Proceedings of the 1st International Symposium on AI and Fluid Mechanics
Paper No S2 P3

# Solving Inverse Fluid Flow Problems with Differentiable Reduced Basis Models

**Marian Staggl[1], Stefan Posch[2], Wolfgang Sanz[1]**

[1] *Institute of Thermal Turbomachinery and Machine Dynamics, Graz University of Technology, Austria*

[2] *Large Engines Competence Center, Graz, Austria*

*Corresponding author: Marian Staggl (marian.staggl@tugraz.at)*

**Keywords:** Reduced Basis Model, Differentiable Simulation, Fluid Flow

**Abstract.** The reduced basis (RB) method is a projection technique assuming that solutions of parameterized partial differential equations (PDEs) can be represented by a small set of coherent flow structures, reducing the solution space to a low-dimensional manifold. The RB methodology enables the creation of compact algebraic systems that are solved significantly faster than traditional computational fluid dynamics (CFD) simulations. Additionally, the compactness of RB models makes them suitable for differentiable computations using a backpropagation method. The present work uses an RB model based on an isentropic version of the Navier-Stokes equations and combines it with a differentiable simulation framework. This approach facilitates highly efficient gradient-based optimization of the investigated system. The method is tested on three fluid flow examples to show its functionality. The first example concerns the viscosity recovery in a sudden expansion channel with known outlet velocity profiles. The second example aims to tune the viscosity so the separation bubble reaches a desired length. The final example expands the same flow setup to a three-dimensional parametric space, including the inlet velocity, the step height, and the viscosity. The goal is to change the parameters so that the velocity at the midsection of the channel approaches a specific target velocity distribution. Overall, the solutions of the solved inverse problems showed accurate results. At the same time, the computational effort was reduced significantly, showing the great potential of combining RB and differentiable simulation for computationally demanding problems. The examples shown in this work are available on GitHub (https://github.com/marianstaggl/ReducedBasis_model.git).

## Acronyms

| | |
|---|---|
| AD | Automatic Differentiation |
| RB | Reduced Basis |
| CFD | Computational Fluid Dynamics |
| PINN | Physics Informed Neural Network |
| PDE | Partial Differential Equation |
| RANS | Reynolds Averaged Navier Stokes |
| POD | Proper Orthogonal Decomposition |

## Variables

| | | |
|---|---|---|
| $\nu$ | $m^2/s$ | kinematic viscosity |
| $H$ | $m$ | step height |
| $u$ | $m/s$ | x-velocity |
| $v$ | $m/s$ | y-velocity |
| $a$ | $m/s$ | local speed of sound |
| $\bar{s}$ | $[m/s, m/s, m/s]$ | full flow field |
| $\mu$ | $-$ | parameter value |

# 1. Introduction

In contrast to classical forward problems, the solution of inverse problems involves the recovery of hidden variables, starting from a set of observations. From a mathematical perspective, inverse problems are often challenging to solve because they are usually ill-posed, yielding no unique solution. One way to solve them is to formulate the underlying problem as an optimization task and solve it with a gradient-based approach. However, this only works if the gradient can be calculated with reasonable effort and sufficient accuracy. A straightforward way to calculate gradients is a finite difference approach, which evaluates the system multiple times, approximating the derivatives along the different parameters in terms of differences. A major drawback of this approach is the effort to evaluate the gradient that scales with the dimensionality of the solution space. Thus, the finite difference approach can be prohibitive if the evaluation of the physical system involves the solution of expensive CFD calculations.

A more convenient way to calculate the gradient was introduced by the automatic differentiation (AD) method [1], which nowadays is an important building block for the training procedure of neural networks. The AD method allows the calculation of the gradient up to machine precision, and more importantly, its computational cost does not scale with the number of input parameters. The use of AD is not limited to classical neural networks, and in a very interesting work, Raissi et al. [2] proposed to use it in order to incorporate PDE systems into the loss function. The authors used this physics-informed neural network (PINN) to solve a fluid flow problem. Furthermore, they successfully applied the approach to an inverse problem. Since then, the field of PINNs has been a rapidly growing research topic with applications to various PDE systems. Still, the training process of PINNs can be quite expensive in terms of computational effort. To reduce the computational costs, it is possible to replace the PDE model with a surrogate model, which can be either data-driven or rely on a reduced basis (RB) methodology. RB models allow a speed-up of multiple orders of magnitude while still bound to governing equations. The methodology has been applied to multiple problems, including fluid flow examples for incompressible flow [3, 4, 5, 6, 7, 8, 9] and also compressible flow examples [10, 11]. Concerning the simulation of turbulent flows, an incompressible RB model to solve the Reynolds Averaged Navier Stokes equations (RANS) was presented by Hijazi et al. [8] who uses a finite volume (FV) based approach to assemble the RB model and a data-driven method to derive the viscosity fields. A possible drawback of RB models are instabilities arising from various sources which can result in spurious oscillations [6], convergence to wrong solutions [4] or unstable longtime behavior [12] if no further stabilization is applied. However, multiple methods have been proposed to stabilize them, ranging from enriching the solution space [4, 6, 8, 7] to Petrov-Galerkin methods [13, 12]. As stated above, the speed-up of RB models can be significant, but a database of converged solutions is necessary to set up the model. The creation of this database and the assembly of the RB equation systems is still a time-consuming step. Yet, if the model can be reused for different tasks, this effort marginalizes over time. This work aims to combine a fast and accurate RB model with optimization methods stemming from the field of machine learning (ML), including the AD method and a quasi-newton gradient-based L-BFGS optimizer. As pointed out by [1], the evaluation time of a backward pass of an AD is of the same order of magnitude as the forward pass. If the forward pass takes only fractions of a second, this allows for fast optimizations once the needed RB model is set up. One big advantage compared to classical surrogate models is that the outlined method is agnostic with respect to the goal of the optimization. If the objective changes, the model can be reused, and no further time-consuming calculation steps are necessary.

# 2. Methodology

## *2.1. Reduced basis model*

In order to reduce the computational effort, RB methods try to reduce the solution space of conventional CFD simulations. The main idea is that for a given parametrized flow problem, the resulting flow fields

are somewhat similar and that there exists a small set of coherent flow structures that, if recombined (see Eq. 1), are sufficient to approximate solutions of the parametrized problem with high accuracy.

$$s(x,\bar{\mu}) = \sum_{i=1}^{\infty} \varphi_i(x)\, a_i(\bar{\mu}) \tag{1}$$

In this case, these coherent flow structures $\varphi(x)$ (they will be referred to as modes for the further course of this work) are independent of the parameter values but are only functions over the domain. In order to approximate a new solution, they are superimposed while only the weighting factors $a_i$ need to be found. In the case of a numerical calculation, the solution $s(x, \bar{\mu})$ can also be expressed in a vector where each vector entry corresponds to one node of the computational mesh. In its discrete form, Eq. 1 can also be expressed by $\bar{s} = \bar{\varphi}_i\, \bar{a}_i$. If this relation is inserted into an algebraic system that arises from the discretization of a PDE system, the number of DoFs can be reduced to the number of modes (see Eq. 2).

$$\bar{N}\,\bar{s} = \bar{f} \quad \rightarrow \quad \bar{N}\,\bar{\Phi}\,\bar{a} = \bar{f} \tag{2}$$

This results in an overdetermined system of equations of the size $h \times n$ where $h$ is the number of nodes times the number of variables while $n$ is the number of modes. Most RB models rely on a Galerkin projection to reduce the number of equations. The system is projected onto the modes (see Eq. 3), and as the modes are independent of the parameters, the projection of the large matrix $\bar{N}$ can be done in advance, leading to a small $n \times n$ system that needs to be solved.

$$\bar{\Phi}^T \bar{N}\,\bar{\Phi}\,\bar{a} = \bar{\Phi}^T \bar{f} \tag{3}$$

In the case of non-linear PDEs, such as the Navier-Stokes equations, further layers of reduction are required following the basic principle outlined above. A description of the implemented RB model can be found in (ArXive Paper).

### *2.2. Proper Orthogonal Decomposition*

The derivation of the modes $\bar{\Phi}$ is an important ingredient for RB models, where the proper orthogonal decomposition (POD) is the most popular method [8, 6, 5, 4, 14, 3, 10, 12]. In order to derive the modes, a collection of results is necessary, where the vector representation of each CFD result is given by Eq. 4. The $x$-velocities $u$ of every mesh node are stored in the first part of the vector, the $y$-velocities $v$ in the second part, and the local speed of sound $a$, which serves as a thermodynamic variable, is stored in the third part of the vector.

$$\bar{s} = [\bar{u};\ \bar{v};\ \bar{a}] \tag{4}$$

If multiple solutions are already available, they can be stored in a so-called snapshot matrix. This snapshot matrix is defined as shown in Eq. 5 and is of dimension $h \times m$, where $h$ is given by three times the number of mesh nodes and $m$ is defined by the number of available CFD results.

$$\bar{S}_{h\times m} = [\bar{s}_1, \bar{s}_2, \bar{s}_3, \ldots, \bar{s}_m] \tag{5}$$

In order to calculate the individual modes $\bar{\varphi}$, a covariance matrix $C$ is calculated as shown in Eq. 6. As a next step, this covariance matrix is subject to an eigenvalue and eigenvector analysis where the identified eigenvectors $\bar{\psi}$ are used to calculate the modes $\bar{\varphi}_i$. The decay of the eigenvalues $\lambda$ shows how important each mode is.

$$\bar{C} = \bar{S} \cdot \bar{S}^{\intercal} \tag{6}$$

$$\bar{C} \cdot \bar{\psi}_1 = \lambda \cdot \bar{\psi} \tag{7}$$

$$\bar{\varphi}_i = \frac{\sqrt{}}{\bar{\lambda}_i} S \cdot \psi_i \tag{8}$$

The outlined procedure results in $m$ modes, which are usually sorted according to their eigenvalue, from the largest to the lowest. These eigenvalues are a measure for how much of the variance in the dataset can be expressed by each of the individual modes. Low-number modes usually capture most of the variance, and thus, usually, only a small subset of the first $m$ modes needs to be considered for an accurate-approximation capability. As for now, the variance is the objective of this decomposition. Each entry of the covariance matrix $C$ is defined by a scalar product as shown in Eq. 9:

$$C_{(1,2)} = \bar{s}_1 \cdot \bar{s}_2 = \bar{u}_1 \cdot \bar{u}_2 + \bar{v}_1 \cdot \bar{v}_2 + \bar{a}_1 \cdot \bar{a}_2 \tag{9}$$

By a proper weighting of the different entries of this vector, the entries of the covariance matrix can be transformed according to Eq. 10, which has the physical meaning of a stagnation enthalpy. This means that the eigenvalues are now a measure of the resolved energy of each mode.

$$C_{(1,2)} = \int (\bar{u}_1 \cdot \bar{u}_2 + \bar{v}_1 \cdot \bar{v}_2 + \frac{1}{\kappa - 1} \bar{a}_1 \cdot \bar{a}_2) \cdot d\Omega \tag{10}$$

This is also the reason why the local speed of sound is chosen as a thermodynamic variable, as it allows this kind of procedure. This energy-based inner product was proposed by Rowley et al. [10], who used it for an RB model to predict cavity oscillations.

### *2.3. Governing Equations*

The RB model uses a simplified version of the compressible Navier Stokes equations, again initially proposed by Rowley et al. [15]. The equations can be derived by closing the system of momentum and continuity equations with isentropic relations for the thermodynamic variables without solving the energy equation. The equations are valid for moderate Mach numbers and small temperature gradients. The momentum equation is given in Eq. 11 while the continuity equation is given in Eq. 12.

$$u_j \frac{\partial u_i}{\partial x_j} + \frac{2}{\kappa - 1} a \frac{\partial a}{\partial x_i} - \nu \frac{\partial^2 u_i}{\partial x_j^2} = 0 \tag{11}$$

$$u_j \frac{\partial a}{\partial x_j} + \frac{\kappa - 1}{2} a \frac{\partial u_j}{\partial x_j} = 0 \tag{12}$$

In both equations, the local speed of sound $a$ is used as a thermodynamic variable. In this way, all flow variables have the same physical dimension ($m/s$), which allows the definition of physically meaningful POD modes that contain all flow variables (see Eq. 10). Furthermore, in contrast to the fully compressible equations, the simplified version only has non-linearities of the second degree, which facilitates its implementation as an efficient and computationally cheap RB model.

### *2.4. Optimization procedure*

Figure 1 outlines the implemented optimization procedure. The goal is to minimize a loss value by changing the model parameters $\bar{\mu}$. The method uses the MATLAB deep learning toolbox, which provides an automatic differentiation methodology and multiple different gradient-based optimizers (see the orange box in Figure 1). At this point, it is assumed that the RB model is already set up, and to evaluate new parameter combinations, its boundary conditions and computational mesh need to be changed accordingly. The mesh deformation and the imposition of the boundary conditions are carried out within a function that wraps the RB model. The function has fixed interfaces, takes a parameter vector $\mu$ as input,

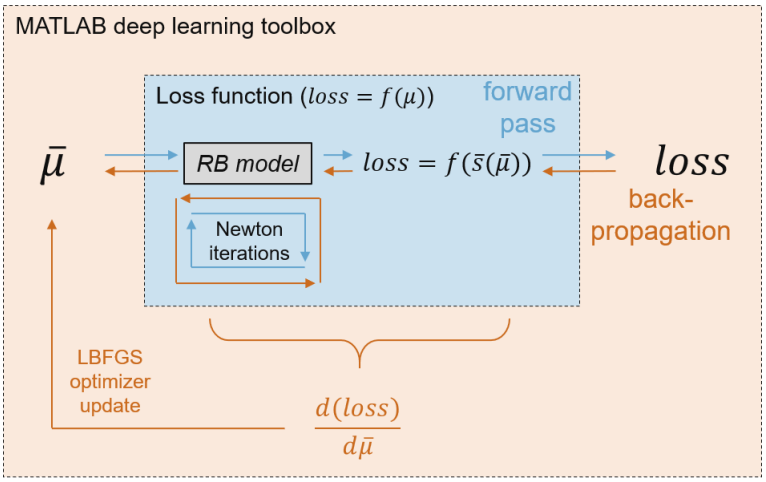


Figure 1: Optimization procedure to solve inverse flow problems by leveraging the MATLAB auto-differentiation and optimization methods.

and returns a flow solution vector $s(\bar{\mu})$ as output (see the blue box in Figure 1). Finally, an objective function $f$ is applied to the solution vector and returns a scalar loss value as subject to the optimizer.
To summarize it, a forward pass (see blue arrows in Figure 1) consists of choosing a parameter set, modifying mesh and boundary conditions, running the numerical simulation, and evaluating the result in terms of an objective function. All basic mathematical operations carried out during this forward pass are tracked and stored alongside their adjoint operations.
During the backward pass, the tracked adjoint operations are carried out, starting from the loss value, which gives the loss gradient with respect to the input parameters. Due to the necessity of this backward pass, it is crucial that every calculation step is derivable and carried out within the MATLAB framework. Until now, the MATLAB backslash operator doesn't support the tracking procedure, which is why a custom QR solver was used.
Once calculated, the loss gradient is handed over to an already implemented quasi-newton L-BFGS optimizer, and the input parameter values are updated accordingly.
Due to the compactness of the RB model, the computational cost for this optimization loop is relatively small. Usually, the time necessary to evaluate the backward pass is similar to the forward pass. It is important to note here that the same RB model can be combined with different objective functions, making the procedure very flexible if the objective of the optimization changes.

### *2.5. RB model parameters*

The model used to solve the outlined optimization problems uses a base of ten POD modes to approximate the solutions within the parameter space. In order to stabilize the RB model, an overdetermined formulation is used, where the dimensionality of the test function space is higher than the number of trial functions (Archive Paper). The work uses two models with a degree of overdetermination of two and three, respectively. The evaluation time of a forward pass is roughly 0.005 seconds on a standard PC for the chosen model, where a higher degree of over-determination leads to slightly higher evaluation times.

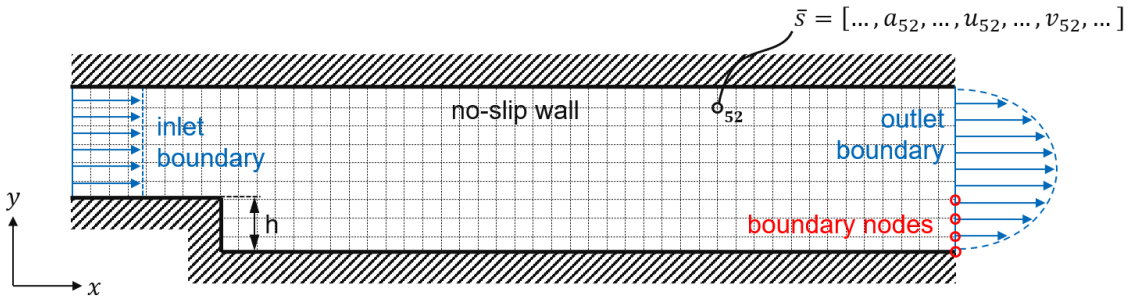


Figure 2: General setup for flow through a sudden expansion channel with velocity inlet at the left side and pressure outlet to the right.

## 3. Creation of the database

This work shows the implemented methods using the example of a sudden expansion channel as shown in Figure 2. The flow enters the left side, where the boundary conditions are specified by setting x- and y-velocities. The flow leaves the domain through a pressure outlet condition where, due to the special formulation of the governing equations, the back pressure is first converted to the local speed of sound; a no-slip condition defines the wall boundaries. The flow separates at the upper corner of the step. It re-attaches further downstream, where the length of the recirculation zone is dependent on three parameters, which are the inlet velocity ($u$), the viscosity ($\nu$), and the height of the step ($H$). All parameters can be varied independently, and the problem is evaluated 400 times to create the database. The sampling is carried out with a random approach by sampling from a uniform distribution for each parameter within a given range. Additionally, to this train data set, which is used to span the solution space of the RB model, a further 200 calculations are carried out to generate a test data set where the methods are tested. The high-fidelity simulations are carried out with a streamwise-upwind-petrov-galerkin stabilized finite element solver, using constant outlet pressure and linear shape functions. The converged solutions are stored and eventually decomposed through an energy-based POD analysis, as outlined in the previous section. The resulting base is truncated, using only the ten most energetic modes. These time-consuming steps are carried out once during the so-called ”offline” phase, while afterwards, the model is reused for the different optimization tasks. An objective function must be defined once the RB model is set up. This objective function calculates a scalar value based on a simulation result and is used as a loss value for the gradient-based optimization. The procedure is tested on three optimization tasks with different objective functions.

## 4. Flow Cases

### *4.1. Outlet Velocity Matching (1st example)*

The objective function of the first investigated inverse problem concerns the recovery of the viscosity $\nu$, given the velocity information at the outlet of the channel (see Figure 3). The objective function is defined as shown in Eq. 13 where $P_{out}$ is constructed by selecting individual rows of the unity matrix. If carried out, the calculations shown in Eq. 13 take $x$- and $y$-velocity at the outlet nodes and compare them to the target outlet profile by calculating the mean squared error; $n$ denotes the number of outlet nodes, multiplied by two for to account for $x$- and $y$-velocity. For this reason, differences in the x-velocity are weighted equally to differences in the y-velocities.

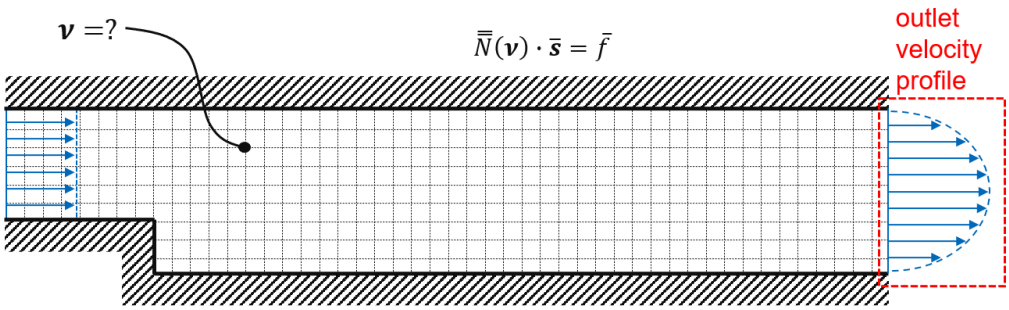


Figure 3: Viscosity recovery by matching outlet velocity profiles with a target distribution (1st example).

$$obj = f(\bar{s}(\mu)) = \frac{1}{n} \sum \left( P_{out}^T \bar{s}(\mu) - P_{out}^T \bar{s}_{target} \right)^2 \quad (13)$$

The example was formulated in terms of an optimization task as outlined in Figure 1 and solved with a quasi-newton L-BFGS solver. On average, the solver converges after 10 steps, and it takes about 45s on a PC to solve the problem for one specific example. The method is tested on the 200 available CFD solutions, and the results are given in Figure 4. In the left diagram, the identified kinematic viscosity (blue and orange) is compared to the known ground truth in yellow for all available test cases. The task was solved with a double and triple overdetermined RB model, where both versions yield similar results. For the double-overdetermined RB model, the largest deviation was 0.72% of the true viscosity value, while the average deviation was 0.18%. In the right diagram of Figure 4, it can be seen that the errors are not centered around zero but are slightly shifted to negative values, meaning that the optimization method, predicts a slightly higher viscosity. The triple overdetermined model shows an average error of 0.13% with a maximum value of 1.08%. For this case, the trend of slightly higher viscosities does not occur.

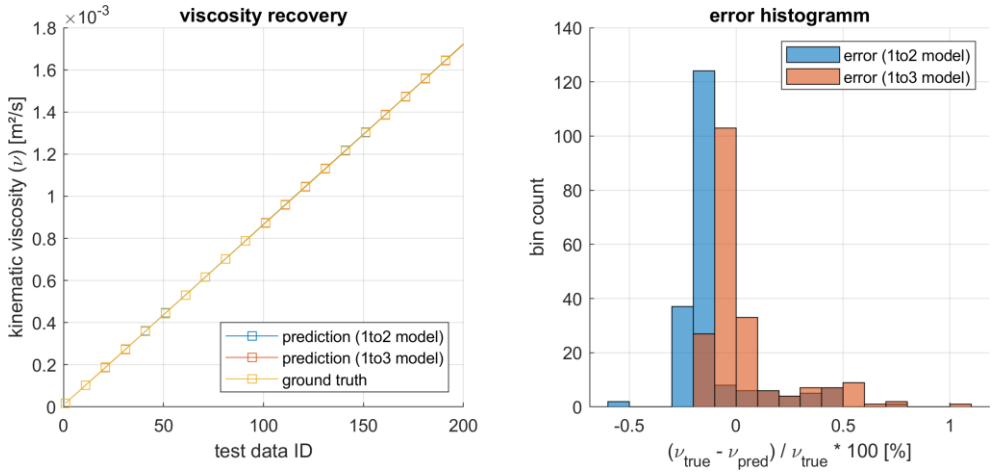


Figure 4: Comparison between predicted and true viscosity values (left) and distribution of the normalized errors for both RB models (right).

### *4.2. Separation Length Matching (2nd example)*

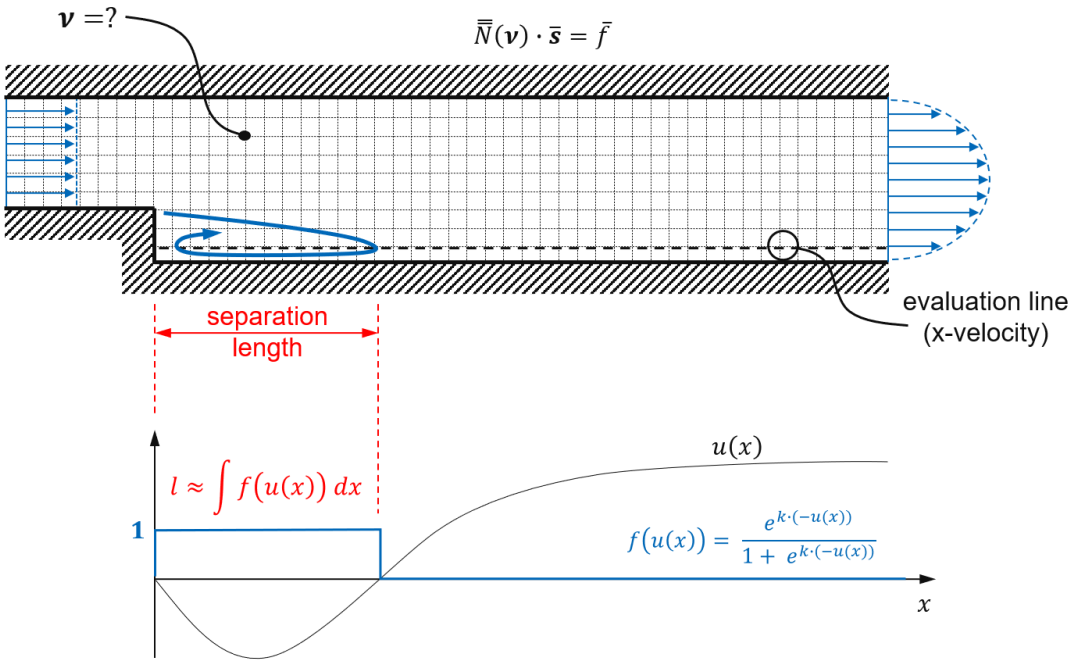


Figure 5: Viscosity adaption to steer the length of a separation bubble (2nd example). The separation length is calculated by evaluating the *x*-velocity along the evaluation line.

The second example uses the same RB model but a different objective function. In this case, the goal is to specify the viscosity $\nu$ such that the length of the separation bubble reaches a prescribed value (see Figure 5). To calculate the length of the separation bubble, the x-velocity of the flow is evaluated along a line near the lower wall (see dotted black line in Figure 5). Within the separation zone, the *x*-velocity along this line is negative and positive elsewhere. A standard approach to determine the length of the separation bubble would be to calculate the position of the zero crossing. Yet, in the MATLAB environment, this includes non-derivable operations. Thus, the x-velocity along the evaluation line (see $u(x)$ in Figure 5) is scaled with a large negative value and plugged into a sigmoid function. The resulting function is approx. one if the fluid is flowing backwards and close to zero elsewhere, and thus, the integral of this function gives the length of the separation bubble. In order to avoid loss gradients approaching zero, this backflow function is slightly extended by adding the x-velocity, scaled with a small negative value (see Eq. 14).

$$obj_2 = f(\bar{s}(\mu)) = \int (f(u(x))\,dx - l_{target}) \tag{14}$$

The goal of the second test case was to set the viscosity value such that the length of the separation bubble reaches the desired length. As shown in Figure 5, the length of the separation zone is calculated by investigating the x-velocity of the fluid along an evaluation line parallel to the lower wall. The results of this example are given in Figure 6. The optimization was carried out for different target lengths specified as a percentage of the step height $H$ ranging from 100% to 700%. To validate the accuracy of the optimization, the resulting parameter values were used to repeat the calculation with the high-fidelity solver. The solution times for this example were slightly higher as derivatives within the objective function became increasingly small, leading to a poorly conditioned problem. This effect becomes more

substantial for large target values. In Figure 6 left, the x-velocities along the evaluation line are shown for different cases. It can be observed how the zero crossing of the line is shifted to higher values as the target length grows. On the right side of the figure, the actual position of the zero crossing is compared to the target values, where the red line marks a perfect match. The results of the RB model optimization are slightly higher, with an average error of 2.84 % and a maximum error of 2.98 %.

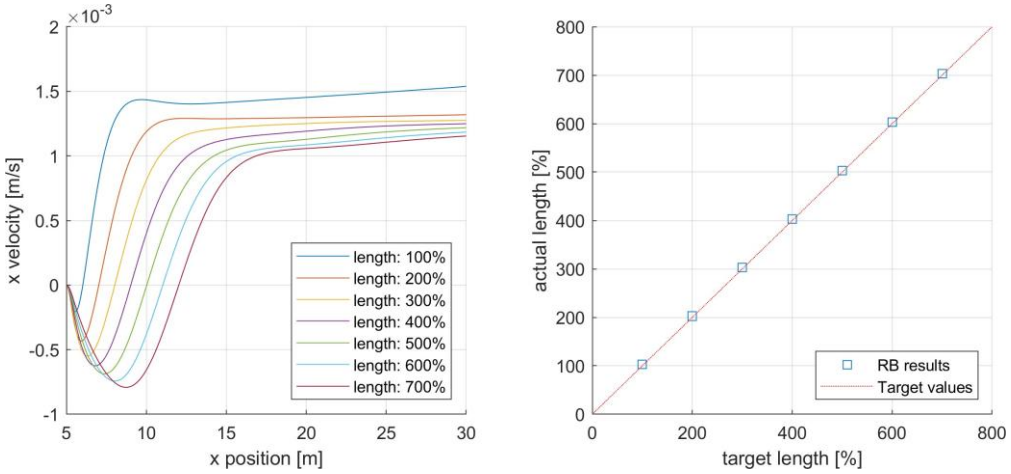


Figure 6: $x$-velocities along evaluation line for different cases (left) and comparison between target length and true value (right).

### 4.3. *Velocity profile matching (3rd example)*

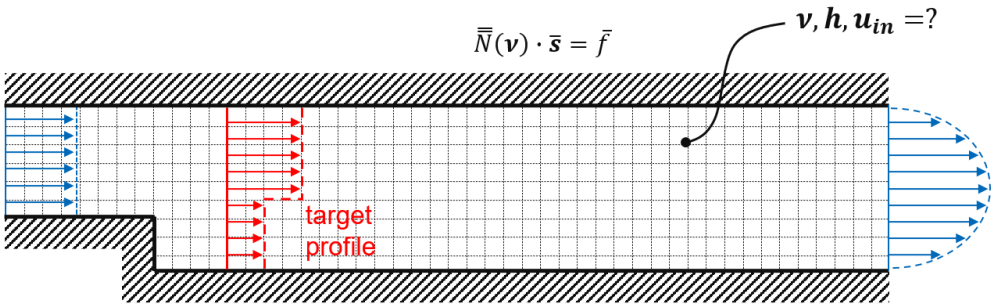


Figure 7: Optimization in three-dimensional parameter space (3rd Example) to reach the target velocity distribution in the channel midsection (marked in red).

The third example uses a larger parametric space, where not only the viscosity $\upsilon$, but also the step height $H$ and the inlet velocity $u$ are subject to the optimization. In this example, the optimization goal is to reach a specific x-velocity profile in the midsection of the channel (see Figure 7). Contrary to the first two examples, the target profiles are defined such that they cannot be reached without violating constraints imposed by the underlying physical system, and it is, at best, possible to approximate them by changing the three available parameters. The objective function for this case is defined by the mean squared error of the actual velocity profile at the position $x = 10$ (see red line in Figure 7), minus the

target velocity profile (see Eq. 15); $n$ denotes the length of vectors $\bar{u}$ .

$$obj_3 = f(\bar{s}(\mu)) = \frac{1}{n} \sum (\bar{u}(x, \mu) - \bar{u}_{target})^2 \tag{15}$$

In the third example, the method was used to approximate velocity profiles that cannot be reached due to the physical restrictions of the system. In this case, the parameter space was three-dimensional, including the step height, the inlet velocity, and the viscosity of the fluid ( $\bar{\mu} = [\nu, H, u_{in}]$). Figure 8 shows the results of this final optimization task. The optimization was carried out for three different target velocity profiles shown by the blue curves, which are impossible to reach with the given degrees of freedom. Nonetheless, they can be approximated in a least-squares sense, and the orange line gives the result of this optimization. In the first case, the target profile is a flat velocity profile, while in cases two and three, the target profile is zero next to the lower wall and jumps to a constant value at a normalized channel height of 0.25 and 0.15, respectively.

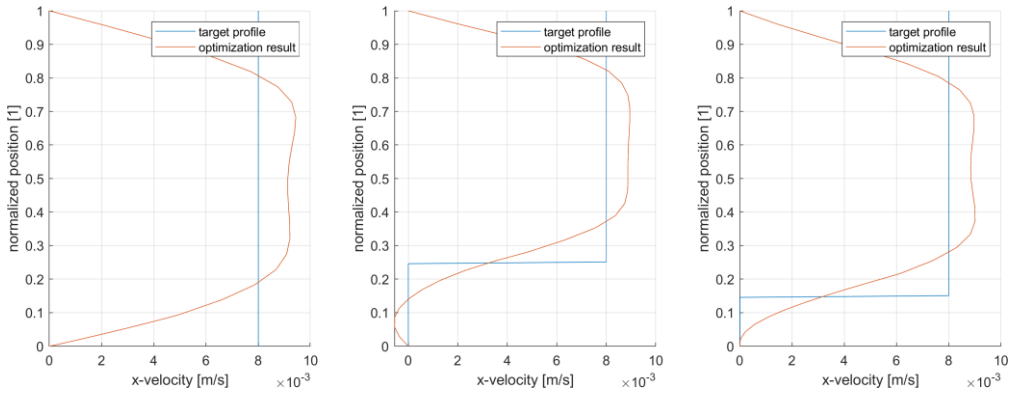


Figure 8: Velocity profile matching results with target distributions (blue lines) and distributions of the optimized physical system (orange).

## 5. Discussion and outlook

Overall, the results of the optimization tasks show that the AD method can be combined with RB models to solve inverse flow problems efficiently. The L-BFGS solver performed well on all tasks using standard settings, suggesting that the method is fairly robust and can be applied to various problems without substantial changes. However, as one forward pass is fairly fast, solution times of≈40 seconds are still relatively high. One reason is surely the usage of a custom QR solver, which performs poorer than the standard backslash operator provided by MATLAB. Furthermore, the solver parameter can be adapted to speed up the optimization process.

The foundation for the shown examples is an accurate and fast RB model, and one might argue that applying the simplified flow equations is restrictive. However, the range of applications for this system is still very broad, and it can provide good approximations of the fully compressible Navier-Stokes equations over a wide Mach number range (ArXive Paper). For future applications, the goal is to extend the model to turbulent flows, following a RANS approach. This will allow fast and fairly accurate optimizations during an early layout stage or a design iteration process. The examples shown in this work are available on GitHub (https://github.com/marianstaggl/ReducedBasis_model.git).

## Acknowledgments

Marian Staggl and Wolfgang Sanz would like to acknowledge the financial support of the ”ARIADNE” project by the ”Take Off” Program, a Research, Technology, and Innovation Funding Program of the Republic of Austria’s Ministry of Climate Action. The ”ARIADNE” Programme is managed by the Austrian Research Promotion Agency (FFG). Stefan Posch would like to acknowledge the financial support of the “COMET Module LEC FFF” within the “COMET - Competence Centers for Excellent Technologies” Programme of the Austrian Federal Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK), the Austrian Federal Ministry of Labour and Economy (BMAW) and the Provinces of Styria and Tyrol. The COMET Programme is managed by the Austrian Research Promotion Agency (FFG).

## References

[1] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, “Automatic differentiation in machine learning: a survey,” *Journal of Machine Learning Research*, vol. 18, no. 153, pp. 1–43, 2018.

[2] M. Raissi, P. Perdikaris, and G. Karniadakis, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,” *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[3] M. Bergmann, C.-H. Bruneau, and A. Iollo, “Enablers for robust pod models,” *Journal of Computational Physics*, vol. 228, no. 2, pp. 516–538, 2009.

[4] F. Ballarin, A. Manzoni, A. Quarteroni, and G. Rozza, “Supremizer stabilization of pod–galerkin approximation of parametrized steady incompressible navier–stokes equations,” *International Journal for Numerical Methods in Engineering*, vol. 102, no. 5, pp. 1136–1161, 2015.

[5] A. Quarteroni, *Reduced Basis Methods for Partial Differential Equations: An Introduction*, vol. 92 of *SpringerLink Bücher*. Cham: Springer, 2016.

[6] S. Ali, *Stabilized reduced basis methods for the approximation of parametrized viscous flows*. Phd, SISSA, International School for Advance Studies, Trieste, Italy, 2018.

[7] N. Akkari, F. Casenave, and V. Moureau, “Time stable reduced order modeling by an enhanced reduced order basis of the turbulent and incompressible 3d navier–stokes equations,” *Mathematical and Computational Applications*, vol. 24, no. 2, p. 45, 2019.

[8] S. Hijazi, G. Stabile, A. Mola, and G. Rozza, “Data-driven pod-galerkin reduced order model for turbulent flows,” *Journal of Computational Physics*, vol. 416, p. 109513, 2020.

[9] Y. Nakamura, S. Sato, and N. Ohnishi, “Application of proper orthogonal decomposition to flow fields around various geometries and reduced-order modeling,” *Computer Methods in Applied Mechanics and Engineering*, vol. 432, p. 117340, 2024.

[10] C. W. Rowley, T. Colonius, and R. M. Murray, “Model reduction for compressible flows using pod and galerkin projection,” *Physica D: Nonlinear Phenomena*, vol. 189, no. 1-2, pp. 115–129, 2004.

[11] I. K. Tezaur, M. Balajewicz, M. F. Barone, K. T. Carlberg, J. A. Fike, and E. E. Mussoni, “Model reduction for compressible cavity simulations towards uncertainty quantification of structural loading,” tech. rep., Sandia National Lab. (SNL-CA), Livermore, CA (United States); Sandia National Lab. (SNL-NM), Albuquerque, NM (United States), 09 2016.

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

[13] K. Carlberg, C. Bou-Mosleh, and C. Farhat, "Efficient non-linear model reduction via a least-squares petrov-galerkin projection and compressive tensor approximations," *International Journal for Numerical Methods in Engineering*, vol. 86, no. 2, pp. 155–181, 2011.

[14] Z. Wang, I. Akhtar, J. Borggaard, and T. Iliescu, "Proper orthogonal decomposition closure models for turbulent flows: A numerical comparison," *Computer Methods in Applied Mechanics and Engineering*, vol. 237-240, pp. 10–26, 2012.

[15] C. Rowley, T. Colonius, and R. Murray, "Dynamical models for control of cavity oscillations," in *7th AIAA/CEAS Aeroacoustics Conference and Exhibit*, (Reston, Virigina), American Institute of Aeronautics and Astronautics, 2001.