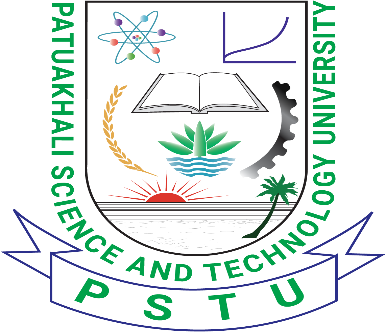
**Lab Problem: 06.**  


**Course code: CCE-312.**

**Course Title: Numerical Methods sessional.**

**Remarks & Signature:**

**Name of the Lab Report:** Solve Real world problem using False-position Method after that implement it by Python.

**Submitted To**

**Professor Dr. Md. Samsuzzaman.**

**Professor,**

**Department of Computer and Communication Engineering.**

**Faculty of Computer Science & Engineering.**

**Submitted By**

**HASAN AHAMMAD**

**ID No: 1902073**

**Reg No: 08779**

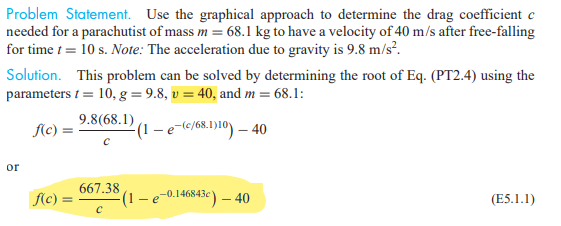
**Level- 3, Semester- 1**

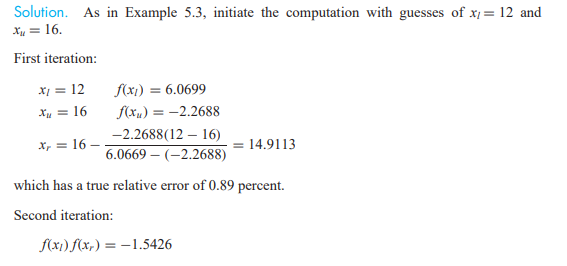
**Session: 2019-2020**

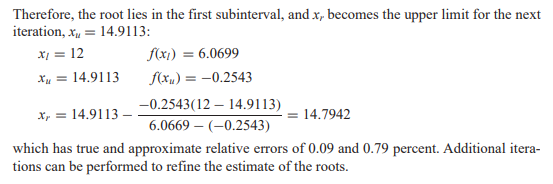
**Faculty of Computer Science & Engineering.**

**Patuakhali Science & Technology University.**

**Dumki, Patuakhali-8602.**

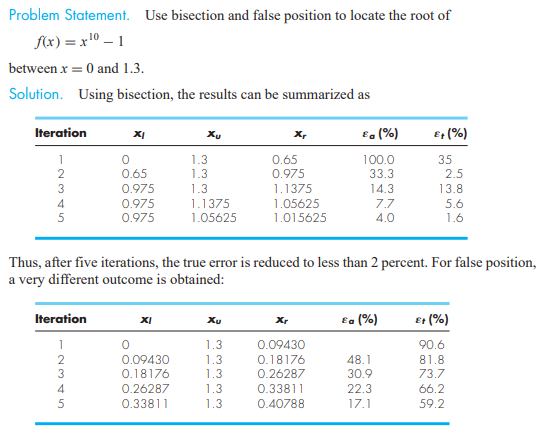






* **Implement using python:**

import math  
  
  
def function\_to\_find\_root(c):  
 return (667.38 / c) \* (1 - math.exp(-0.146843 \* c)) - 40  
  
  
def false\_position\_method(func, a, b, tol=1e-6, max\_iter=100):  
  
 if func(a) \* func(b) > 0:  
 raise ValueError("The function must have different signs at the interval endpoints.")  
  
 iterations = 0  
 while iterations < max\_iter:  
 c = (a \* func(b) - b \* func(a)) / (func(b) - func(a))  
  
 if abs(func(c)) < tol:  
 return c, iterations  
  
 if func(c) \* func(a) < 0:  
 b = c  
 else:  
 a = c  
  
 iterations += 1  
  
 raise ValueError("False-position method did not converge within the maximum number of iterations.")  
  
x1 = 12  
x2 = 16  
tolerance = 1e-6  
  
root, iterations = false\_position\_method(function\_to\_find\_root, x1, x2, tol=tolerance)  
print(f"Approximated root: {root:.6f}")  
print(f"Iterations: {iterations}")

****.

* **Implement using Python:**
* def false\_position\_method(func, a, b, tol=1e-6, max\_iter=100):  
   if func(a) \* func(b) > 0:  
   raise ValueError("The function must have different signs at the interval endpoints.")  
    
   iterations = 0  
   while iterations < max\_iter:  
   c = (a \* func(b) - b \* func(a)) / (func(b) - func(a))  
    
   if abs(func(c)) < tol:  
   return c, iterations  
    
   if func(c) \* func(a) < 0:  
   b = c  
   else:  
   a = c  
    
   iterations += 1  
    
   raise ValueError("False-position method did not converge within the maximum number of iterations.")  
    
    
  # Example usage:  
  def example\_function(x):  
   return x \*\* 10 - 1  
    
    
  a = 0  
  b = 1.3  
  tolerance = 1e-6  
    
  root, iterations = false\_position\_method(example\_function, a, b, tol=tolerance)  
  print(f"Approximated root: {root:.6f}")  
  print(f"Iterations: {iterations}")