NNEOSB

November 14, 2022

Contents

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['figure.dpi'] = 300
import random
import csv
import pandas as pd
import torch
from torch import nn # pytorch neural networks
from torch.utils.data import Dataset, DataLoader # pytorch dataset structures
from torchvision.transforms import ToTensor # pytorch transformer
# from torch.utils.data import DataLoader
# from torchvision import datasets
# from torchvision.transforms import ToTensor
```

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:

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

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

2 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 $\epsilon \in (0, 2.02)$. Below, we first focus on the implementation of **NNEOSB** as called in the paper, meaning we also make the derivatives of the EOS part of the output. So we 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

[2]:
```python
# 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
```

[3]:
```python
# 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 in comment below was used to generate the data. It has now been saved separately in a folder called "data".

[4]:
```python
# 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)
```

[5]:
```
# 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)
```

[6]:
```
# Import data
data_train = pd.read_csv("data/NNEOS_data_train.csv")
data_test = pd.read_csv("data/NNEOS_data_test.csv")
print("The training data has " + str(len(data_train)) + " instances")
print("The test data has " + str(len(data_test)) + " instances")
data_train
```

```
The training data has 80000 instances
The test data has 10000 instances
```

[6]:

	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 useful, nothing significant happening).

[7]:
```
# 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()
```

3 Getting data into PyTorch's DataLoader

Below: all_data is of the type $(\rho, \epsilon, p, \chi, \kappa)$ as generated above.

```
[8]: class CustomDataset(Dataset):
         """See PyTorch tutorial: the following three methods HAVE to be␣
     ↪implemented"""

         def __init__(self, all_data, transform=None, target_transform=None):
             self.transform = transform
             self.target_transform = target_transform

             # Separate features (rho and eps) from the labels (p, chi, kappa)
             # (see above to get how data is organized)
             features = []
             labels = []

             for i in range(len(all_data)):
                 # Separate the features
                 new_feature = [all_data['rho'][i], all_data['eps'][i]]
                 features.append(torch.tensor(new_feature, dtype = torch.float32))
                 # Separate the labels
                 new_label = [all_data['p'][i], all_data['chi'][i],␣
     ↪all_data['kappa'][i]]
                 labels.append(torch.tensor(new_label, dtype = torch.float32))

             # Save as instance variables to the dataloader
             self.features = features
             self.labels = labels

         def __len__(self):
             return len(self.labels)

         # TODO: I don't understand transform and target_transform --- but this is␣
     ↪not used now!
         def __getitem__(self, idx):
             feature = self.features[idx]
             if self.transform:
                 feature = transform(feature)
```

```
        label = self.labels[idx]
        if self.target_transform:
            feature = target_transform(label)

        return feature, label
```

Note that the following cell may be confusing. "data_train" refers to the data that was generatd above, see the pandas table. "training_data" is defined similarly as in the PyTorch tutorial, see this page and this is an instance of the class CustomDataset defined above.

[9]:
```
# Make training and test data, as in the tutorial
training_data = CustomDataset(data_train)
test_data = CustomDataset(data_test)
```

[10]:
```
# Check if this is done correctly
print(training_data.features[:2])
print(training_data.labels[:2])
print(training_data.__len__())
print(test_data.__len__())
```

```
[tensor([9.7708, 0.8098]), tensor([10.0934,  0.5753])]
[tensor([5.2747, 0.5398, 6.5139]), tensor([3.8714, 0.3836, 6.7289])]
80000
10000
```

[11]:
```
# Now call DataLoader on the above CustomDataset instances:
train_dataloader = DataLoader(training_data, batch_size=32)
test_dataloader = DataLoader(test_data, batch_size=32)
```

4 Building the neural networks

We will follow this part of the PyTorch tutorial. For more information, see the documentation page of torch.nn. We take the parameters of NNEOSB in the paper, see Table 1. **To do:** check other activation functions and architectures.

[16]:
```
# Define hyperparameters of the model here. Will first of all put two hidden␣
 ↪layers
device = "cpu"
size_HL_1 = 400
size_HL_2 = 600

# Implement neural network
class NeuralNetwork(nn.Module):
    def __init__(self):
        super(NeuralNetwork, self).__init__()
        #self.flatten = nn.Flatten()
        self.stack = nn.Sequential(
            nn.Linear(2, size_HL_1),
```

```python
            nn.Sigmoid(),
            nn.Linear(size_HL_1, size_HL_2),
            nn.Sigmoid(),#nn.ReLU(),
            nn.Linear(size_HL_2, 3)
        )

    def forward(self, x):
        # No flatten needed, as our input and output are 1D?
        #x = self.flatten(x)
        logits = self.stack(x)
        return logits
```

5 Training the neural network

Now we generate an instance of the above neural network in `model` (note: running this cell will create a 'fresh' model!).

Save hyperparameters and loss function - note that we follow the paper. I think that their loss function agrees with MSELoss. The paper uses the Adam optimizer. More details on optimizers can be found here. Required argument `params` can be filled in by calling `model` which contains the neural network. For simplicity we will train for 10 epochs here. **Question:** how many epochs should be used? What size for the batches,…

[91]:
```python
model = NeuralNetwork().to(device)
print(model)

# Save hyperparameters, loss function and optimizer here (see paper for details)
# learning_rate = 6e-4
learning_rate = 4.6875e-6
batch_size = 32
epochs = 200
loss_fn = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)

### not sure how this works
# Adaptive learning rate:
# scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer, 'min',␣
 ↪factor=0.5, verbose=True)
```

```
NeuralNetwork(
  (stack): Sequential(
    (0): Linear(in_features=2, out_features=400, bias=True)
    (1): Sigmoid()
    (2): Linear(in_features=400, out_features=600, bias=True)
    (3): Sigmoid()
    (4): Linear(in_features=600, out_features=3, bias=True)
  )
)
```

The train and test loops are implemented below (copy pasted from this part of the tutorial):

[101]:
```
def train_loop(dataloader, model, loss_fn, optimizer, report_progress = False):
    """The training loop of the algorithm"""
    size = len(dataloader.dataset)
    for batch, (X, y) in enumerate(dataloader):
        # Compute prediction and loss
        pred = model(X)
        loss = loss_fn(pred, y)

        # Backpropagation
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        # If we want to report progress during training (not recommended -␣
↪obstructs view)
        if report_progress:
            if batch % 100 == 0:
                loss, current = loss.item(), batch * len(X)
                print(f"loss: {loss:>7f}  [{current:>5d}/{size:>5d}]")


def test_loop(dataloader, model, loss_fn):
    """The testing loop of the algorithm"""
    num_batches = len(dataloader)
    test_loss = 0

    # Predict and compute losses
    with torch.no_grad():
        for X, y in dataloader:
            pred = model(X)
            test_loss += loss_fn(pred, y).item()

    average_test_loss = test_loss/num_batches
    return average_test_loss
```

[102]:
```
def get_subset_train_dataloader(data_train, size = 10000):
    """Creates a 'subset' of dataloader for computing loss on training data.
        This way we can 'test' on training data too - to check the claim of the␣
↪paper about overfitting. """

    # Get random ids to sample
    random_ids =  np.random.choice(len(data_train), size, replace=False)

    # the following is a pandas dataframe
    sampled_train_data = data_train.iloc[random_ids]
```

```
    # relabel the indices
    sampled_train_data.index = [i for i in range(len(sampled_train_data))]
    new_dataset = CustomDataset(sampled_train_data)

    # Make it a dataloader and return it
    new_dataloader = DataLoader(new_dataset, batch_size=32)

    return new_dataloader
```

continue with NNEOSBv1

[103]:
```
model = torch.load('NNEOSBv1.pth')
```

[104]:
```
# Restart training by changing this parameter:
restart = True
abort = False
update_lr = True
batch_size = 32
max_number_epochs = 400
adaptation_threshold = 0.9995
adaptation_multiplier = 0.9

# Initialize the loss function
loss_fn = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)

# Initialize lists in case we start a new training loop
if restart:
    confirmation = input("Are you sure you want to restart? Press y >> ")
    if confirmation == "y":
        test_losses = []
        train_losses = []
        train_losses_subset = []
        adaptation_indices = []
        counter = -5 # we skip the very first few iterations before changing␣
 ↪learning rate
    else:
        print("Aborting training.")
        abort = True

# Acutal training loop is done:
if abort is False:
    epoch_counter = len(train_losses) + 1

    print("Training the model . . .")
    if restart is False:
        print("(Continued)")
```

```
    # Training:
    while epoch_counter < max_number_epochs:
        print(f"\n Epoch {epoch_counter} \n --------------")
        # Train
        train_loop(train_dataloader, model, loss_fn, optimizer)
        # Test on the training data
        average_train_loss = test_loop(train_dataloader, model, loss_fn)
        train_losses.append(average_train_loss)
        # Test on SUBSET of the training data
        train_subset_dataloader = get_subset_train_dataloader(data_train)
        average_train_loss = test_loop(train_subset_dataloader, model, loss_fn)
        train_losses_subset.append(average_train_loss)
        # Test on testing data
        average_test_loss = test_loop(test_dataloader, model, loss_fn)
        test_losses.append(average_test_loss)

        # Update the learning rate - see Appendix B of the paper
        # only check if update needed after 10 new epochs
        if counter >= 10 and update_lr is True:
            current = np.min(train_losses[-5:])
            previous = np.min(train_losses[-10:-5])

            # If we did not improve the test loss sufficiently, going to adapt␣
↪LR
            if current/previous >= adaptation_threshold:
                # Reset counter (note: will increment later, so set to -1 st it␣
↪becomes 0)
                counter = -1
                learning_rate = adaptation_multiplier*learning_rate
                print(f"Adapting learning rate to {learning_rate}")
                # Change optimizer
                optimizer = torch.optim.Adam(model.parameters(),␣
↪lr=learning_rate)
                # Add the epoch time for plotting later on
                adaptation_indices.append(epoch_counter)

        # Report progress:
#         print(f"Average loss of: {average_test_loss} for test data")
        print(f"Average loss of: {average_train_loss} for train data")

        # Another epoch passed - increment counter
        counter += 1
        epoch_counter += 1

    print("Done!")
```

```
Are you sure you want to restart? Press y >> y
Training the model . . .

 Epoch 1
 --------------
Average loss of: 3.9656275366807545e-08 for train data

 Epoch 2
 --------------
Average loss of: 3.716019254323629e-08 for train data

 Epoch 3
 --------------
Average loss of: 3.605011615034167e-08 for train data

 Epoch 4
 --------------
Average loss of: 3.869103654999537e-08 for train data

 Epoch 5
 --------------
Average loss of: 3.802104947104466e-08 for train data

 Epoch 6
 --------------
Average loss of: 3.487937339733266e-08 for train data

 Epoch 7
 --------------
Average loss of: 3.417244890182027e-08 for train data

 Epoch 8
 --------------
Average loss of: 3.8948198015706376e-08 for train data

 Epoch 9
 --------------
Average loss of: 3.611024425447031e-08 for train data

 Epoch 10
 --------------
Average loss of: 3.756165950089083e-08 for train data

 Epoch 11
 --------------
Average loss of: 3.804749886253245e-08 for train data

 Epoch 12
```

```
 --------------
Average loss of: 3.7600973157941954e-08 for train data

 Epoch 13
 --------------
Average loss of: 3.710415834572001e-08 for train data

 Epoch 14
 --------------
Average loss of: 3.7186421669313614e-08 for train data

 Epoch 15
 --------------
Average loss of: 3.511040774127833e-08 for train data

 Epoch 16
 --------------
Average loss of: 3.587256612028535e-08 for train data

 Epoch 17
 --------------
Average loss of: 3.210988081176723e-08 for train data

 Epoch 18
 --------------
Average loss of: 3.1800146578475666e-08 for train data

 Epoch 19
 --------------
Average loss of: 3.410315440480728e-08 for train data

 Epoch 20
 --------------
Average loss of: 3.274497183732988e-08 for train data

 Epoch 21
 --------------
Average loss of: 3.533321528862826e-08 for train data

 Epoch 22
 --------------
Average loss of: 3.837813405658735e-08 for train data

 Epoch 23
 --------------
Average loss of: 3.322278134529921e-08 for train data

 Epoch 24
```

```
 --------------
Average loss of: 3.2989946689235476e-08 for train data

 Epoch 25
 --------------
Average loss of: 3.4498897035521075e-08 for train data

 Epoch 26
 --------------
Average loss of: 3.373054959376224e-08 for train data

 Epoch 27
 --------------
Average loss of: 3.646424172194738e-08 for train data

 Epoch 28
 --------------
Average loss of: 3.243444333984538e-08 for train data

 Epoch 29
 --------------
Average loss of: 3.5955106518656704e-08 for train data

 Epoch 30
 --------------
Average loss of: 3.471433224229131e-08 for train data

 Epoch 31
 --------------
Average loss of: 3.2624217758247445e-08 for train data

 Epoch 32
 --------------
Average loss of: 3.3159731426445716e-08 for train data

 Epoch 33
 --------------
Average loss of: 3.133581688344136e-08 for train data

 Epoch 34
 --------------
Average loss of: 3.2883910214639965e-08 for train data

 Epoch 35
 --------------
Average loss of: 3.137121819116974e-08 for train data

 Epoch 36
```

```
 --------------
Average loss of: 3.4862964857204325e-08 for train data

 Epoch 37
 --------------
Average loss of: 2.8229605992525552e-08 for train data

 Epoch 38
 --------------
Average loss of: 3.1416802356650004e-08 for train data

 Epoch 39
 --------------
Average loss of: 3.214741090439689e-08 for train data

 Epoch 40
 --------------
Average loss of: 3.151867600425789e-08 for train data

 Epoch 41
 --------------
Average loss of: 3.076069082661646e-08 for train data

 Epoch 42
 --------------
Average loss of: 3.0987181661223406e-08 for train data

 Epoch 43
 --------------
Average loss of: 3.1727118979598125e-08 for train data

 Epoch 44
 --------------
Average loss of: 2.927234164540717e-08 for train data

 Epoch 45
 --------------
Average loss of: 2.850116378617179e-08 for train data

 Epoch 46
 --------------
Average loss of: 2.9115289456616863e-08 for train data

 Epoch 47
 --------------
Average loss of: 2.823471736568911e-08 for train data

 Epoch 48
```

```
 --------------
Average loss of: 2.8568788936237985e-08 for train data

 Epoch 49
 --------------
Average loss of: 2.85102714911773e-08 for train data

 Epoch 50
 --------------
Average loss of: 2.85499862331445e-08 for train data

 Epoch 51
 --------------
Average loss of: 2.6542487392926167e-08 for train data

 Epoch 52
 --------------
Average loss of: 2.852271720053213e-08 for train data

 Epoch 53
 --------------
Average loss of: 2.685605900249886e-08 for train data

 Epoch 54
 --------------
Average loss of: 2.6848865324825437e-08 for train data

 Epoch 55
 --------------
Average loss of: 2.894451507014951e-08 for train data

 Epoch 56
 --------------
Average loss of: 2.8696734950688568e-08 for train data

 Epoch 57
 --------------
Average loss of: 2.9144240423905963e-08 for train data

 Epoch 58
 --------------
Average loss of: 2.604699807207158e-08 for train data

 Epoch 59
 --------------
Average loss of: 2.6580363784335324e-08 for train data

 Epoch 60
```

```
 --------------
Average loss of: 2.8261188638147483e-08 for train data

 Epoch 61
 --------------
Average loss of: 2.8461491007087883e-08 for train data

 Epoch 62
 --------------
Average loss of: 2.6075002036855867e-08 for train data

 Epoch 63
 --------------
Average loss of: 2.7555899100335023e-08 for train data

 Epoch 64
 --------------
Average loss of: 2.4245537403113083e-08 for train data

 Epoch 65
 --------------
Average loss of: 2.64057220845826e-08 for train data

 Epoch 66
 --------------
Average loss of: 2.5773773224284177e-08 for train data

 Epoch 67
 --------------
Average loss of: 2.6858814744885012e-08 for train data

 Epoch 68
 --------------
Average loss of: 2.40518719204564e-08 for train data

 Epoch 69
 --------------
Average loss of: 2.448724737641337e-08 for train data

 Epoch 70
 --------------
Average loss of: 2.5458800656554758e-08 for train data

 Epoch 71
 --------------
Average loss of: 2.662722840514557e-08 for train data

 Epoch 72
```

```
 --------------
Average loss of: 2.6049230824925322e-08 for train data

 Epoch 73
 --------------
Average loss of: 2.7272617203991943e-08 for train data

 Epoch 74
 --------------
Average loss of: 2.859144890456641e-08 for train data

 Epoch 75
 --------------
Average loss of: 2.4215464067739817e-08 for train data

 Epoch 76
 --------------
Average loss of: 2.4441993000376992e-08 for train data

 Epoch 77
 --------------
Average loss of: 2.4986604299991664e-08 for train data

 Epoch 78
 --------------
Average loss of: 2.4487391088221888e-08 for train data

 Epoch 79
 --------------
Average loss of: 2.277483091857905e-08 for train data

 Epoch 80
 --------------
Average loss of: 2.4838377735848506e-08 for train data

 Epoch 81
 --------------
Average loss of: 2.2422933116236677e-08 for train data

 Epoch 82
 --------------
Average loss of: 2.4840709428373047e-08 for train data

 Epoch 83
 --------------
Average loss of: 2.672523824110441e-08 for train data

 Epoch 84
```

```
 --------------
Average loss of: 2.3676443961944757e-08 for train data

 Epoch 85
 --------------
Average loss of: 2.3550350245007273e-08 for train data

 Epoch 86
 --------------
Average loss of: 2.4110217717233994e-08 for train data

 Epoch 87
 --------------
Average loss of: 2.281514604929455e-08 for train data

 Epoch 88
 --------------
Average loss of: 2.3572611262317932e-08 for train data

 Epoch 89
 --------------
Average loss of: 2.561007488150818e-08 for train data

 Epoch 90
 --------------
Average loss of: 2.4343603741975095e-08 for train data

 Epoch 91
 --------------
Average loss of: 2.2956524340690103e-08 for train data

 Epoch 92
 --------------
Average loss of: 2.2325116341322448e-08 for train data

 Epoch 93
 --------------
Average loss of: 2.352275113351268e-08 for train data

 Epoch 94
 --------------
Average loss of: 2.305670661566144e-08 for train data

 Epoch 95
 --------------
Average loss of: 2.2042823418880365e-08 for train data

 Epoch 96
```

```
 --------------
Average loss of: 2.3145481898845705e-08 for train data

 Epoch 97
 --------------
Average loss of: 2.3301973706643424e-08 for train data

 Epoch 98
 --------------
Average loss of: 2.337100169621552e-08 for train data

 Epoch 99
 --------------
Average loss of: 2.1640068355643382e-08 for train data

 Epoch 100
 --------------
Average loss of: 2.4113521814733678e-08 for train data

 Epoch 101
 --------------
Average loss of: 2.5146560386706363e-08 for train data

 Epoch 102
 --------------
Average loss of: 2.094976783602227e-08 for train data

 Epoch 103
 --------------
Average loss of: 2.0629460298340357e-08 for train data

 Epoch 104
 --------------
Average loss of: 2.4196617001127442e-08 for train data

 Epoch 105
 --------------
Average loss of: 2.192571642172442e-08 for train data

 Epoch 106
 --------------
Average loss of: 2.2183948780898086e-08 for train data

 Epoch 107
 --------------
Average loss of: 2.136465272356079e-08 for train data

 Epoch 108
```

```
 --------------
Average loss of: 2.105262002356295e-08 for train data

 Epoch 109
 --------------
Average loss of: 2.261402906183597e-08 for train data

 Epoch 110
 --------------
Average loss of: 2.1230688616447732e-08 for train data

 Epoch 111
 --------------
Average loss of: 2.236114529324232e-08 for train data

 Epoch 112
 --------------
Average loss of: 2.0900970536133127e-08 for train data

 Epoch 113
 --------------
Average loss of: 2.2561442062118466e-08 for train data

 Epoch 114
 --------------
Average loss of: 2.2520775926412893e-08 for train data

 Epoch 115
 --------------
Average loss of: 2.065608349187296e-08 for train data

 Epoch 116
 --------------
Average loss of: 2.1398720344844247e-08 for train data

 Epoch 117
 --------------
Average loss of: 2.2929078560572213e-08 for train data

 Epoch 118
 --------------
Average loss of: 2.3302839453592174e-08 for train data

 Epoch 119
 --------------
Average loss of: 2.5711605766024462e-08 for train data

 Epoch 120
```

```
 --------------
Average loss of: 2.0563829676352536e-08 for train data

 Epoch 121
 --------------
Average loss of: 2.3243066714473227e-08 for train data

 Epoch 122
 --------------
Average loss of: 2.0699608944905282e-08 for train data

 Epoch 123
 --------------
Average loss of: 2.2524418005596394e-08 for train data

 Epoch 124
 --------------
Average loss of: 2.1414034118522567e-08 for train data

 Epoch 125
 --------------
Average loss of: 2.1803329862193688e-08 for train data

 Epoch 126
 --------------
Average loss of: 2.306037215921348e-08 for train data

 Epoch 127
 --------------
Average loss of: 2.052748999251678e-08 for train data

 Epoch 128
 --------------
Average loss of: 2.252601659905847e-08 for train data

 Epoch 129
 --------------
Average loss of: 2.1239227240284826e-08 for train data

 Epoch 130
 --------------
Average loss of: 2.0332748979708826e-08 for train data

 Epoch 131
 --------------
Average loss of: 2.1940679191831484e-08 for train data

 Epoch 132
```

```
 --------------
Average loss of: 1.995270434071235e-08 for train data

 Epoch 133
 --------------
Average loss of: 2.0067068807490844e-08 for train data

 Epoch 134
 --------------
Adapting learning rate to 4.21875e-06
Average loss of: 2.0767464780161985e-08 for train data

 Epoch 135
 --------------
Average loss of: 2.0728052921224056e-08 for train data

 Epoch 136
 --------------
Average loss of: 1.8683102099837274e-08 for train data

 Epoch 137
 --------------
Average loss of: 2.0060848664169593e-08 for train data

 Epoch 138
 --------------
Average loss of: 1.851899518080006e-08 for train data

 Epoch 139
 --------------
Average loss of: 2.085253944951597e-08 for train data

 Epoch 140
 --------------
Average loss of: 2.0984559353931722e-08 for train data

 Epoch 141
 --------------
Average loss of: 2.125548222486363e-08 for train data

 Epoch 142
 --------------
Average loss of: 1.97036386298098e-08 for train data

 Epoch 143
 --------------
Average loss of: 2.0035366087524405e-08 for train data
```

```
 Epoch 144
 --------------
Average loss of: 1.8834248481013244e-08 for train data

 Epoch 145
 --------------
Average loss of: 1.8873680413350984e-08 for train data

 Epoch 146
 --------------
Average loss of: 1.9938520006493986e-08 for train data

 Epoch 147
 --------------
Average loss of: 1.901866052220637e-08 for train data

 Epoch 148
 --------------
Average loss of: 2.026511635816091e-08 for train data

 Epoch 149
 --------------
Average loss of: 1.9039122877481246e-08 for train data

 Epoch 150
 --------------
Average loss of: 1.7351085170705976e-08 for train data

 Epoch 151
 --------------
Average loss of: 1.9473223692413608e-08 for train data

 Epoch 152
 --------------
Average loss of: 1.8900701215702015e-08 for train data

 Epoch 153
 --------------
Average loss of: 1.8441214331945725e-08 for train data

 Epoch 154
 --------------
Average loss of: 1.836047869509742e-08 for train data

 Epoch 155
 --------------
Average loss of: 1.7416910654215106e-08 for train data
```

```
 Epoch 156
 --------------
Average loss of: 1.7952485357012827e-08 for train data

 Epoch 157
 --------------
Average loss of: 1.751544282705128e-08 for train data

 Epoch 158
 --------------
Average loss of: 1.7332085842503683e-08 for train data

 Epoch 159
 --------------
Average loss of: 1.8427248745005377e-08 for train data

 Epoch 160
 --------------
Average loss of: 1.9020149716490763e-08 for train data

 Epoch 161
 --------------
Average loss of: 1.7659883665467928e-08 for train data

 Epoch 162
 --------------
Average loss of: 2.1020089319837567e-08 for train data

 Epoch 163
 --------------
Average loss of: 1.796696507440243e-08 for train data

 Epoch 164
 --------------
Average loss of: 1.769769929185255e-08 for train data

 Epoch 165
 --------------
Average loss of: 1.8690176777209428e-08 for train data

 Epoch 166
 --------------
Average loss of: 1.829210694452173e-08 for train data

 Epoch 167
 --------------
Average loss of: 1.854306248060384e-08 for train data
```

```
 Epoch 168
 --------------
Average loss of: 1.950000353167561e-08 for train data

 Epoch 169
 --------------
Average loss of: 1.7778413569854324e-08 for train data

 Epoch 170
 --------------
Average loss of: 1.6845578221661867e-08 for train data

 Epoch 171
 --------------
Average loss of: 1.900248635760731e-08 for train data

 Epoch 172
 --------------
Average loss of: 1.703475856930046e-08 for train data

 Epoch 173
 --------------
Average loss of: 1.745092205777728e-08 for train data

 Epoch 174
 --------------
Average loss of: 1.9531199050687046e-08 for train data

 Epoch 175
 --------------
Average loss of: 1.625457582944685e-08 for train data

 Epoch 176
 --------------
Average loss of: 1.827222562535355e-08 for train data

 Epoch 177
 --------------
Average loss of: 1.8469707978152865e-08 for train data

 Epoch 178
 --------------
Average loss of: 2.100681660854403e-08 for train data

 Epoch 179
 --------------
Average loss of: 1.825709856493989e-08 for train data
```

```
 Epoch 180
 --------------
Average loss of: 1.8102942349990157e-08 for train data

 Epoch 181
 --------------
Average loss of: 1.6437446915133364e-08 for train data

 Epoch 182
 --------------
Average loss of: 1.7624167465857017e-08 for train data

 Epoch 183
 --------------
Average loss of: 1.7508332226692854e-08 for train data

 Epoch 184
 --------------
Average loss of: 1.8030641831560295e-08 for train data

 Epoch 185
 --------------
Average loss of: 1.7319891897679268e-08 for train data

 Epoch 186
 --------------
Average loss of: 1.7357603842712138e-08 for train data

 Epoch 187
 --------------
Average loss of: 1.7023809565438577e-08 for train data

 Epoch 188
 --------------
Average loss of: 1.796971604815013e-08 for train data

 Epoch 189
 --------------
Average loss of: 1.8232820392283385e-08 for train data

 Epoch 190
 --------------
Average loss of: 1.7822572820480396e-08 for train data

 Epoch 191
 --------------
Average loss of: 1.6852466964699e-08 for train data
```

```
 Epoch 192
 --------------
Average loss of: 1.769570750350665e-08 for train data

 Epoch 193
 --------------
Average loss of: 1.718251275447417e-08 for train data

 Epoch 194
 --------------
Average loss of: 1.7708177202191707e-08 for train data

 Epoch 195
 --------------
Average loss of: 1.8179436577342925e-08 for train data

 Epoch 196
 --------------
Adapting learning rate to 3.796875e-06
Average loss of: 1.8517390494944795e-08 for train data

 Epoch 197
 --------------
Average loss of: 1.900607153945697e-08 for train data

 Epoch 198
 --------------
Average loss of: 1.844973889674135e-08 for train data

 Epoch 199
 --------------
Average loss of: 1.5591153183554336e-08 for train data

 Epoch 200
 --------------
Average loss of: 1.6758489047309143e-08 for train data

 Epoch 201
 --------------
Average loss of: 1.5947463701740836e-08 for train data

 Epoch 202
 --------------
Average loss of: 1.589696324737233e-08 for train data

 Epoch 203
 --------------
Average loss of: 1.747598792976801e-08 for train data
```

```
 Epoch 204
 --------------
Average loss of: 1.6678891585567004e-08 for train data

 Epoch 205
 --------------
Average loss of: 1.6519428012230628e-08 for train data

 Epoch 206
 --------------
Average loss of: 1.7154041369480138e-08 for train data

 Epoch 207
 --------------
Average loss of: 1.703549449038669e-08 for train data

 Epoch 208
 --------------
Average loss of: 1.6844158705400144e-08 for train data

 Epoch 209
 --------------
Average loss of: 1.6701002123319847e-08 for train data

 Epoch 210
 --------------
Average loss of: 1.537543044959355e-08 for train data

 Epoch 211
 --------------
Average loss of: 1.569023027797086e-08 for train data

 Epoch 212
 --------------
Average loss of: 1.6031628476287114e-08 for train data

 Epoch 213
 --------------
Average loss of: 1.6147126155155623e-08 for train data

 Epoch 214
 --------------
Average loss of: 1.514381461835789e-08 for train data

 Epoch 215
 --------------
Average loss of: 1.5388845944790353e-08 for train data
```

```
 Epoch 216
 --------------
Average loss of: 1.7123805344224005e-08 for train data

 Epoch 217
 --------------
Average loss of: 1.6557145775825317e-08 for train data

 Epoch 218
 --------------
Average loss of: 1.6177929208554336e-08 for train data

 Epoch 219
 --------------
Average loss of: 1.4907728257389536e-08 for train data

 Epoch 220
 --------------
Average loss of: 1.4674044716997695e-08 for train data

 Epoch 221
 --------------
Average loss of: 1.5479410420836644e-08 for train data

 Epoch 222
 --------------
Average loss of: 1.4910836132723972e-08 for train data

 Epoch 223
 --------------
Average loss of: 1.5801835514915033e-08 for train data

 Epoch 224
 --------------
Average loss of: 1.53029285998443e-08 for train data

 Epoch 225
 --------------
Average loss of: 1.6048359567063337e-08 for train data

 Epoch 226
 --------------
Average loss of: 1.5853687298697543e-08 for train data

 Epoch 227
 --------------
Average loss of: 1.4155368800439678e-08 for train data
```

```
 Epoch 228
 --------------
Average loss of: 1.5660119614984103e-08 for train data

 Epoch 229
 --------------
Average loss of: 1.6376609924026498e-08 for train data

 Epoch 230
 --------------
Average loss of: 1.4813735625465507e-08 for train data

 Epoch 231
 --------------
Average loss of: 1.5030884458991322e-08 for train data

 Epoch 232
 --------------
Average loss of: 1.676006838460906e-08 for train data

 Epoch 233
 --------------
Average loss of: 1.5255790882614595e-08 for train data

 Epoch 234
 --------------
Average loss of: 1.4057086578887878e-08 for train data

 Epoch 235
 --------------
Average loss of: 1.5049632432700843e-08 for train data

 Epoch 236
 --------------
Average loss of: 1.458426894906336e-08 for train data

 Epoch 237
 --------------
Average loss of: 1.4692760181390712e-08 for train data

 Epoch 238
 --------------
Average loss of: 1.5421000371586346e-08 for train data

 Epoch 239
 --------------
Average loss of: 1.54420819021438e-08 for train data
```

```
 Epoch 240
 --------------
Average loss of: 1.4278305884452641e-08 for train data

 Epoch 241
 --------------
Average loss of: 1.6284880268561158e-08 for train data

 Epoch 242
 --------------
Average loss of: 1.4110007980618026e-08 for train data

 Epoch 243
 --------------
Average loss of: 1.4428435549630743e-08 for train data

 Epoch 244
 --------------
Average loss of: 1.477765795568238e-08 for train data

 Epoch 245
 --------------
Average loss of: 1.4798109027118641e-08 for train data

 Epoch 246
 --------------
Average loss of: 1.4625710778087529e-08 for train data

 Epoch 247
 --------------
Average loss of: 1.4258346123239271e-08 for train data

 Epoch 248
 --------------
Average loss of: 1.5144945635540536e-08 for train data

 Epoch 249
 --------------
Average loss of: 1.644012906356643e-08 for train data

 Epoch 250
 --------------
Average loss of: 1.4931360230587173e-08 for train data

 Epoch 251
 --------------
Average loss of: 1.526338022107295e-08 for train data
```

```
 Epoch 252
 --------------
Average loss of: 1.4902732421198937e-08 for train data

 Epoch 253
 --------------
Average loss of: 1.579725743897752e-08 for train data

 Epoch 254
 --------------
Average loss of: 1.6046368299422678e-08 for train data

 Epoch 255
 --------------
Average loss of: 1.3845588458486316e-08 for train data

 Epoch 256
 --------------
Average loss of: 1.458594237368552e-08 for train data

 Epoch 257
 --------------
Average loss of: 1.5730043809214428e-08 for train data

 Epoch 258
 --------------
Average loss of: 1.5238876883836527e-08 for train data

 Epoch 259
 --------------
Average loss of: 1.3904996142086614e-08 for train data

 Epoch 260
 --------------
Average loss of: 1.4148846164263383e-08 for train data

 Epoch 261
 --------------
Average loss of: 1.4950317496589096e-08 for train data

 Epoch 262
 --------------
Average loss of: 1.5269772804376266e-08 for train data

 Epoch 263
 --------------
Average loss of: 1.5440002509043066e-08 for train data
```

```
 Epoch 264
 --------------
Average loss of: 1.4695798854005626e-08 for train data

 Epoch 265
 --------------
Average loss of: 1.561745091077347e-08 for train data

 Epoch 266
 --------------
Average loss of: 1.4377443343787766e-08 for train data

 Epoch 267
 --------------
Average loss of: 1.3709935778059424e-08 for train data

 Epoch 268
 --------------
Adapting learning rate to 3.4171875e-06
Average loss of: 1.5425981128607225e-08 for train data

 Epoch 269
 --------------
Average loss of: 1.3663480837348747e-08 for train data

 Epoch 270
 --------------
Average loss of: 1.481852299937069e-08 for train data

 Epoch 271
 --------------
Average loss of: 1.4515318026704368e-08 for train data

 Epoch 272
 --------------
Average loss of: 1.5028832830688508e-08 for train data

 Epoch 273
 --------------
Average loss of: 1.3548998647202812e-08 for train data

 Epoch 274
 --------------
Average loss of: 1.3549849781772098e-08 for train data

 Epoch 275
 --------------
```

```
Average loss of: 1.2831937470703458e-08 for train data

 Epoch 276
 --------------
Average loss of: 1.2886662422288328e-08 for train data

 Epoch 277
 --------------
Average loss of: 1.3548161564580922e-08 for train data

 Epoch 278
 --------------
Average loss of: 1.3134104957333757e-08 for train data

 Epoch 279
 --------------
Average loss of: 1.3572830398117005e-08 for train data

 Epoch 280
 --------------
Average loss of: 1.4104440265355108e-08 for train data

 Epoch 281
 --------------
Average loss of: 1.4001873944991008e-08 for train data

 Epoch 282
 --------------
Average loss of: 1.3583604852218373e-08 for train data

 Epoch 283
 --------------
Average loss of: 1.3170343737858247e-08 for train data

 Epoch 284
 --------------
Average loss of: 1.2593763321503845e-08 for train data

 Epoch 285
 --------------
Average loss of: 1.4119965997279908e-08 for train data

 Epoch 286
 --------------
Average loss of: 1.3399046576563007e-08 for train data

 Epoch 287
 --------------
```

```
Average loss of: 1.3002022938600056e-08 for train data

 Epoch 288
 --------------
Average loss of: 1.3579290971052704e-08 for train data

 Epoch 289
 --------------
Average loss of: 1.305385013005581e-08 for train data

 Epoch 290
 --------------
Average loss of: 1.3323238262730044e-08 for train data

 Epoch 291
 --------------
Average loss of: 1.2407190982101317e-08 for train data

 Epoch 292
 --------------
Average loss of: 1.2649126284564104e-08 for train data

 Epoch 293
 --------------
Average loss of: 1.2502652268119673e-08 for train data

 Epoch 294
 --------------
Average loss of: 1.439781003118008e-08 for train data

 Epoch 295
 --------------
Average loss of: 1.31039716273527e-08 for train data

 Epoch 296
 --------------
Average loss of: 1.3724014546739226e-08 for train data

 Epoch 297
 --------------
Average loss of: 1.332018992514485e-08 for train data

 Epoch 298
 --------------
Average loss of: 1.3337239808127055e-08 for train data

 Epoch 299
 --------------
```

```
Average loss of: 1.3836114518926588e-08 for train data

 Epoch 300
 --------------
Average loss of: 1.3670450469887579e-08 for train data

 Epoch 301
 --------------
Average loss of: 1.3039874953342394e-08 for train data

 Epoch 302
 --------------
Average loss of: 1.3890083014910426e-08 for train data

 Epoch 303
 --------------
Average loss of: 1.3945222462054223e-08 for train data

 Epoch 304
 --------------
Average loss of: 1.3649813705314908e-08 for train data

 Epoch 305
 --------------
Average loss of: 1.3077651562304426e-08 for train data

 Epoch 306
 --------------
Average loss of: 1.2036814478761472e-08 for train data

 Epoch 307
 --------------
Average loss of: 1.2904518034910445e-08 for train data

 Epoch 308
 --------------
Average loss of: 1.2342287132678241e-08 for train data

 Epoch 309
 --------------
Average loss of: 1.2845482821557907e-08 for train data

 Epoch 310
 --------------
Average loss of: 1.3470098520444183e-08 for train data

 Epoch 311
 --------------
```

```
Average loss of: 1.3790089929036563e-08 for train data

 Epoch 312
 --------------
Average loss of: 1.2622005711116468e-08 for train data

 Epoch 313
 --------------
Average loss of: 1.353432201242843e-08 for train data

 Epoch 314
 --------------
Average loss of: 1.2254808672278028e-08 for train data

 Epoch 315
 --------------
```

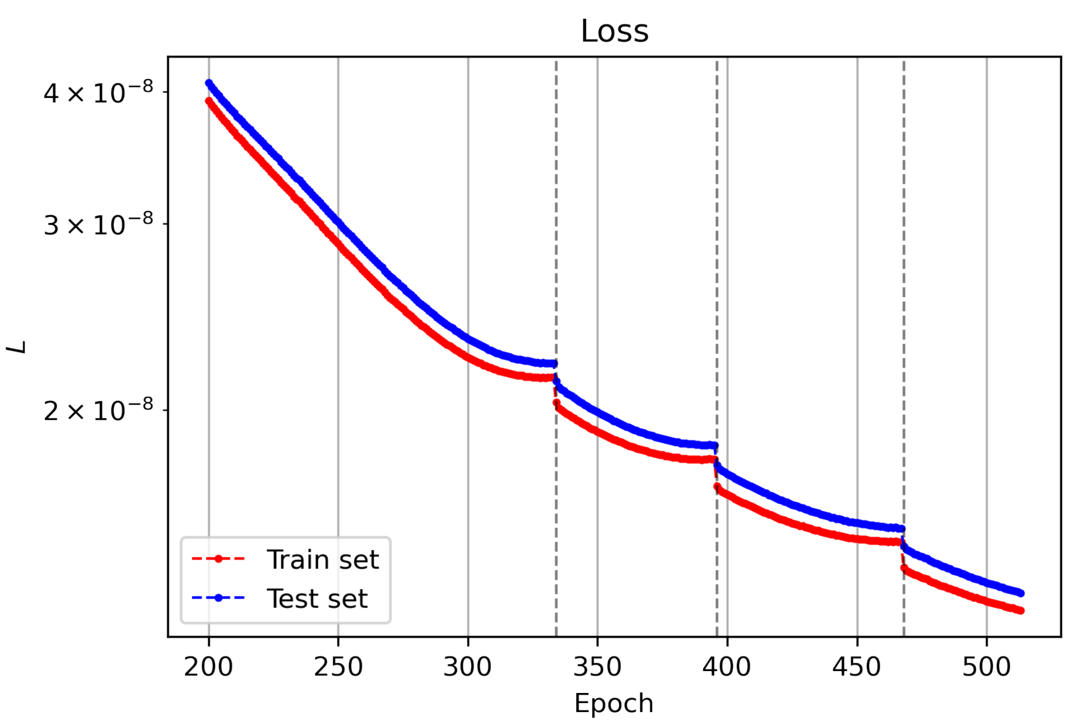
```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Input In [104], in <cell line: 28>()
     39 train_loop(train_dataloader, model, loss_fn, optimizer)
     40 # Test on the training data
---> 41 average_train_loss = test_loop(train_dataloader, model, loss_fn)
     42 train_losses.append(average_train_loss)
     43 # Test on SUBSET of the training data

Input In [101], in test_loop(dataloader, model, loss_fn)
     27 with torch.no_grad():
     28     for X, y in dataloader:
---> 29         pred = model(X)
     30         test_loss += loss_fn(pred, y).item()
     32 average_test_loss = test_loss/num_batches

File D:\Anaconda3\lib\site-packages\torch\nn\modules\module.py:1190, in Module.
↪_call_impl(self, *input, **kwargs)
   1186 # If we don't have any hooks, we want to skip the rest of the logic in
   1187 # this function, and just call forward.
   1188 if not (self._backward_hooks or self._forward_hooks or self.
↪_forward_pre_hooks or _global_backward_hooks
   1189         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1190     return forward_call(*input, **kwargs)
   1191 # Do not call functions when jit is used
   1192 full_backward_hooks, non_full_backward_hooks = [], []

Input In [16], in NeuralNetwork.forward(self, x)
     19 def forward(self, x):
     20     # No flatten needed, as our input and output are 1D?
```

```
     21     #x = self.flatten(x)
---> 22     logits = self.stack(x)
     23     return logits

File D:\Anaconda3\lib\site-packages\torch\nn\modules\module.py:1190, in Module._call_impl(self, *input, **kwargs)
   1186 # If we don't have any hooks, we want to skip the rest of the logic in
   1187 # this function, and just call forward.
   1188 if not (self._backward_hooks or self._forward_hooks or self._forward_pre_hooks or _global_backward_hooks
   1189         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1190     return forward_call(*input, **kwargs)
   1191 # Do not call functions when jit is used
   1192 full_backward_hooks, non_full_backward_hooks = [], []

File D:\Anaconda3\lib\site-packages\torch\nn\modules\container.py:204, in Sequential.forward(self, input)
    202 def forward(self, input):
    203     for module in self:
--> 204         input = module(input)
    205     return input

File D:\Anaconda3\lib\site-packages\torch\nn\modules\module.py:1190, in Module._call_impl(self, *input, **kwargs)
   1186 # If we don't have any hooks, we want to skip the rest of the logic in
   1187 # this function, and just call forward.
   1188 if not (self._backward_hooks or self._forward_hooks or self._forward_pre_hooks or _global_backward_hooks
   1189         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1190     return forward_call(*input, **kwargs)
   1191 # Do not call functions when jit is used
   1192 full_backward_hooks, non_full_backward_hooks = [], []

File D:\Anaconda3\lib\site-packages\torch\nn\modules\activation.py:294, in Sigmoid.forward(self, input)
    293 def forward(self, input: Tensor) -> Tensor:
--> 294     return torch.sigmoid(input)

KeyboardInterrupt:
```

5.1 Results of training

[108]:
```
# Plot it
plt.figure()
lw = 1
ms = 2
```

```python
epochs = [i for i in range(200, 200+314)]
plt.plot(epochs, train_losses, 'o--', color = 'red', label = 'Train set', lw =␣
 ↪lw, ms = ms)
plt.plot(epochs, test_losses, 'o--', color = 'blue', label = "Test set", lw =␣
 ↪lw, ms = ms)
plt.legend()
plt.grid()
plt.xlabel("Epoch")
# xt_step = 20
# xt = [i*xt_step for i in range(len(train_losses)//xt_step+2)]
# plt.xticks(xt)
plt.ylabel(r'$L$')
plt.axhline(0, color = 'black', alpha = 0.7)
plt.title("Loss")
# Plot when we adapted learning rate
for t in adaptation_indices:
    plt.axvline(t+200, linestyle = "--", color = 'black', alpha = 0.5, lw = 1)
plt.yscale('log')
# plt.ylim(10**(-9))
plt.savefig("Plots/NNEOSBv1_part2.pdf", bbox_inches = 'tight')
plt.show()
```



5.2 Estimate the performance of the network

[110]:
```
def L1_norm(predictions, y):
    """Here, predictions and y are arrays for one specific quantity, eg␣
↪pressure. See table 1"""
    return sum(abs(predictions - y))/len(predictions)
```

[111]:
```
def Linfty_norm(predictions, y):
    """Here, predictions and y are arrays for one specific quantity, eg␣
↪pressure. See table 1"""
    return max(abs(predictions - y))
```

Get rho, chi and kappa back out of custom dataset objects:

[112]:
```
# Get features and labels
test_features = test_data.features
test_labels = test_data.labels
test_features[:4]
```

[112]:
```
[tensor([7.2904, 0.7552]),
 tensor([5.5853, 0.9099]),
 tensor([5.5768, 1.4810]),
 tensor([3.9533, 0.8353])]
```

[113]:
```
with torch.no_grad():
    p_hat= np.array([])
    chi_hat = np.array([])
    kappa_hat = np.array([])
    for input_values in test_features:
        prediction = model(input_values)

        p_hat = np.append(p_hat, prediction[0].item())
        chi_hat = np.append(chi_hat, prediction[1].item())
        kappa_hat = np.append(kappa_hat, prediction[2].item())
```

[114]:
```
# Get features as np arrays
rho = np.array([])
eps = np.array([])
for value in test_features:
    rho = np.append(rho, value[0].item())
    eps = np.append(eps, value[1].item())


# Get labels as np arrays
p = np.array([])
chi = np.array([])
kappa = np.array([])
for value in test_labels:
```

```
    p = np.append(p, value[0].item())
    chi = np.append(chi, value[1].item())
    kappa = np.append(kappa, value[2].item())
```

[115]:
```
print(p[0])
print(p_hat[0])
```

```
3.6703102588653564
3.6704049110412598
```

[116]:
```
# Get the errors:
delta_p_L1 = L1_norm(p_hat, p)
delta_chi_L1 = L1_norm(chi_hat, chi)
delta_kappa_L1 = L1_norm(kappa_hat, kappa)

delta_p_Linfty = Linfty_norm(p_hat, p)
delta_chi_Linfty = Linfty_norm(chi_hat, chi)
delta_kappa_Linfty = Linfty_norm(kappa_hat, kappa)
```

[117]:
```
print("Errors for p: %e  with L1 and %e with Linfty" % (delta_p_L1,␣
 ↪delta_p_Linfty) )
print("Errors for chi: %e  with L1 and %e with Linfty" % (delta_chi_L1,␣
 ↪delta_chi_Linfty) )
print("Errors for kappa: %e  with L1 and %e with Linfty" % (delta_kappa_L1,␣
 ↪delta_kappa_Linfty) )
```

```
Errors for p: 9.858099e-05  with L1 and 3.545761e-03 with Linfty
Errors for chi: 6.180509e-05  with L1 and 7.337332e-04 with Linfty
Errors for kappa: 7.404387e-05  with L1 and 9.305319e-04 with Linfty
```

5.3 Save the neural network if desired

[109]:
```
# torch.save(model, 'NNEOSBv1_part2.pth')
```

Testing the loading of models

[27]:
```
# test = torch.load('NNEOSBv0.pth')
```

6 Archive

The following plots the difference between the train and test loss. However, I put it in the archive of this notebook, as the differences are usually very small and hence unimportant for practical aspects.

[ ]:
```
# Get the difference (need np.array)
test_losses_as_array = np.array(test_losses)
train_losses_as_array = np.array(train_losses)
difference = test_losses_as_array - train_losses_as_array
```

```python
# Plot it
plt.figure()
plt.plot(difference[20:], 'o--', color = 'red', label = "Difference", lw = 1,␣
 ↪ms = 1.5)
plt.grid()
plt.xlabel("Epoch")
plt.ylabel(r'$L_{test} - L_{train}$')
plt.axhline(0, color = 'black', alpha = 0.7)
plt.title("Difference in losses")
# Plot when we adapted learning rate
# for t in adaptation_indices:
#     plt.axvline(t, color = 'black', alpha = 0.7)
plt.show()
```

6.1 Line search to find the optimal learning rate parameter

For now, we are **not** using a self-adaptive algorithm, but rather, we are going to do a line search to find the optimal value for the learning rate using a log scale as recommended by the book of Goodfellow.

[63]:
```python
lr_list = [10**(-3), 10**(-4), 10**(-5), 10**(-6), 10**(-7)]
all_train_losses = []
all_test_losses = []

# Hyperparameters:
batch_size = 32
max_number_epochs = 100

for learning_rate in lr_list:
    ## Do a run: for one single learning rate

    # Make a new model, empty train and test loss arrays again
    model = NeuralNetwork().to(device)
    test_losses = []
    train_losses = []

    # Initialize the loss function
    loss_fn = nn.MSELoss()
    optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)

    # ----- Train the model -----
    print(f"Training the model with lr {learning_rate} . . .")
    for t in range(max_number_epochs):
        print(f"\n Epoch {t+1} \n --------------")
        # Train
        train_loop(train_dataloader, model, loss_fn, optimizer)
```

```
        # Test on the training data
        average_train_loss = test_loop(train_dataloader, model, loss_fn)
        train_losses.append(average_train_loss)
        # Test on testing data
        average_test_loss = test_loop(test_dataloader, model, loss_fn)
        test_losses.append(average_test_loss)

    # ----- Train the model -----
    print('Finished a run')
    all_test_losses.append(test_losses)
    all_train_losses.append(train_losses)

    print("Done!")
```

```
Training the model with lr 0.001 . . .

 Epoch 1
 --------------

 Epoch 2
 --------------

 Epoch 3
 --------------

 Epoch 4
 --------------

 Epoch 5
 --------------

 Epoch 6
 --------------

 Epoch 7
 --------------

 Epoch 8
 --------------

 Epoch 9
 --------------

 Epoch 10
 --------------

 Epoch 11
 --------------
```

```
Epoch 12
--------------

Epoch 13
--------------

Epoch 14
--------------

Epoch 15
--------------

Epoch 16
--------------

Epoch 17
--------------

Epoch 18
--------------

Epoch 19
--------------

Epoch 20
--------------

Epoch 21
--------------

Epoch 22
--------------

Epoch 23
--------------

Epoch 24
--------------

Epoch 25
--------------

Epoch 26
--------------

Epoch 27
--------------
```

```
Epoch 28
--------------

Epoch 29
--------------

Epoch 30
--------------

Epoch 31
--------------

Epoch 32
--------------

Epoch 33
--------------

Epoch 34
--------------

Epoch 35
--------------

Epoch 36
--------------

Epoch 37
--------------

Epoch 38
--------------

Epoch 39
--------------

Epoch 40
--------------

Epoch 41
--------------

Epoch 42
--------------

Epoch 43
--------------
```

```
Epoch 44
--------------

Epoch 45
--------------

Epoch 46
--------------

Epoch 47
--------------

Epoch 48
--------------

Epoch 49
--------------

Epoch 50
--------------

Epoch 51
--------------

Epoch 52
--------------

Epoch 53
--------------

Epoch 54
--------------

Epoch 55
--------------

Epoch 56
--------------

Epoch 57
--------------

Epoch 58
--------------

Epoch 59
--------------
```

```
Epoch 60
--------------

Epoch 61
--------------

Epoch 62
--------------

Epoch 63
--------------

Epoch 64
--------------

Epoch 65
--------------

Epoch 66
--------------

Epoch 67
--------------

Epoch 68
--------------

Epoch 69
--------------

Epoch 70
--------------

Epoch 71
--------------

Epoch 72
--------------

Epoch 73
--------------

Epoch 74
--------------

Epoch 75
--------------
```

```
Epoch 76
--------------

Epoch 77
--------------

Epoch 78
--------------

Epoch 79
--------------

Epoch 80
--------------

Epoch 81
--------------

Epoch 82
--------------

Epoch 83
--------------

Epoch 84
--------------

Epoch 85
--------------

Epoch 86
--------------

Epoch 87
--------------

Epoch 88
--------------

Epoch 89
--------------

Epoch 90
--------------

Epoch 91
--------------
```

```
 Epoch 92
 --------------

 Epoch 93
 --------------

 Epoch 94
 --------------

 Epoch 95
 --------------

 Epoch 96
 --------------

 Epoch 97
 --------------

 Epoch 98
 --------------

 Epoch 99
 --------------

 Epoch 100
 --------------
Finished a run
Done!
Training the model with lr 0.0001 . . .

 Epoch 1
 --------------

 Epoch 2
 --------------

 Epoch 3
 --------------

 Epoch 4
 --------------

 Epoch 5
 --------------

 Epoch 6
 --------------
```

```
Epoch 7
--------------

Epoch 8
--------------

Epoch 9
--------------

Epoch 10
--------------

Epoch 11
--------------

Epoch 12
--------------

Epoch 13
--------------

Epoch 14
--------------

Epoch 15
--------------

Epoch 16
--------------

Epoch 17
--------------

Epoch 18
--------------

Epoch 19
--------------

Epoch 20
--------------

Epoch 21
--------------

Epoch 22
--------------
```

```
Epoch 23
--------------

Epoch 24
--------------

Epoch 25
--------------

Epoch 26
--------------

Epoch 27
--------------

Epoch 28
--------------

Epoch 29
--------------

Epoch 30
--------------

Epoch 31
--------------

Epoch 32
--------------

Epoch 33
--------------

Epoch 34
--------------

Epoch 35
--------------

Epoch 36
--------------

Epoch 37
--------------

Epoch 38
--------------
```

```
Epoch 39
--------------

Epoch 40
--------------

Epoch 41
--------------

Epoch 42
--------------

Epoch 43
--------------

Epoch 44
--------------

Epoch 45
--------------

Epoch 46
--------------

Epoch 47
--------------

Epoch 48
--------------

Epoch 49
--------------

Epoch 50
--------------

Epoch 51
--------------

Epoch 52
--------------

Epoch 53
--------------

Epoch 54
--------------
```

```
Epoch 55
--------------

Epoch 56
--------------

Epoch 57
--------------

Epoch 58
--------------

Epoch 59
--------------

Epoch 60
--------------

Epoch 61
--------------

Epoch 62
--------------

Epoch 63
--------------

Epoch 64
--------------

Epoch 65
--------------

Epoch 66
--------------

Epoch 67
--------------

Epoch 68
--------------

Epoch 69
--------------

Epoch 70
--------------
```

```
Epoch 71
--------------

Epoch 72
--------------

Epoch 73
--------------

Epoch 74
--------------

Epoch 75
--------------

Epoch 76
--------------

Epoch 77
--------------

Epoch 78
--------------

Epoch 79
--------------

Epoch 80
--------------

Epoch 81
--------------

Epoch 82
--------------

Epoch 83
--------------

Epoch 84
--------------

Epoch 85
--------------

Epoch 86
--------------
```

```
 Epoch 87
 --------------

 Epoch 88
 --------------

 Epoch 89
 --------------

 Epoch 90
 --------------

 Epoch 91
 --------------

 Epoch 92
 --------------

 Epoch 93
 --------------

 Epoch 94
 --------------

 Epoch 95
 --------------

 Epoch 96
 --------------

 Epoch 97
 --------------

 Epoch 98
 --------------

 Epoch 99
 --------------

 Epoch 100
 --------------
Finished a run
Done!
Training the model with lr 1e-05 . . .

 Epoch 1
 --------------
```

```
Epoch 2
--------------

Epoch 3
--------------

Epoch 4
--------------

Epoch 5
--------------

Epoch 6
--------------

Epoch 7
--------------

Epoch 8
--------------

Epoch 9
--------------

Epoch 10
--------------

Epoch 11
--------------

Epoch 12
--------------

Epoch 13
--------------

Epoch 14
--------------

Epoch 15
--------------

Epoch 16
--------------

Epoch 17
--------------
```

```
Epoch 18
--------------

Epoch 19
--------------

Epoch 20
--------------

Epoch 21
--------------

Epoch 22
--------------

Epoch 23
--------------

Epoch 24
--------------

Epoch 25
--------------

Epoch 26
--------------

Epoch 27
--------------

Epoch 28
--------------

Epoch 29
--------------

Epoch 30
--------------

Epoch 31
--------------

Epoch 32
--------------

Epoch 33
--------------
```

```
Epoch 34
--------------

Epoch 35
--------------

Epoch 36
--------------

Epoch 37
--------------

Epoch 38
--------------

Epoch 39
--------------

Epoch 40
--------------

Epoch 41
--------------

Epoch 42
--------------

Epoch 43
--------------

Epoch 44
--------------

Epoch 45
--------------

Epoch 46
--------------

Epoch 47
--------------

Epoch 48
--------------

Epoch 49
--------------
```

```
Epoch 50
--------------

Epoch 51
--------------

Epoch 52
--------------

Epoch 53
--------------

Epoch 54
--------------

Epoch 55
--------------

Epoch 56
--------------

Epoch 57
--------------

Epoch 58
--------------

Epoch 59
--------------

Epoch 60
--------------

Epoch 61
--------------

Epoch 62
--------------

Epoch 63
--------------

Epoch 64
--------------

Epoch 65
--------------
```

```
Epoch 66
--------------

Epoch 67
--------------

Epoch 68
--------------

Epoch 69
--------------

Epoch 70
--------------

Epoch 71
--------------

Epoch 72
--------------

Epoch 73
--------------

Epoch 74
--------------

Epoch 75
--------------

Epoch 76
--------------

Epoch 77
--------------

Epoch 78
--------------

Epoch 79
--------------

Epoch 80
--------------

Epoch 81
--------------
```

```
Epoch 82
--------------

Epoch 83
--------------

Epoch 84
--------------

Epoch 85
--------------

Epoch 86
--------------

Epoch 87
--------------

Epoch 88
--------------

Epoch 89
--------------

Epoch 90
--------------

Epoch 91
--------------

Epoch 92
--------------

Epoch 93
--------------

Epoch 94
--------------

Epoch 95
--------------

Epoch 96
--------------

Epoch 97
--------------
```

```
 Epoch 98
 --------------

 Epoch 99
 --------------

 Epoch 100
 --------------
Finished a run
Done!
Training the model with lr 1e-06 . . .

 Epoch 1
 --------------

 Epoch 2
 --------------

 Epoch 3
 --------------

 Epoch 4
 --------------

 Epoch 5
 --------------

 Epoch 6
 --------------

 Epoch 7
 --------------

 Epoch 8
 --------------

 Epoch 9
 --------------

 Epoch 10
 --------------

 Epoch 11
 --------------

 Epoch 12
 --------------
```

```
Epoch 13
--------------

Epoch 14
--------------

Epoch 15
--------------

Epoch 16
--------------

Epoch 17
--------------

Epoch 18
--------------

Epoch 19
--------------

Epoch 20
--------------

Epoch 21
--------------

Epoch 22
--------------

Epoch 23
--------------

Epoch 24
--------------

Epoch 25
--------------

Epoch 26
--------------

Epoch 27
--------------

Epoch 28
--------------
```

```
Epoch 29
--------------

Epoch 30
--------------

Epoch 31
--------------

Epoch 32
--------------

Epoch 33
--------------

Epoch 34
--------------

Epoch 35
--------------

Epoch 36
--------------

Epoch 37
--------------

Epoch 38
--------------

Epoch 39
--------------

Epoch 40
--------------

Epoch 41
--------------

Epoch 42
--------------

Epoch 43
--------------

Epoch 44
--------------
```

```
Epoch 45
--------------

Epoch 46
--------------

Epoch 47
--------------

Epoch 48
--------------

Epoch 49
--------------

Epoch 50
--------------

Epoch 51
--------------

Epoch 52
--------------

Epoch 53
--------------

Epoch 54
--------------

Epoch 55
--------------

Epoch 56
--------------

Epoch 57
--------------

Epoch 58
--------------

Epoch 59
--------------

Epoch 60
--------------
```

```
Epoch 61
--------------

Epoch 62
--------------

Epoch 63
--------------

Epoch 64
--------------

Epoch 65
--------------

Epoch 66
--------------

Epoch 67
--------------

Epoch 68
--------------

Epoch 69
--------------

Epoch 70
--------------

Epoch 71
--------------

Epoch 72
--------------

Epoch 73
--------------

Epoch 74
--------------

Epoch 75
--------------

Epoch 76
--------------
```

```
Epoch 77
--------------

Epoch 78
--------------

Epoch 79
--------------

Epoch 80
--------------

Epoch 81
--------------

Epoch 82
--------------

Epoch 83
--------------

Epoch 84
--------------

Epoch 85
--------------

Epoch 86
--------------

Epoch 87
--------------

Epoch 88
--------------

Epoch 89
--------------

Epoch 90
--------------

Epoch 91
--------------

Epoch 92
--------------
```

```
 Epoch 93
 --------------

 Epoch 94
 --------------

 Epoch 95
 --------------

 Epoch 96
 --------------

 Epoch 97
 --------------

 Epoch 98
 --------------

 Epoch 99
 --------------

 Epoch 100
 --------------
Finished a run
Done!
Training the model with lr 1e-07 . . .

 Epoch 1
 --------------

 Epoch 2
 --------------

 Epoch 3
 --------------

 Epoch 4
 --------------

 Epoch 5
 --------------

 Epoch 6
 --------------

 Epoch 7
 --------------
```

```
Epoch 8
--------------

Epoch 9
--------------

Epoch 10
--------------

Epoch 11
--------------

Epoch 12
--------------

Epoch 13
--------------

Epoch 14
--------------

Epoch 15
--------------

Epoch 16
--------------

Epoch 17
--------------

Epoch 18
--------------

Epoch 19
--------------

Epoch 20
--------------

Epoch 21
--------------

Epoch 22
--------------

Epoch 23
--------------
```

```
Epoch 24
--------------

Epoch 25
--------------

Epoch 26
--------------

Epoch 27
--------------

Epoch 28
--------------

Epoch 29
--------------

Epoch 30
--------------

Epoch 31
--------------

Epoch 32
--------------

Epoch 33
--------------

Epoch 34
--------------

Epoch 35
--------------

Epoch 36
--------------

Epoch 37
--------------

Epoch 38
--------------

Epoch 39
--------------
```

```
Epoch 40
--------------

Epoch 41
--------------

Epoch 42
--------------

Epoch 43
--------------

Epoch 44
--------------

Epoch 45
--------------

Epoch 46
--------------

Epoch 47
--------------

Epoch 48
--------------

Epoch 49
--------------

Epoch 50
--------------

Epoch 51
--------------

Epoch 52
--------------

Epoch 53
--------------

Epoch 54
--------------

Epoch 55
--------------
```

```
Epoch 56
--------------

Epoch 57
--------------

Epoch 58
--------------

Epoch 59
--------------

Epoch 60
--------------

Epoch 61
--------------

Epoch 62
--------------

Epoch 63
--------------

Epoch 64
--------------

Epoch 65
--------------

Epoch 66
--------------

Epoch 67
--------------

Epoch 68
--------------

Epoch 69
--------------

Epoch 70
--------------

Epoch 71
--------------
```

```
Epoch 72
--------------

Epoch 73
--------------

Epoch 74
--------------

Epoch 75
--------------

Epoch 76
--------------

Epoch 77
--------------

Epoch 78
--------------

Epoch 79
--------------

Epoch 80
--------------

Epoch 81
--------------

Epoch 82
--------------

Epoch 83
--------------

Epoch 84
--------------

Epoch 85
--------------

Epoch 86
--------------

Epoch 87
--------------
```

```
 Epoch 88
 --------------

 Epoch 89
 --------------

 Epoch 90
 --------------

 Epoch 91
 --------------

 Epoch 92
 --------------

 Epoch 93
 --------------

 Epoch 94
 --------------

 Epoch 95
 --------------

 Epoch 96
 --------------

 Epoch 97
 --------------

 Epoch 98
 --------------

 Epoch 99
 --------------

 Epoch 100
 --------------
Finished a run
Done!
```