

1. We write a Python program to approximate the function

$$f(x) = \log(1+x) \exp(\sin(x))$$

on the interval $I = [0, 10]$.

- a) Plot the function on the Interval $[0, 10]$.

hint1a.py shows how we can use matplotlib to plot points

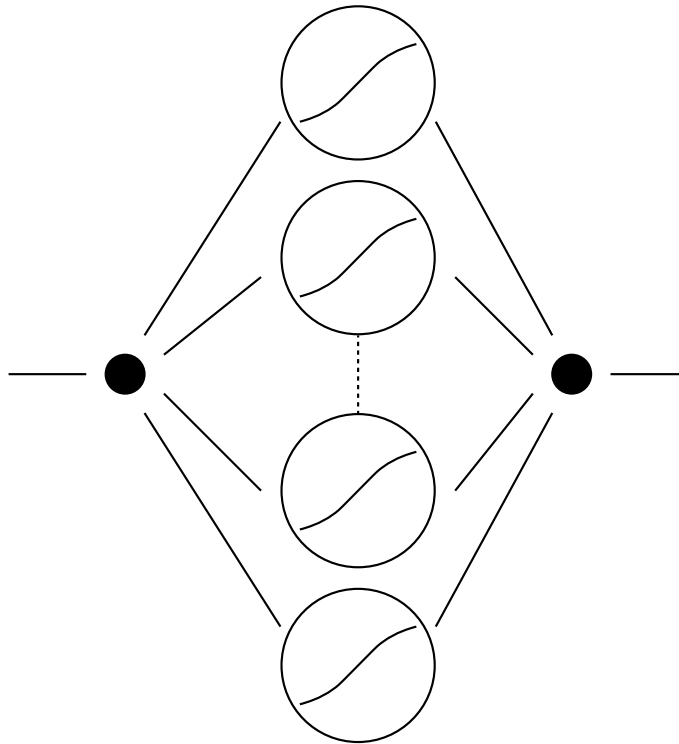
solution1a.py is the complete solution

- b) Train a neural network with 1 input, one hidden layer with 5 neurons (sigmoid activation) and 1 output (no activation function) to minimize

$$\frac{1}{N} \sum_{i=1}^N \|y_i - f[\mathbf{P}](x_i)\|^2 \rightarrow \min$$

with the network

$$f[\mathbf{P}](x) = \mathbf{w}_2 \cdot \sigma(\mathbf{w}_1 x + \mathbf{b}_1) + b_2$$



For training, use 20 random data points x_i and $y_i = f(x_i)$ from the interval $[0, 10]$.

Plot the function $f(x)$ in and the network $f[\mathbf{P}](x)$ as a function on $I = [0, 10]$ and also the 20 data points.

hint1b.py constructs the network and shows how the training works

solution1b.py is the complete solution

- c) Experiment with the program. Some suggestions:

- Change the number of training points
- Change the size of the network. Add more neurons, add more layers
- You can use different activation functions. Ask google for *pytorch activation functions*.
- Plot the resulting network on the larger interval $[0, 20]$. Try to explain the shape of the neural network function $f[\mathbf{P}](x)$ for $x > 10$.

2. We solve the Lorenz system

$$\mathbf{x}'(t) = \begin{pmatrix} -10x_1(t) + 10x_2(t) \\ 28x_1(t) - x_2(t) - x_1(t)x_3(t) \\ x_1(t)x_2(t) - \frac{8}{3}x_3(t) \end{pmatrix}, \quad \mathbf{x}(0) = \begin{pmatrix} -10 \\ -10 \\ 20 \end{pmatrix}$$

a) Approximate the ODE on the interval $I = [0, 10]$ using Heun's method

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{h}{2}f(\mathbf{x}_n) + \frac{h}{2}f(\mathbf{x}_n + hf(\mathbf{x}_n))$$

with the step size $h = 0.01$.

Plot the solution as a parametric 3d-plot $t \mapsto (x_1(t), x_2(t), x_3(t))$ and also plot three separate plot $(t, x_1(t))$, $(t, x_2(t))$, $(t, x_3(t))$.

hint2a.py implements the ordinary differential equation and Heun's method
solution2a.py is the complete solution with both types of plots

b) We prepare the data generation for the neural network solution. Write a Python-program that generates $N_{train} = 10$ random data pairs

$$\mathbf{x}_i \mapsto \mathbf{y}_i,$$

where \mathbf{x}_i are chosen randomly in the interval

$$\mathbf{x}_i = (\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, \mathbf{x}_{i,3}), \quad \mathbf{x}_{i,1} \in [-20, 20], \quad \mathbf{x}_{i,2} \in [-20, 20], \quad \mathbf{x}_{i,3} \in [0, 40].$$

\mathbf{y}_i is the solution to the Lorenz system after the trainig-interval $t_{train} = 0.1$, i.e.

$$\mathbf{y}_i := \bar{\mathbf{x}}_i(0.1), \quad \bar{\mathbf{x}}'_i(t) = f(t, \bar{\mathbf{x}}_i(t)), \quad \bar{\mathbf{x}}_i(0) = \mathbf{x}_i.$$

The solution $\bar{\mathbf{x}}_i(t)$ should be approximated with Heun's method using the step size $h = 0.001$.

For each of these random data points plot \mathbf{x}_i and \mathbf{y}_i as points but also the complete solution $\bar{\mathbf{x}}_i(t)$.

hint2b.py creates the random initial data \mathbf{x}_i and prepares the main loop for computing the solutions $\bar{\mathbf{x}}_i(t)$.

solution2b.py is the complete solution

c) We train a neural network that is learning the Lorenz system:

- (i) Create $N_{\text{train}} = 1000$ data pairs $\mathbf{x}_i \mapsto \mathbf{y}_i$ such as done in b)
- (ii) Generate a Neural network. We suggest to use 3 inputs, and 3 layer with 10 neurons each (sigmoid activation), finally, 3 outputs.
- (iii) Train the network to learn the data points generated above:

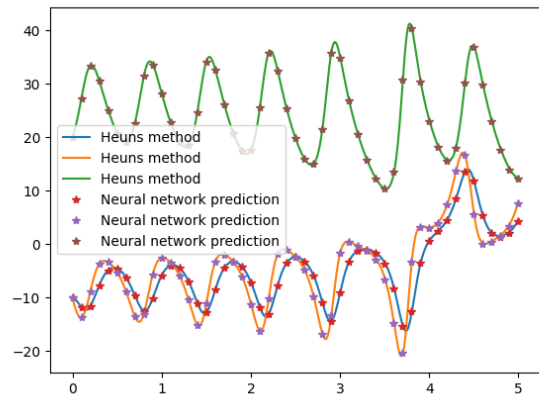
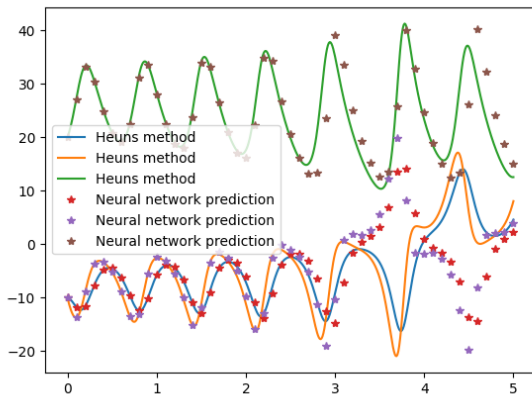
$$\frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} \|\mathbf{y}_i - f[\mathbf{P}](\mathbf{x}_i)\|^2 \rightarrow \min$$

For the initial value

$$\mathbf{x}(0) = \begin{pmatrix} -10 \\ -10 \\ 20 \end{pmatrix}$$

compute the solution with Heun's method ($h = 0.001$) and with the neural network $\mathbf{x}_{n+1} = f[\mathbf{P}](\mathbf{x}_n)$. Plot both solutions.

Further steps: If you have done everything you the result might look like this (on the left)



- Try to improve the result by using more data points
- Try to improve the result by making the network better
- What of the abover is more important? Larger network, more data or maybe both? I get the result on the right with 10 000 data points and 50 instead of 10 neurons in each layer.
- How far can you get? Increase the time interval from $[0, 5]$ to a larger interval.

hint2c.py does steps c) (1) - (3).

solution2c.py is the complete solution