Bayesian Inference in Physics-Based Nonlinear Flame Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

This study uses a Bayesian machine learning method to infer the parameters of a physics-based model of a bluff-body-stabilised flame in real-time. An ensemble of neural networks is trained on a library of simulated flame fronts with known parameters, generated using a level-set solver, LSGEN2D. The ensemble learns a surrogate of the approximate Bayesian posterior of the parameters given the observations, from which the flame can be re-simulated beyond the observation window of the experiment. The method is general: once trained, the ensemble can be used to infer the parameters from any bluff-body-stabilised flame as long as the flame is qualitatively similar and the parameters lie within the ranges in the training library. Amortized inference takes milliseconds, which is fast enough to work in real-time.

1 Introduction

The modelling of complex physical phenomena often relies on the solution to partial differential equations. Starting from a complete model description of the physical system, including model parameters, simulation can be used to solve the forward problem. It is often desirable, however, to infer model parameters from measurements, which are usually noisy: this constitutes the inverse problem. If assumptions about the statistics of the measurement noise are made and prior knowledge about the un-observed parameters is considered, the inverse problem can addressed through a Bayesian inference framework [1]. A key advantage of Bayesian inference for solving inverse problems is that uncertainties are also quantified, through the inferred probability distributions of the un-observed quantities.

Traditional filtering-based techniques for parameter inference, such as the ensemble Kalman filter [2, 3], require the system to be simulated in parallel, which becomes infeasible if the underlying model is computationally expensive to simulate. Instead, deep learning-based methods are made possible in part because synthetic data sets can be created cheaply at scale [4]. Amortized inference using neural networks involves an expensive offline training phase, where a surrogate of the approximate posterior $p(\mathbf{t}|\mathbf{z})$ is learnt from a library of simulator-generated observations $\mathbf{z}_i$ corresponding to input parameters $\mathbf{t}_i$ [5]. The surrogate can then be rapidly evaluated online to perform parameter inference on new observed data.

In this study, the physical system is a version of the Volvo burner: a partially-observed bluff-body-stabilised flame inside a rectangular duct [6, 7]. The G-equation model [8] for the flame front models how acoustic perturbations at the base of the flame cause wrinkles that propagate down the flame. The flame's surface area variation in time is a proxy for the heat release rate - a key quantity in determining the thermoacoustic stability of the burner. Inferring the model parameters allows the flame to be simulated beyond the observation window so that the surface area variation can be calculated, with

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

uncertainties, arbitrarily far downstream. Bayesian neural network ensembles are used to learn the surrogate posterior of the parameters of the model from images of the partially-observed flame [9, 10].

2 Volvo burner experiment, model and simulations

2.1 The Volvo burner

Experiments are performed on a version of the Volvo burner shown schematically in Figure 1. Premixed air and propane flow into the combustor and are burnt by a flame stabilized on a triangular bluff body with side length $D = 3.8$ cm. As the air-fuel mixture flows through the combustor, vortices are shed periodically from the bluff body. Images of the flame are recorded at 10kHz using OH planar laser induced fluorescence (OH PLIF) through a window $3D$ tall and $3.4D$ wide. The images are processed to find discretisations of the position $y = f(x)$ of the flame front by thresholding and interpolating the magnitude of the OH gradient vector at each point. The vectors of positions $\mathbf{y}$ are smoothed using splines with 10 knots. To create an observation vector $\mathbf{z}$ representing a sequence of 10 flame front positions, 10 position vectors are appended together.

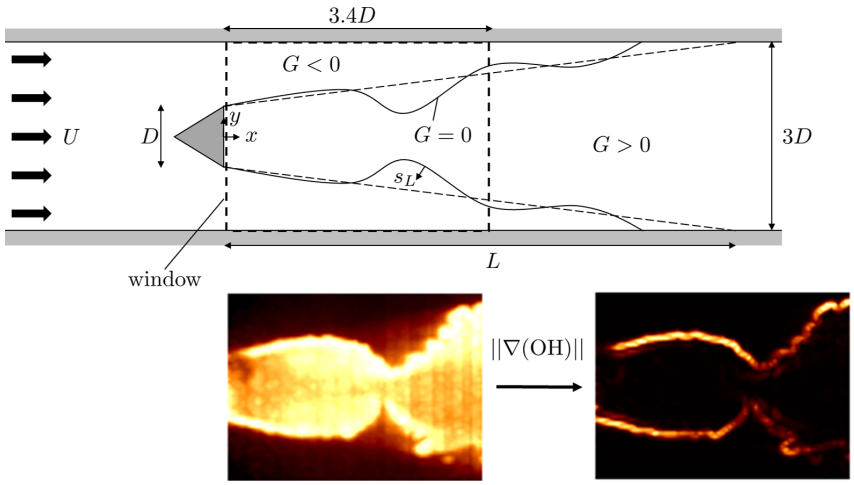

Figure 1: Diagram of the Volvo combustor rig and G-equation model of the flame. As the air-fuel mixture flows through the combustor, vortex shedding causes wrinkling and cusping of the flame front, represented by the $G = 0$ contour of a continuous scalar field $G(x, y, t)$. An example OH intensity image taken through the window is shown, as well as the pre-processing step to find the flame front.

2.2 The G-equation model of the flame front

The flame front is assumed to be a thin boundary between unburnt and burnt gases (see Fig. 1). The flame travels normal to itself into the unburnt gases with laminar flame speed s_L which depends on the gas composition. The velocity in the burnt gases does not affect the flame kinematics. The unburnt and burnt gases are assumed to travel with velocity $\mathbf{u}(x, y, t)$. Under these assumptions, the flame front is modelled by the $G(x, y, t) = 0$ contour (or level-set) of a continuous scalar field G whose motion is governed by the G-equation:

$$\frac{\partial G}{\partial t} + \mathbf{u} \cdot \nabla G = s_L |\nabla G|. \tag{1}$$

The flow velocity field $\mathbf{u}$ is assumed to comprise a constant and uniform base flow U and superimposed physics-based (continuity-obeying) velocity perturbations $u'(x, y, t)$ and $v'(x, y, t)$:

$$\frac{\mathbf{u}(x, y, t)}{U} = (1 + u'(x, y, t))\,\mathbf{i} + v'(x, y, t)\mathbf{j}, \tag{2}$$

$$u'(x, y, t) = \varepsilon \left(\frac{x}{D}\right)^{\eta} \sin\left(\mathrm{St}\left(K\left(\frac{x}{D}\right) - t\right)\right), \tag{3}$$

where v' is calculated using $\nabla.\mathbf{u} = 0$. In the above, U is a characteristic speed, ε is a non-dimensional amplitude, St is the Strouhal number of the flame with characteristic length D, excitation frequency f, and nominal aspect ratio β: $\text{St} = 2\pi f \beta D/U$, and K is the ratio of the characteristic speed U to the perturbation phase speed. The parameter η is introduced to the flame perturbation model to allow for the horizontal velocity perturbations to increase in size with distance from the flame holder, which is the qualitative behaviour observed in the experiment. This has proven to be a versatile and physically descriptive flame front model in several previous studies, despite having only a few parameters [11]. To make the G-equation model quantitatively accurate, the parameters $K, \varepsilon, \eta, \text{St}$ and β must be tuned to fit an observed flame shape.

2.3 The simulated flame front library

LSGEN2D [12] is a level-set solver that iterates the G field of the G-equation model for known parameters $K, \varepsilon, \eta, \text{St}$ and β. In this study, the G field is iterated until convergence to a set of 200 different periodic solutions. This is repeated for 2400 combinations of parameters sampled from the ranges shown in Table 1. The forced cycle states are processed to find a $y = f(x)$ discretisation of the $G = 0$ contour, for all x in the range of the experiment observation window. This is done by interpolating the G field values for every vertical coordinate, and recording the positions y in vectors $\mathbf{y}$. To create a single observation vector $\mathbf{z}$ representing a sequence of 10 flame front positions, 10 position vectors are appended together. This is repeated for every state in the forced cycle. The result is a library of 4.8×10^5 observation - parameter pairs.

Table 1: Parameters of the G-equation model that are varied in this study and the range over which they are varied in order to generate the synthetic flame front library. v_p is the perturbation phase speed.

Parameter	Range	Description
K	0 - 2	Ratio U/v_p
ε	0 - 1	Perturbation amplitude
η	0 - 3	Spatial growth rate
St	5 - 30	Strouhal number
β	4 - 8	Flame aspect ratio

3 Inference using a heteroscedastic Bayesian neural network ensemble

The posterior probability distribution $p(\mathbf{t}|\mathbf{z})$ of the G-equation parameters $\mathbf{t}$, given the observations $\mathbf{z}$ is assumed to be a multivariate Gaussian with mean vector $\boldsymbol{\mu}(\mathbf{z})$ and diagonal covariance matrix $\boldsymbol{\Sigma}(\mathbf{z}) = \text{diag}(\boldsymbol{\sigma}^2(\mathbf{z}))$. An ensemble of M neural networks are trained on the synthetic flame front library to predict the mean and variance vectors, $\boldsymbol{\mu}(\mathbf{z})$ and $\boldsymbol{\sigma}^2(\mathbf{z})$. Each neural network in the ensemble produces estimates $\boldsymbol{\mu}_j(\mathbf{z}_i)$ and $\boldsymbol{\sigma}_j^2(\mathbf{z}_i)$ for each observation vector $\mathbf{z}_i$. These estimates are combined as follows:

$$\boldsymbol{\mu}(\mathbf{z}_i) = \frac{1}{M} \sum_j \boldsymbol{\mu}_j(\mathbf{z}_i), \tag{4}$$

$$\boldsymbol{\sigma}^2(\mathbf{z}_i) = \frac{1}{M} \sum_j \boldsymbol{\sigma}_j^2(\mathbf{z}_i) + \frac{1}{M} \sum_j \boldsymbol{\mu}_j^2(\mathbf{z}_i) - \boldsymbol{\mu}^2(\mathbf{z}_i), \tag{5}$$

following Ref. [13]. Each neural network comprises 4 fully connected layers 600 hidden units wide, and two output layers (one for the mean vector, one for the variance vector) each 6 units wide. ReLU activations are used for the hidden layers, a sigmoid activation is used for the output layer for the mean and an exponential activation is used for the variance layer, to ensure positivity. The architecture of one such neural network is shown in Appendix A.1. Further hyperparameter details are listed in Appendix A.2. The weights $\boldsymbol{\theta}_j$ of each neural network are initialised by sampling from Gaussian prior distributions with means $\mathbf{0}$ and covariance matrices $\boldsymbol{\Sigma}_{prior}$ according to He normalisation [14]. During training, the weights are anchored to their initial values $\boldsymbol{\theta}_{j,anc}$. The loss function used for

training is:

$$\mathcal{L}_j = \left(\boldsymbol{\mu}_j(\mathbf{z}) - \mathbf{t}\right)^T \boldsymbol{\Sigma}_j\left(\mathbf{z}\right)^{-1}\left(\boldsymbol{\mu}_j(\mathbf{z}) - \mathbf{t}\right) + \log\left(\left|\boldsymbol{\Sigma}_j\left(\mathbf{z}\right)\right|\right) + \left(\boldsymbol{\theta}_j - \boldsymbol{\theta}_{anc,j}\right)^T \boldsymbol{\Sigma}_{prior}^{-1}\left(\boldsymbol{\theta}_j - \boldsymbol{\theta}_{anc,j}\right). \tag{6}$$

Training the ensemble in this way is known as Bayesian ensembling with maximum a-posteriori (MAP) sampling [15]. An ensemble of size $M = 20$ is trained for 100 epochs on a Tesla P100 GPU. This takes approximately 3 hours. Once converged, the ensemble is evaluated on the observations, which takes milliseconds.

4 Results

Fig. 2 shows the results of inferring the parameters from a sequence of 430 experimental images. These parameter estimates are used to re-simulate the flames, which match the experiments well in the observation window (top left quadrant of each re-simulation image). The appropriately-tuned flame models accurately predict the oscillations in the flame front position and their evolution with time. This shows that the method is working well in the observed region. The model then extrapolates beyond the observed region, using the physics-based velocity field and the G-equation. With the flame fully re-simulated, the surface area variation can be calculated.

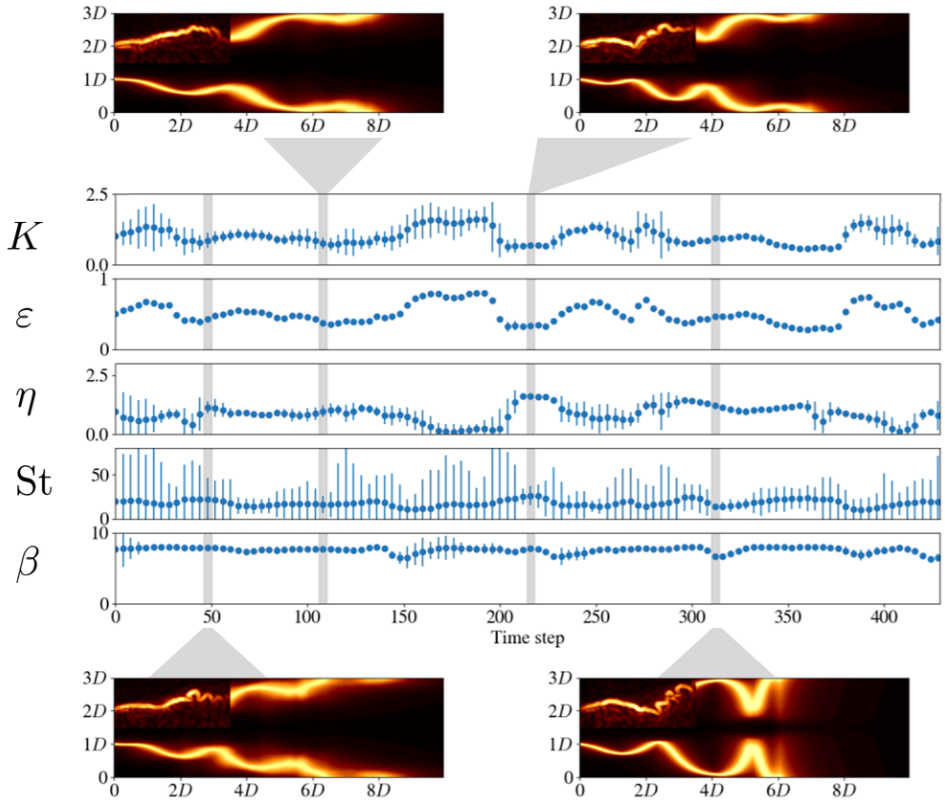

Figure 2: Results of inference using the neural networks. In the centre, the parameter means (dots) and uncertainties (bars) are plotted for every time step. The parameters are used to re-simulate the flames, as shown in the outer four images. The top left quadrant of each re-simulation image is the experimental data. The other three quadrants show the G-field domain, which extrapolates the flame downstream.

5 Conclusions

This study uses a Bayesian machine learning method to infer the parameters of a physics-based model of the flame front of a bluff-body-stabilized flame. Using this method, parameters can be inferred with known uncertainty in milliseconds, as long as the parameters fall within the range of the training data set. The method is demonstrated here for a physics-based nonlinear flame model, but this could in principle be performed for any computational fluid dynamics (CFD) solution. Firstly, this provides a cheap way to store CFD data: for example the parameters of the most relevant CFD solution for a given experiment can be extracted cheaply, and the CFD solution then re-calculated. Secondly, it shows how sparse experimental results can be combined with complete numerical results to extrapolate, with defined confidence levels, beyond experimental observations.

Broader Impact

This work will lead to cost savings in high-energy density combustor design by enabling engineers to quickly and reliably tune their models to match experiments.

Disclosure of Funding

This project has received funding from the UK Engineering and Physical Sciences Research Council (EPSRC) award EP/N509620/1 and from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement number 766264.

References

[1] A. Tarantola. *Inverse problem theory and methods for model parameter estimation*, volume 1. Society for Industrial and Applied Mathematics, 2005.

[2] G. Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using monte carlo methods to forecast error statistics. *Journal of Geophysical Research: Oceans*, 1(99): 10143–10162, 5 1994.

[3] H. Yu, M. P. Juniper, and L. Magri. Combined state and parameter estimation in level-set methods. *J. Comp. Phys.*, 399, 2019.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, volume 1. MIT Press, 2016.

[5] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, Lille, France, 2015. JMLR: W&CP.

[6] B. Paxton, C. A. Fugger, B. A. Rankin, and A. W. Caswell. Development and characterization of an experimental arrangement for studying bluff-body-stabilized turbulent premixed propane-air flames. *AIAA SciTech Forum*, 2019.

[7] B. Paxton, C. A. Fugger, A. S. Tomlin, and A. W. Caswell. Experimental investigation of fuel type on combustion instabilities in a premixed bluff-body combustor. *AIAA SciTech Forum*, 2020.

[8] F. A. Williams. Turbulent combustion. *The mathematics of combustion*, pages 97–131, 1985.

[9] U. Sengupta, M. Amos, J. Scott Hosking, C. E. Rasmussen, M. P. Juniper, and P. J. Young. Ensembling geophysical models with bayesian neural networks. *34th Conference on Neural Information Processing Systems (NeurIPS), Vancouver, Canada*, 2020.

[10] M. L. Croci, U. Sengupta, and M. P. Juniper. Data assimilation using heteroscedastic bayesian neural network ensembles for reduced-order flame models. In *Paszynski M., Kranzlmüller D., Krzhizhanovskaya V.V., Dongarra J.J., Sloot P.M. (eds) Computational Science – ICCS 2021*, Krakow, Poland, 2021. Springer, Cham.

[11] K. Kashinath, L. K. B. Li, and M. P. Juniper. Forced synchronization of periodic and aperiodic thermoacoustic oscillations: lock-in, bifurcations and open-loop control. *Journal of Fluid Mechanics*, 838:690–714, 2018.

[12] S. Hemchandra. *Dynamics of Turbulent Premixed Flames in Acoustic Fields*. PhD Thesis, 2009.

[13] B. Lakshminarayanan, A Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *31st Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, USA*, 2017.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[15] T. Pearce, M. Zaki, A. Brintrup, N. Anastassacos, and A. Neely. Uncertainty in neural networks: Bayesian ensembling. *International Conference on Artificial Intelligence and Statistics*, 2020.

A Appendix

A.1 Neural network architecture

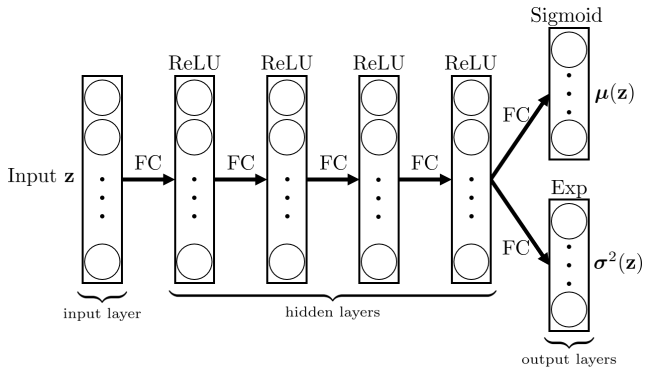

Figure 3: Architecture of each neural network in the ensemble of 20. The input and hidden layers have 600 units each, while each output layer has 6 units each. All layers are fully connected (FC). Rectified Linear Unit (ReLU) activation functions are used for the hidden layers and sigmoid and exponential (Exp) activation functions are used for the mean and variance output layers respectively.

A.2 Hyperparameters

Table 2: Hyperparameter settings used for neural network training.

Hyperparameter	Value
Training	
Train-test split	80:20
Batch size	256
Epochs	100
Optimiser	Adam
Learning rate	10^{-4}