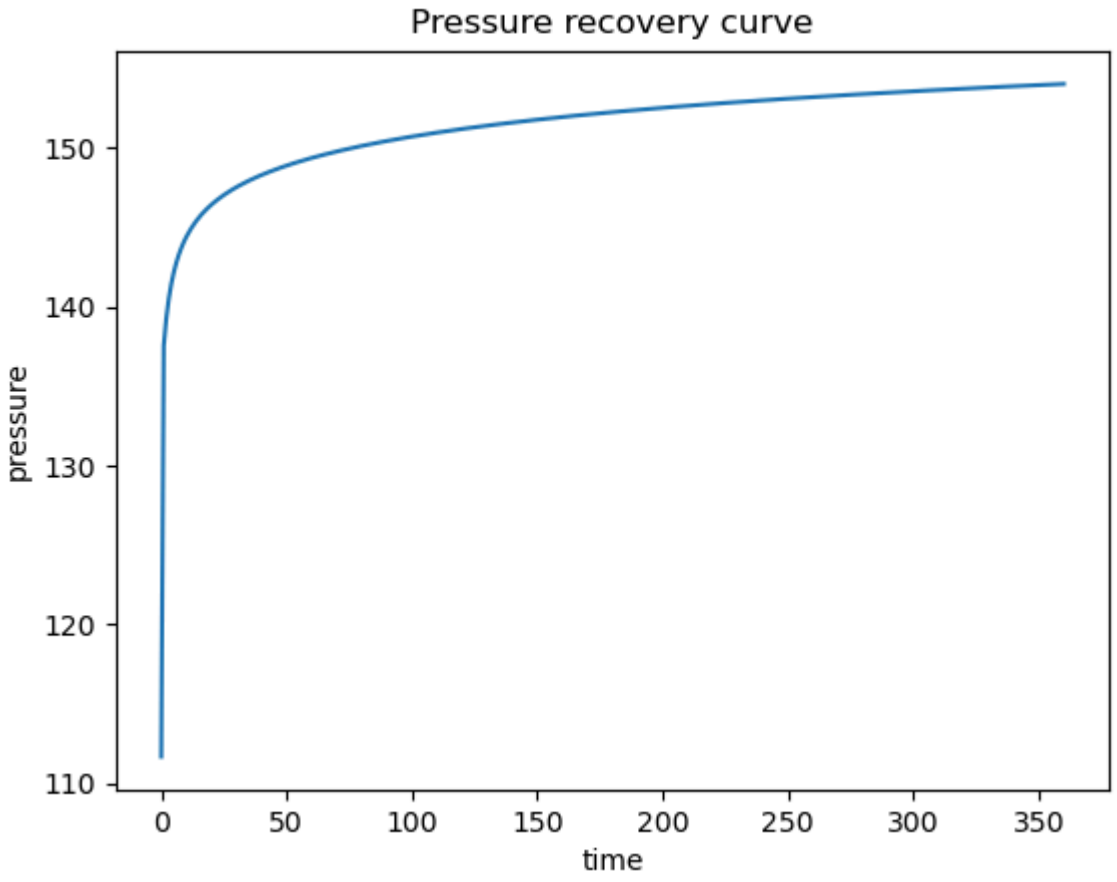
Project report

1. Introduction

Imagine, you have permeability map, which provide us opportunity for numerical calculations of pressure recovery curve.

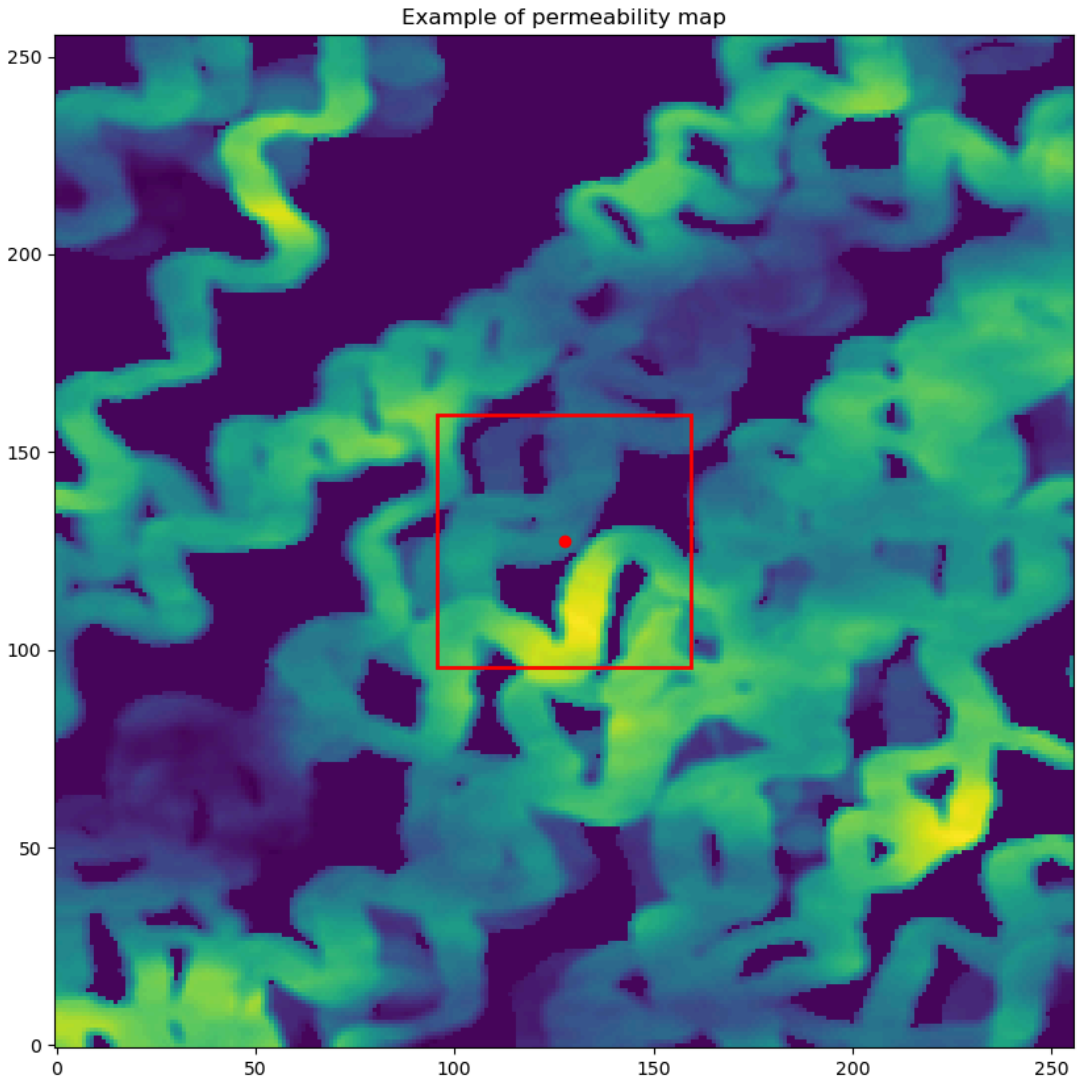


How a fluid's pressure changes through time? We can describe it using pressure recovery curve.
Their primary purpose is to quantify how much pressure is regained downstream of a constriction or flow disturbance, which has direct implications on the efficiency and safety of various fluid systems.

Permeability is one of the property which determines pressure response in a well. Given that the other characteristics were the same for each scenario, it is assumed that the mapping from a permeability field and pressure build up curve could be learned via neural network.

The data was generated in a numerical solver where all properties were the same (initial reservoir pressure, porosity, well characteristics, boundary conditions, reservoir thickness, e.t.c), except the permeability field in each scenario. Initially, a production well operated with constant production rate q which causes reservoir pressure decrease and after that the well was shut down and pressure build up curve was obtained.

For the model I will consider this red square part for the input, where red dot is exactly well placement.
**



2. Methodology

For this project I consider only MLP, but of course in future works it could be more complex architectures.

```
MLP(
  (layers): Sequential(
    (0): Linear(in_features=64*64, out_features=64, bias=True)
    (1): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): Tanh()
    (3): Linear(in_features=64, out_features=64, bias=True)
    (4): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True,
```

```
track_running_stats=True)
    (5): Tanh()
    (6): Linear(in_features=64, out_features=361, bias=True)
  )
)
```

Through several iterations I decided to use Tanh as activation functions, because previous steps with ReLU was worse than last type of architecture.

Losses

I implement 2 types of losses for the baseline model:

1. MSE for the output pressure vector
2. Monotonicity loss

$$L = MSE(p, \hat{p}) + MON\_loss(\hat{p}) + PI\_loss(q, \hat{q})$$

$$q_f = q + \frac{24 C_s}{B} \frac{dP}{dt}$$

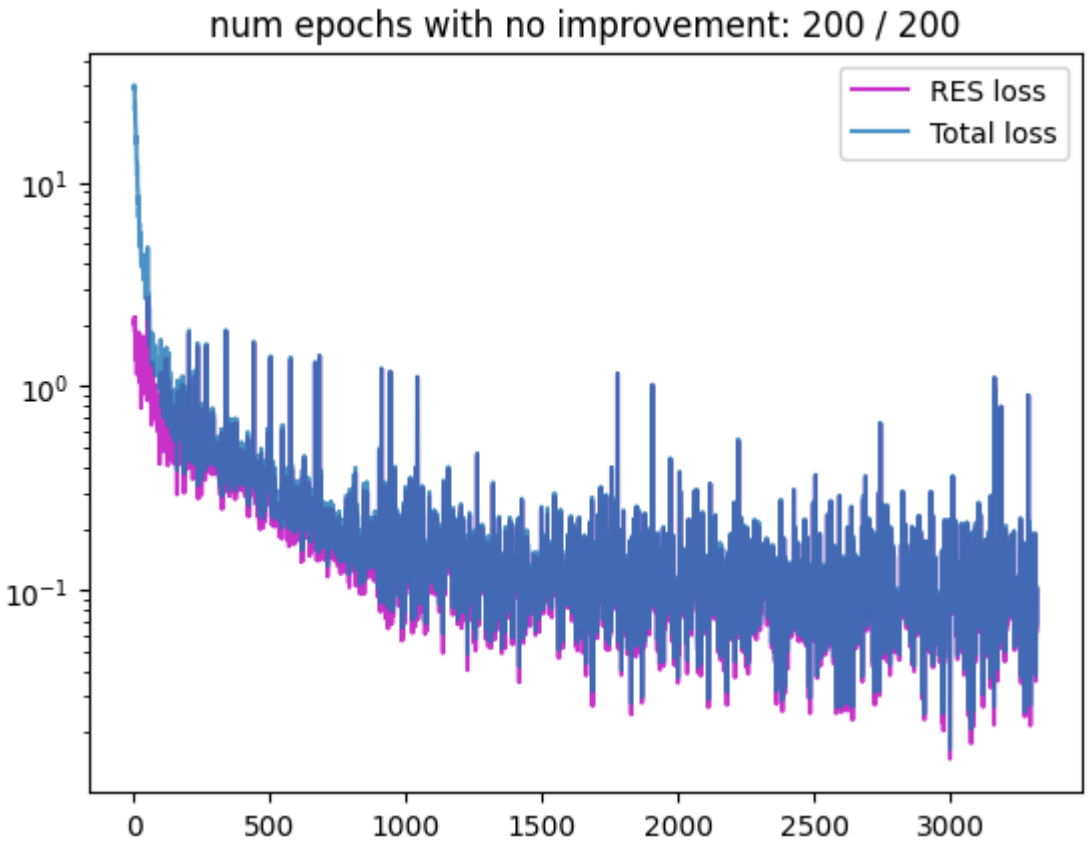
$q_f:$ inflow from the formation under surface conditions, $[\mathrm{m}^3/\mathrm{day}]$
$q:$ surface flow rate, $[\mathrm{m}^3/\mathrm{day}]$
$B:$ volumetric coefficient
$t:$ time, $[\mathrm{hours}]$
$P:$ pressure, $[\mathrm{atm}]$
$C_s:$ coefficient of influence of the wellbore volume, $[\mathrm{m}^3/\mathrm{atm}]$

Physical loss

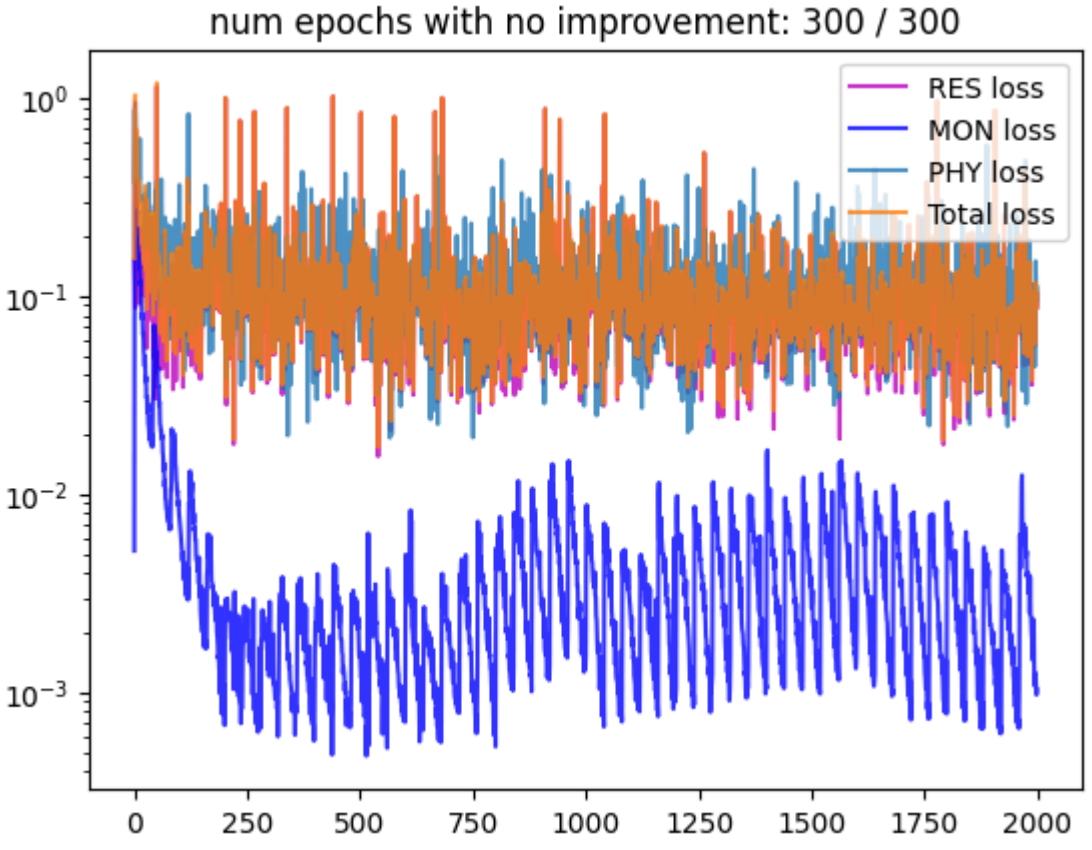
Despite the fact that the well is shut down and surface flow (production rate) equals to zero q=0, the fluid flow from a reservoir to the bottom part of the well occurs anyway q_f. It is because the bottomhole pressure (pressure in the bottom part of the well) is not equal to the reservoir pressure instantly (the system is compressible). I tried to implement the effect in the form of physics-informed loss in neural network training.
As B and Cs are dependent on pressure, the equation implicitly takes into account the PVT properties of oil.

3. Results and analysis

Graph describe us baseline model with 2 losses (the first one is mse for curve values, the second is monotonicity loss).



This graph shows the losses picture with Physical loss



Also approach with checking loss on validation dataset was provided to stop the training after n_patience step with higher results on loss.

Metrics on test sample

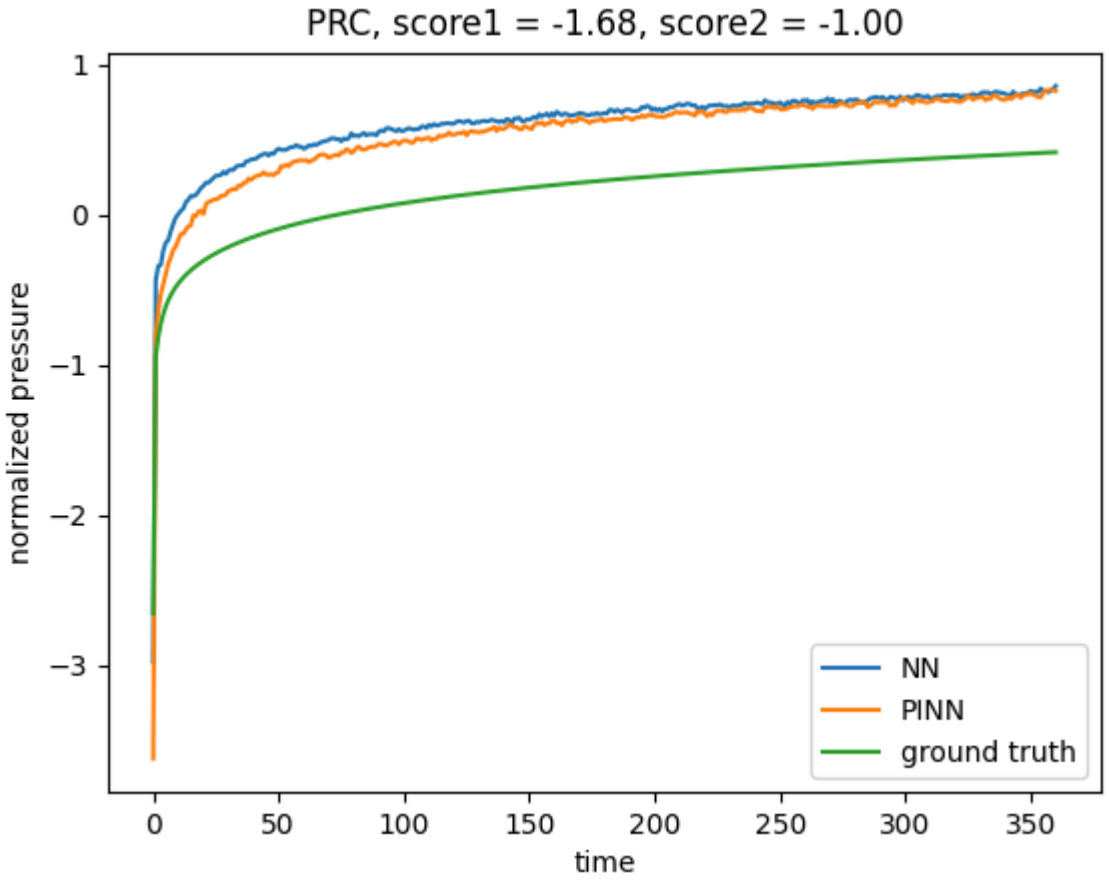
As we can see, models provided us results, where PINN achieved a little bit lower results,. To my mind, the problem was in selecting hyperparameters. The next step is create grid search for it or provide some algorithm with automatic selection, because it consume much more time and compute for it.

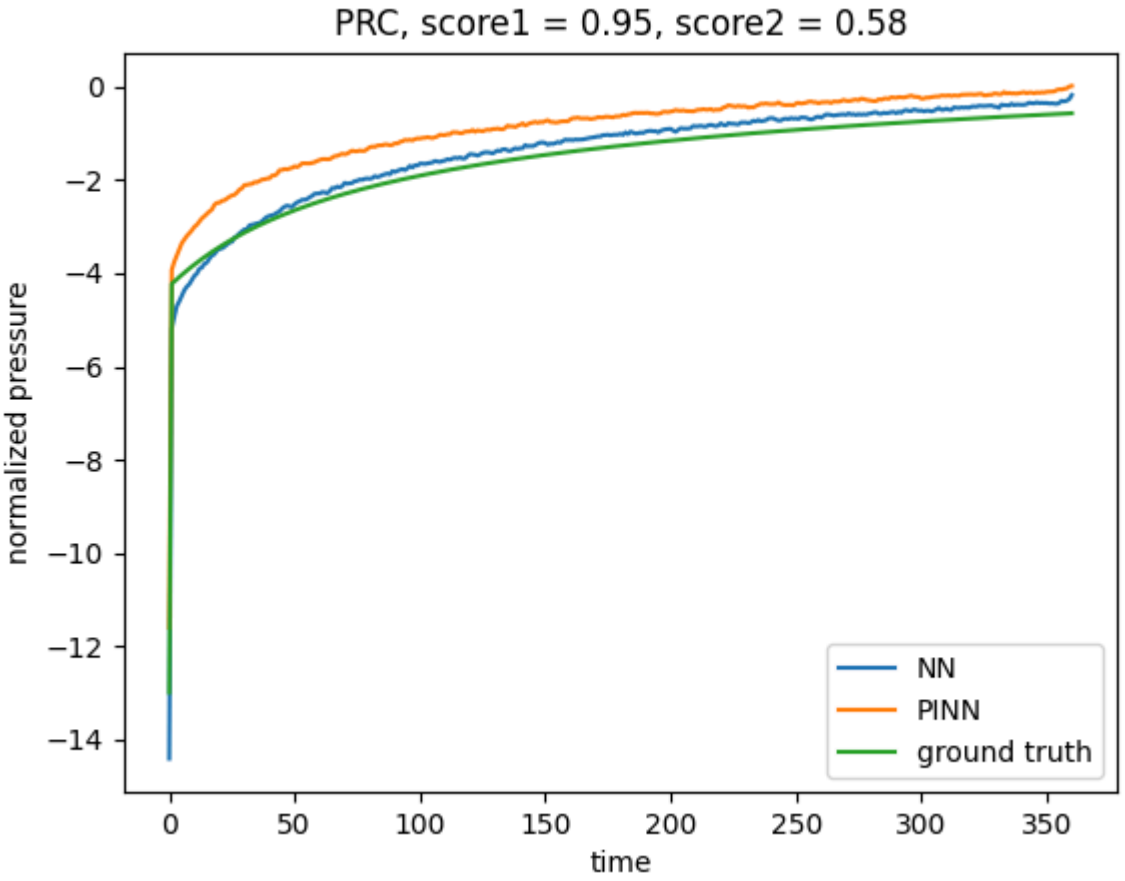
Tablet with results

	NN	PINN
r2_mean	0.788	0.681
rmse_mean	0.191	0.238

Examples of curves

Here is 2 first test samples and predictions on it.





4. Discussion and conclusion

For this project was provided a solid proof of concept, which provide us understanding for the future steps for the work on this task.

Next steps:

- Search hyperparameters for the model (batch_size, coeff for loss, n_patience, etc.)
- Try different architectures
  CNN (motivation for considering maps like images)
  RNN (step by step generation for curve)
- Augmentation techniques (maps reflection)
- Predict Bourdet pressure derivative instead of pressure

Appendix: training configs for NN and PINN

```
# Tanh, with batch norm

mlp = trainer.train(
    mlp,
    n_epochs=3000,
    n_patience=200,
    learning_rate=5e-3,
    batch_size=20,
```

```
    lambda_mon = 25 # coeff for monotonicity loss
)


mlp_phy = trainer.train_physics(
    mlp_phy,
    n_epochs=3000,
    n_patience=300,
    learning_rate=1e-2,
    batch_size=20,
    lambda_mon = 50, # coeff for monotonicity loss
    lambda_phy=2 # coeff for physical loss
)
```