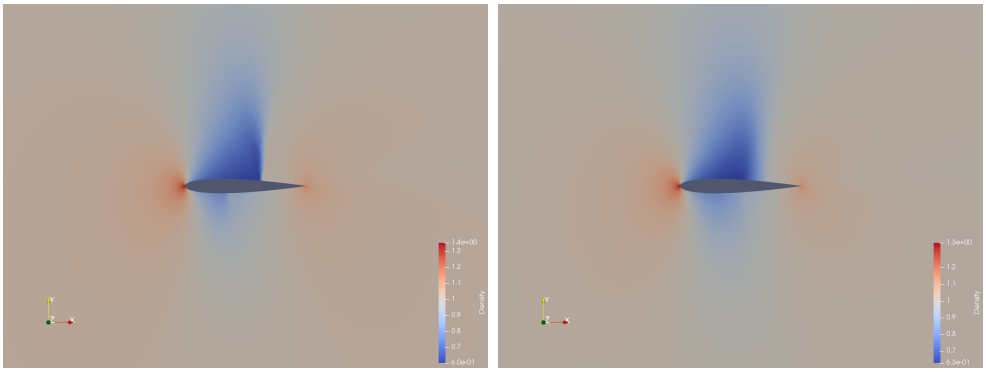
(c) 10 POD Modes

(d) 20 POD Modes



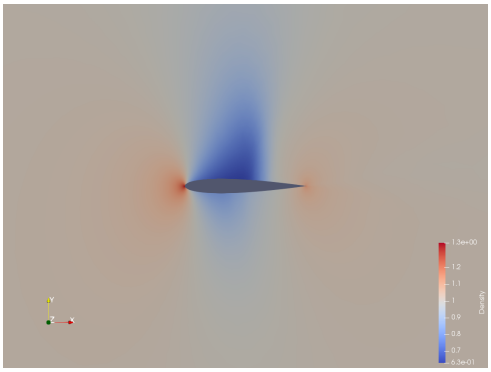
(e) Density distribution along the airfoil surface

Figure 4.1: Density distribution computed by the surrogate model using different numbers of POD modes. The same geometry as in Figure 3.2 has been used. The coefficients for the modes are obtained as output of the Kriging model. They have been part of the training data, therefore a good fit is expected.

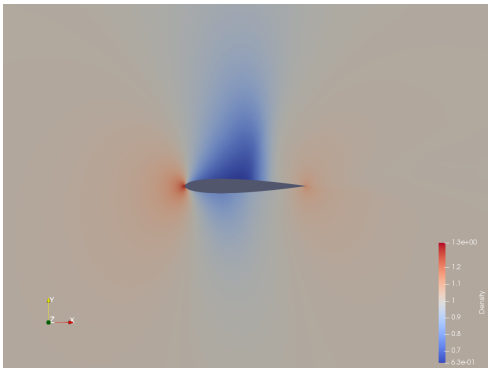


(a) Full Model Solution

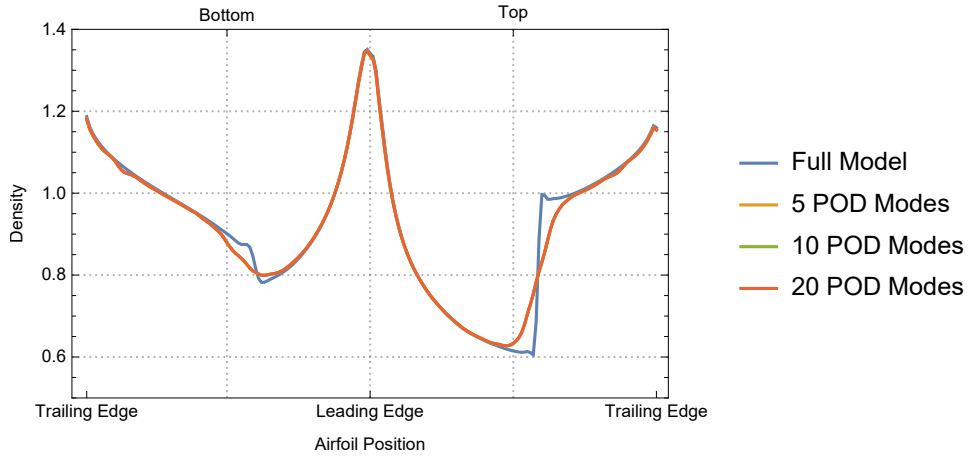
(b) 5 POD Modes



(c) 10 POD Modes



(d) 20 POD Modes



(e) Density distribution along the airfoil surface

Figure 4.2: Density distribution computed by the surrogate model using different numbers of POD modes. The undeformed NACA 0012 airfoil (which was not part of the training data) has been used.

proper deformed mesh may be obtained and included separately by the SU^2 deformation algorithms if needed.

Figure 4.1 shows the surrogate model output using different numbers of POD modes, when provided with design values that have been part of the randomly generated training data set. It is the same sample that is also shown in figure 3.2, but now the POD coefficients predicted by the Kriging model are used rather than the exact values obtained by projecting the snapshot vector onto the modes. The density distribution of the flow field obtained is able describe the shock front. This is what should be expected for inputs near given training data points.

Figure 4.2 shows the output of the surrogate model for the undeformed mesh, the case where all design parameters have been set to zero. Those parameter settings have not been part of the training data, so this is as test case for the quality of the surrogate model on unknown location. It can be seen, that the model is still able to describe the general flow field, but is not giving a precise description of the shock fronts anymore, even if many modes are used. It is possible, that tuning of the regularization parameter can further improve the surrogate model.

5. Conclusion

A surrogate model for aerodynamic shape optimization using POD has successfully been implemented. Using the NACA 0012 airfoil as test case, a set of training data has been created using SU^2. It could be shown, that POD is able to achieve a significant model order reduction. However, the quality of the reduced order model suffers significantly near shock fronts in the flow field. Kriging regression has been used to train a surrogate model, that is able to predict the flow field at arbitrary design sites.

The computational costs for the training process increases with the number of design parameters and POD modes used and is therefore relatively high for the NACA 0012 airfoil test case using Hicks-Henne bump functions. The full CFD solution is quite cheap for the test case on hand, while a large number of POD modes is needed due to the characteristic shock fronts. While the final evaluation of the surrogate is indeed faster than the CFD solution, there is no substantial gain in terms of computational costs when also taking the training process in account - even if a large number of evaluations are performed. However, larger scale problems may have a similar number of design parameters and POD modes, while requiring much more computational resources when solved using CFD. If the number of design parameters decreases, less sample points need to be created, reducing the cost of the sample data acquisition and model training.

The methods discussed can be applied to other problems as well. Parts of the surrogate model may be valuable for the creation of future surrogate models employing other techniques, as the general work flow of data acquisition, model order reduction and model training remains identical.

Bibliography

[Baglama et al., 2019] Baglama, J., Reichel, L., and Lewis, B. W. (2019). Fast truncated singular value decomposition and principal components analysis for large dense and sparse matrices. `https://cran.r-project.org/web/packages/irlba/irlba.pdf`.

[Economon et al., 2015] Economon, T. D., Palacios, F., Copeland, S. R., Lukaczyk, T. W., and Alonso, J. J. (2015). Su2: An open-source suite for multiphysics simulation and design. *AIAA Journal*, 54(3):828–846.

[Forrester et al., 2008] Forrester, A., Sobester, A., and Keane, A. (2008). *Engineering Design Via Surrogate Modelling: A Practical Guide*.

[Iuliano and Quagliarella, 2013] Iuliano, E. and Quagliarella, D. (2013). Proper orthogonal decomposition, surrogate modelling and evolutionary optimization in aerodynamic design. *COMPUTERS & FLUIDS*, 84:327–350.

[Lee, 2019] Lee, A. (2019). pydoe: design of experiments for python. `https://pythonhosted.org/pyDOE/index.html`.

[McKay et al., 1979] McKay, M. D., Beckman, R. J., and Conover, W. J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245.

[Sanderson and Curtin, 2016] Sanderson, C. and Curtin, R. (2016). Armadillo: a template-based C++ library for linear algebra. *The Journal of Open Source Software*, 1:26.

[SU2 Test Cases, 2018] SU2 Test Cases (2018). Unconstrained shape design of a transonic inviscid airfoil at a cte. aoa. `https://su2code.github.io/tutorials/Inviscid_2D_Unconstrained_NACA0012/`.

[Tesfahunegn et al., 2015] Tesfahunegn, Y. A., Kozeil, S., Gramanzini, J.-R., Hosder, S., Han, Z.-H., and Leifsson, L. (2015). Application of direct and surrogate-based optimization to two-dimensional benchmark aerodynamic problems: A comparative study.

[Volkwein, 2013] Volkwein, S. (2013). Proper orthogonal decomposition: Theory and reduced-order modelling. `http://www.math.uni-konstanz.de/numerik/personen/volkwein/teaching/POD-Book.pdf`.

[xkcd, 2018] xkcd (2018). Memorable quotes. `https://xkcd.com/1942/`.

[Özkava, 2018] Özkava, E. (2018). Rodeo: A package for robust design optimization. `https://github.com/eoezkaya/RoDeO`.