First notebook

October 29, 2022

Contents

1 First notebook **1**

1.1 Introduction 1

1.2 Step 1: Generating training data 2

1.3 Building neural networks 4

1 First notebook

[7]:
```
import numpy as np
import matplotlib.pyplot as plt
import random
import csv
import pandas as pd
import torch
from torch import nn
# from torch.utils.data import DataLoader
# from torchvision import datasets
# from torchvision.transforms import ToTensor
```

1.1 Introduction

The conserved variables are (D, S_i, τ) and they are related to primitive variables, $w = (\rho, v^i, \epsilon, p)$, defined in the local rest frame of the fluid through (in units of light speed $c = 1$). The P2C is explicitly given:

$$D = \rho W\,, \quad S_i = \rho h W^2 v_i\,, \quad \tau = \rho h W^2 - p - D\,, \tag{1}$$

where we used

$$W = (1 - v^2)^{-1/2}\,, \quad h = 1 + \epsilon + \frac{p}{\rho}\,. \tag{2}$$

Our first goal is to reproduce the results from this paper. We first focus on what they call **NNEOS** networks. These are networks which are trained to infer information on the equation of state (EOS). In its simplest form, the EOS is the thermodynamical relation connecting the pressure to the fluid's rest-mass density and internal energy $p = \bar{p}(\rho, \epsilon)$. We consider an **analytical Γ-law EOS** as a benchmark:

$$\bar{p}(\rho, \varepsilon) = (\Gamma - 1)\rho\epsilon\,, \tag{3}$$

and we fix $\Gamma = 5/3$ in order to fully mimic the situation of the paper.

1.2 Step 1: Generating training data

We generate training data for the NNEOS networks as follows. We create a training set by randomly sampling the EOS on a uniform distribution over $\rho \in (0, 10.1)$ and $\epsilon \in (0, 2.02)$. We then compute three quantities: - p, using the EOS defined above - $\chi := \partial p/\partial \rho$, inferred from the EOS - $\kappa := \partial p/\partial \epsilon$, inferred from the EOS

Define the relevant functions

[2]:
```
# Define the three functions determining the output
def eos(rho, eps, Gamma = 5/3):
    """Computes the analytical gamma law EOS from rho and epsilon"""
    return (Gamma - 1) * rho * eps

def chi(rho, eps, Gamma = 5/3):
    """Computes dp/drho from EOS"""
    return (Gamma - 1) * eps

def kappa(rho, eps, Gamma = 5/3):
    """Computes dp/deps from EOS"""
    return (Gamma - 1) * rho
```

[5]:
```
# Define ranges of parameters to be sampled (see paper Section 2.1)
rho_min = 0
rho_max = 10.1
eps_min = 0
eps_max = 2.02
```

Note: the code is in comment, as the data has been generated already and we want to use the same dataset for reproducibility.

[21]:
```
# number_of_datapoints = 10000 # 80 000 for train, 10 000 for test
# data = []

# for i in range(number_of_datapoints):
#     rho = random.uniform(rho_min, rho_max)
#     eps = random.uniform(eps_min, eps_max)

#     new_row = [rho, eps, eos(rho, eps), chi(rho, eps), kappa(rho, eps)]

#     data.append(new_row)
```

[22]:
```
# header = ['rho', 'eps', 'p', 'chi', 'kappa']

# with open('NNEOS_data_test.csv', 'w', newline = '') as file:
#     writer = csv.writer(file)
#     # write header
#     writer.writerow(header)
#     # write data
```

```
#     writer.writerows(data)
```

[24]:
```
data_train = pd.read_csv("NNEOS_data_train.csv")
data_train
```

[24]:

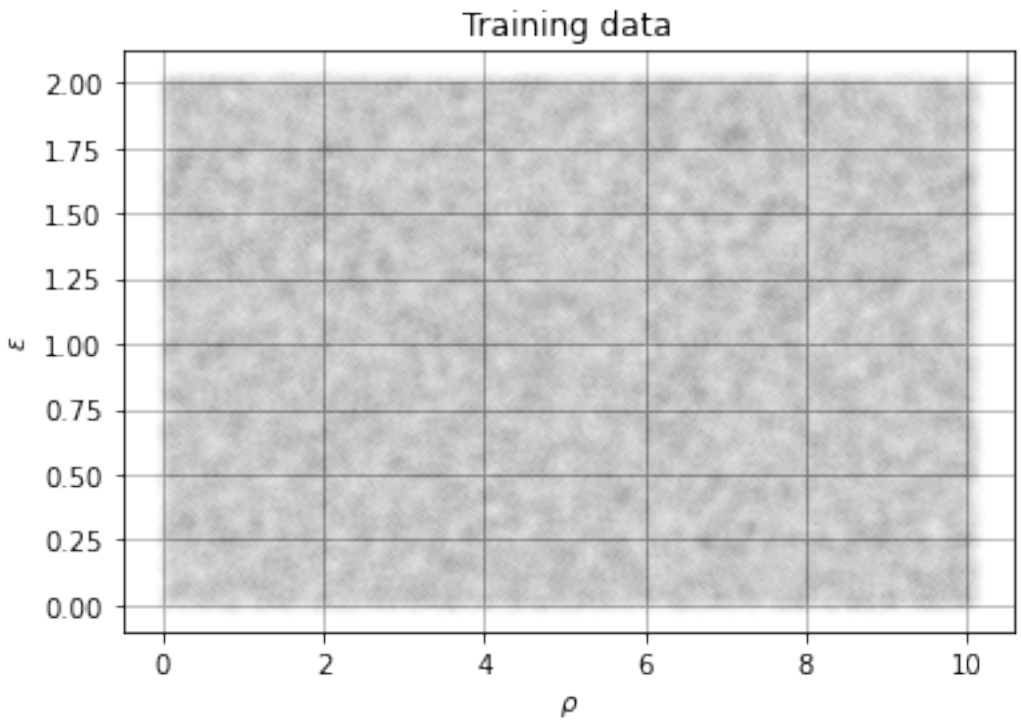
	rho	eps	p	chi	kappa
0	9.770794	0.809768	5.274717	0.539845	6.513863
1	10.093352	0.575342	3.871421	0.383561	6.728901
2	1.685186	1.647820	1.851255	1.098547	1.123457
3	1.167718	0.408377	0.317913	0.272251	0.778479
4	7.750848	1.069954	5.528700	0.713303	5.167232
…	…	…	…	…	…
79995	3.985951	1.642317	4.364131	1.094878	2.657301
79996	6.948815	0.809021	3.747824	0.539347	4.632543
79997	8.423227	1.125142	6.318217	0.750095	5.615485
79998	4.748173	0.774870	2.452810	0.516580	3.165449
79999	2.927483	0.616751	1.203686	0.411167	1.951655

[80000 rows x 5 columns]

In case we want to visualize the datapoints (not recommended).

[34]:
```
# rho = data_train['rho']
# eps = data_train['eps']

# plt.figure(figsize = (12,10))
# plt.plot(rho, eps, 'o', color = 'black', alpha = 0.005)
# plt.grid()
# plt.xlabel(r'$\rho$')
# plt.ylabel(r'$\epsilon$')
# plt.title('Training data')
# plt.show()
```



1.3 Building neural networks

[ ]:

[ ]: