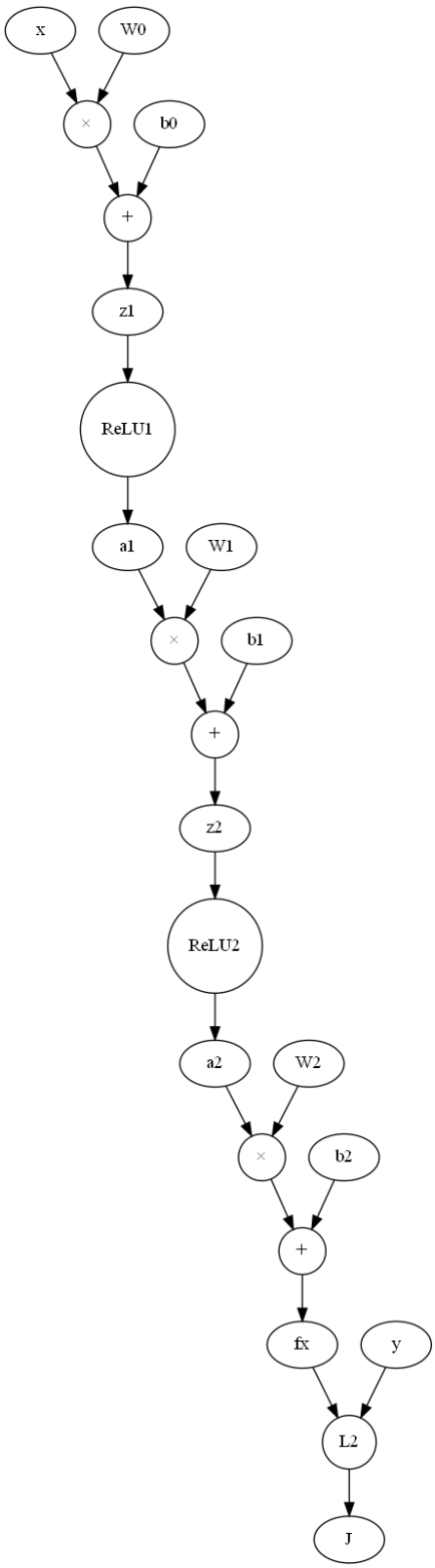
Problem 1

The computing graph of $J(\theta, \boldsymbol{x})$ depending on (x, y) and $\theta = (\boldsymbol{W}_0, \boldsymbol{b}_0, \boldsymbol{W}_1, \boldsymbol{b}_1, \boldsymbol{W}_2, \boldsymbol{b}_2)$ is plotted as below:



Problem 2

In the graph in Prob.1, the circle nodes are operators, and the nodes with shape as ellipse are parameters. Herein we compute only the partial derivatives of the parameter nodes.

The derivative of ReLU function can be expressed as

$$\frac{\partial \mathrm{ReLU}(x)}{\partial x} = \begin{cases} 0 & x \leq 0, \\ 1 & x > 0 \end{cases} \tag{1}$$

And the partial derivatives lists upstream from the last node are shown below:

$$\begin{aligned} \frac{\partial J}{\partial y} &= -(f(\boldsymbol{x}) - \boldsymbol{y}) \\ \frac{\partial J}{\partial f(\boldsymbol{x})} &= (f(\boldsymbol{x}) - \boldsymbol{y}) \end{aligned} \tag{2}$$

$$\begin{aligned} \frac{\partial J}{\partial \boldsymbol{b}_2} &= 1 \cdot \frac{\partial J}{\partial f(\boldsymbol{x})} \\ \frac{\partial J}{\partial \boldsymbol{a}_2} &= \boldsymbol{W}_2^T \cdot \frac{\partial J}{\partial f(\boldsymbol{x})} \\ \frac{\partial J}{\partial \boldsymbol{W}_2} &= \boldsymbol{a}_2 \cdot \frac{\partial J}{\partial f(\boldsymbol{x})} \end{aligned} \tag{3}$$

$$\frac{\partial J}{\partial \boldsymbol{z}_2} = \frac{\partial\, \mathrm{ReLu}(\boldsymbol{z}_2)}{\partial \boldsymbol{z}_2} \cdot \frac{\partial J}{\partial \boldsymbol{a}_2} \tag{4}$$

where $\frac{\partial\, \mathrm{ReLu}(\boldsymbol{z}_2)}{\partial \boldsymbol{z}_2}$ takes the partial derivative for all the components of $\boldsymbol{z}_2$, and then transpose itself.

$$\begin{aligned} \frac{\partial J}{\partial \boldsymbol{b}_1} &= 1 \cdot \frac{\partial J}{\partial \boldsymbol{z}_2} \\ \frac{\partial J}{\partial \boldsymbol{a}_1} &= \boldsymbol{W}_1^T \cdot \frac{\partial J}{\partial \boldsymbol{z}_2} \\ \frac{\partial J}{\partial \boldsymbol{W}_1} &= \boldsymbol{a}_1 \cdot \frac{\partial J}{\partial \boldsymbol{z}_2} \end{aligned} \tag{5}$$

$$\frac{\partial J}{\partial \boldsymbol{z}_1} = \frac{\partial\, \mathrm{ReLu}(\boldsymbol{z}_1)}{\partial \boldsymbol{z}_1} \cdot \frac{\partial J}{\partial \boldsymbol{a}_1} \tag{6}$$

$$\begin{aligned} \frac{\partial J}{\partial \boldsymbol{b}_0} &= 1 \cdot \frac{\partial J}{\partial \boldsymbol{z}_1} \\ \frac{\partial J}{\partial \boldsymbol{x}} &= \boldsymbol{W}_0^T \cdot \frac{\partial J}{\partial \boldsymbol{z}_1} \\ \frac{\partial J}{\partial \boldsymbol{W}_0} &= \boldsymbol{x} \cdot \frac{\partial J}{\partial \boldsymbol{z}_1} \end{aligned} \tag{7}$$

Problem 3

The blank in `train.py` is filled as below:

```
y_hat = np.dot(a2, W2) + b2

    grad_yhat = y_hat - y

    grad_w2 = np.dot(a2.T, grad_yhat) / x.shape[0]
    grad_b2 = np.mean(grad_yhat, axis=0)

    grad_a2 = np.dot(grad_yhat, W2.T) * relu_derivative(a2)

    grad_w1 = np.dot(a1.T, grad_a2) / x.shape[0]
```

```
        grad_b1 = np.mean(grad_a2, axis=0)

        grad_a1 = np.dot(grad_a2, W1.T) * relu_derivative(a1)

        grad_w0 = np.dot(x.T, grad_a1) / x.shape[0]
        grad_b0 = np.mean(grad_a1, axis=0)

        return grad_w0, grad_w1, grad_w2, grad_b0, grad_b1, grad_b2
```

And the toy regression code returns results as below:

```
0: loss is 7.383711411547159
1: loss is 5.565479595565317
2: loss is 4.2623618094099704
3: loss is 3.2768154757390007
4: loss is 2.5111091054934076
5: loss is 1.912952345381044
6: loss is 1.448889682590686
7: loss is 1.0947256163614174
8: loss is 0.8298958540221288
9: loss is 0.6361123294130413
10: loss is 0.4978888735075733
```

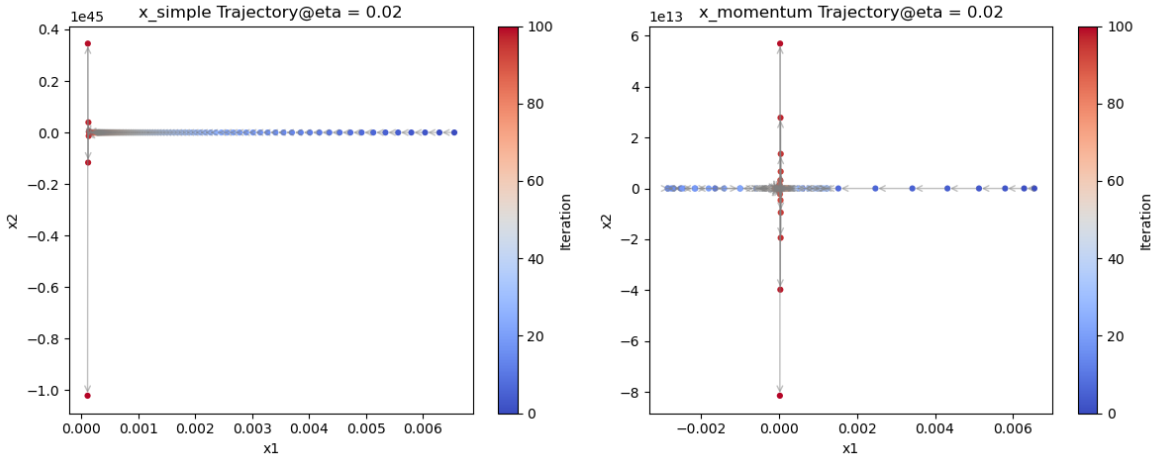
```
57: loss is 0.12287513248513712
58: loss is 0.12181950585517677
59: loss is 0.12078690878329554
60: loss is 0.11977596540614181
61: loss is 0.11878591471510976
62: loss is 0.11781701235499563
63: loss is 0.11686844544088743
64: loss is 0.11593873397074103
65: loss is 0.1150284300352336
66: loss is 0.11413631683770625
67: loss is 0.11326127244741759
```

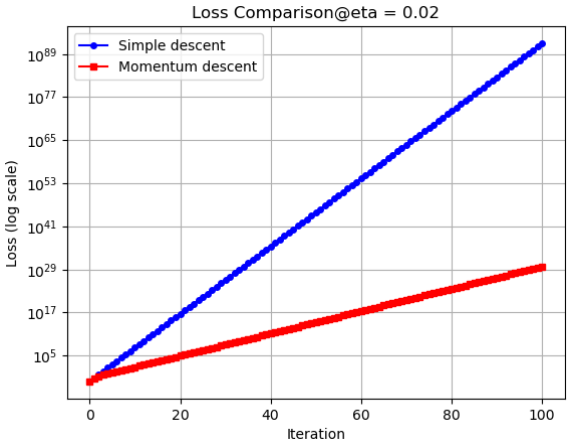
```
91: loss is 0.09640916723940746
92: loss is 0.09584220405065876
93: loss is 0.09528430234806674
94: loss is 0.09473485191874413
95: loss is 0.09419370111668125
96: loss is 0.09366065527349664
97: loss is 0.09313535326928932
98: loss is 0.09261779193767472
99: loss is 0.09210800793719595
Test loss is 0.09615876007241468
```

which is a perfect gradient descent result.

Problem 4

In our program, the initial value is randomly set. When $\eta = 0.02$, the results' comparison for simple gradient descent and momentum descent are as below:

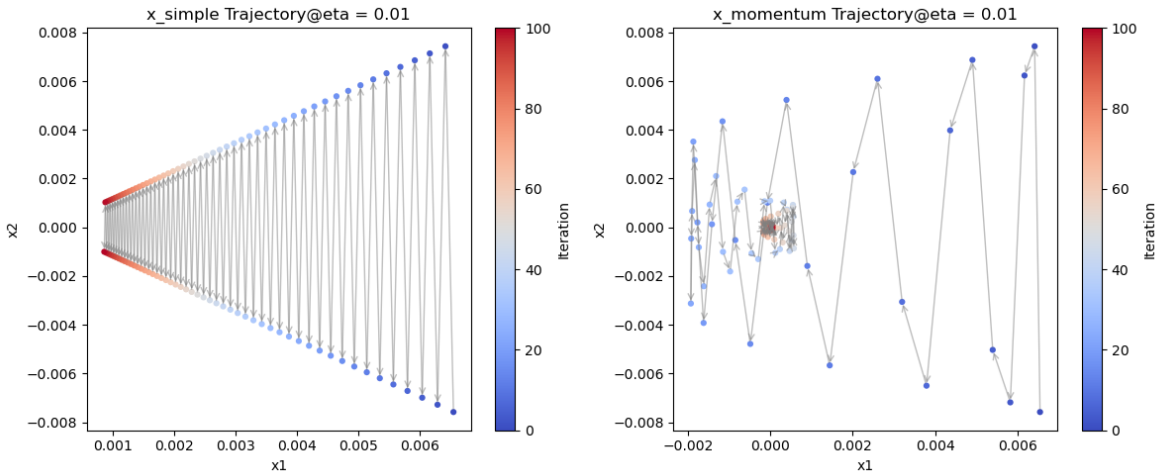


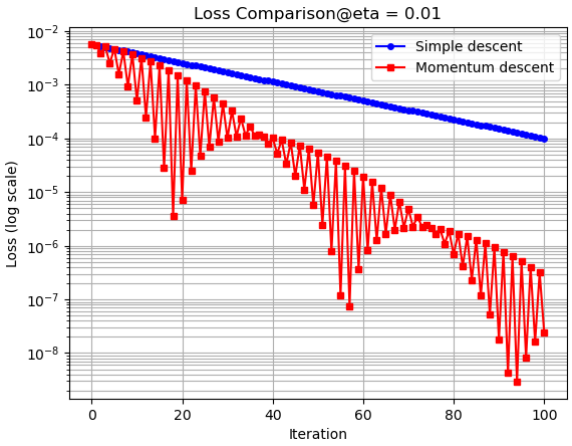


It's clear that the results diverged no matter if the momentum is introduced. In this problem, the coefficient for x_2 is 98 times larger than that of x_1, which leads to the divergence in x_2 values using gradient descent at such a η value. Say, the gradient descent

$$\begin{aligned} x_t =& x_{t-1} - \eta \nabla_{x_{t-1}} \mathcal{L}(x) \\ =& x_{t-1} - 2\eta [x_1, 99x_2]^T \end{aligned} \tag{8}$$

would cause a fatal fluctuation on the 2nd direction (too large speed in a sharp valley). However, if we take the parameter $\eta = 0.01$, there would no more be divergence, but convergent results instead:





In this regime, we can clearly find that the introduction of momentum accelerates the convergence for several order of magnitude, especially when the value is near to the convergent point (origin). The result shows that how the momentum influence the behavior of the 'ball' in the valley, especially near the bottom.

Problem 5

(1)

The total number of parameters should be $3 \times 1 \times 8 + 1 = 25$. The 1 is the stride. If we do not consider stride as a parameter, then the result should be 24.

(2)

The result of $\boldsymbol{k}_1$ and $\boldsymbol{k}_2$ are listed as below:

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
[[-6.  6. -6.]
 [-6.  6. -6.]
 [-6.  6. -6.]]
```

The 0 matrix is for $\boldsymbol{k}_1$, and the second is for $\boldsymbol{k}_2$.

Actually, the kernel $\boldsymbol{k}_1$ and $\boldsymbol{k}_2$ are operators for the strength of wave with certain period and direction. In our case, the period is `3`, and the wave direction of $\boldsymbol{k}_1$ is along `axis = 0`, while the wave direction of $\boldsymbol{k}_2$ is along `axis = 1`.

The given matrix A is periodic along `axis =1` with period `3`, and uniform in `axis = 0`, so the convolution of kernel $\boldsymbol{k}_1$ and A would be 0, since their wave directions are orthogonal, and the wave strength of A along `axis = 0` is 0. However, the absolute value of the convolution of kernel $\boldsymbol{k}_2$ and A is uniformly 6, which indicates that the wave strength and the period of A is uniform along `axis = 1`, and the alternative of symbols indicates the period of A is odd times of that of $\boldsymbol{k}_2$.