OpenFOAM meeting

1. VS code remote

2. Dask

Dask natively scales Python

Dask provides advanced parallelism for analytics, enabling performance at scale for the tools you love
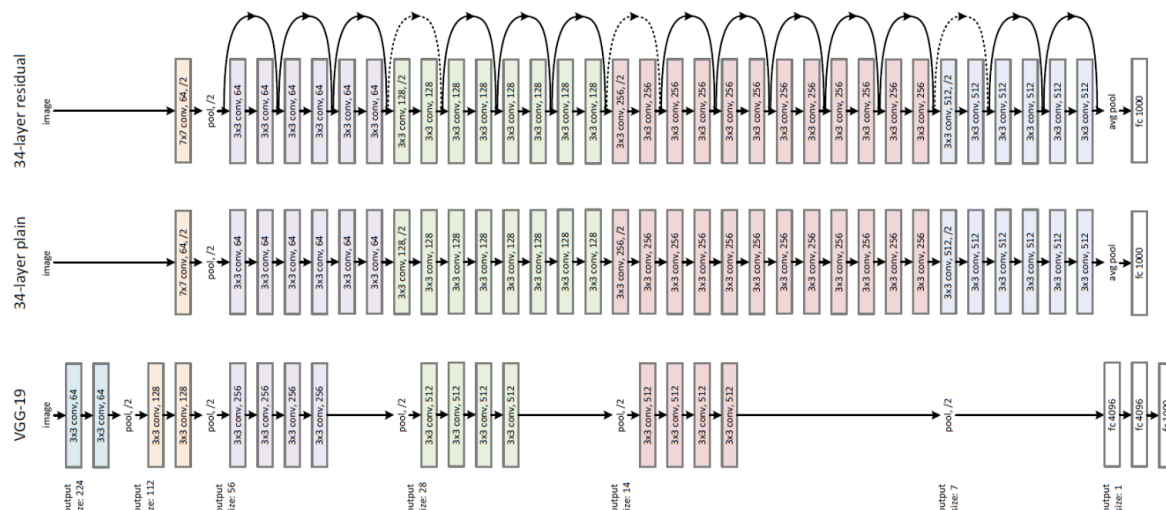
REMOTE-SSH

ralODE_post.py   dataGen.py   tf2.py

▲ CONNECTIONS

▲ 🖵 gpu133
  🗁 neuralODE_combustion /home/edis...
  🗀 LRT10Foam /home/edison/repos
▲ 🖵 gpu134
  🗀 LRT10Foam /home/edison/repos
🖵 gcp

src ▸ 🐍 dataGen.py

```python
132    dask.config.set(scheduler='processes')
133
134    s = time.time()
135        You, a month ago • add neuralode
136    a = [delayed(ignite_f)([x[0], x[1], 'H2']
137    a = dask.compute(*a)
138
139    print('a[0][0]={}'.format(len(a[0][0])))
140    e = time.time()
```

▲ FORWARDED PORTS

No ports are being forwarded from "gpu133".
Forward a port.

TERMINAL   ···   1: bash

```
edison@Node133:~/repos/neuralODE_combustion$ ls
base_neuralODE.h5     img                   README.md
boost_test.py         input_preprocessing.py rk4Model.h5
chem_test             LICENSE               savedModel
clustering.py         neuralODE_post.py     src
data                  neuralODE_train.py    stiffODE.py
delta_res_nets.py     notebooks             tf2.py
distributed.py        pb2uff                tmp
draft.md              pca_filter.py         tmp.png
fig                   postODENet.h5         wudi.h5
edison@Node133:~/repos/neuralODE_combustion$
```

▲ HELP AND FEEDBACK

★ Getting Started
📖 Read Documentation
🐦 Provide Feedback
ⓘ Review Issues
💬 Report Issue

Dask in action
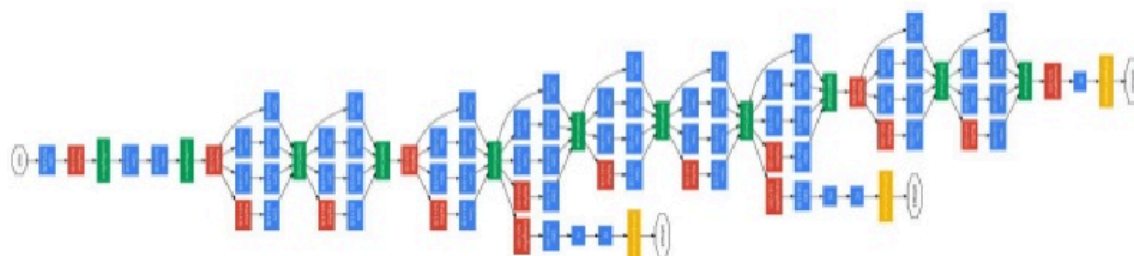
3. Neural ODE for chemical reactions

- **MLP**

  mlp

- **Deep Neural Networks**

GoogleNet or Inception (2014)

- **Neural ODE**

Computer Science > Machine Learning

Neural Ordinary Differential Equations

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, David Duvenaud

(Submitted on 19 Jun 2018 (v1), last revised 15 Jan 2019 (this version, v4))

We introduce a new family of deep neural network models. Instead of specifying a discrete sequence of hidden layers, we parameterize the derivative of the hidden state using a neural network. The output of the network is computed using a black-box differential equation solver. These continuous-depth models have constant memory cost, adapt their evaluation strategy to each input, and can explicitly trade numerical precision for speed. We demonstrate these properties in continuous-depth residual networks and continuous-time latent variable models. We also construct continuous normalizing flows, a generative model that can train by maximum likelihood, without partitioning or ordering the data dimensions. For training, we show how to scalably backpropagate through any ODE solver, without access to its internal operations. This allows end-to-end training of ODEs within larger models.

- **Runge kutta network**

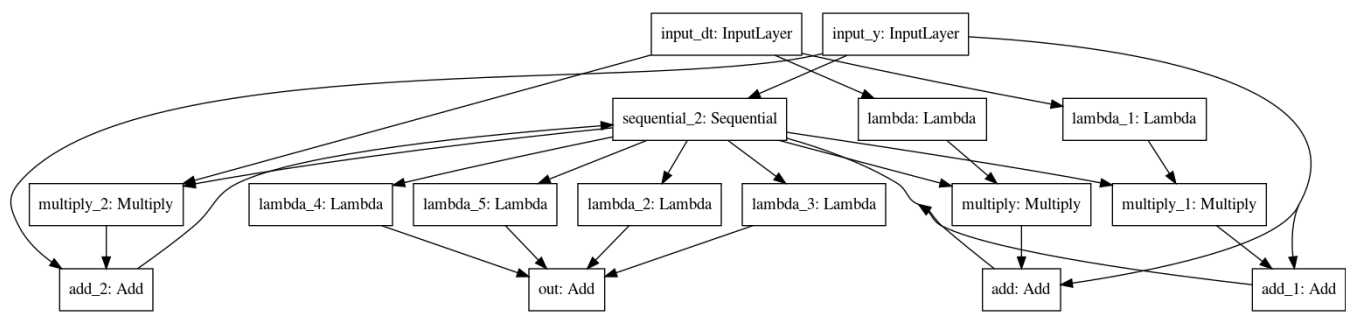**Runge kutta 4th order method**

$$k_1 = hf(x_n, y_n)$$

$$k_2 = hf(x_n + h/2, y_n + k_1/2)$$

$$k_3 = hf(x_n + h/2, y_n + k_2/2)$$
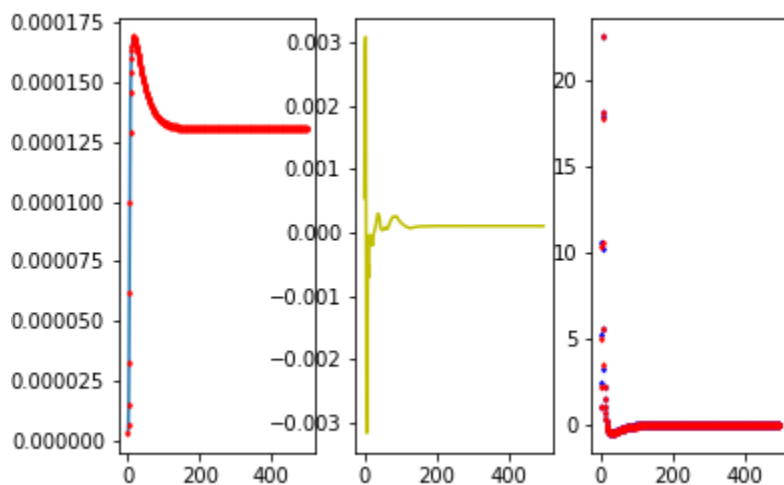
$$k_4 = hf(x_n + h, y_n + k_3)$$

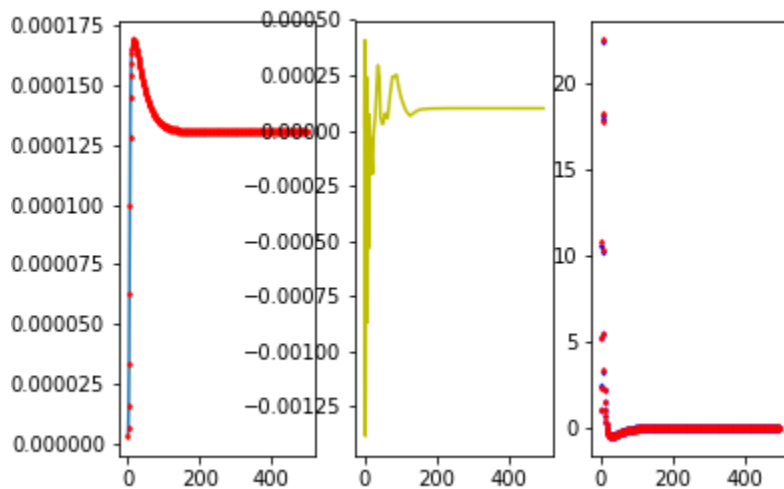$$y_{n+1} = y_n + 1/6(k_1 + 2k_2 + 2k_3 + k4)$$
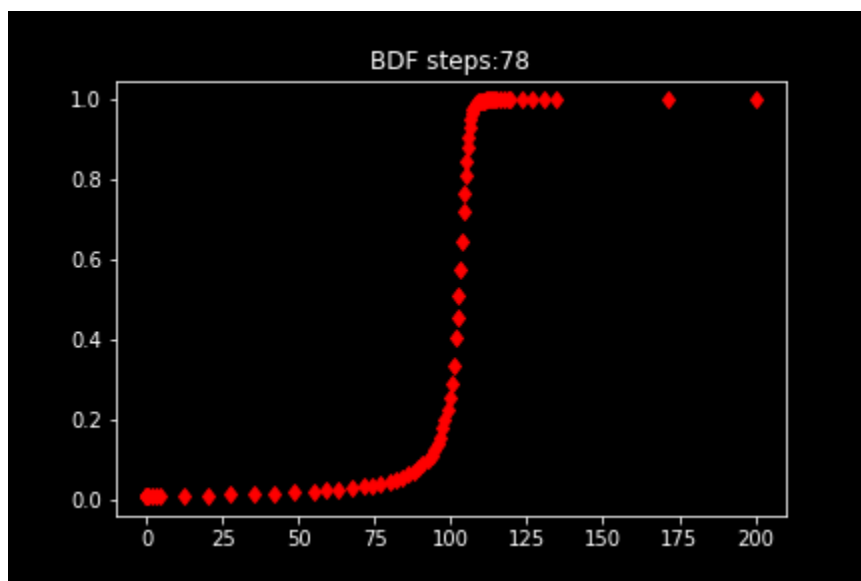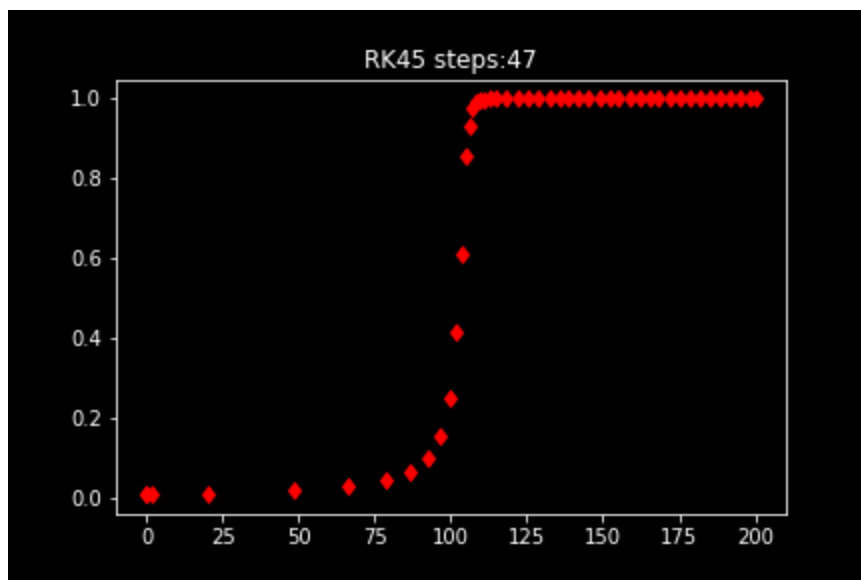
RK45Network

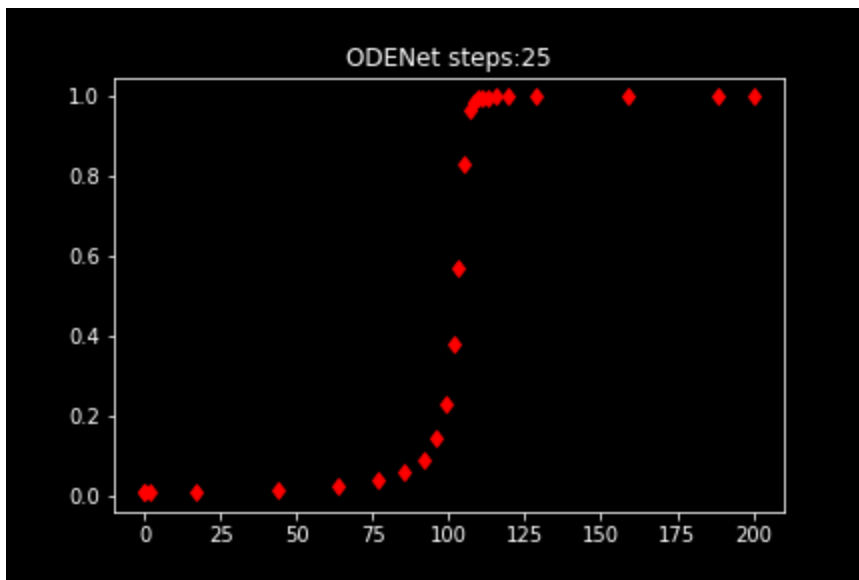**H2 auto ignition**

EULER: OH, T=1401

RK4: OH, T=1401

- **explicit or implicit flame expansion**

RK45 steps:47

BDF steps:78

Star me on github

https://github.com/uqyge/neuralODE_combustion