Linear Regression

import library

In [1]:

```
import numpy as np
import matplotlib.image as img
import matplotlib.pyplot as plt
import matplotlib.colors as colors
from mpl_toolkits.mplot3d import Axes3D
```

load point data

In [2]:

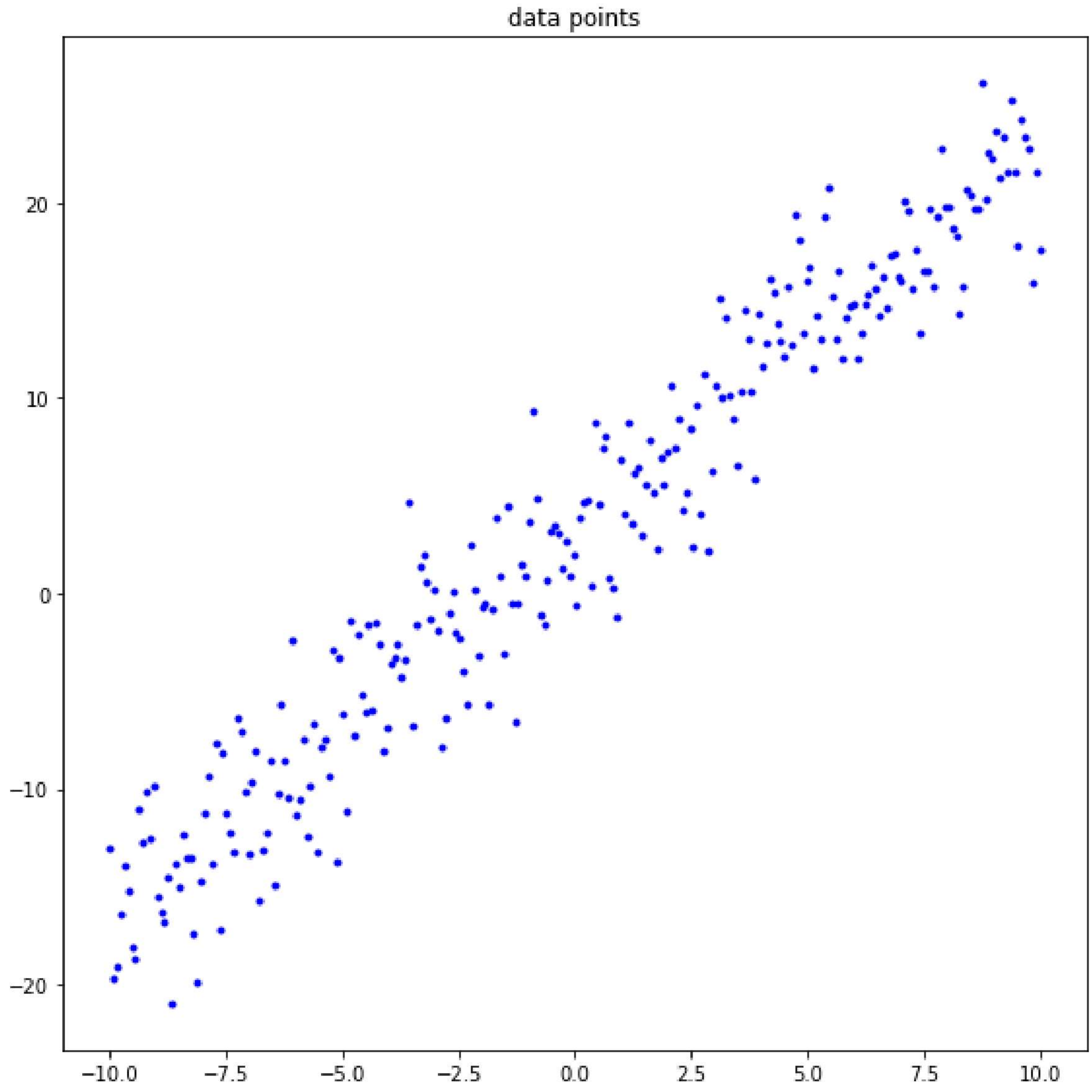
```
filename    = 'assignment_06_data.csv'
data_load   = np.loadtxt(filename, delimiter = ',')

x   = data_load[0, :]
y   = data_load[1, :]

plt.figure(figsize=(8,8))

plt.plot(x, y, '.', color = 'blue')
plt.title('data points')

plt.tight_layout()
plt.show()
```



compute the loss function

In [3]:

```
def compute_loss(x, y, theta0, theta1):
    loss = 0.0
    n = len(x)
    for i in range(0, n) :
        loss += (theta0 + theta1*x[i] - y[i])**2
    loss = loss / (2*n)
    return loss
```

compute the gradient for each model parameter

In [4]:

```
def compute_gradient_theta0(x, y, theta0, theta1):
    dL = 0.0
    n = len(x)
    for i in range(0, n) :
        dL += (2 * theta0) + (2 * theta1 * x[i]) - (2 * y[i])
    dL = dL / (2*x.size)
    return dL
```

In [5]:

```
def compute_gradient_theta1(x, y, theta0, theta1):
    dL = 0.0
    n = len(x)
    for i in range(0, n) :
        dL += (2 * x[i] * x[i] * theta1) + (2 * theta0 * x[i]) - (2 * x[i] * y[i])
    dL = dL / (2*x.size)
    return dL
```

gradient descent for each model parameter

In [6]:

```
num_iteration       = 1000
learning_rate       = 0.01

theta0              = 0
theta1              = 0

theta0_iteration    = np.zeros(num_iteration)
theta1_iteration    = np.zeros(num_iteration)
loss_iteration      = np.zeros(num_iteration)

for i in range(num_iteration):
    ntheta0  = theta0 - (learning_rate * compute_gradient_theta0(x, y, theta0, theta1))
    ntheta1  = theta1 - (learning_rate * compute_gradient_theta1(x, y, theta0, theta1))
    loss     = compute_loss(x, y, ntheta0, ntheta1)

    theta0_iteration[i] = ntheta0
    theta1_iteration[i] = ntheta1
    loss_iteration[i]   = loss

    theta0 = ntheta0
    theta1 = ntheta1

    print("iteration = %4d, loss = %5.5f" % (i, loss))
```

```
iteration =    0, loss = 38.92372
iteration =    1, loss = 22.45805
iteration =    2, loss = 15.14235
iteration =    3, loss = 11.86256
iteration =    4, loss = 10.36354
iteration =    5, loss = 9.65082
iteration =    6, loss = 9.28587
iteration =    7, loss = 9.07528
iteration =    8, loss = 8.93374
iteration =    9, loss = 8.82363
iteration =   10, loss = 8.72832
iteration =   11, loss = 8.64046
iteration =   12, loss = 8.55681
iteration =   13, loss = 8.47590
iteration =   14, loss = 8.39708
iteration =   15, loss = 8.32004
iteration =   16, loss = 8.24463
iteration =   17, loss = 8.17075
iteration =   18. loss = 8.09837
```

In [7]:

```
f = theta0 + (theta1 * x)
```

plot the results

In [8]:

```
def plot_data_regression(x, y, f):

    plt.figure(figsize=(8,6))
    plt.title('linear regression result')
    plt.plot(x, f, color = 'red')
    plt.plot(x, y, '.', color = 'blue')

    plt.tight_layout()
    plt.show()
```

In [9]:

```
def plot_loss_curve(loss_iteration):

    plt.figure(figsize=(8,6))
    plt.title('loss curve')
    plt.plot(loss_iteration, '-', color = 'red')

    plt.tight_layout()
    plt.show()
```

In [10]:

```

def plot_model_parameter(theta0_iteration, theta1_iteration):

    plt.figure(figsize=(8,6))
    plt.title('model parameter')
    plt.plot(theta0_iteration, '-', color = 'blue')
    plt.plot(theta1_iteration, '-', color = 'green')

    plt.tight_layout()
    plt.show()
```

In [11]:

```

X0  = np.arange(-10, 10, 0.1)
X1  = np.arange(-10, 10, 0.1)

grid_theta0, grid_theta1 = np.meshgrid(X0, X1)

grid_loss   = compute_loss(x, y, grid_theta0, grid_theta1)


def plot_loss_surface(grid_theta0, grid_theta1, grid_loss):

    fig = plt.figure(figsize=(8,8))
    ax = fig.add_subplot( projection = '3d')
    ax.plot_surface(grid_theta0, grid_theta1, grid_loss)
    plt.title('loss surface')

    plt.tight_layout()
    plt.show()
```

* *

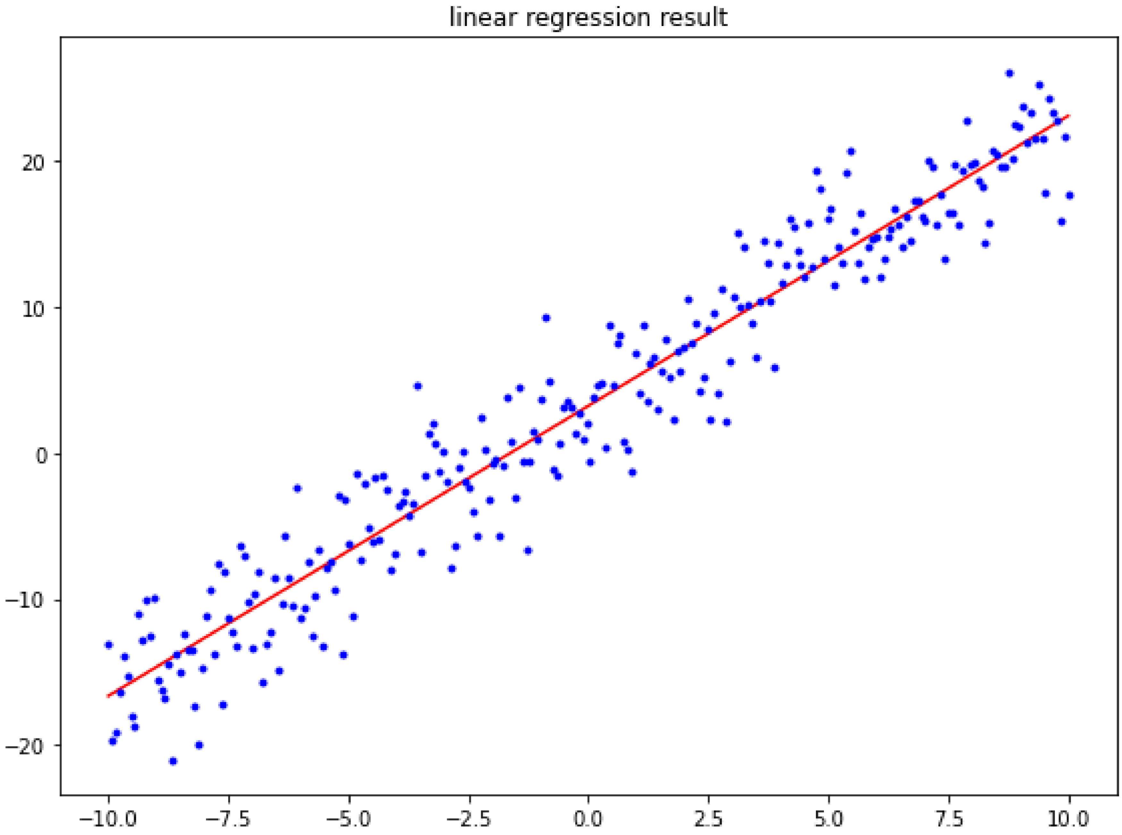
* results

* *

01. plot the input data in blue point and the regression result in red curve

In [12]:

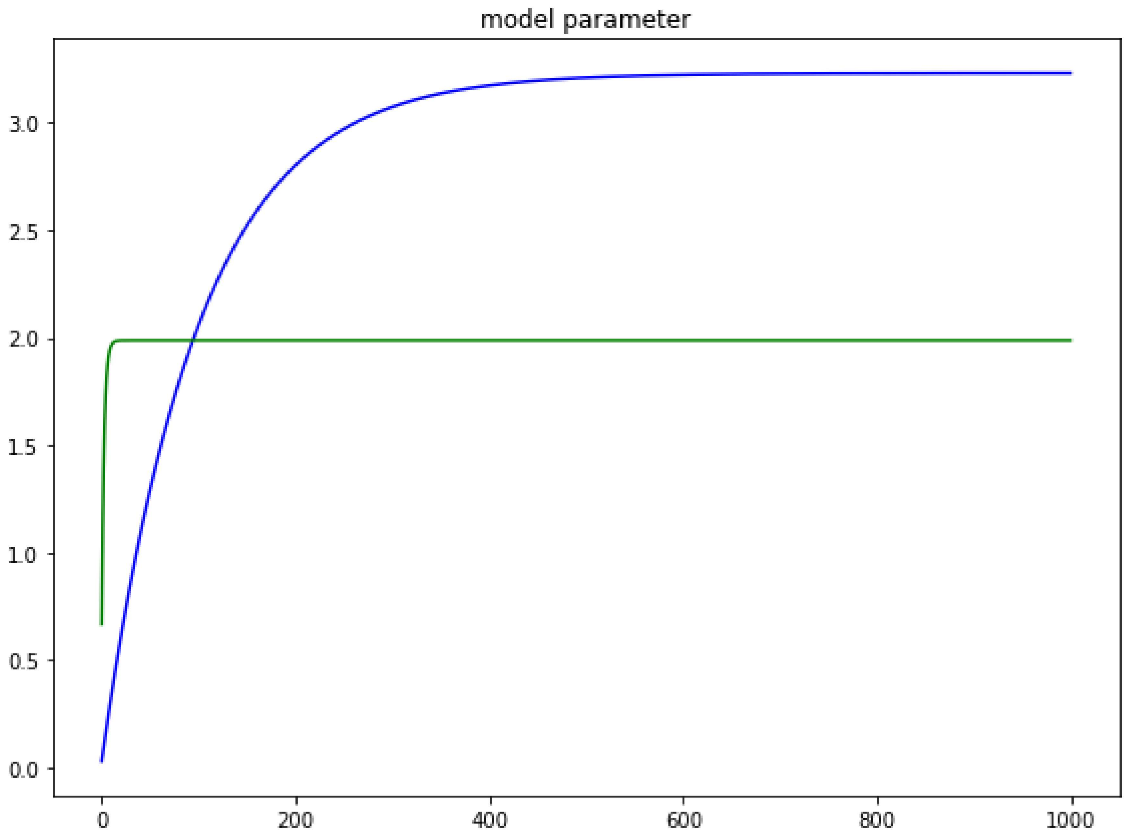
```
plot_data_regression(x, y, f)
```



02. plot the values of the model parameters θ_0 in blue curve and θ_1 in green curve over the gradient descent iterations

In [13]:

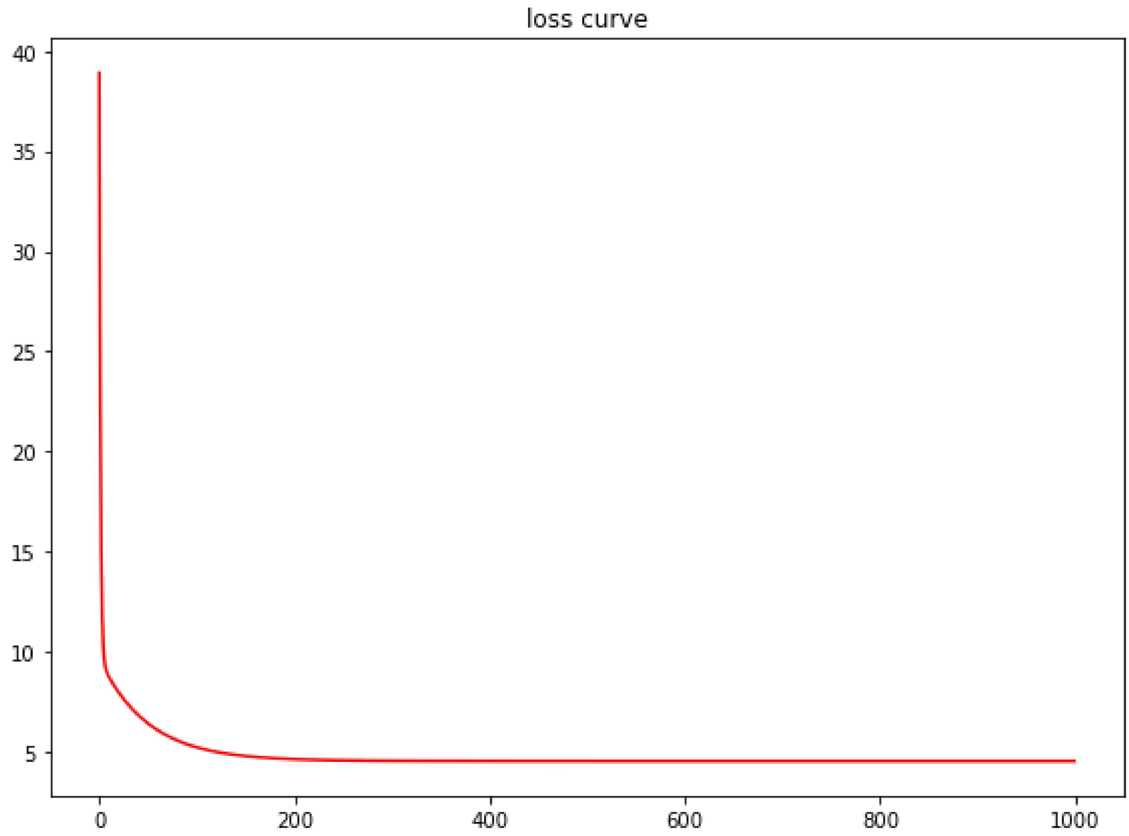
```
plot_model_parameter(theta0_iteration, theta1_iteration)
```



03. plot the loss values $\mathcal{L}(\theta)$ in red curve over the gradient descent iterations

In [14]:

```
plot_loss_curve(loss_iteration)
```

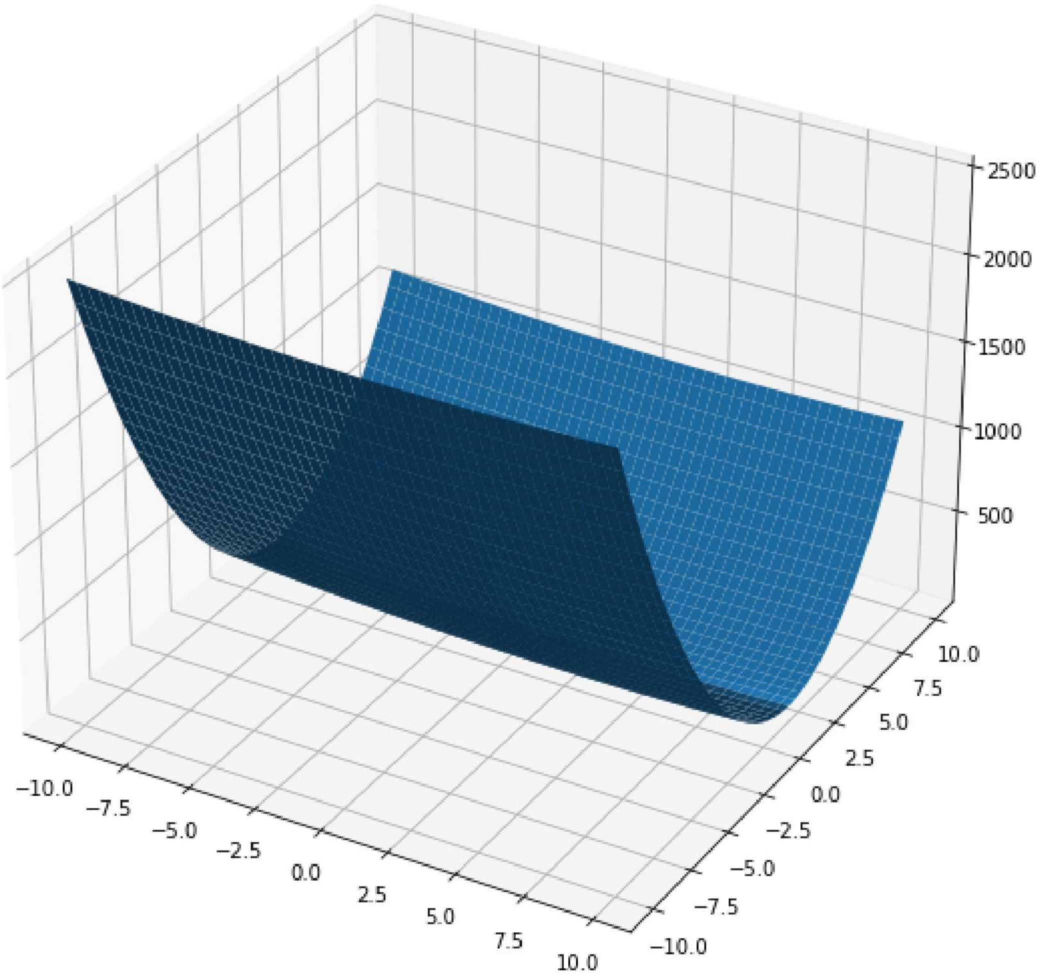


04. plot the loss surface in 3-dimension surface where x-axis represents θ_0, y-axis represents θ_1 and z-axis represents $\mathcal{L}$

In [15]:

```
plot_loss_surface(grid_theta0, grid_theta1, grid_loss)
```

loss surface


2500
2000
1500
1000
500
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0