**Mid Exam Simulation #1**

1. A sample of the data of the amount of water in liter used by some washing machines are of the following.

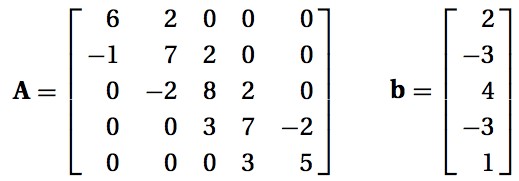
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5.5 | 1.1 | 6.5 | 4.9 | 6.4 |
| 7.0 | 1.5 | 5.7 | 5.9 | 5.4 |
| 6.1 | 1.2 | 7.3 | 6.1 | 4.4 |

Write a python program to compute the statistic descriptive of the sample data.

* 1. What are the sample mean (𝑥̅) and the sample standard deviation (𝑠).
  2. What are the first (𝑄%), second (𝑄&), and third (𝑄’) quantiles.

**import** numpy **as** np  
**import** math **as** m  
  
**from** sympy **import** symbols, integrate  
  
data = np.array([  
 5.5, 1.1, 6.5, 4.9, 6.4,  
 7.0, 1.5, 5.7, 5.9, 5.4,  
 6.1, 1.2, 7.3, 6.1, 4.4,  
 ])  
  
*#1 a)*mean = np.mean(data)  
print(mean)  
  
ssd = np.std(data, ddof=1)  
print(ssd)  
  
*#1 b)*Q = np.percentile(data, [25, 50, 75])  
print(Q)

1. We consider the linear algebra problem 𝐴𝑥 = 𝑏 where the system matrix and the right-hand-side vector are:

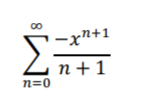


Use python, to compute the following. Copy your solutions to the answer sheet.

* 1. The solution vector 𝑥.
  2. The inverse of the matrix 𝐴.
  3. The determinant of the matrix 𝐴.

*#----------------------  
#2*A = np.array([  
 [6, 2, 0, 0, 0],  
 [-1, 7, 2, 0, 0],  
 [0, -2, 8, 2, 0],  
 [0, 0, 3, 7, -2],  
 [0, 0, 0, 3, 5]  
])  
  
b = np.array([  
 [2],  
 [-3],  
 [4],  
 [-3],  
 [1]  
])  
  
*#2 a)*x = np.linalg.solve(A, b)  
print(**"Solution vector x "**,x)  
  
*#2 b)*A\_inverse = np.linalg.inv(A)  
print(**"Inverse of matrix A "**,A\_inverse)  
  
*#2 c)*det\_A = np.linalg.det(A)  
print(**"Determinant of matrix A"**, det\_A)  
  
*# test*transpose\_A = np.transpose(A)  
print(**"Transpose A : "**,transpose\_A)

1. The Taylor/Maclaurin series expansion of the ln(1-x) function is:



* 1. Write a python function that compute the ln(1-x). Your function should have the following interface that allows the user to adjust the number of the terms of the series 𝑛.

def ln1\_x(x, n):

# Your codes

* 1. Use your function to compute ln(1-x) with x = 0.5 by using the first 8 and 16 terms of the series.
  2. Compute also ln(1-x) by using the math log function
  3. Compute the relative errors of the results in (b) by assuming the solution in (c) as the exact solution.

*#-------------------------  
#3  
  
#a)***def** ln1\_x(x, n):  
 total = 0  
 **for** i **in** range(0, n):  
 value = m.pow(-x, i+1) / (i + 1)  
 total = total + value  
 **return** total  
  
*#b)*ln\_1\_8 = ln1\_x(0.5, 8)  
ln\_1\_16 = ln1\_x(0.5, 16)  
print(**"ln1\_x 8 "**, ln\_1\_8)  
print(**"ln1\_x 16"**, ln\_1\_16)  
  
*#c)*original = m.log(0.5)  
print(**"ln(1-x) using math log "**, original)  
  
*#d)*error\_8 = original - ln\_1\_8  
error\_16 = original - ln\_1\_16  
print(**"Error 8 "**, error\_8)  
print(**"Error 16"**, error\_16)  
print(**"Percentage error 8 "**, error\_8 / original \* 100, **"%"**)  
print(**"Percentage error 16 "**, error\_16 / original \* 100, **"%"**)

1. Compute the exact solution and the numerical solution for the problem: &

As for the numerical solution, you can solve by either the trapezoidal rule or simpson rule. Use four panels for the integration. Compute the relative error of the numerical solution compared to the exact integration result.

*#------------------------------  
#4***def** f(x):  
 **return** 2\*x\*\*4 + 6\*x\*\*3 - 2\*x  
  
**def** trapezoidal(a, b, n):  
 h = (b-a)/n  
 s = (f(a) + f(b)) \* 0.5  
 **for** i **in** range(1,n):  
 s = s + f(a + i \* h)  
  
 **return** h \* s  
  
**def** simpson1\_3(a, b, n):  
 h = (b-a)/n  
 x = a  
 s = 0  
 **for** i **in** range (1,n):  
 x = x + h  
 **if** i % 2 == 0:  
 m = 2  
 **else**:  
 m = 4  
 s = s + m\*f(x)  
 **return** (b-a)\*(f(a)+s+f(b))/(3\*n)  
  
**def** simpson(a, b, n):  
 h = (b-a)/ n  
 x = a  
 s = 0  
 **for** i **in** range(1, n):  
 x = x + h  
 **if**(i%2==0):  
 m = 2  
 **else**:  
 m = 4  
 s = s + m \* f(x)  
 **return** (b-a) \* (f(a) + s + f(b)) / (3\*n)  
  
trape = trapezoidal(0, 2, 4)  
print(**"Trapezoidal "**, trape)  
  
simp = simpson1\_3(0, 2, 4)  
print(**"Simpson "**, simp)  
  
x = symbols(**'x'**)  
  
exact\_int = integrate(2\*x\*\*4 + 6\*x\*\*3 - 2\*x, x)  
exact = abs(exact\_int.subs(x, 0) - exact\_int.subs(x, 2))  
  
print(**"Exact : "**, float(exact))  
  
error\_trapezoid = exact - trape  
error\_simpson = exact - simp  
print(**"Error Trapezoid : "**, error\_trapezoid)  
print(**"Error simpson "**, error\_simpson)