DeepFlame

Release 0.1

DeepModeling

Nov 11, 2022

QUICK START

DeepFlame is a computational fluid dynamics suite for single or multiphase, laminar or turbulent reacting flows at all speeds with machine learning capabilities. It aims to provide an open-source platform bringing together the individual strengths of OpenFOAM, Cantera and Torch libraries for deep learning assisted reacting flow simulations. It is also has the scope to incorporate next-generation heterogenous supercomputing and AI acceleration infrustructures such as GPU and FPGAs.

Note: This project is under active development.

CHAPTER

ONE

INSTALLATION

1.1 Prerequisites

The installation of DeepFlame is simple and requires **OpenFOAM-7**, **LibCantera**, and **PyTorch**.

Note: If Ubuntu is used as the subsystem, please use Ubuntu:20.04 instead of the latest version. OpenFOAM-7 accompanied by ParaView 5.6.0 is not available for Ubuntu-latest.

First install OpenFOAM-7 if it is not already installed.

```
sudo sh -c "wget -O - https://dl.openfoam.org/gpg.key | apt-key add -"
sudo add-apt-repository http://dl.openfoam.org/ubuntu
sudo apt-get update
sudo apt-get -y install openfoam7
```

OpenFOAM 7 and ParaView 5.6.0 will be installed in the `/opt` directory.

LibCantera and **PyTorch** are installed via conda. DNN version aims to support computation on CUDA. If you have compatible platform, run the following command to install DeepFlame.

```
conda create -n df-pytorch python=3.8
conda activate df-pytorch
conda install -c cantera libcantera-devel
conda install pytorch torchvision torchaudio pytorch-cuda=11.6 -c pytorch -c nvidia
conda install pybind11 easydict
```

Note: Please go to PyTorch's official website to check your system compatability and choose the installation command line that is suitable for your platform.

Note: Check your `Miniconda3/envs/libcantera` directory and make sure the install was successful (lib/ include/ etc. exist).

1.2 Configure

You need to source your OpenFOAM-7. .. code-block:: bash

source $HOME/OpenFOAM/OpenFOAM-7/etc/bashrc

This depends on your own path for OpenFOAM bashrc.

1.3 Build and Install

DeepFlame with DNN capatibility is installed through: .. code-block:: bash

git clone https://github.com/deepmodeling/deepflame-dev.git cd deepflame-dev source ./bashrc . install.sh –use_pytorch

Note: You may come accross an error regarding shared library `libmkl_rt.so.2` when libcantera is installed through cantera channel. If so, go to your conda environment and check the existance of `libmkl_rt.so.2` and `libmkl_rt.so.1`, and then link `libmkl_rt.so.2` to `libmkl_rt.so.1`.

```
cd ~/miniconda3/envs/df-pytorch/lib
ln -s libmkl_rt.so.1 libmkl_rt.so.2
```

1.4 Other Options

DeepFlame also provides users with LibTorch integraotr and CVODE integrator.

If you choose to use LibTorch (C++ API for Torch), first create the conda env and install LibCantera:

```
conda create -n df-libtorch
conda activate df-libtorch
conda install -c cantera libcantera-devel
```

Then you can pass your own libtorch path to DeepFlame. .. code-block:: bash

. install.sh –libtorch_dir /path/to/libtorch/

Note: Some compiling issues may happen due to system compatability. Instead of using conda installed Cantera C++ lib and the downloaded Torch C++ lib, try to compile your own Cantera and Torch C++ libraries.

If your are using DeepFlame's CVODE solver without DNN model, just install LibCantera via conda.

```
conda create -n df-notorch
conda activate df-notorch
conda install -c cantera libcantera-devel
```

If the conda env `df-notorch` is activated, install DeepFlame by running:

```
. install.sh
```

If `df-notorch` not activated (or you have a self-complied libcantera), specify the path to your libcantera:

```
. install.sh --libcantera_dir /your/path/to/libcantera/
```

CHAPTER

TWO

TWO EXAMPLES

2.1 DeepFlame with DNN

If you choose to use PyTorch as the integratgor and use the compilation flag *–use_pytorch*, you can run examples stored in *$HOME/deepflame-dev/examples/.../pytorchIntegratgor*. To run an example, you first need to source your OpenFOAM:

```
source $HOME/OpenFOAM/OpenFOAM-7/etc/bashrc
```

Then, source your DeepFlame:

```
source $HOME/deepflame-dev/bashrc
```

Next, you can go to the directory of any example case that you want to run. For example:

```
cd $HOME/deepflame-dev/examples/zeroD_cubicReactor/H2/pytorchIntegratgor
```

This is an example for the zero-dimensional hydrogen combustion with PyTorch as the integrator. All files needed by DNN are stored in *pytorchDNN* folder, and the inference file is *inference.py*. Configurations regarding DNN are included in *constant/CanteraTorchProperties*.

The case is run by simply typing:

```
./Allrun
```

Note: Users can go to *constant/CanteraTorchProperties* and check if *torch* is switched on. Switch it *on* to run DNN cases, and switch *off* to run CVODE cases.

If you plot PyTorch's result together with CVODE's result, the graph is expected to look like:

2.2 DeepFlame without DNN

CVODE Integrator is the one without the application of Deep Neural Network (DNN). Follow the steps below to run an example of CVODE. Examples are stored in the directory: .. code-block:: bash

    $HOME/deepflame-dev/examples

To run these examples, first source your OpenFOAM, depending on your OpenFOAM path:

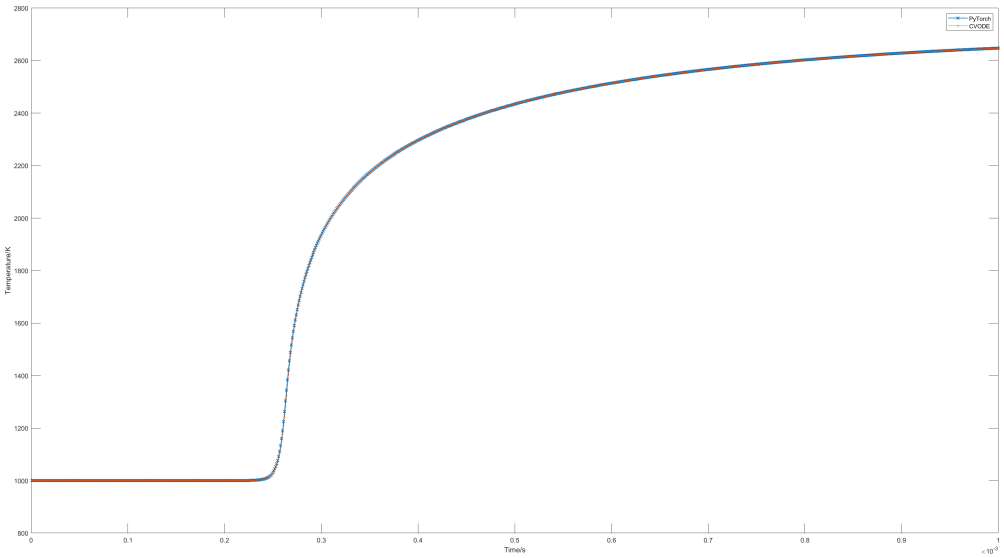


Fig. 1: Visualisation of 0D results from PyTorch and CVODE integrators

```
source $HOME/OpenFOAM/OpenFOAM-7/etc/bashrc
```

Then, source your DeepFlame:

```
source $HOME/deepflame-dev/bashrc
```

Next, you can go to the directory of any example case that you want to run. For example:

```
cd $HOME/deepflame-dev/examples/zeroD_cubicReactor/H2/cvodeIntegrator
```

This is an example for the zero-dimensional hydrogen combustion with CVODE integrator.

The case is run by simply typing:

```
./Allrun
```

The probe used for post processing is defined in `/system/probes`. In this case, the probe is located at the coordinates (0.0025 0.0025 0.0025) to measure temperature variation with time. If the case is successfully run, the result can be found in `/postProcessing/probes/0/T`, and it can be visualized by running:

```
gunplot
plot "/your/path/to/postProcessing/probes/0/T"
```

You will get a graph:

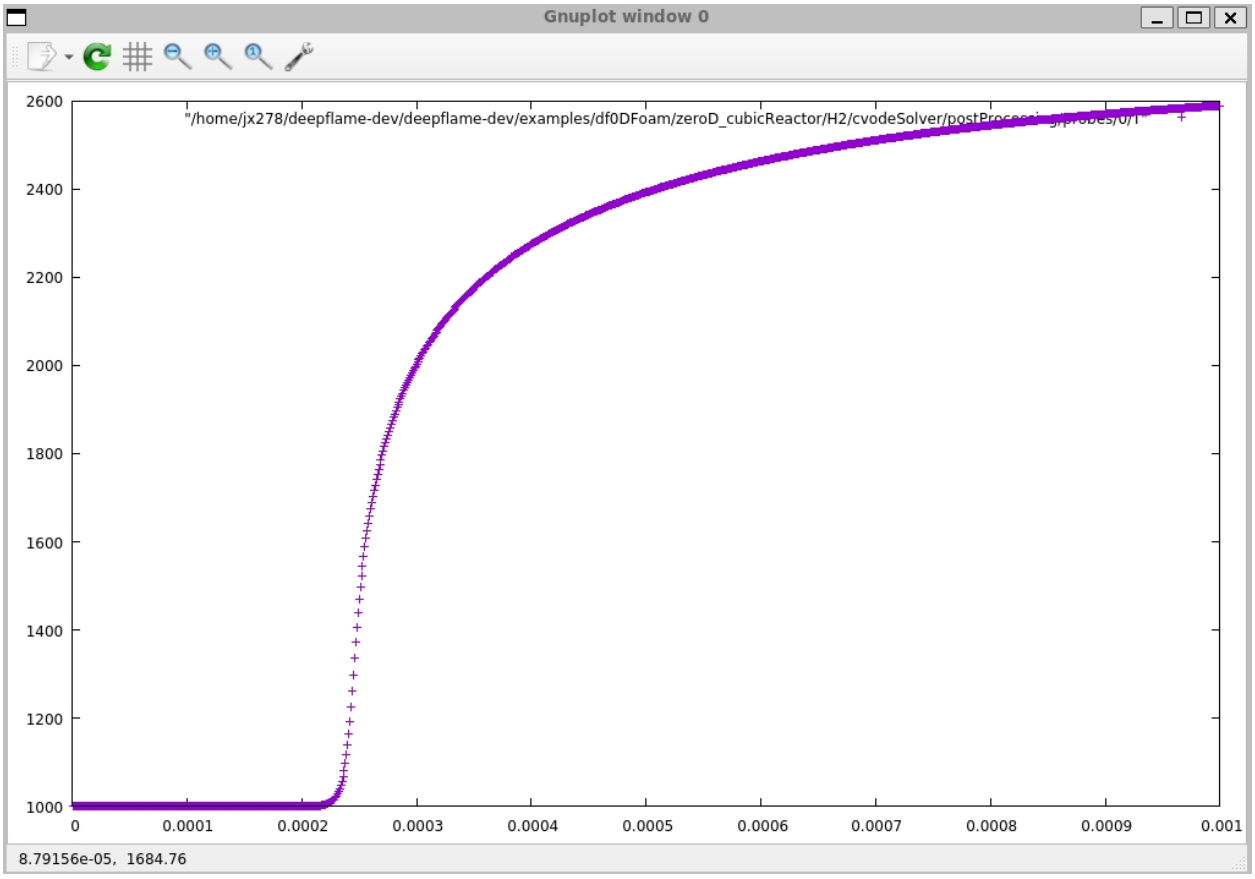


Fig. 2: Visualisation of the zero-dimensional hydrogen combustion result with CVODE integrator

CHAPTER

THREE

BRIEF INTRODUCTION TO INPUTS

The dictionary `CanteraTorchProperties` is the original dictionay of DeepFlame. It read in netowrk realted parameters and configurations. It typically looks like:

```
chemistry           on;
CanteraMechanismFile "ES80_H2-7-16.yaml";
transportModel "Mix";//"UnityLewis";
odeCoeffs
{
    //"relTol"   1e-15;
    //"absTol"   1e-24;
}
inertSpecie        "N2";

zeroDReactor
{
    constantProperty "pressure";
}

torch on;
GPU on;
torchModel1 "ESH2-sub1.pt";
torchModel2 "ESH2-sub2.pt";
torchModel3 "ESH2-sub3.pt";

torchParameters1
{
    Tact 700  ;
    Qdotact  3e7;
    coresPerGPU 4;
}
torchParameters2
{
    Tact 2000;
    Qdotact  3e7;
}
torchParameters3
{
    Tact 2000;
    Qdotact  7e8;
}
```

(continues on next page)

(continued from previous page)

```
loadbalancing
{
        active  false;
        //log   true;
}
```

In the above example, the meanings of the parameters are:

- `CanteraMechanismFile`: the name of the reaction mechanism file
- `odeCoeffs`: the ode torlerance. 1e-15 and 1e-24 are used for network training, so it should keep the same when comparing results with nd without DNN.
- `torch`: the switch used to control the on and off of DNN. If users are running CVODE, this needs to be switched off.
- `GPU`: the switch used to control whether GPU or CPU is used to carry out inference.
- `torchModel`: name of network.
- `torchParameters`: thresholds used to decide when to use network.
- `coresPerGPU`: number of CPU cores on one node.

CHAPTER

FOUR

DF0DFOAM

4.1 Zero-Dimensional ignition reactor

Problem Description

This case simulates the zero-dimensional autoignition under constant-pressure or constant-volume condition. This case confirm the validity of the implementation of chemical reaction source terms in DeepFlame.

Table 1: Operating Conditions in Brief

Mixture	Hydrogen-Air
Equivalence Ratio	1.0
Initial Gas Temperature	1400 K
Initial Gas Pressure	1 atm

Output

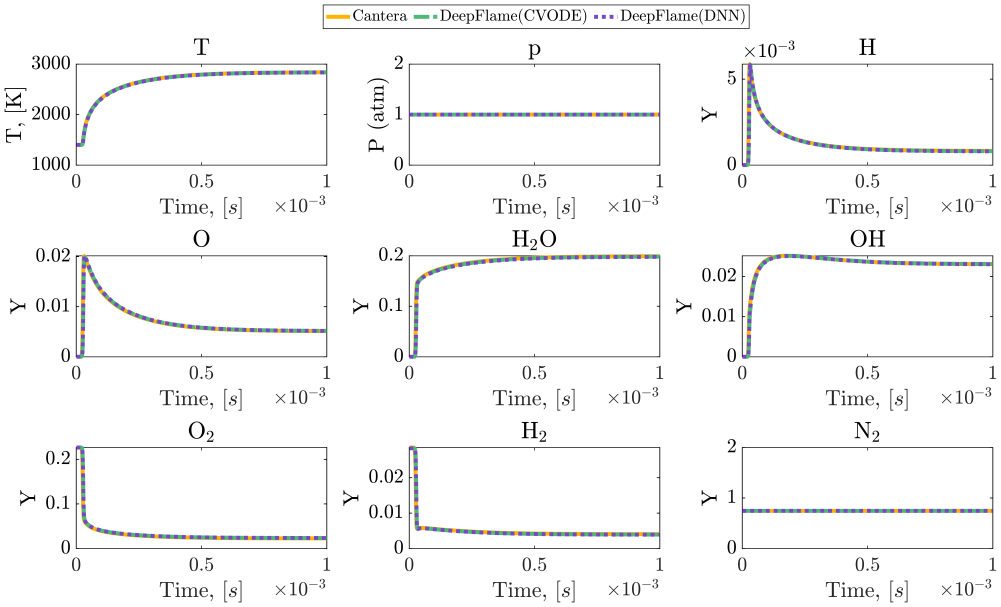


Fig. 1: Results of zero-dimensional constant-pressure autoignition

CHAPTER

FIVE

DFLOWMACHFOAM

5.1 One-Dimensional Planar Flame

Problem Description

The case simulates the steady-state 1D freely-propagating flame. The results are able to catch the flame thickness, laminar fame speed and the detailed 1D flame structure. This case demonstrate that the convection-diffusion-reaction algorithms implemented in our solver are stable and accurate.

Table 1: Operating Conditions in Brief

Computational Domain length	0.06 m
Mixture	Hydrogen-Air
Equivalence Ratio	1.0
Inlet Gas Temperature	300 K

Output

5.2 Two-Dimensional Jet Flame

Problem Description

This case simulates the evolution of a 2D non-premixed planar jet flame to validate the capability of our solver for multi-dimensional applications.

Table 2: Operating Conditions in Brief

Computational Domain size (x)	0.03 m * 0.05 m
Jet Composition	H2/N2= 1/3 (fuel jet), Air (co-flow)
Initial Velocity	5 m/s (fuel jet), 1 m/s (co-flow)
Initial Gas Temperature	1400 K (ignition region), 300 K (other area)

Output

The initial condition and the evolution of the jet flame are presented in this figure.

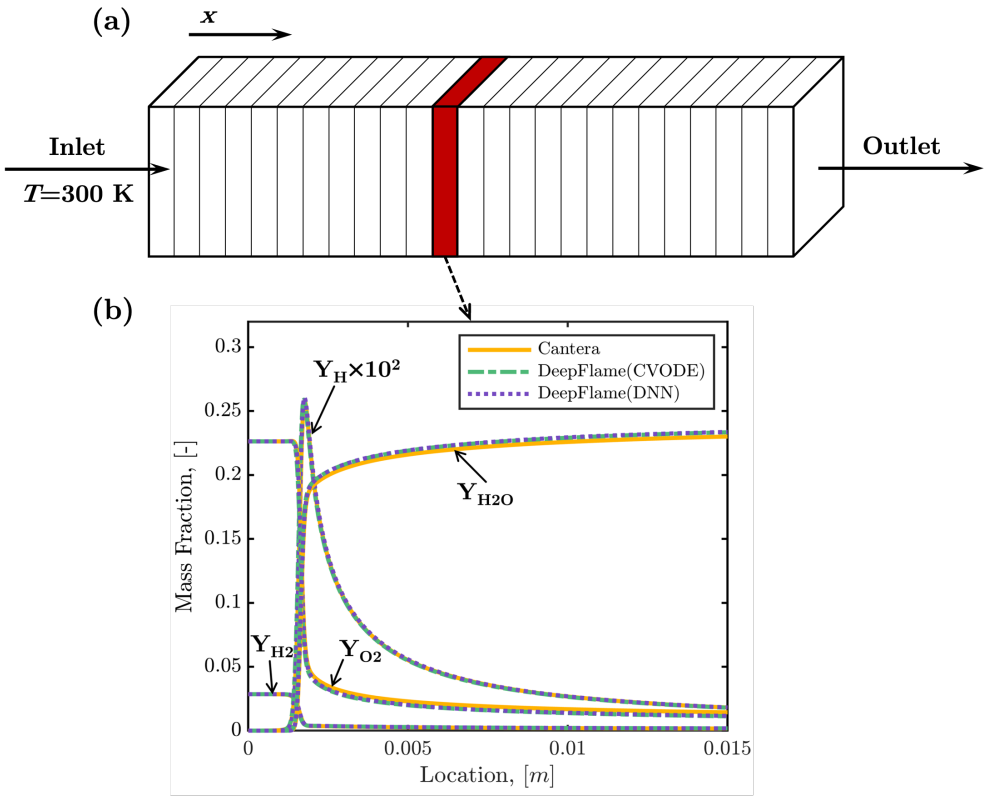


Fig. 1: Numerical setup of one-dimensional premixed flame and the detailed flame structure obtained by our solver

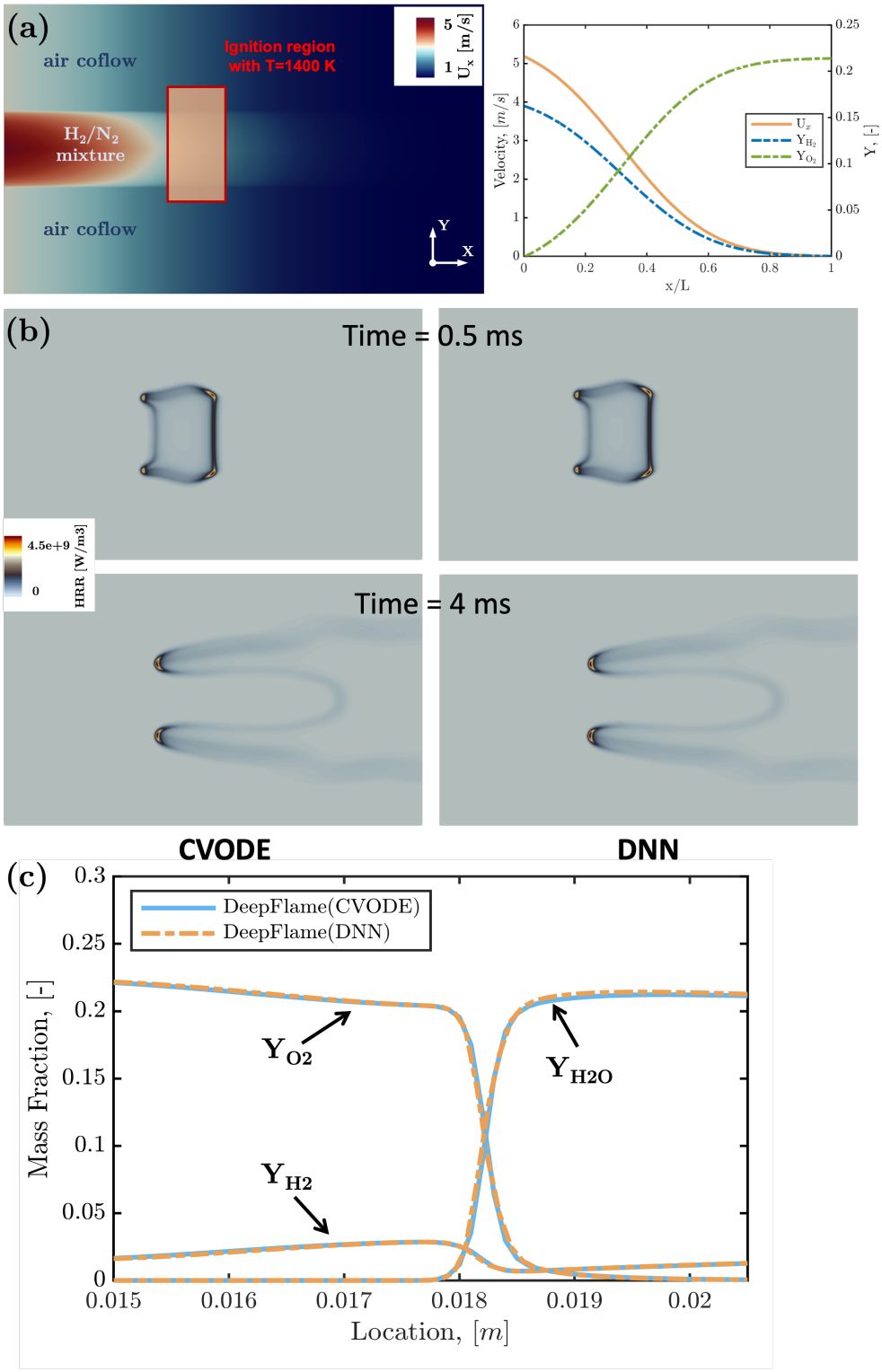


Fig. 2: Simulation results of the two-dimensional jet flame.

5.3 Three-Dimensional reactive Taylor-Green Vortex

3D reactive Taylor-Green Vortex (TGV) which is a newly established benchmark case for reacting flow DNS codes is simulated here to evaluate the computational performance of our solver.

The initial fields are set according to a benchmark case established by Abdelsamie et al. The figure below shows contours of vorticity magnitude and temperature as well as the x-direction profiles of species at initial time.

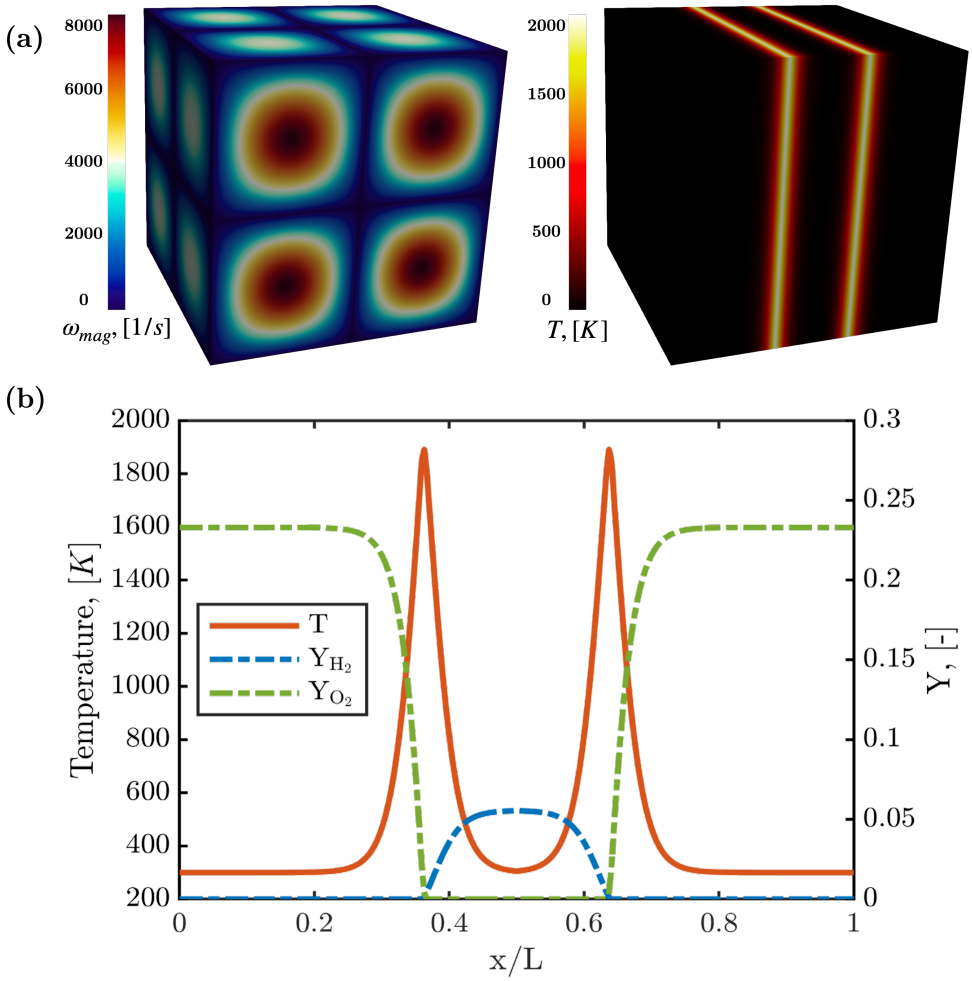


Fig. 3: Initial contours and profiles of vorticity magnitude, temperature, and species mass fraction for the reactive TGV

Output

The developed TGV are displayed in the figures below.

Reference

A.Abdelsamie, G.Lartigue, C.E.Frouzakis, D.Thevenin, The taylor-green vortex as a benchmark for high-fidelity combustion simulations using low-mach solvers, Computers & Fluids 223 (2021): 104935.

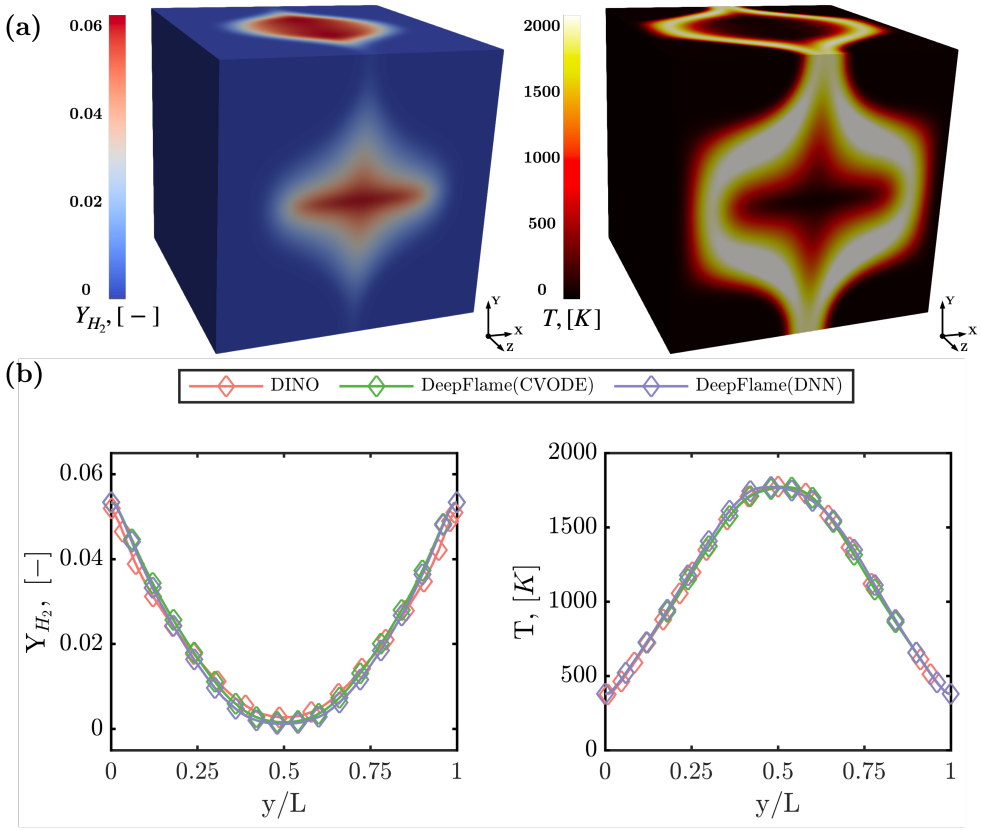


Fig. 4: Contours and profiles of temperature and species mass fraction at t = 0.5 ms

CHAPTER

SIX

DFHIGHSPEEDFOAM

6.1 One-Dimensional Reactive Shock Tube

Problem Description

The case simulates supersonic inlet flow hitting the wall and then reflected to ignite the premixed gas. The reactive wave will catch the reflected shock wave. This case can also verify the accuracy of our solver in capturing the interaction of convection and reaction.

Table 1: Operating Conditions in Brief

Chamber size (x)	0.12 m
Initial Gas Density	0.072 kg/m^3 (x<=0.06 m), 0.18075 kg/m^3 (x>0.06 m)
Initial Gas Pressure	7173 Pa (x<=0.06 m), 35594 Pa (x>0.06 m)
Initial Gas Velocity	0 m/s (x<=0.06 m), -487.34 m/s (x>0.06 m)
Ideal Gas Composition (mole fraction)	H2/O2/Ar = 2/1/7

Output

Reference

E S Oran, T R Young, J P Boris, A Cohen, Weak and strong ignition. i. Numerical simulations of shock tube experiments, Combustion and Flame 48 (1982) 135-148.

R J Kee, J F Grcar, M D Smooke, J A Miller, E Meeks, Premix: A fortran program for modeling steady laminar one-dimensional premixed flames, Sandia National Laboratories.

6.2 One-Dimensional H2/Air Detonation

Problem Description

Detonation propagation contains a complex interaction of the leading shock wave and auto-igniting reaction, showing the coupling of shock wave and chemical reaction. This case aims to validate the accuracy of this solver in capturing this process and the propagation speed.

Table 2: Operating Conditions in Brief

Chamber size (x)	0.5 m
Initial Gas Pressure	90 atm (hot spot), 1 atm (other area)
Initial Gas Temperature	2000 K (hot spot), 300 K (other area)
Ideal Gas Composition (mole fraction)	H2/O2/N2 = 2/1/3.76 (homogeneous stoichiometric mixture)

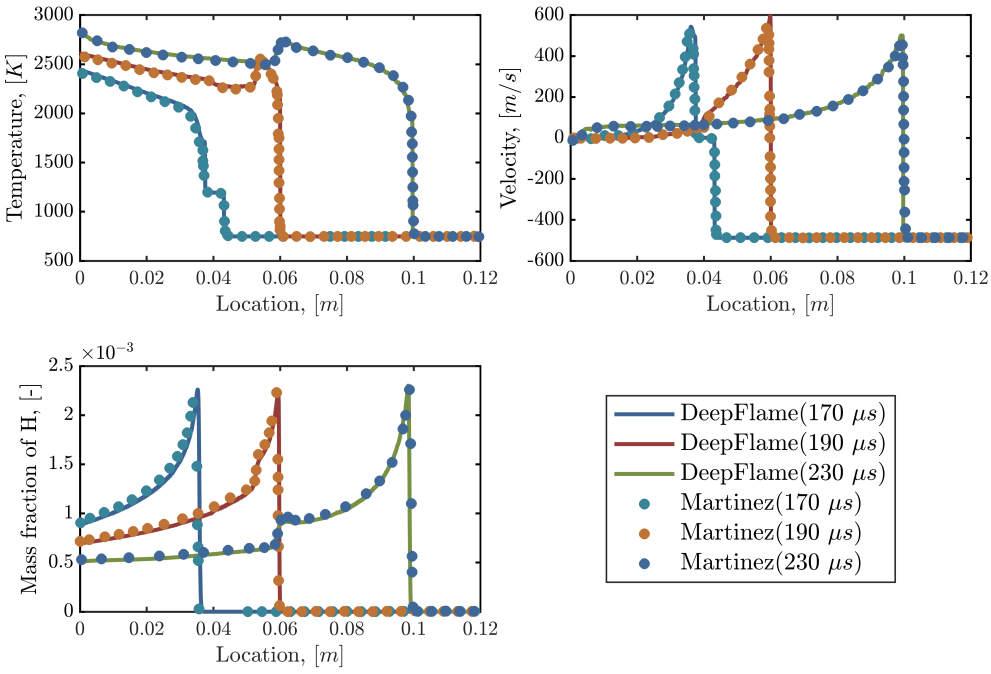


Fig. 1: Result of one-dimensional reactive shock tube

Output

Reference

J Li, Z Zhao, A Kazakov, F L Dryer, An updated comprehensive kinetic model of hydrogen combustion, International Journal of Chemical Kinetics 36 (2004) 566-575.

6.3 Two-Dimensional H2/Air Detonation

Problem Description

Detonation propagation contains a complex interaction of the leading shock wave and auto-igniting reaction, and two-dimensional detonation can further reveal the interaction of shear waves and shock waves.

Table 3: Operating Conditions in Brief

Chamber size (x)	0.2 m * 0.01 m
Initial Gas Pressure	100 atm (three hot spot), 1 atm (other area)
Initial Gas Temperature	2000 K (three hot spot), 300 K (other area)
Ideal Gas Composition (mole fraction)	H2/O2/N2 = 2/1/7 (homogeneous stoichiometric mixture)

Output

Triple points can be seen clearly in the picture below.

In the picture below, during the propagation of detonation wave, we can see that the size of cells gradually became stable.

Reference

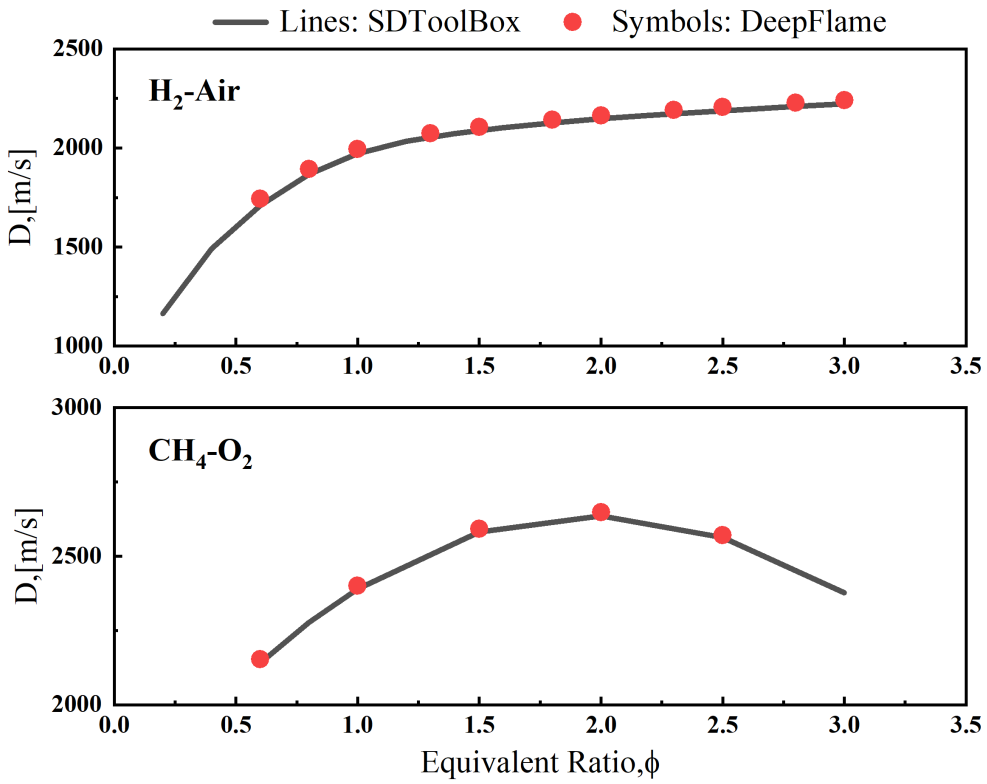


Fig. 2: Result of one-dimensional H2/air detonation

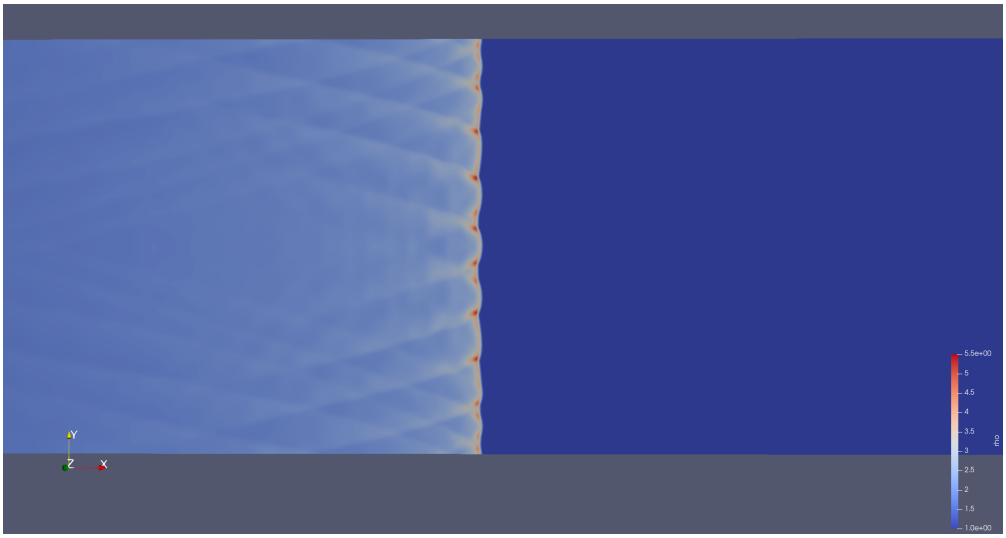


Fig. 3: Density field of two-dimensional H2 detonation

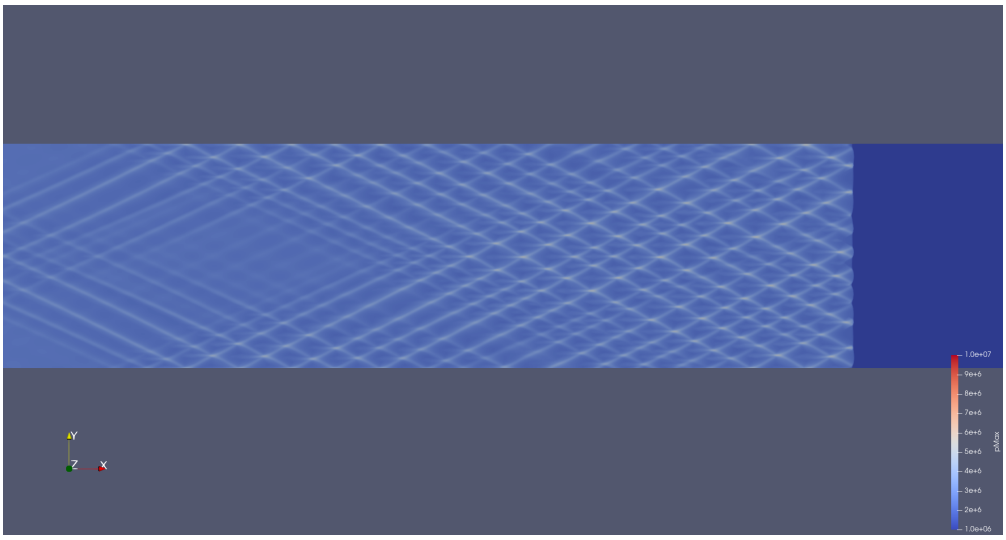


Fig. 4: History of maximum pressure during detonation propagation

C J Jachimowski, An Analytical Study of the Hydrogen-Air Reaction Mechanism with Application to Scramjet Combustion, NASA TP-2791, Feb. 1988.

CHAPTER

SEVEN

DFSPRAYFOAM

7.1 aachenBomb

Problem Description

This case simulates combustion inside a constant volume chamber based on an experimental setup at RWTH Aachen University. It can mimic, for example, the beginning of power stroke in a four-stroke diesel engine.

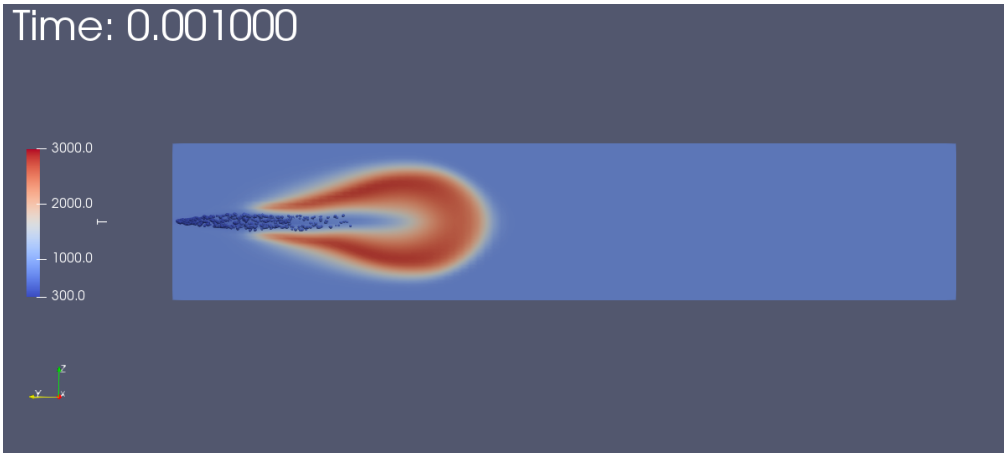
Table 1: Operating Conditions in Brief

Chamber size (xyz)	0.02×0.1×0.02m^3
Initial Gas Temperature	800K
Initial Gas Pressure	5MPa
Initial Gas Composition (mass fraction)	23.4% O2, 76.6% N2
Fuel	n-heptane
Fuel Temperature at the Nozzle	320K
Fuel Injection Duration	1.25ms
Total Injection Mass	6mg

Configurations Different from OpenFOAM Case

Cantera is used instead of the built-in modules of OpenFOAM to solve the chemical reactions. Therefore, a chemical mechanism file in YAML format is required in the case directory, and the full name of the mechanism file ("xxx.yaml") should be the entry after the keyword **CanteraMechanismFile** in *constant/CanteraTorchProperties*. Non-reacting simulation can be conducted by switching the entry after the keyword **chemistry** from **on** to **off** in *constant/CanteraTorchProperties*.

Results



CHAPTER

EIGHT

REACTION MECHANISM CONVERTION

DeepFlame uses *yaml* reaction mechanisms, which are compatible with Cantera. The following command lines can be used to convert *chemkin* mechanisms into *yaml* format.

```
conda create --name ct-env --channel conda-forge cantera
conda activate ct-env
ck2yaml --input=chem.inp --thermo=therm.dat --transport=tran.dat
```

Note: Users will need to create a new conda environment other than the one used for DeepFlame's dependencies, and the channel needs to be `conda-forge`. Otherwise, there might be an error regarding shared library, `libmkl_rt.so.2`.

More detailed instruction of converting mechanisms can be found on Cantera official website.

CHAPTER

NINE

FLAME SPEED

`flameSpeed.C` is another utility in DeepFlame. The flame is located at the maximum temperature gradient at a certain time, and its speed is equal to the maximum gradient porpagation speed subtracting the inlet speed. To use this utility, simply run the commands below after the simulation.

```
runApplication reconstructPar
flameSpeed
```

A log containing flame thickness, flame location, flame proagation speed, and flame speed at each time step will be presented.

Note: This utility only applies to one-dimensional cases. Similar logs can also exit when it is run for two or three dimensional cases, but results are not physical.

CHAPTER

TEN

DEVELOPERS TEAM

The current developers team consists the following research groups/affiliations:

- Peking University (Lead PI: Zhi X. Chen)
- Southern University of Science and Technology (Lead PI: Tianhan Zhang)
- Shanghai Jiao Tong University (Lead PI: Zhi-Qin John Xu)
- Beijing AI for Science Institute

CHAPTER

ELEVEN

HOW TO CITE

If you use DeepFlame for a publication, please use the citation:

Runze Mao, Minqi Lin, Yan Zhang, Tianhan Zhang, Zhi-Qin John Xu, Zhi X. Chen. DeepFlame: A deep learning empowered open-source platform for reacting flow simulations (2022). doi:10.48550/arXiv.2210.07094

If you have used the DNN model provided from us, please use the citation:

Tianhan Zhang, Yuxiao Yi, Yifan Xu, Zhi X. Chen, Yaoyu Zhang, Weinan E, Zhi-Qin John Xu. A multi-scale sampling method for accurate and robust deep neural network to predict combustion chemical kinetics. Combust. Flame 245:112319 (2022). doi:10.1016/j.combustflame.2022.112319

CHAPTER

TWELVE

LICENSE

The project DeepFlame is licensed under GNU General Public License v3.0

CHAPTER

THIRTEEN

SUBMITTING A PULL REQUEST

We welcome contributions from the open source community. The main approach to communicate with and to make contribution to DeepFlame is to open a pull request.

1. Fork the DeepFlame repository.
2. Pull your forked repository, and create a new git branchmake to your changes in it:

```
git checkout -b my-fix-branch
```

3. Coding your patch
4. After tests passed, commit your changes with a proper message.
5. Push your branch to GitHub:

```
git push origin my-fix-branch
```

6. In GitHub, send a pull request with `deepmodeling/deepflame-dev` as the base repository.
7. After your pull request is merged, you can safely delete your branch and sync the changes from the main (upstream) repository:

- Delete the remote branch on GitHub either through the GitHub web UI or your local shell as follows:

```
git push origin --delete my-fix-branch
```

- Check out the master branch:

```
git checkout develop -f
```

- Delete the local branch:

```
git branch -D my-fix-branch
```

- Update your master with the latest upstream version:

```
git pull --ff upstream develop
```