Efficient and accurate time-integration of combustion chemical kinetics using artificial neural networks

Wen Yu Peng (wypeng), Nicolas H. Pinkowski (npinkows)

Abstract An artificial neural network (ANN) was designed and trained to approximately calculate the time evolution of reacting H_2/O_2 chemical systems. The ANN uses the temperature, pressure, and chemical composition at the current time step and outputs the chemical composition at the next time step. Training data sets were generated by numerical integration of the H_2 oxidation sub-mechanism of the GRI 3.0 chemical kinetics mechanism under a variety of combustion-relevant thermodynamic conditions. A systematic search for the optimal ANN architecture was conducted, yielding a system of 8 individual neural networks for each of the 8 species in the reaction mechanism—each network consisted of 2 hidden layers containing 30 ReLU-activated neurons per layer. The ANN was successfully trained against 29038 input-output pairs with a fixed time-increment of 100 ns and matched unseen test data within 0.22% on average. When used to compute the full species time-history of a mixture, the ANN matched numerical simulations to within 0.3% by mole for the best cases, although for several cases, ANN prediction errors at early times compounded to produce inaccurate results at later times. On average, the ANN was 35 times faster than direct numerical solvers, demonstrating the potential of ANNs for approximate computation of chemistry in practical reactive flow simulations.

1 Introduction

Reactive computational fluid dynamics (CFD) models are used extensively to simulate and optimize the design of practical combustion systems such as internal combustion and rocket engines. Central to CFD simulations are chemical kinetics mechanisms, which are collections of expressions that detail the reaction rates of elementary reactions within a system of molecular species that determine how the various chemical species react and evolve in time under prescribed initial conditions. Using these mechanisms, the most accurate method for computing species time-histories involves solving systems of nonlinear ordinary differential equations (ODEs) which, owing to the vastly different time scales of the various reaction rates, are often numerically stiff and expensive to solve, precluding widespread use of complex mechanisms in practical reacting CFD simulations.

Due to the intractability of directly integrating chemical reactions, approximate methods for computing species time-histories are sought. Modelers often use simplified chemical models with fewer species and reactions than the full set; but these simplifications require human input and may limit their applicability to arbitrary conditions. Other methods rely on generating high-dimensional lookup tables which, although both fast and accurate, require significant computational overhead and memory to generate and store these tables. Artificial neural networks (ANNs) that approximately advance the chemistry forward in time, on the other hand, provide a good compromise between the computational complexity of the full reaction mechanism and the overhead/memory requirements of lookup tables.

ANNs have been used in Blasco et al. [1] and Shenvi et al. [2] to time-advance simplified CH_4 and H_2 oxidation mechanisms, respectively. Laubscher [3] trained an ANN to replicate a simplified 5-step CH_4 reaction mechanism and applied the network to a turbulent flame CFD simulation. Comparisons between the ANN-based simulations and the mechanism-based simulations showed no significant difference in results while the ANN-based simulation was able to reduce computation time by a factor of 100. However, it is unclear whether these ANNs generalize to more complex detailed kinetics mechanisms involving thousands of elementary reactions and hundreds of chemical species.

We build on these previous results by designing and training an ANN to replicate the time-evolution of the 8-species, 20-reaction mechanism describing H_2 oxidation [4] over a range of initial temperature, pressures, and species compositions. The full ANN was composed of 8 individually trained networks for each species, each with 2 hidden layers and 30 neurons per layer. Training data consisting of 29038 example input-output pairs generated from directly integrating the mechanism were used to train the ANN. When tested on initial conditions the ANN was never trained on, ANN-computed species time-histories matched the numerical solutions within 0.3% on average.

2 Dataset and features

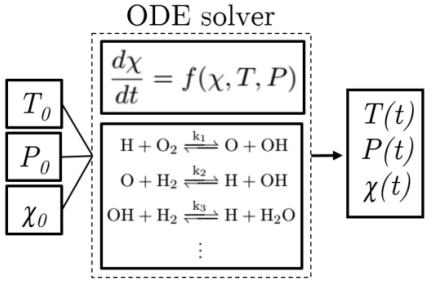

Fig. 1 Flow-chart for computing the time-evolution of chemical systems via numerical integration subject to a prescribed reaction mechanism.

Fig. 1 shows the direct method for time-advancing a reactive chemical system: given some initial temperature, T_0 (K), pressure, P_0 (atm), and species mole fraction vector, χ_0, a numerical solver integrates the ODEs describing the time-evolution of the species (top middle) as prescribed by some reaction mechanism (bottom middle) and outputs the temperature, pressure, and species time-history (right). The reaction mechanism used in this work is the H_2/O_2 sub-mechanism of the GRI 3.0 C_1-C_3 oxidation mechanism [4] involving 8 species (H_2, O_2, H_2O, H, O, OH, HO_2, H_2O_2) and 20 elementary reactions.

Training data was generated by initializing T between 1300 and 1500 K while P was kept constant at 1.5 atm. This initialization scheme was chosen because our primary goal was to develop a neural network that works at least over some restricted range of conditions; future work will expand this range to generalize the results shown in this work to arbitrary T and P. χ was initialized with only the stable species (H_2, O_2, H_2O) to produce physically reasonable species time-histories (it is impossible, for example, for a combustion mixture to begin with a very high concentration of radical species). Each initialization was time-advanced using the Cantera chemical kinetics ODE solver [4] at constant T and P with a fixed $\Delta t = 100$ ns until χ_{H_2O} reached within 1% of its equilibrium value.

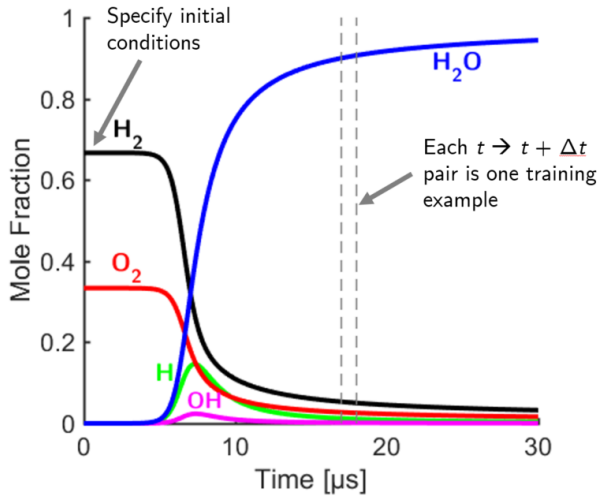

Fig. 2 Sample time-evolution to equilibrium of a stoichiometric H_2/O_2 mixture at constant $T = 1300$ K, $P = 1.5$ atm with time increment, $\Delta t = 100$ ns. Each $t \rightarrow t + \Delta t$ data pair in the time-series is stored as data to train the artificial neural network integrator.

Fig. 2 shows an example species time-history initialized with $T = 1300$ K, $P = 1.5$ atm, 67% H_2, and 33% O_2 by mole corresponding to an equivalence ratio of 1. Each $t \rightarrow t + \Delta t$ input-output pair was stored as a training example for later use; in the case of Fig. 2, the chemistry was advanced over 1432 time steps resulting in 1431 training examples. For the results presented in this work, 29038 training examples were generated over 20 evenly-spaced temperature initializations all with an equivalence ratio of 1.

3 Methods

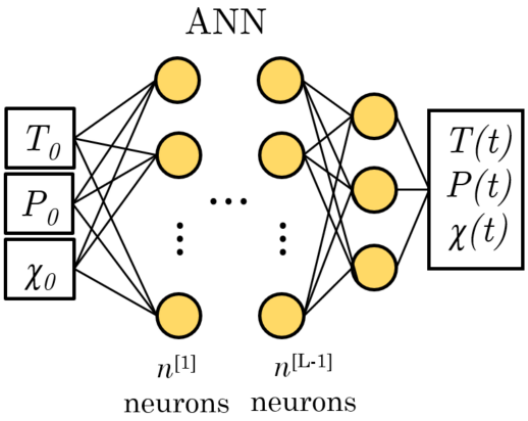

Fig. 3 Flow-chart for approximate time-integration of chemical reactions using a multi-layer perceptron artificial neural network.

In this work, explicit time-integration of the chemical system was replaced by an approximate ANN-based time-integrator. Here, given a set of initial conditions at time t_0, the ANN computes the approximate mole fraction at a later time $t_0 + \Delta t$, with Δt being some fixed time increment as dictated by the training set. The outputs are fed repeatedly back into the ANN as inputs to compute the full time-history of each chemical species. Energy conservation based on thermodynamic constraints is enforced between adjacent time steps to explicitly compute temperature and pressure and hence these variables are not considered outputs of the ANN.

This work uses the multi-layer perceptron (MLP) architecture, a type of ANN that is well suited for nonlinear regression problems. MLPs are essentially nonlinear mappings between the input and output variables and is described by a set of unknown model coefficients that are determined by training the model on pairs of input-output data. Fig. 3 shows the structure of the MLP, which is characterized by an input layer consisting of the current T, P, and χ separated from T, P, and χ at the next time-step by $L - 1$ hidden layers consisting of $n^{[l]}$ neurons each. The output, $a_j^{[l]}$, of the j^{th} neuron in the l^{th} layer can be described mathematically as a feed-forward operation:

$$a_j^{[l]} = g\left(W_j^{[l]} a^{[l-1]} + b_j^{[l]}\right) \tag{1}$$

Here, g is some nonlinear activation function, $W_j^{[l]}$ is the weight vector, $a^{[l-1]}$ is the vector of outputs from the previous layer, and $b_j^{[l]}$ is the bias constant. The feed-forward operation propagates layer-by-layer through the MLP until it reaches the output layer, where $a_j^{[L]}$ is stored as the j^{th} output, $\hat{y}_j$. g, L, and $n^{[l]}$ are user-defined MLP architecture parameters that significantly impact the performance of the ANN; the methods used to select the optimal architecture parameters and to train the ANN are

described in the following subsection.

3.1 *Training and ANN architecture selection*

The cost function, J, used to evaluate the performance of the ANN for some set of network coefficients, $\mathbf{\Theta}$, was chosen to be as follows:

$$J(\mathbf{\Theta}) = \frac{1}{mK}\sum_{i=1}^{m}\sum_{j=1}^{K}\left\{\left[\left|\hat{y}_j^{(i)} - y_j^{(i)}\right| + \log\left(\frac{\hat{y}_j^{(i)}}{y_j^{(i)}}\right)\right]^2\right\} \tag{2}$$

Here, $\hat{y}_j^{(i)}$ is the ANN-predicted mole fraction for the j^{th} species and i^{th} training example, $y_j^{(i)}$ is the target value, and K is the total number of species (8 in this case). The first term of the cost function directly penalizes the ANN for not fitting the data well. The second term is a separate incentive to encourage the training algorithm to more accurately fit the low-concentration species. In testing, we have found that this second term is particularly important for improving ANN performance for the radical species (H, O, OH, HO_2, H_2O_2) because any mismatch between the ANN and the data at low concentrations will result in compounding error and non-physical results as the chemical system moved forward in time. This penalty scheme is also meaningful from a physical perspective because combustible mixtures are ignited by the exponential growth of radical species concentrations, therefore making the overall behavior of a combustion system highly sensitive to the early-time behavior of the radical species.

Prior to training, the following steps were taken to prepare the training algorithm:

1. Input layer parameters were log-normalized as follows:
$$\hat{x} = \frac{\log(x) - \langle\log(x)\rangle}{\sqrt{\text{var}(\log(x))}}$$
where x are the input layer parameters, $\hat{x}$ are the log-normalized parameters, $\langle\cdot\rangle$ denotes the ensemble average, and var denotes the variance. Log-normalization was done because the radical species mole fraction distributions are highly skewed towards zero.

2. A random 10% subset of the training data was used for cross-validation.

3. $W^{[l]}$ was randomly initialized according to the Xavier method [5]: $W^{[l]} \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n^{[l]}+n^{[l-1]}}}\right)$. All $b^{[l]}$ were initialized to zero.

Network coefficients were trained using mini-batch Polak-Ribiere conjugate gradient-descent [6], with coefficient gradients computed using the backpropagation method and verified by comparing the results against a numerically-computed gradient. The Polak-Ribiere scheme eliminates the need to specify a learning rate, thereby eliminating the risk of training instability. The training set was split into mini-batches of 100 training examples each.

In early attempts at training the ANN, the network was trained to simultaneously fit all species. This was the favored approach because (1) the softmax function, which guarantees that $\sum\chi = 1$ and $\chi \succeq 0$ (both are required for conservation of mass), could be used as the output activation function and (2) previous researchers mentioned in Sec. 1 have succeeded with this approach. However, it was quickly realized that, due to the non-convex nature of MLPs, the learning algorithm would often get trapped in local minima where the ANN would perform well only for certain species. We therefore opted to train individual MLP networks for each species independently. This made the ANN both faster to train and easier to modify in case one of the species networks did not perform as well as needed.

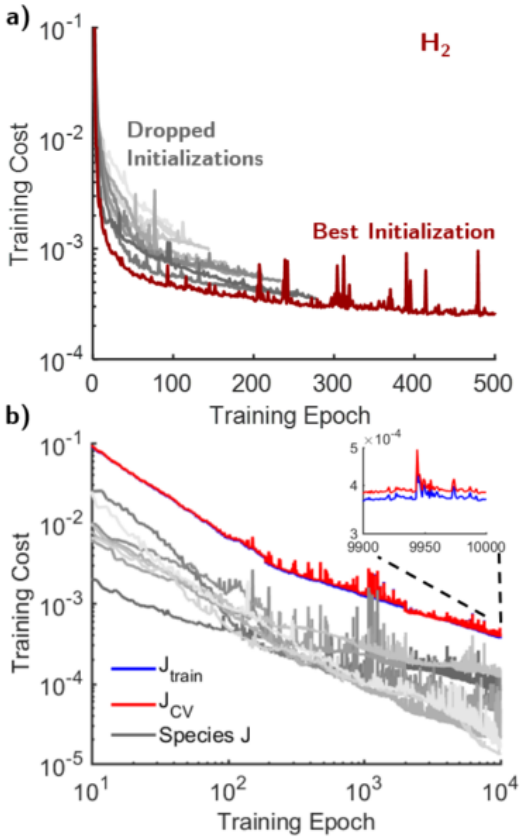

Fig. 4 (a) Example simultaneous learning drop-out initialization for the H_2 network using the method outlined in [7]. (b) Learning curves for the training, cross-validation, and individual species as a function of training epoch.

To accelerate the training process and to reduce the risk of local minima, the simultaneous learning drop-out initialization scheme proposed by Atakulreka & Sutivong [7] was implemented. In this method, 10 ANNs were initialized for each species and simultaneously trained for 100 epochs. An error metric that combines the current training error and an extrapolated future error was used

to judge the performance of each initialization. Once every 20 epochs after the 100th epoch, the worst-performing network was removed from consideration until only one network survived at 300 epochs. This network was then trained for a minimum of 10000 epochs. Fig. 4a shows the behavior drop-out initialization for H_2 in a semi-logarithmic scale, with J_{train} of the dropped initializations plotted in various shades of gray and J_{train} of the surviving network plotted in cardinal. As expected, all of the dropped initializations never perform better than the surviving initialization, therefore only the best initialization is worthy of being trained to completion.

Fig. 4b shows the learning curves on a log-log scale for the best-performing ANN architecture, with the training and cross-validation costs plotted in blue and red, respectively. Note that although the log-log scaling suggests that the ANN has not yet converged onto the minimum after 10000 epochs, the training algorithm has reached a point of diminishing returns and in practice the ANN cannot perform much better than its current state. Training costs of the individual species are plotted in various shades of gray to illustrate the relative performance of each species network. This optimal architecture consisted of 2 hidden layers with 30 neurons each using ReLU activation and linear weights. Because mole fractions are, by definition, bounded between 0 and 1, the sigmoid function was used as the activation function for the output layer. In separate testing, it was determined that this architecture was able to fit the training data to a very high degree of accuracy while remaining a relatively simple network that does not require excessive computational cost. Additionally, as seen in the figure inset in the two right corner of Fig. 4b, J_{train} and J_{CV} lie almost exactly on top of each other, demonstrating that the ANN does not overfit the training data and precluding the need for adding regularization to the cost function.

4 Results

Fig. 5 compares the ANN predictions species mole fractions against a testing data set consisting of 10000 examples that the ANN has never seen before. Although there are eight species overall, in the interest of concision only two species are shown to represent the ANN's performance on the two classes of species: a stable species (H_2O, 5a) and a radical species (OH, 5b). The dotted black line represents the ideal prediction = test example line. As can be seen, the ANN was able to replicate the test examples with almost perfect accuracy; the ANN predictions matched within 0.22% of the test values for all species.

We now assess how well the ANN performs when used to compute the full time-history of a combusting H_2/O_2 mixture. Up to this point, the ANN has been trained and assessed using perfectly known conditions at the input layer. Problems can arise, however, if small ANN errors near the beginning of a time-series compound with

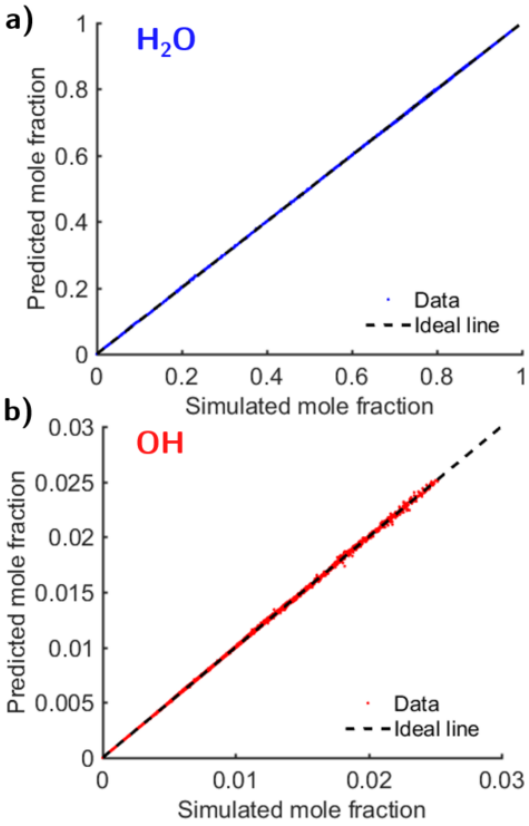

Fig. 5 Comparison between the ANN model prediction and 10000 simulated examples that the ANN was never trained on (a) H_2O and (b) OH

successive increments in time, potentially causing significant deviation from numerical simulations at later times. These deviations can be assessed by calculating the mean squared error between the ANN-computed time-history and the exact numerical solutions for a given initial condition. This figure of merit distinguishes itself from the cost function used to train the network because it measures the ANN's performance over many time steps rather than just one step step forward in the future.

Fig. 6 compares the ANN-predicted time-series against the numerical ODE solution for a stoichiometric mixture of H_2 and O_2 at constant $T = 1394$ K and $P = 1.5$ atm. This temperature was randomly chosen within the range of initial temperatures in the training set but has never been seen by the training algorithm. The solid lines in this figure show the exact numerical solution for the species time-history while the solid dots represent the ANN predictions at 1.5 μs increments (equal to 15 ANN iterations time increments). As can be seen, the ANN predictions match qualitatively very well with the numerical solution, but a few species such as atomic oxygen and hydrogen start to deviate from the numerical solution near 20 μs, indicating the propagation and growth of earlier errors. The mean squared error between the numerical solution and the ANN approximation for this case was 1.37×10^{-5}, corresponding to an average difference of 0.37% in species concentration by mole.

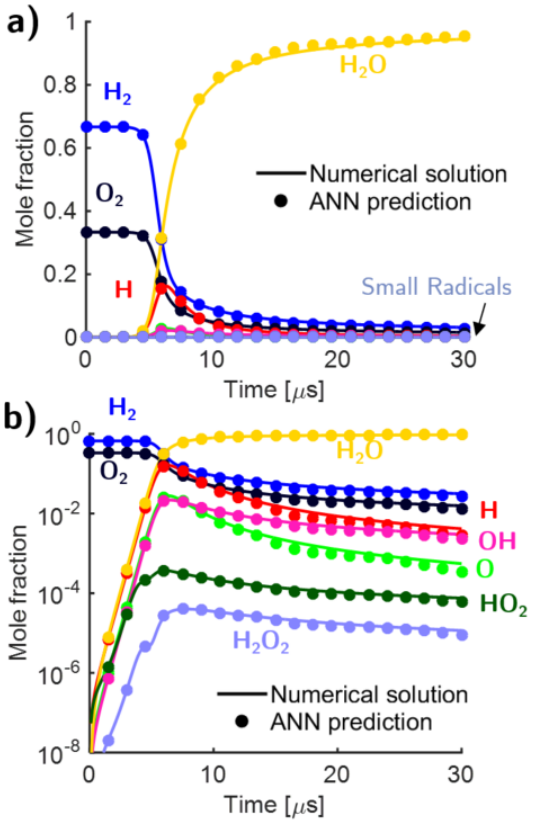

Fig. 6 ANN-predicted species time histories for a stoichiometric H_2/O_2 mixture plotted on (a) linear scale and (b) semi-logarithmic scale in the abscissa to highlight the lower-concentration radical species.

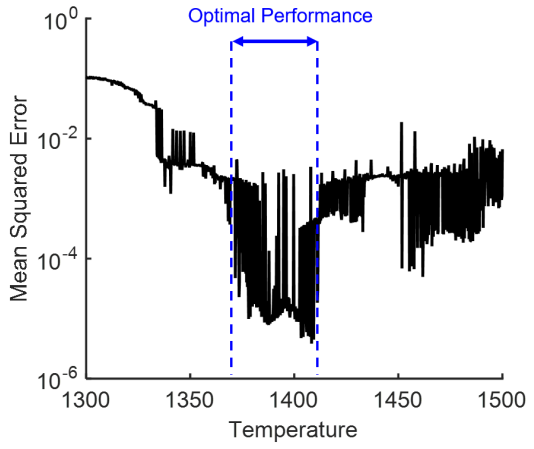

Fig. 7 Mean squared error of the ANN-compute species time-history of a stoichiometric H_2/O_2 mixture plotted on a semi-logarithmic scale in the abscissa.

The case study shown in Fig. 6 was repeated for 1000 random temperature initializations within the training range of 1300-1500 K. Mean squared errors were calculate for each initialization and is shown in Fig. 7 on a semi-logarithmic plot as a function of temperature. Most notably, the ANN performed poorly at lower temperatures while at higher temperatures (notably between 1375 and 1410 K) the ANN approximates the solution of the ODE solver with a high degree of accuracy. These results show that the ANN is certainly capable of producing accurate species time-histories over a certain range of conditions, but additional work needs to be done to reduce the ANN's sensitivity to minor errors at early times as well as to generalize the ANN's capabilities to a larger range of thermodynamic conditions. Nonetheless, the ANN computed species time-histories on average 35 times faster than the numerical solver, demonstrating the ANN's potential for reducing the computational cost of time-advancing chemical systems while maintaining a high level of accuracy in practical CFD simulations.

5 Conclusion

The high computational expense required to combine chemical kinetics and CFD currently impedes design optimization of combustion systems. Methods such as mechanism reduction and lookup tables do not scale well with the growing complexity of CFD problems and provide only a temporary solution to a growing challenge. ANNs have been proposed as an promising alternative to current methods, however, they have only been applied to small mechanisms and there remains open questions regarding how these networks scale with increasing mechanism complexity. This work puts forth a method that uses an ANN for each species in the mechanism as an effective approach that scales well to larger problems. Each ANN presented in this work was trained using only the H_2/O_2 sub-mechanism of GRI 3.0 and a limited temperature range of 1300-1500K, pressure 1.5 atm, and with stoichiometric mixtures of H_2 and O_2.

The ANN used in this work relied on training individual neural network for each species. By limiting the network to focus on one species at a time, strong agreement between the model and training data can be achieved. When compared against numerical solvers, the ANN produced results with mean squared error of less than 0.01 between $T = 1325 - 1500$ K while taking on average 35 times less time to compute the same results.

Future work will focus on improving the training efficiency for each species and generalization of this method to a wider temperature range. Additionally, this method could be extended to approximate the entire 53 species GRI 3.0 mechanism by training networks for each added species. Lastly, other deep learning methods that better suited for time-series problems such as recursive neural networks will be investigated as potential methods for improving the ANN performance. More advanced methods should focus on reducing the system sensitivity to the compounding of small errors in the time series integration.

References

[1] J. A. Blasco, N. Fueyo, C. Dopazo, and J. Ballester, "Modelling the temporal evolution of a reduced combustion chemical system with an artificial neural network," *Combustion and Flame*, vol. 113, no. 1-2, pp. 38–52, 1998.

[2] N. Shenvi, J. M. Geremia, and H. Rabitz, "Efficient chemical kinetic modeling through neural network maps.," *The Journal of chemical physics*, vol. 120, no. 21, pp. 9942–9951, 2004.

[3] R. Laubscher, *Utilization of artificial neural networks to resolve chemical kinetics in turbulent fine structures of an advanced CFD combustion model.* PhD thesis, Stellenbosch University, 2017.

[4] G. Smith, D. Golden, M. Frenklach, N. Moriarty, B. Eiteneer, M. Goldenberg, C. Bowman, R. Hanson, S. Song, W. Gardiner Jr, and V. Lissianski, "The GRI 3.0 chemical kinetic mechanism," 1999.

[5] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 9, pp. 249–256, 2010.

[6] E. Polak and G. Ribiere, "Note sur la convergence de méthodes de directions conjuguées," *Revue française d'informatique et de recherche opérationnelle*, vol. 3, no. 16, pp. 35–43, 1969.

[7] A. Atakulreka and D. Sutivong, "Avoiding Local Minima in Feedforward Neural Networks by Simultaneous Learning," *Lecture Notes in Artificial Intelligence*, vol. 4830, pp. 100–109, 2007.

Contributions

W.Y. Peng and N.H. Pinkowski both equally contributed to all parts of this project.