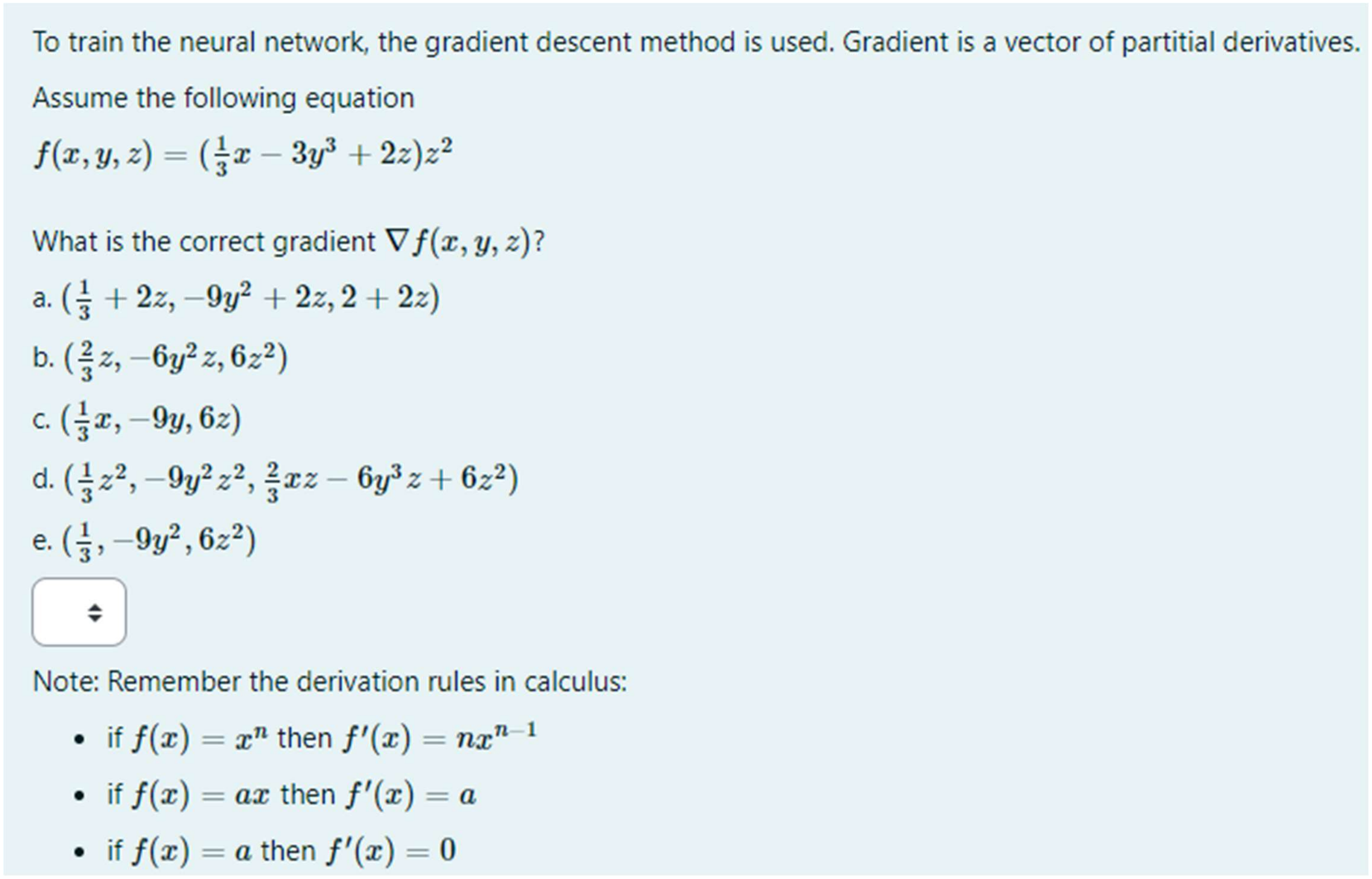
Exercise 6.1



To train the neural network, the gradient descent method is used. Gradient is a vector of partitial derivatives.

Assume the following equation

$f(x, y, z) = (\frac{1}{3}x - 3y^3 + 2z)z^2$

What is the correct gradient $\nabla f(x, y, z)$?

a. $(\frac{1}{3} + 2z, -9y^2 + 2z, 2 + 2z)$

b. $(\frac{2}{3}z, -6y^2 z, 6z^2)$

c. $(\frac{1}{3}x, -9y, 6z)$

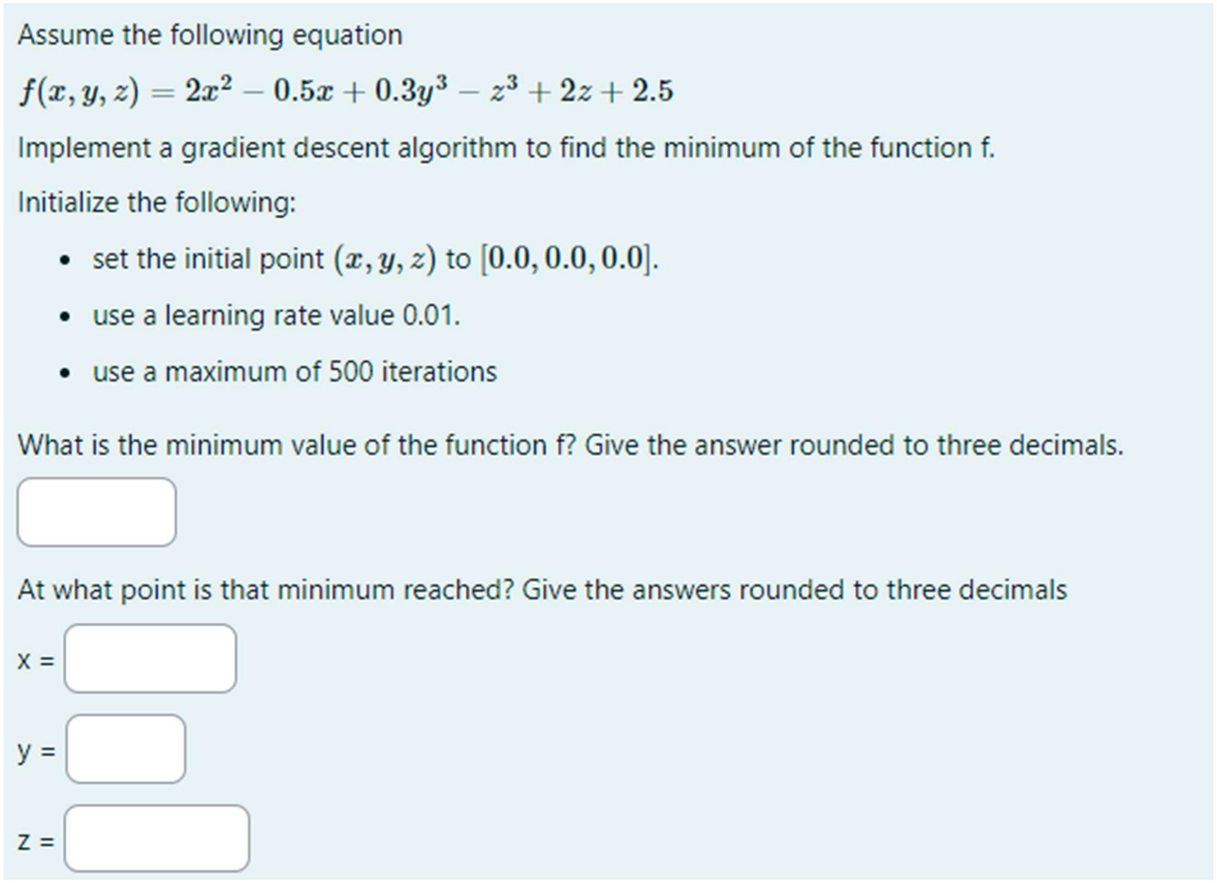
d. $(\frac{1}{3}z^2, -9y^2 z^2, \frac{2}{3}xz - 6y^3 z + 6z^2)$

e. $(\frac{1}{3}, -9y^2, 6z^2)$

Note: Remember the derivation rules in calculus:

- if $f(x) = x^n$ then $f'(x) = nx^{n-1}$
- if $f(x) = ax$ then $f'(x) = a$
- if $f(x) = a$ then $f'(x) = 0$

Exercise 6.2



Assume the following equation

$$f(x, y, z) = 2x^2 - 0.5x + 0.3y^3 - z^3 + 2z + 2.5$$

Implement a gradient descent algorithm to find the minimum of the function f.

Initialize the following:

- set the initial point (x, y, z) to $[0.0, 0.0, 0.0]$.
- use a learning rate value 0.01.
- use a maximum of 500 iterations

What is the minimum value of the function f? Give the answer rounded to three decimals.

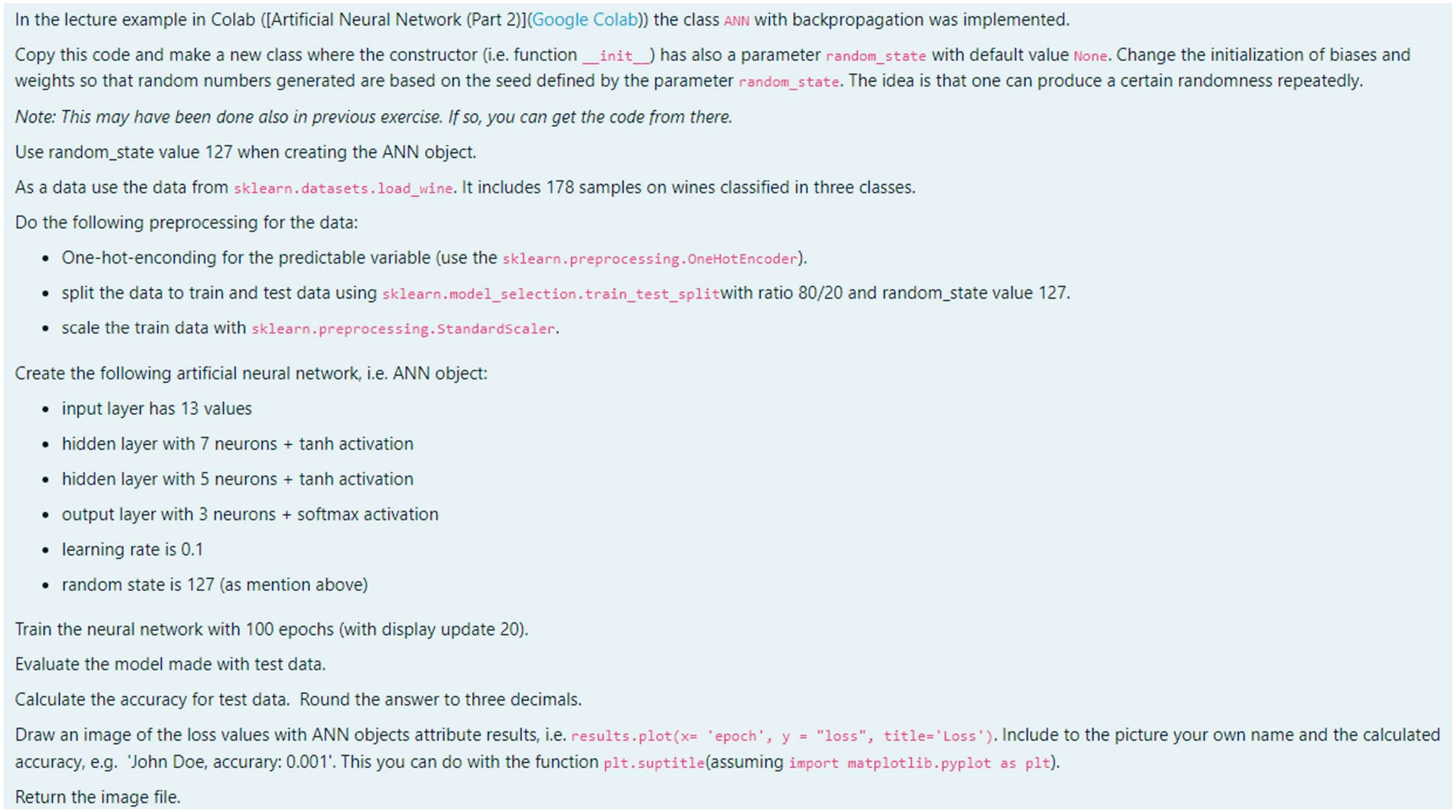
At what point is that minimum reached? Give the answers rounded to three decimals

x =

y =

z =

Exercise 6.3



In the lecture example in Colab ([Artificial Neural Network (Part 2)](Google Colab)) the class `ANN` with backpropagation was implemented.

Copy this code and make a new class where the constructor (i.e. function `__init__`) has also a parameter `random_state` with default value `None`. Change the initialization of biases and weights so that random numbers generated are based on the seed defined by the parameter `random_state`. The idea is that one can produce a certain randomness repeatedly.

Note: This may have been done also in previous exercise. If so, you can get the code from there.

Use random_state value 127 when creating the ANN object.

As a data use the data from `sklearn.datasets.load_wine`. It includes 178 samples on wines classified in three classes.

Do the following preprocessing for the data:

- One-hot-enconding for the predictable variable (use the `sklearn.preprocessing.OneHotEncoder`).
- split the data to train and test data using `sklearn.model_selection.train_test_split`with ratio 80/20 and random_state value 127.
- scale the train data with `sklearn.preprocessing.StandardScaler`.

Create the following artificial neural network, i.e. ANN object:

- input layer has 13 values
- hidden layer with 7 neurons + tanh activation
- hidden layer with 5 neurons + tanh activation
- output layer with 3 neurons + softmax activation
- learning rate is 0.1
- random state is 127 (as mention above)

Train the neural network with 100 epochs (with display update 20).

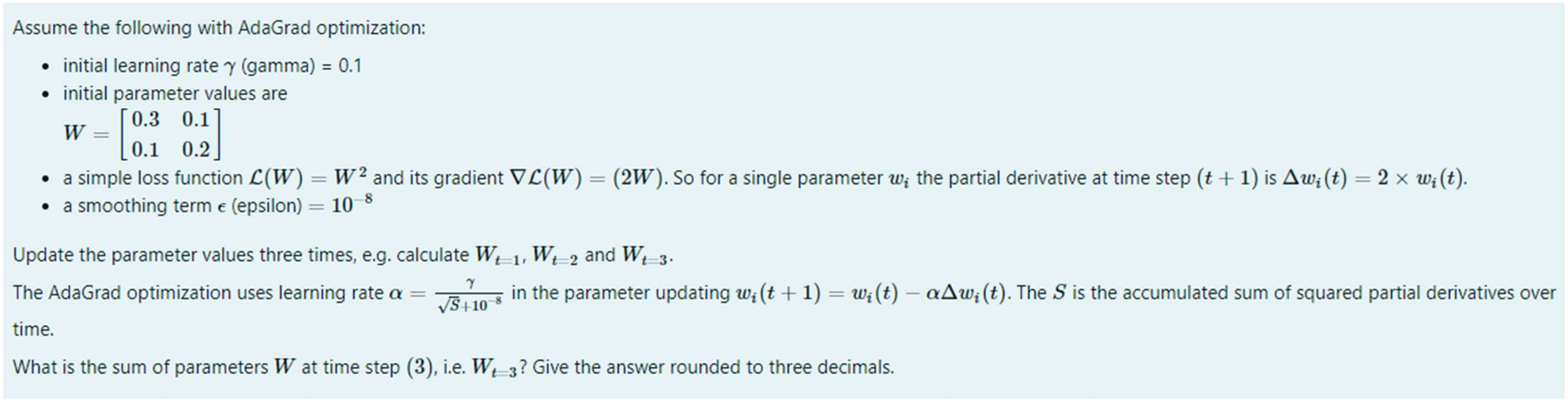
Evaluate the model made with test data.

Calculate the accuracy for test data. Round the answer to three decimals.

Draw an image of the loss values with ANN objects attribute results, i.e. `results.plot(x= 'epoch', y = "loss", title='Loss')`. Include to the picture your own name and the calculated accuracy, e.g. 'John Doe, accurary: 0.001'. This you can do with the function `plt.suptitle`(assuming `import matplotlib.pyplot as plt`).

Return the image file.

Exercise 6.4



Assume the following with AdaGrad optimization:

- initial learning rate γ (gamma) = 0.1
- initial parameter values are
  $W = \begin{bmatrix} 0.3 & 0.1 \\ 0.1 & 0.2 \end{bmatrix}$
- a simple loss function $\mathcal{L}(W) = W^2$ and its gradient $\nabla\mathcal{L}(W) = (2W)$. So for a single parameter w_i the partial derivative at time step $(t+1)$ is $\Delta w_i(t) = 2 \times w_i(t)$.
- a smoothing term ϵ (epsilon) $= 10^{-8}$

Update the parameter values three times, e.g. calculate $W_{t=1}$, $W_{t=2}$ and $W_{t=3}$.

The AdaGrad optimization uses learning rate $\alpha = \frac{\gamma}{\sqrt{S}+10^{-8}}$ in the parameter updating $w_i(t+1) = w_i(t) - \alpha\Delta w_i(t)$. The S is the accumulated sum of squared partial derivatives over time.

What is the sum of parameters W at time step (3), i.e. $W_{t=3}$? Give the answer rounded to three decimals.