Investigating inconsistent uncertainty quantification encountered in neural network modelling of nonlinear response of a laminar flame

Marcin Rywik[1], David Sören da Cruz [2] and Wolfgang Polifke [3]
Department of Engineering Physics and Computation, School of Engineering & Design, Technical University Munich
Boltzmannstraße 15, 85748 Garching, Germany

ABSTRACT

Some investigations on neural network flame modelling report contradictory results. Conclusions on the performance and uncertainty are disparate, despite using an identical dataset for model derivation. Contrasting approaches to data preprocessing are determined as one of the major differences between the studies. Shuffling of the time series data is identified as a probable reason for sub-optimal results of some previously designed networks. Incorporating shuffling is shown to produce training, validation and test sets of extremely high similarity. It is illustrated that a close-to-linear dependence between examples from different sets is the cause for this effect. A 'reference test set' is introduced to visualise that shuffling leads to a loss of overtraining indication from the default loss functions, increasing the chance of producing overfitting models. We conclude that for flame models, that rely on a history of velocity perturbations, shuffling before performing the data split into training, validation and tests sets is detrimental for the network design process.

1. INTRODUCTION

Today, the fast development of computer technology and increased digitalisation and online activity leads us into an era of Big Data [1]. Large amounts of information on internet traffic, GPS-tracked movement or user accounts activity are constantly being gathered, resulting in enormous readily available datasets. The reality of Big Data supports the quick development of data-driven methods of machine and deep learning. Their applications are already omnipresent in our lives in the form of recommended adds, traffic and estimated time of arrival predictions as well as blocking suspicious login attempts or financial activity.

Every day machine and deep learning become more established in scientific research. The spectrum of applications is constantly growing. Deep learning in brain tumour diagnosis increases the chance of an early detection and a successful patient treatment [2]. Further, in material science it facilitates the design of new materials with tailored properties and helps to reduce the costs of this process [3]. Finally, a lot of scientific effort is dedicated to shaping the travel and city traffic

[1]rywik@tfd.mw.tum.de
[2]david.da-cruz@tum.de
[3]polifke@tum.de

of tomorrow through replacing drivers with autonomous vehicles [4]. Similarly, the approach gains increasing attention in fluid and combustion dynamics as computational simulations or experimental measurements are complicated and costly. Studies often include pursuit of an efficient order reduction of a flow field, with an example of a hierarchical autoencoder using energetically ordered nonlinear mode decomposition [5]. Others focus on combustion stability monitoring and robustness against noisy data, even with a model relying on unlabelled data [6]. Finally, applications aim to reduce the reliance on expensive experimental measurement equipment, like in the case of a 3D velocity field reconstruction from chemiluminescence images [7].

Furthermore, the deep learning framework is an appealing tool for creating data-driven flame models for investigation of thermoacoustic effects. In thermoacoustics, we are interested in the relation between acoustics and the response of the flame to it in the form of heat release rate fluctuations. At large amplitudes, this is a highly nonlinear interaction, which needs to be captured correctly to accurately model flame behaviour in enclosed spaces such as combustion chambers. In general, neural networks are capable of simulating nonlinear trends, thanks to their nonlinear activation functions [8]. Hence, neural networks can naturally account for nonlinear flame phenomena, including generation of higher order harmonics and saturation of heat release rate, resulting in limit cycle oscillations.

For those reasons, Jaensch et al. [9] and Tathawadekar et. al. [10] developed a neural network flame model with a multi-layer perceptron architecture. Such model is intended to be used in a reduced order model simulation of a combustion system. Recently, Yadav and Ghani [11] further introduced a more involved LSTM-based recurrent architecture, including early work on physics-informed loss function. All authors focused on laminar flames as turbulent flame modelling comes with its own additional challenges and extra uncertainty due to combustion noise. All of those authors verified their proposed networks against the data obtained through computational fluid dynamics (CFD), in the temporal domain as well as in the frequency domain, using a so called Flame Describing Function (FDF). As presented in all of the studies, the FDF predictions were accurate. However, when tested in the reduced order model framework of the full combustor, the network of [9] was unable to provide correct results, in contrast to the one developed in [10]. To further improve the neural network flame models, it is first necessary to investigate the contradictory results of Jaensch [9] and Tathawadekar et al. [10], despite similar approaches. We do not focus directly on [11] as a completely different architecture was used. In this short paper we demonstrate how randomising (i.e. shuffling) the order of the data before splitting them into training, validation and test sets affects the network performance in this application. We believe it to be at least one of the reasons for the discrepancies in the two aforementioned studies.

2. NUMERICAL DATA

In this work, we are analysing exactly the same training dataset used in related investigations by Jaensch [9] and Tathawadekar et al. [10]. This enables us to exclude any data-based reasons for the mismatch, facilitating any comparisons.

The time series data utilised in the training process were generated with a computational fluid dynamics (CFD) simulation, which was a computational representation of the experimental configuration investigated by Kornilov et al. [12]. Those numerical results were produced by Jaensch et al. [13], who also provide a more detailed description of the simulation parameters. Here, it is only briefly summarised. The numerical simulation is a 2D CFD of a laminar slit flame, subjected to

axial velocity perturbations. The excitation signal is a low-pass filtered broadband time series, with a constant amplitude across the frequencies of interest. The monitored variable was the unsteady heat release rate $\dot{q}'$ integrated over the entire flame. The data was produced using highly nonlinear excitation amplitudes of u'. The normalised amplitudes were 0.5, 1.0 and 1.5, which in [10] were referred to as A, B and C, respectively.

The numerical configuration included a symmetry condition, a mean inlet velocity of 0.4 m/s and temperature of 293 K. The walls were set to no-slip condition with a fixed temperature of $373K$. The structured grid consisted of $122,000$ cells, with a cell size of 0.025 mm in the vicinity of the flame. The average timestep was equal to 10^{-6} s, providing very high resolution time-series data for network training.

3. FLAME DESCRIBING FUNCTION

In all of the studies [9–11], the quality of the network was evaluated both in the temporal and frequency domain. The latter is crucial for engineering application, thus the network results were compared against available Flame Describing Function (FDF) data. FDF is a similar concept to a Flame Transfer Function (FTF). The key difference is that FDF, apart from the dependence on frequency ω, also explicitly depends on the amplitude of velocity forcing $|u'|$:

$$FDF(\omega, |u'|/\bar{u}) = \frac{\dot{q}'(\omega, |u'|)/\bar{\dot{q}}}{u'(\omega)/\bar{u}}, \tag{1}$$

where $\bar{u}$, $\bar{q}$ represent the corresponding time-averaged values, used for normalisation.

FTF inherently assumes linear response in forcing amplitude, while FDF does not. Their predictions begin to differ significantly with increasing amplitude of velocity forcing, due to weakly nonlinear effects. In contrast to an FTF, an FDF can model nonlinear phenomena in the form of amplitude saturation, marginal stability and limit cycle formation. However, the FDF still assumes linear behaviour in the output frequency. Namely, the $\dot{q}'$ response is assumed to be exactly at the frequency of the input u'. This neglects any representation of higher order harmonics generated.

4. NEURAL NETWORK

In this section a brief summary on neural networks, namely multi-layer perceptron architectures, is provided, especially in the context of flame modelling. More general details can be found in [8].

Neural networks are a flexible tool, with different architectures being suitable for different applications. They are a data-driven method, analysing a dataset to 'self-tune' its parameters. In this and other studies [9, 10] the focus is set to feedforward networks, as the gathered data represent a forced flame response to the axial velocity forcing u', with no recurrent behaviour expected. Wherever possible, same parameters as in [10] are kept.

A neural network takes an input matrix X_0 and multiplies it by a corresponding set of layer weights stored in the weight matrix W_1. The outcome is then summed at each neuron and a bias term, included in the bias vector b_1, is added. Finally, the sum obtained at every neuron is modulated by an activation function g_1. Figure 1 schematically depicts the process. This sequence is analogously reiterated for other layers, with the output of the previous layer becoming the input for the following layer. The process is repeated until the output layer of the network is reached. The general compact formulation to process the operations for every neuron in the *ith* layer at once, is as follows

$$X_i = g_i(W_i^T X_{i-1} + b_i). \tag{2}$$

After the input X_0 is processed by all the layers, the network produces a prediction $\hat{y}$. This is compared against the true value, available from the dataset. Then, a loss function often in the form of a Mean-Squared-Error (MSE) between the estimate $\hat{y}$ and the correct value y penalises incorrect

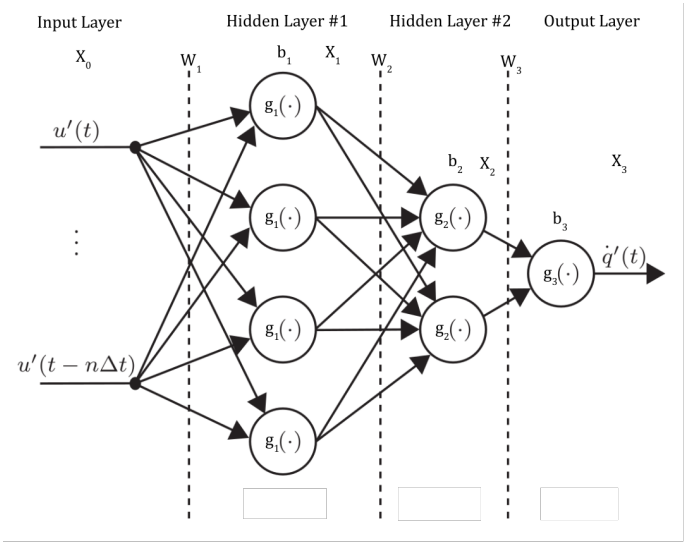

Figure 1: An overview of a multi-layer perceptron neural network architecture. Adapted form [9].

predictions. The next step is referred to as 'backpropagation' and is the mechanic by which a network 'learns' and 'self-tunes'. Using the chain rule, partial derivatives of the loss function with respect to all parameters (weights W, biases b) are calculated. Adjusting the parameter values by stepping in the opposite direction of the gradient, decreases the loss function and helps the network produce better predictions. This is referred to as gradient descent. The step size itself is governed by a learning rate hyperparameter, usually set by the user. Once the parameters are adjusted, the whole process is reiterated, starting by a recalculation of a new prediction $\hat{y}$ and its associated loss. It is continued until a prescribed stopping criterion is met - maximum number of iterations, lack of loss value improvement, etc.

In order for the network to be able to generalise well from the provided data and stop overfitting, the data is split into three sets: training, validation and test. The general idea behind this partition is as follows. The training set is fed to the network to iteratively evaluate the loss and perform gradient descent. The validation set is used only to evaluate the validation loss value and it does not participate in the backpropagation calculation. It serves only to monitor whether the learning process is still generalising well, that is having a similar loss on the training and validation sets. If this is no longer fulfilled, the network is said to be overfitting and the learning process is stopped. The test set is not seen at any point by the learning algorithm - neither in the backpropagation step, nor as a stopping criterion. Hence, the test loss value is used as a final indication of the predictive capabilities of the network.

When splitting the data into those three sets, one can choose to shuffle the order of the corresponding input and output (label) pairs first or immediately divide them chronologically into three blocks. We are highlighting this decision step as this turns out to be an important factor in the training process of a neural network based flame model.

In this study, we are looking for a mapping $\mathcal{F}$, linking the time history U' of u' perturbations to a flame response $\dot{q}'(t)$:

$$U'(t) = [u'(t), u'(t - \Delta t), ..., u'(t - n\Delta t)], \tag{3}$$

$$\dot{q}'(t) \approx \mathcal{F}(U'(t)) \qquad \forall t. \tag{4}$$

$\Delta t = 15\ \mu$s was selected to match the preprocessing of [10], resulting in a downsampling of the initial high resolution time-series CFD data by a factor of 15. This greatly reduces the number of weights in

the first layer, reducing the size of the network - Figure 1. Variable n determines the number of past time steps of u' included in the history vector U'.

The history of u' is necessary to account for the convective nature of a flame response, with perturbations reaching different parts of the flame at different time instants. An important choice is the number of u' regressors used, which is governed by n. Ideally, the delays should span the finite impulse response length of the flame. Shorter than that and the neural network model would be missing physical information, preventing it from correctly modelling the flame. On the other hand, choosing a too long history of regressors results in providing redundant information to the network, hindering the training process. Eventually, 667 regressors were used, spanning a history of 10 ms, which is approximately the length of the flame response for this configuration [14].

5. RESULTS

Shuffling the data before splitting into training, validation and test sets is recommended in many, if not most, neural network applications. Set of images for image labelling, housing prices and surface area for market modelling or user information on whether an add was clicked on are all examples, where there are no coherent temporal relations. In such cases, shuffling would result in a more generalised and accurate model. However, one needs to be wary when to apply it while working with a time series data.

One of the differences between the two studies by Jaensch [9] and Tathawadekar et al. [10] was their approach to data preprocessing. Jaensch et al. shuffled the data before splitting into training, validation and test set. Thus, each input and output data pair, that is a U' history vector and a corresponding $\dot{q}'$ value, was randomly assigned to one of the three sets. On the other hand, Tathawadekar et al. used a different policy and divided the time-series data by blocks along the simulated timeline - the first continuous block was assigned to training, the second to validation and the final one to the test set. In the following paragraphs, we investigate the influence of those different approaches on the resultant network performance.

5.1. Influence of data split policy on monitored loss values

We used a two hidden layer network with 73 and 36 units. Only the output layer had a linear activation function, otherwise tanh was implemented. Additionally, MSE was selected as a loss function. This configuration was tested previously by [10]. Finally, a learning rate of 10^{-4} was used as it provided good training and convergence. We follow the same ratio of training, validation and test set sizes - 56%, 14% and 30% of the whole data, respectively. However, in order to highlight the impact of the aforementioned data splitting approaches, we extract half of the test set (15% of the total data) to serve as a 'reference test set'. This set is never shuffled, independently from the data division policy - it is always kept as a single continuous block. To further clarify this: in case of by block division, we select first 56% of the data pairs as the training set, the next 14% as the validation set, further 15% as the test set and finally the last 15% as the reference test set. When testing the influence of shuffling, again the last 15% of the time series data is selected as the reference test set, while the remainder is randomly sampled into training, validation and test sets, in the end comprising 56%, 14% and 15% of the whole data, respectively. Most importantly, in order to visualise the issue with monitored loss functions, in contrast to [10] we are not using dropout or any kind of regularisation.

We train the networks for both data division policies and estimate the loss function, evaluated on each of those sets. The aim of the loss on the reference test set is to show the impact of different policies on the generalisation abilities of the network. The continuity of the reference test is always preserved. This better emulates the datasets of any future applications, to which the network could be subjected.

Table 1: Loss functions estimated for different data split policies.

Data Split	Training	Validation	Test	Reference Test
Shuffled	1.30e-5	1.73e-5	1.73e-5	2.82e-4
By Block	4.28e-5	3.98e-4	3.79e-4	3.38e-4

As presented in Table 1, in the case of shuffling, the loss values for training, validation and test sets are all of the order of 10^{-5}. This would normally indicate that the network is generalising well and not overfitting as there is no discrepancy between the training and other losses. However, the artificially introduced reference test set displays a loss, which is an order of magnitude greater. The performance on this set is significantly worse than on the training set, indicating that the network has issues with generalisation. Despite that, the loss on the three default sets (training, validation & test) is substantially lower. Such results suggest a critical problem. This indicates that data shuffling is not suited for time series applications, as the generalisation performance is overestimated. The results would suggest that no regularization is needed, which could be untrue as depicted by the reference test loss. This lack of training oversight could lead to selection faulty networks, which on 'paper' were the best performing ones.

On the contrary, when the by block policy is utilised, the validation, test and reference test loss values coincide. On the other hand, the training loss is an order of magnitude lower than the others, clearly indicating overfitting and a need to address it via regularisation. When training on a time series data, data division by blocks preserves the monitoring functions of test and validation sets. Although the comparison of the network results in [9] against FDFs are reasonable, the overtraining consequences could have only revealed themselves in the analysis of the higher harmonic response. Unfortunately, that was not conducted then, contrary to the more promising similar studies [10, 11]. It suggests that outside of the default loss monitors, extra verification of the network results in the form of harmonic analysis (through xFDFs [15] or otherwise) is often beneficial.

5.2. Similarity between data sets

The cause of the mismatch between the loss values on training, validation and test sets for the two data split policies is the different similarity between the sets. To verify this, average maximum cosine similarity between the input examples from different sets are calculated, using the following formulation:

$$\cos\theta = \frac{\mathbf{x}\cdot\mathbf{y}}{||\mathbf{x}||||\mathbf{y}||}, \tag{5}$$

where θ is the angle between vectors $\mathbf{x}$ and $\mathbf{y}$. Cosine value of 0 indicates orthogonality, whereas 1 - a full linear dependence. For each U' history input from set A (vector $\mathbf{x}$), an analogous input with the greatest cosine similarity to $\mathbf{x}$ is found in set B (vector $\mathbf{y}$). The cosine similarity value is then recorded, the process reiterated for all U' vectors in set A and the average is calculated. This average maximum cosine similarity is referred to in Table 2 as 'A to B'.

When shuffling, it is immediately visible that the similarity values for the training, validation and test sets remain close to 1. This shows that for every example vector from one set, we can find a very

Table 2: Cosine similarity values between the sets and the training set for both data split policies. Shuffling results in greater similarity of the validation and test to training set.

Data Split	Validation to Train	Test to Train	Reference Test to Train
Shuffled	0.992	0.992	0.561
By Block	0.548	0.552	0.552

similar, almost linearly dependent vector in another set. Then for every validation/test set example the network has to evaluate, it has already seen an almost identical one within the training examples. However, when the network is given example from a reference test set, for which the similarity to the other sets is considerably lower, the prediction performance decreases - a sign of overfitting. On the other hand, in the case of by block division, all the similarity values remain significantly lower and consistent with the reference test of around 0.5. This enables the training process monitors to give correct indications of overtraining.

5.3. Leakage of input information

The origin of the similarity between between the sets in the case of shuffling can be further visualised. Using this policy, at each time step the time series data pair is in fact randomly assigned into a training, validation or test set. Thus, there is a high probability for every test example to have a neighbouring training example a few or even just a single time step away. Figure 2 depicts such a occurrence, highlighting how it would cause the increased similarity and hence the loss of overtraining supervision.

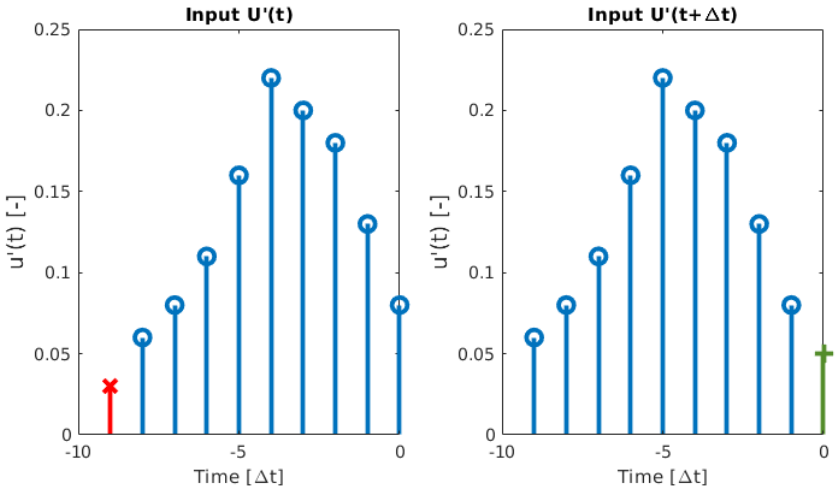

Figure 2: Visualisation of the difference between two subsequent $U'(t)$ and $U'(t + \Delta t)$ input vectors.

The figure presents two hypothetical velocity perturbation histories, described by input vectors U' - one being a training example $U'(t)$ and the other a test example a single time step later $U'(t + \Delta t)$. Note that those are not actual data used and serve only an illustrative purpose. It can be observed that there are three differences between $U'(t)$ and $U'(t + \Delta t)$. The most 'past' constituent of $U'(t)$ is discarded (red 'x'), the remainder gets shifted one time step towards the past by one time step (blue 'o') and a new most recent contribution is added (green '+') to $U'(t + \Delta t)$. Unless the sampling rate is extremely low, the shifted constituents will each replace a neighbour of a very similar magnitude,

producing only minor differences between $U'(t)$ and $U'(t + \Delta t)$ vectors. This results in an unsolicited leakage of input data into the validation and test sets from the training set. It causes the mentioned above similarity between the sets, emanating itself in generalisation issues.

6. CONCLUSIONS

In this work we have investigated inconsistent results of previous studies [9, 10] concerning laminar flame modelling by a means of neural networks. A difference in the data treatment could have been one of the reasons explaining the discrepancies. One study performed shuffling of the data before splitting them into the training, validation and test sets, whereas the other opted to divide the data into chronological blocks. The former policy is demonstrated to result in extremely high similarity of the validation and test sets to the training set. This is caused by the training examples often neighbouring the examples from the validation and test sets. Hence, they share most of the information included in U' time history as their input vectors mostly overlap. As a consequence of this similarity, all of the training, validation and test loss values always remain close of each other. Thus, no feedback is provided on the actual overtraining or generalisation capabilities, giving a false impression of the performance of a network. We conclude that for time series investigations, where a time history of a variable is used, shuffling should not be implemented. This then by induction applies to flame response modelling. In future work, we aim to expand the investigation and look for ways to further improve network architectures for thermoacoustic purposes.

ACKNOWLEDGEMENTS

This work is part of the Marie Skłodowska-Curie Innovative Training Network Pollution Know-How and Abatement (POLKA). We gratefully acknowledge the financial support from the European Union's under the Horizon 2020 Marie Skłodowska-Curie grant agreement No. 813367.

REFERENCES

[1] James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, and Angela Hung Byers. Big data: The next frontier for innovation, competition, and productivity. *McKinsey Global Institute*, 2011.

[2] Gopal S. Tandel, Mainak Biswas, Omprakash G. Kakde, Ashish Tiwari, Harman S. Suri, Monica Turk, John Laird, Christopher Asare, Annabel A. Ankrah, N. N. Khanna, B. K. Madhusudhan, Luca Saba, and Jasjit S. Suri. A Review on a Deep Learning Perspective in Brain Cancer Classification. *Cancers*, 11(1):111, January 2019.

[3] Kamal Choudhary, Brian DeCost, Chi Chen, Anubhav Jain, Francesca Tavazza, Ryan Cohn, Cheol Woo Park, Alok Choudhary, Ankit Agrawal, Simon J. L. Billinge, Elizabeth Holm, Shyue Ping Ong, and Chris Wolverton. Recent advances and applications of deep learning methods in materials science. *npj Computational Materials*, 8(1):59, December 2022.

[4] Sajjad Mozaffari, Omar Y. Al-Jarrah, Mehrdad Dianati, Paul Jennings, and Alexandros Mouzakitis. Deep Learning-Based Vehicle Behavior Prediction for Autonomous Driving Applications: A Review. *IEEE Transactions on Intelligent Transportation Systems*, 23(1):33–47, January 2022.

[5] Kai Fukami, Taichi Nakamura, and Koji Fukagata. Convolutional neural network based hierarchical autoencoder for nonlinear mode decomposition of fluid field data. *Physics of Fluids*, 32(9):095110, September 2020.

[6] Zhezhe Han, Md. Moinul Hossain, Yuwei Wang, Jian Li, and Chuanlong Xu. Combustion stability monitoring through flame imaging and stacked sparse autoencoder based deep neural network. *Applied Energy*, 259:114159, February 2020.

[7] Shivam Barwey, Malik Hassanaly, Venkat Raman, and Adam Steinberg. Using machine learning to construct velocity fields from OH-PLIF images. *arXiv:1909.13669 [physics]*, September 2019.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[9] Stefan Jaensch and Wolfgang Polifke. Uncertainty Encountered When Modelling Self-Excited Thermoacoustic Oscillations with Artificial Neural Networks. *Int. J. Spray Combust. Dyn.*, 9(4):367–379, 2017.

[10] N. Tathawadekar, A. K. Doan, C. F. Silva, and N. Thuerey. Modelling of the nonlinear flame response of a Bunsen-type flame via multi-layer perceptron. *Proceedings of the Combustion Institute*, 38(4):6261–6269, January 24th to 29th, 2021.

[11] Vikas Yadav and Abdulla Ghani. Physics-informed Recurrent Neural Networks for Linear and Nonlinear Flame Dynamics. In *39th International Symposium on Combustion [Accepted]*, volume 39, Vancouver, BC, Canada, 2022. Combustion Institute.

[12] V. N. Kornilov, R. Rook, J. H. M. ten Thije Boonkkamp, and L. P. H. de Goey. Experimental and Numerical Investigation of the Acoustic Response of Multi-Slit Bunsen Burners. *Combustion and Flame*, 156(10):1957–1970, October 2009.

[13] S. Jaensch, M. Merk, E.A. Gopalakrishnan, S. Bomberg, T. Emmert, R.I. Sujith, and W. Polifke. Hybrid CFD/Low-Order Modeling of Nonlinear Thermoacoustic Oscillations. *Proceedings of the Combustion Institute*, 36(3):3827–3834, 2017.

[14] Camilo F. Silva, Thomas Emmert, Stefan Jaensch, and Wolfgang Polifke. Numerical Study on Intrinsic Thermoacoustic Instability of a Laminar Premixed Flame. *Combustion and Flame*, 162(9):3370–3378, 2015.

[15] Matthias Haeringer, Malte Merk, and Wolfgang Polifke. Inclusion of higher Harmonics in the Flame Describing Function for Predicting Limit Cycles of self-excited Combustion Instabilitites. *Proceedings of the Combustion Institute*, 37(4):5255–5262, 2019.