Efficient and Accurate Time-integration of Combustion Chemical Kinetics using Artificial Neural Networks

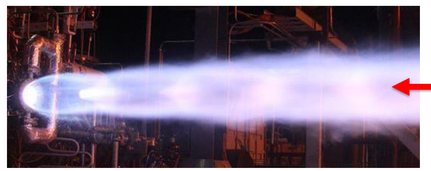
Wen Yu Peng (wypeng@stanford.edu) & Nicolas H. Pinkowski (npinkows@stanford.edu)

Stanford | ENGINEERING
MECHANICAL ENGINEERING

Background and Motivation

Goal: Reduce computation time for time-integrating chemical kinetics mechanisms

- Chemical kinetics: central to design of practical combustion systems such as internal combustion and rocket engines
- Currently: computational fluid dynamics (CFD) models plus reaction mechanism with expensive ODE solvers are used in the design process

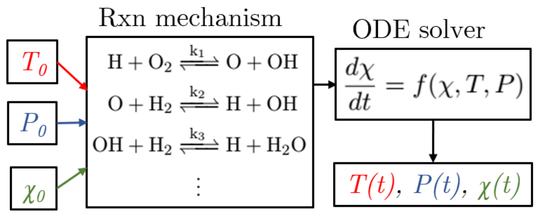


Problem:
Chemistry difficult and slow to model together with CFD

***If** we can increase computational efficiency*
***then** we can design more efficient engines*

Problem Description

Typical ODE-based method for implementing reactive CFD

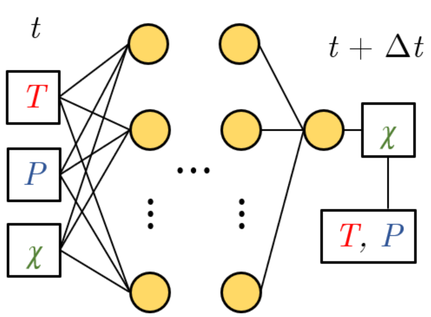


T = Temp [K]
P = Pressure [atm]
χ = Mole fraction

H_2 oxidation:
8 chemical species
20 reactions

- Accurate outputs ✓
- Major problem: stiff ODEs require expensive implicit integration
- Intractable with modern mechanisms (e.g. gasoline surrogate → 1550 species + 6000 reactions)

Proposed neural network-based method

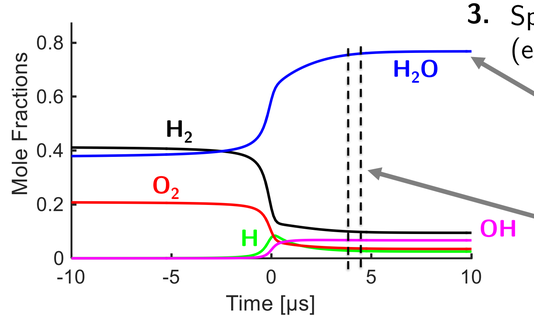


- Replace time-integration of mechanism with artificial neural network (ANN)
- Computationally cheap ✓
- Approximate outputs good enough for CFD ✓
- Tradeoff: High up-front cost for generating training data

Methods

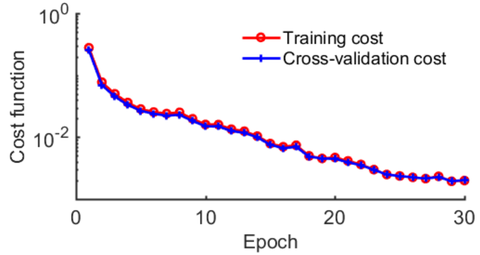
Training data generation:

1. Specify reaction mechanism (H_2 oxidation in GRI 3.0[2])
2. Bounded Monte-Carlo method to generate initial conditions
3. Specify reactor constraint (eg. const. UV, HP, TP)
4. Use ODE solver[3] to advance system to equilibrium
5. Each $t + \Delta t$ is one training and target example



- Training set size: 165,000 training examples, 300 initiations
- Breakdown: 90% train, 10% cross validation, random permutation

Training & cost function:



- Cost: $J = \frac{1}{2mK} \sum_{i=1}^{m} \sum_{j=1}^{K} \left[\log \left(\frac{y_j^{(i)}}{\hat{y}_j^{(i)}} \right) \right]^2$
- Trained using mini-batch P-R conjugate gradient descent[4] with backpropagation
- Model complexity insufficient to require regularization

ANN Architecture Selection

- Tested various network parameters: number of hidden layers, number of neurons per layer, cost function, and activation types
- Defined a **figure of merit** = $||x_{sim} - x_{ANN}||$ to quantify model performance for each architecture

Figure of merit for network architectures

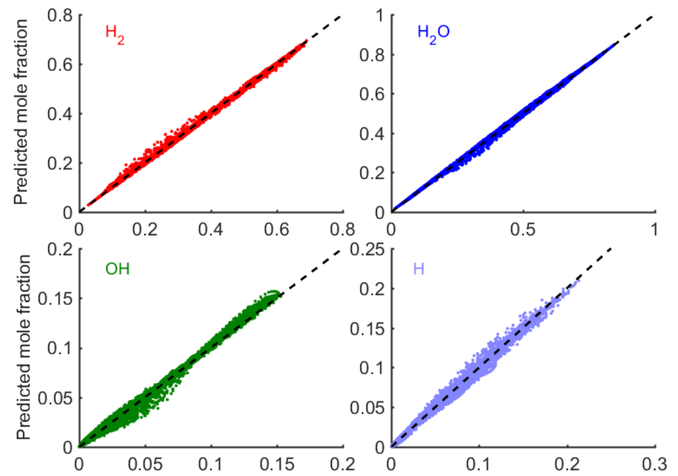
Too complex: local minima

# of Layers	Number of Hidden Units per Layer: 6	10	20	200	500	700
2	37.3	28.8	16.0	22.6	27.8	43.2
4	18.3	7.4	15.9	17.2	3.0	44.5
6	28.2	2.3	27.7			
8	5.1	3.7	2.4	*computationally expensive*		

Optimal network architecture determined to be 6 layers, 10 neurons per layer, sigmoidal activation, and softmax output

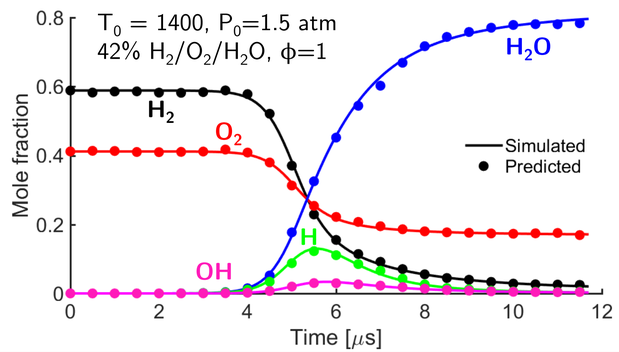
Results

Sample comparisons between ANN and training mole fractions:



Case study on model performance compared to ODE solver:

ANN 35x faster than ODE solver



Discussion & Future Work

Discussion

- ANN prediction agrees with ODE solver for low Temp.
- Limited generalizability to unseen conditions

Future Work

- Constrained ANN parameter optimization (atom balance)
- Evaluate the computational efficiency vs. ODE solver

[1] J.A. Blasco, N. Fueyo, C. Dopazo, J. Ballester, Combust. Flame 113 (1–2) (1998) 38–52.
[2] G.P. Smith, D.M. Golden, M. Frenklach, N.W. Moriarty, et al. , available at <http://combustion.berkeley.edu/gri-mech/version30/text30.html>.
[3] D.G. Goodwin, H.K. Moffat, R.L. Speth, (2016).
[4] E. Polak, G. Ribiere, Rev. Française D'informatique Rech. Opérationnelle 3 (16) (1969) 35–43.