**Problem#1**

Suppose your cost function is as follows.

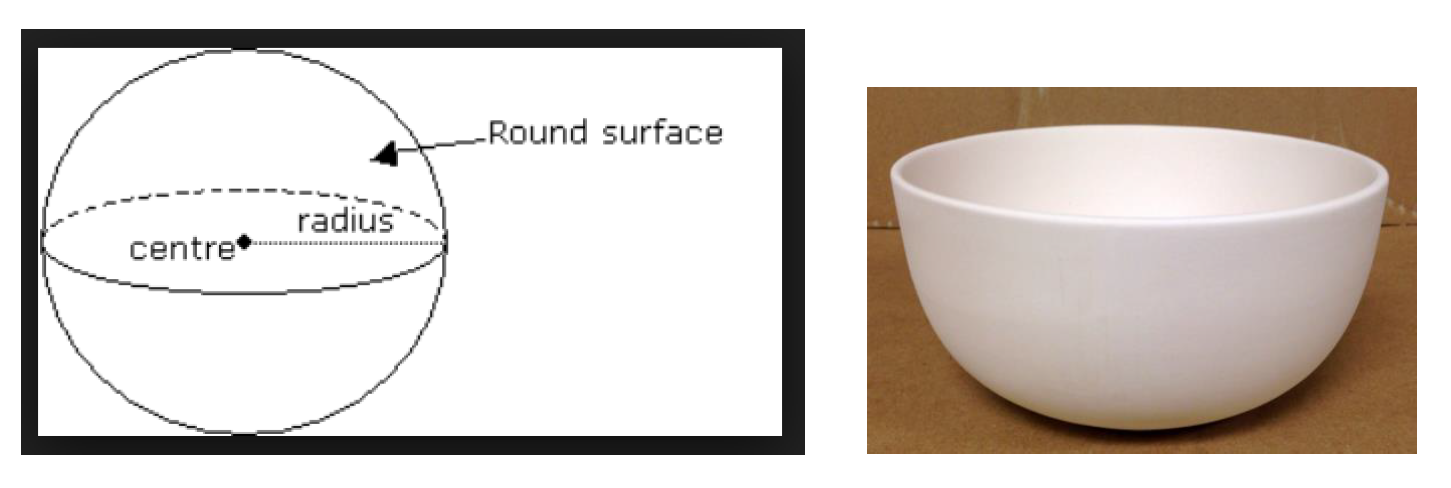
𝑓(𝑥,𝑦)=𝑧=−√25−(𝑥−2)2−(𝑦−3)2

This cost function represents the **bottom half** of a sphere with the following specifications.

Center of the sphere = x=2, y=3, z=0

Radius of the sphere = 5

The cost function is **convex** which means that it has a minimum point.



Find the values of ‘x’ and ‘y’ where the cost (z-value) is minimum using the Gradient Descent algorithm written in Python. The gradient (vector of partial derivatives) of this cost function is as follows.

𝜕𝑧𝜕𝑥=(𝑥−2)√25−(𝑥−2)2−(𝑦−3)2

𝜕𝑧𝜕𝑦=(𝑦−3)√25−(𝑥−2)2−(𝑦−3)2

Assuming learning rate = 0.01.

How many steps are needed for the Gradient Descent algorithm to converge? The criterion for convergence is when the incremental value of ‘x’ and ‘y’ is less than epsilon (0.000001).

Answer: The values of ‘x’ and ‘y’ where ‘z’ is minimum are as follows.

x = approximately 2, y = approximately 3. At this point the z = -5.

============================================================================ 2

**Problem#2**

**Dataset:**

Generate a synthetic dataset using the following Python code.

---------------------------------------------------------------

import tensorflow.compat.v1 as tf

tf.disable\_v2\_behavior()

RANDOM\_SEED = 42

tf.set\_random\_seed(RANDOM\_SEED)

import numpy as np

import matplotlib.pyplot as plt

n\_samples = 30

train\_x = np.linspace(0,20,n\_samples)

train\_y = 3.7 \* train\_x + 14 + 4 \* np.random.randn(n\_samples)

plt.plot(train\_x, train\_y,'o')

---------------------------------------------------------------

The value of RANDOM\_SEED can be any integer. However, choose the value of RANDOM\_SEED as 42. This will allow all students to get identical datasets. The ‘train\_x’ is the predictor variable and ‘train\_y’ is the response variable.

**Model Building:**

Compute the regression equation between ‘train\_x’ and ‘train\_y’ variables using Scikit-Learn package.

**Gradient Descent (GD):**

Using the Gradient Descent (GD) algorithm compute the regression equation - coefficient and intercept values.

Do not use the TensorFlow Gradient Descent function but write your own Python code to implement the GD algorithm.

Vary the learning rate and the number of iterations, till the answer computed by your GD algorithm (implemented in Python) matches with the answer computed by Scikit-Learn.