Assignment02

March 21, 2019

Assignemt02 : First-order Taylor approximation
Software Engineering
20154652 Lee Dong Jae

```
In [53]: import matplotlib.pyplot as plt
         from scipy.misc import derivative
         import random
         import numpy as np
```

1. Define a differentiable function that maps from real number to real number.

Function: $f(x) = \frac{e^x + e^{-x}}{2}$

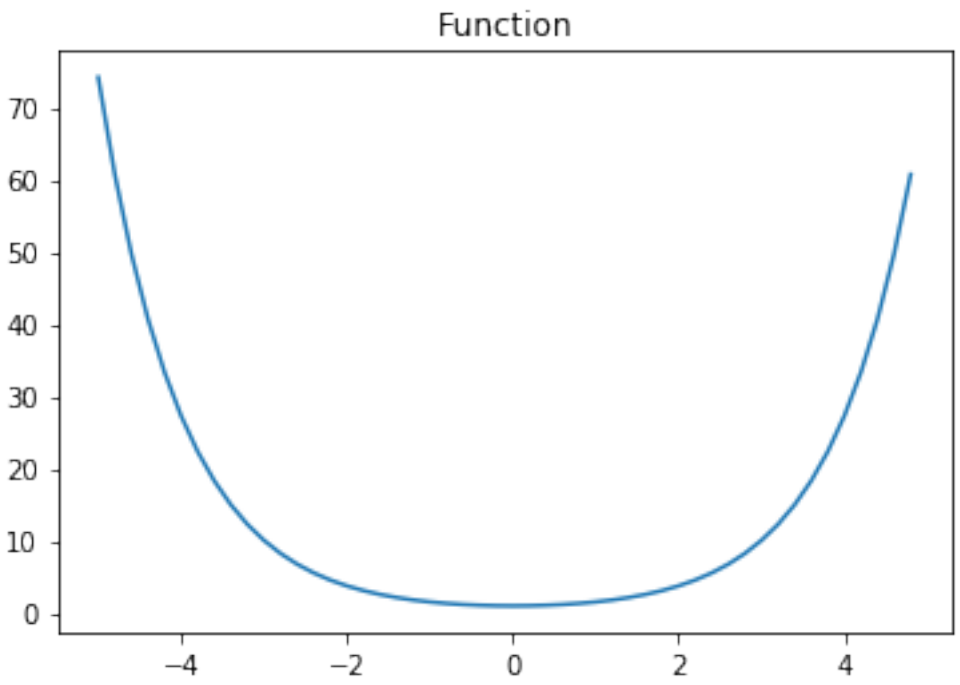
```
In [54]: def func(x):
             result = (np.exp(-x) + np.exp(x)) / 2
             return result
```

2. Define a domain of the function.

```
In [55]: x = np.arange(-5,5,0.2)
         y = func(x)
```

3. Plot the function.

```
In [56]: plt.title("Function")
         plt.plot(x, y)
         plt.show()
```

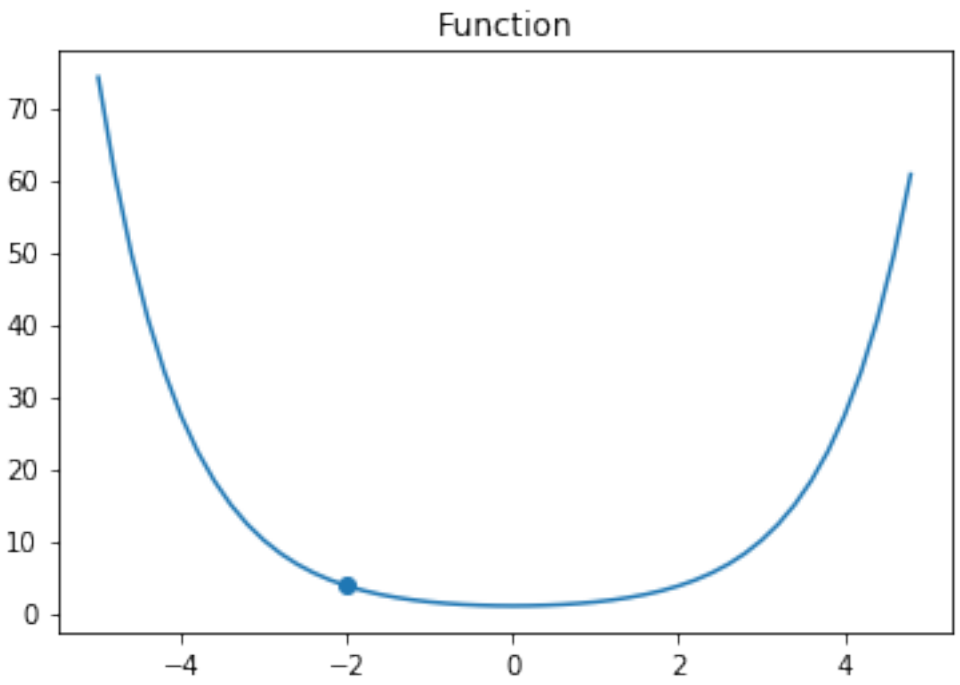


4. Select a point within the domain.

```
In [64]: #Specify a value in the range of the definition as a random value

         x_point = random.randrange(-5,5)
```

5. Mark the selected point on the function.

```
In [65]: y_point = func(x_point)
         plt.title("Function")
         plt.scatter(x_point, y_point)
         plt.plot(x,y)
         plt.show()
```

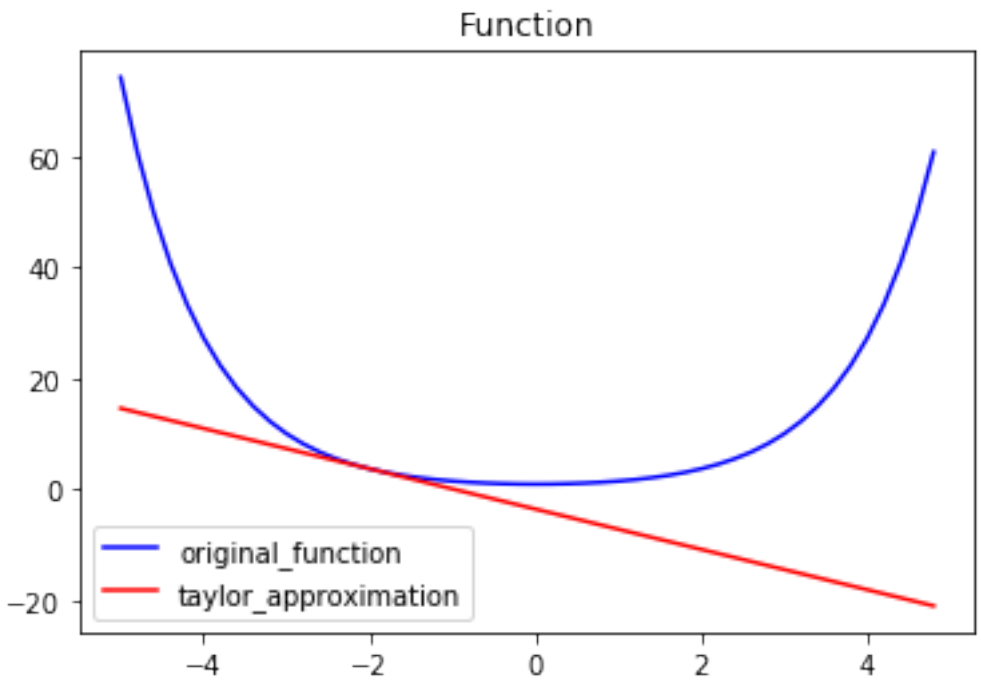


6. Define the first-order Taylor approximation at the selected point.

```
In [66]: #find a derived function

         def derivative_func(x):
             return derivative(func, x, dx = 1e-5)

         #find Taylor approximation

         def taylor_approximation(domain, point):
             taylor_approxiamtion_form = derivative_func(point) * (domain - point) + func(poin
             return taylor_approxiamtion_form
```

7. Plot the Taylor approximation with the same domain of the original function.

```
In [67]: plt.title("Function")
         plt.plot(x,y, 'b', label = "original_function")
         plt.plot(x,taylor_approximation(x, x_point), 'r', label = "taylor_approximation")
         plt.legend()
         plt.show()
```


Function
60
40
20
0
−20
−4
−2
0
2
4
original_function
taylor_approximation

In [ ]: