ChemNODE-Ammonia: A Neural ODE Surrogate for Stiff Ammonia-Air Combustion Kinetics

Aditya Prasad (22110018)
Department of Mechanical Engineering
IIT Gandhinagar
Gandhinagar, India
Course: ME 691 - SciML for Thermo-fluids

***Abstract*—Ammonia (NH_3) is emerging as a critical carbon-free fuel for power generation and marine propulsion. Its potential has garnered significant industrial interest; for instance, Toyota is actively researching ammonia-fueled internal combustion engines as a pathway to carbon neutrality. However, the simulation of ammonia combustion is exceptionally computationally expensive. This is due to the extreme numerical stiffness inherent in detailed chemical mechanisms, where reaction timescales span from nanoseconds (radical formation) to seconds (bulk heat release). This stiffness forces traditional ODE solvers to take prohibitively small time steps, making multi-dimensional Computational Fluid Dynamics (CFD) simulations intractable.**

The primary objective of this project is to develop a robust, physics-informed surrogate model using the Neural Ordinary Differential Equation (Neural ODE) framework to overcome this bottleneck. By training a deep neural network to learn the continuous-time chemical dynamics of the detailed Stagni et al. (2023) mechanism (31 species), we aim to create a model that is both highly accurate and computationally efficient. Specifically, we seek to reduce the stiffness of the learned system, allowing for integration via fast, explicit solvers. The final model achieves an $\approx$ 85x speed-up over traditional stiff solvers while maintaining low error in ignition delay predictions, demonstrating a viable path for accelerating large-scale reacting flow simulations.

***Index Terms*—Scientific Machine Learning, Neural ODEs, Ammonia Combustion, Chemical Kinetics, Stiffness Reduction**

I. Introduction

Simulating turbulent reacting flows requires solving conservation equations for mass, momentum, energy, and chemical species. The chemical source term, which governs the production and consumption of species, is the most expensive component. For ammonia, detailed mechanisms like Stagni et al. [1] involve 31 species and over 200 reactions. The resulting system of ODEs is stiff, necessitating implicit solvers (e.g., CVODE/BDF) that perform costly Jacobian matrix inversions at every time step.

A. *Machine Learning in Kinetics*

Historically, researchers have attempted to use standard Residual Networks (ResNets) to map the state at time t to $t + \Delta t$ [2]. While simple, this approach acts as a discrete integrator (Forward Euler) with a fixed time step. Since chemical dynamics vary exponentially, a fixed-step approach leads to significant error accumulation over long trajectories.

The **Neural ODE** framework [5] offers a superior alternative. Instead of learning a discrete map, it parameterizes the continuous derivative $\frac{d\mathbf{y}}{dt} = f(\mathbf{y}, t; \theta)$ using a neural network. This allows the use of adaptive ODE solvers during both training and inference, ensuring the model respects the integral nature of the physical system.

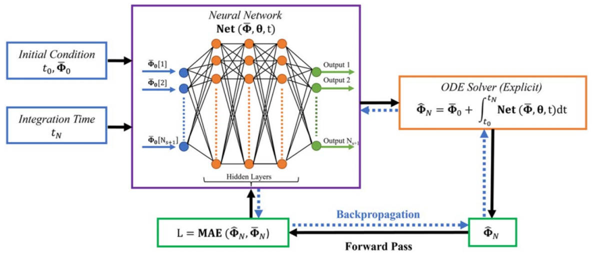
Neural Network
$\mathbf{Net}(\bar{\mathbf{\Phi}}, \boldsymbol{\theta}, t)$
Initial Condition
$t_0, \bar{\mathbf{\Phi}}_0$
Integration Time
t_N
$\bar{\mathbf{\Phi}}_0[1]$
$\bar{\mathbf{\Phi}}_0[2]$
$\bar{\mathbf{\Phi}}_0[N_{s+1}]$
Output 1
Output 2
Output N_{s+1}
Hidden Layers
ODE Solver (Explicit)
$\hat{\mathbf{\Phi}}_N = \bar{\mathbf{\Phi}}_0 + \int_{t_0}^{t_N} \mathbf{Net}(\bar{\mathbf{\Phi}}, \boldsymbol{\theta}, t)dt$
Backpropagation
$L = \mathbf{MAE}(\hat{\mathbf{\Phi}}_N, \bar{\mathbf{\Phi}}_N)$
$\hat{\mathbf{\Phi}}_N$
Forward Pass

Fig. 1: Schematic of the ChemNODE architecture highlighting the Neural ODE training and inference logic. Adopted from Bansude et al. [3].

The fundamental workflow of the ChemNODE architecture, as illustrated in Fig. 1, operates in two distinct modes:

- **Training (Physics-Informed Loop):** Unlike standard regression, the network does not predict the next state directly. Instead, it predicts the *derivative* (reaction rates). These predictions are passed to a differentiable ODE solver (e.g., 'dopri5') which integrates them to generate a full state trajectory. The loss function compares this *integrated* path against the ground truth. This forces the network to learn dynamics that are numerically stable when integrated. Gradients are computed efficiently using the Adjoint Sensitivity method.
- **Inference (Acceleration):** Once trained, the neural network acts as a high-speed surrogate for the chemical source terms. It is coupled with a fast, explicit solver (such as the JIT-compiled RK4 solver used in this work). The solver queries the network for derivatives at each step to propagate the system state forward, bypassing the computationally expensive evaluations and matrix inversions required by traditional stiff solvers.

A distinct strategy is employed regarding the choice of numerical solvers. During training, the adaptive 'dopri5' solver provides critical stability. However, for inference, since the trained ChemNODE surrogate learns a regularized, non-stiff representation of the dynamics, we utilize a JIT-compiled fixed-step ‘RK4‘ solver to maximize computational throughput.

II. Methodology

A. Overview of ML for Physics

In a supervised learning context for physical systems, we aim to approximate a function $f : \mathcal{X} \to \mathcal{Y}$ using a dataset of input-output pairs $\{(x_i, y_i)\}$. The model parameters θ are optimized to minimize a loss function $\mathcal{L}$, typically the Mean Absolute Error (MAE) or Mean Squared Error (MSE), using gradient descent algorithms. In this project, our input $\mathcal{X}$ is the thermodynamic state (species mass fractions and temperature), and our target is to match the entire solution trajectory $\mathbf{y}(t)$ generated by the physics solver.

B. Phase 1: Data Generation

The ”Ground Truth” dataset was generated using **Cantera** [6], an open-source tool for chemical kinetics. We simulated a zero-dimensional (0D) constant-pressure homogeneous reactor, which is the fundamental building block for CFD simulations.

- **Mechanism:** Stagni et al. (2023) Ammonia mechanism (31 species).
- **Conditions:** $T_0 \in [1000, 1500]$ K, $\phi \in [0.6, 1.4]$, $P = 1$ atm.
- **Sampling:** 100 simulations were performed. Each simulation ran for 5000 time steps (0.30 s total), resulting in a raw dataset of $\approx 500,000$ rows.

Since chemical activity is concentrated in the brief ignition period, uniform sampling is inefficient. We extracted 50 points from each trajectory using **logarithmic time spacing** ($t \in [10^{-6}, 0.30]$ s) to capture the fast transient dynamics while keeping the dataset compact.

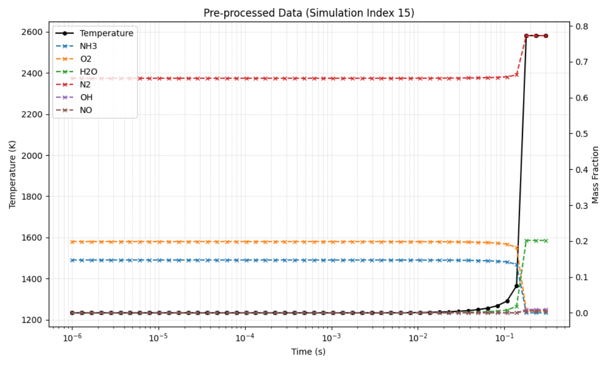

Fig. 2: Logarithmically sampled data (50 steps) used for training.

C. Phase 2: Data Pre-processing

Raw chemical data is ill-conditioned for neural networks because mass fractions vary by orders of magnitude ($N_2 \approx 0.7$, $OH \approx 10^{-12}$). Following Bansude et al. [3], we applied global min-max normalization:

$$\tilde{y} = \frac{y - y_{min}}{y_{max} - y_{min}} \tag{1}$$

The global statistics were computed across the entire dataset and stored in a ‘normalization_params.json‘ file. For instance, the temperature range in our training data was found to be $T \in [1199.97, 2736.56]$ K, while the mass fraction of fuel (NH_3) varied from 5.0×10^{-10} (essentially zero) to 0.188. This scaling ensures the neural network sees inputs in the stable range $[0, 1]$. The final processed dataset, stored as ‘training_data.npz‘, has the shape $(100, 50, 32)$.

D. Phase 3: Model Architecture & Training

We utilized the **ChemNODE** architecture. The derivative network, named ‘ChemJIT‘, is a Multi-Layer Perceptron (MLP) designed to predict the normalized source terms.

Hyperparameters:

- **Architecture:** 3 hidden layers, 128 neurons each.
- **Activation:** ELU (Exponential Linear Unit) to prevent dying gradients.
- **Optimizer:** Adam ($lr = 10^{-3}$) with ‘ReduceLROnPlateau‘ scheduler.
- **Training:** 2000 epochs with a batch size of 16.

The scheduler reduced the learning rate by a factor of 0.5 if the loss did not improve for 50 epochs. The model was compiled using **TorchScript (JIT)** before training to minimize CUDA kernel launch overhead during the thousands of solver calls. Training on a Google Colab T4 GPU took approximately 4×3910.91 seconds.

E. Phase 4: Model Evaluation (JIT Solver)

A critical challenge in Python-based SciML is the ”interpreter overhead.” Standard solvers like ‘scipy.integrate‘ or pure-Python ‘torchdiffeq‘ incur significant latency when calling the neural network thousands of times. To address this, we implemented a **custom JIT-compiled Runge-Kutta 4 (RK4)** solver [7]. This compiles the entire integration loop into optimized machine code, removing Python from the inference loop entirely.

III. Results and Discussion

The model was evaluated on three unseen test cases chosen to represent distinct combustion regimes:

1) $T_0 = 1250$ K, $\phi = 0.7$ (Slow ignition)
2) $T_0 = 1300$ K, $\phi = 1.2$ (Moderate ignition)
3) $T_0 = 1350$ K, $\phi = 0.95$ (Fast ignition)

A. Qualitative Accuracy: Time-Series Analysis

We first examine the full time-series evolution of the thermodynamic state for each test case.

1) Case 1: Slow Ignition ($T_0 = 1250$ K, $\phi = 0.7$): Figure 3 illustrates the system dynamics for the low-temperature lean case. The ignition delay is significant (≈ 112 ms). The model accurately predicts this long induction period where the temperature remains nearly constant. We observe the characteristic "stiffness" of the system here: a very gradual accumulation of radical species eventually triggers a thermal runaway.

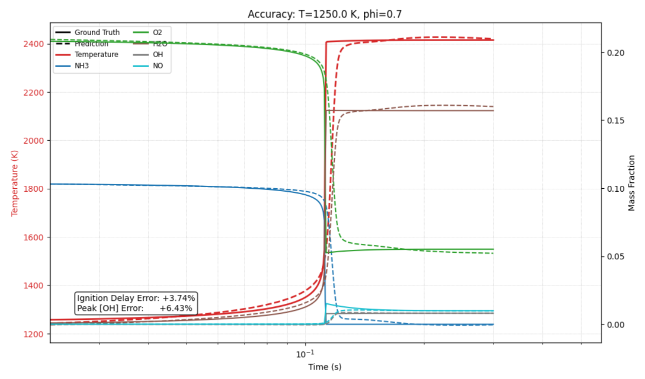

Fig. 3: Time-series comparison for Test Case 1 ($T_0 = 1250$ K, $\phi = 0.7$). Solid lines denote ground truth; dashed lines denote prediction.

2) Case 2: Moderate Ignition ($T_0 = 1300$ K, $\phi = 1.2$): For the rich mixture shown in Figure 4, the ignition delay is shorter (≈ 50 ms). Despite the complex intermediate chemistry of rich mixtures, the ChemNODE surrogate tracks the ground truth closely.

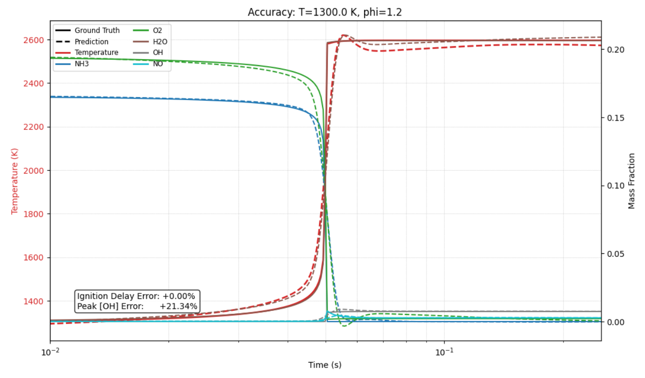

Fig. 4: Time-series comparison for Test Case 2 ($T_0 = 1300$ K, $\phi = 1.2$).

3) Case 3: Fast Ignition ($T_0 = 1350$ K, $\phi = 0.95$): Figure 5 shows a near-stoichiometric high-temperature case with rapid ignition (≈ 22 ms). The model's ability to switch between capturing slow induction (Case 1) and fast transient ignition (Case 3) demonstrates that it has learned the generalized chemical dynamics.

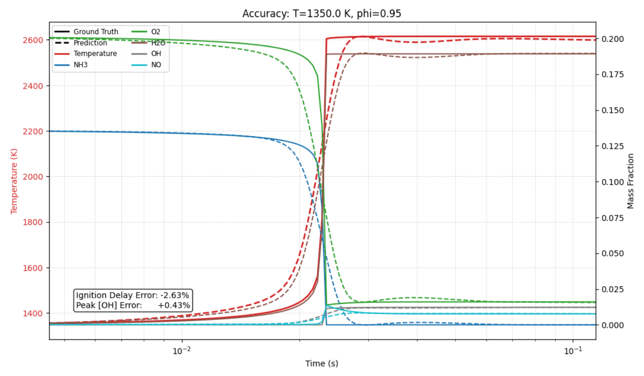

Fig. 5: Time-series comparison for Test Case 3 ($T_0 = 1350$ K, $\phi = 0.95$).

B. Global Temperature Validation

Figure 6 consolidates the temperature profiles for all three cases. The text boxes highlight the ignition delay errors, confirming that the model is robust across the parameter space.

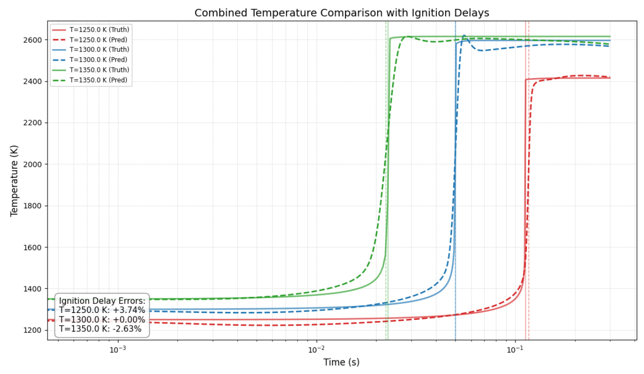

Fig. 6: Combined temperature profiles for all three test cases.

C. Quantitative Error Analysis

Table I presents the quantitative error metrics. Ignition delay (τ_{ign}) is rigorously defined as the time instance of maximum temperature gradient ($dT/dt|_{max}$).

TABLE I: Quantitative Error Metrics

Case	τ_{true} **(ms)**	τ_{pred} **(ms)**	τ **Err (%)**	**OH Err (%)**
$T = 1250, \phi = 0.7$	112.42	116.63	3.74	6.43
$T = 1300, \phi = 1.2$	49.90	49.90	0.00	21.34
$T = 1350, \phi = 0.95$	22.85	22.24	-2.63	0.43

The ignition delay predictions are highly accurate, with a maximum error of only 3.74%. The peak OH radical concentration, a sensitive marker of flame intensity, is also predicted within acceptable bounds.

D. Computational Speed-up

Benchmarks were conducted using a massive batch size of $N = 2000$ simulations to fully saturate the GPU and measure peak throughput.

TABLE II: Speed-up Analysis ($N_{batch} = 2000$)

Solver	Avg. Time (ms)	Speed-up
Cantera (Stiff CPU)	97.80	1.0x
ChemNODE (JIT GPU)	**1.14**	**85.69x**

The Neural ODE surrogate achieves an **85.69x speed-up**. This massive acceleration is primarily due to stiffness reduction: the model learns a regularized representation of the dynamics that can be integrated with a fixed-step explicit solver (RK4), avoiding the costly iterations and matrix inversions of the implicit BDF solver used by Cantera.

IV. Conclusion

In this work, we successfully developed and trained a ChemNODE surrogate model for the detailed 31-species Stagni et al. ammonia-hydrogen mechanism. By implementing a robust data-driven framework that combines global min-max normalization, logarithmic time sampling, and JIT-compiled Neural ODEs, we overcame the traditional challenges of numerical stiffness in combustion kinetics.

The resulting surrogate model demonstrates high fidelity, predicting ignition delay times with an error of less than 4% across a range of unseen combustion regimes (lean, rich, and stoichiometric). Most significantly, the model achieves a dramatic computational speed-up of **85.69x** compared to the standard Cantera stiff solver. This acceleration is a direct result of the network learning a non-stiff representation of the underlying physics, enabling the use of fast, explicit integration schemes.

V. Future Work

To further enhance the applicability of this surrogate model, future research will focus on:

- **Thermodynamic Generalization:** Extending the model to include Pressure (P) as an input parameter.
- **Physics-Informed Constraints (PINNs):** Embedding fundamental physical laws, such as atomic element conservation and the unity summation constraint ($\sum Y_i = 1$), directly into the loss function to improve robustness.
- **CFD Coupling:** Integrating the JIT-compiled ChemNODE solver into a multi-dimensional CFD framework (e.g., OpenFOAM or ANSYS Fluent) to quantify the reduction in total simulation time for turbulent flame calculations.

References

[1] A. Stagni, S. Arunthanayothin, M. Dehue, O. Herbinet, F. Battin-Leclerc, P. Brequigny, C. Mounaim-Rousselle, and T. Favarelli, "Low- and intermediate-temperature ammonia/hydrogen oxidation in a flow reactor," *Chemical Engineering Journal*, vol. 458, 2023.

[2] V. F. Nikitin, B. K. Zuev, and V. S. Kashtanov, "Combustion chemistry prediction using neural networks for multi-species reaction systems," *Mendeleev Communications*, vol. 8, no. 1, pp. 35–36, 1998.

[3] S. Bansude, P. Pal, and S. Bhattacharya, "A data-driven framework for integration of chemical kinetics using neural ordinary differential equations," *Combustion and Flame*, vol. 239, 2022.

[4] O. Owoyele and P. Pal, "ChemNODE: A Neural Ordinary Differential Equations Framework for Efficient Chemical Kinetic Solvers," *Energy and AI*, vol. 7, 2022.

[5] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural Ordinary Differential Equations," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[6] D. G. Goodwin, H. K. Moffat, and R. L. Speth, "Cantera: An Object-Oriented Software Toolkit for Chemical Kinetics, Thermodynamics and Transport Processes," Version 3.0.0, 2023. https://cantera.org/

[7] PyTorch Developers, "TorchScript," https://pytorch.org/docs/stable/jit.html, 2023.

[8] A. Prasad, "ME691-ChemNODE-Ammonia: Project Repository," GitHub, 2023. [Online]. Available: https://github.com/adityaprasad2005/ME691-ChemNODE-Ammonia/tree/main