Quiz - Computational Methods in Materials Science

Time - 5 hours

Answer all questions. Write all explanations as markdowns in one single Jupyter notebook. Name the file as <First Name>_<RollNo>.ipynb and submit.

1. Cramer's rule is a method of computing the determinant of a matrix. Consider an $n \times n$ square matrix M. Let M_{ij} denote the element in the i-th row and j-th column of M, and let C_{ij} be the cofactor of M by removing the i-th row and j-th column from M and using appropriate sign.

The cofactor C_{ij} is defined as

$$C_{ij} = (-1)^{i+j} \det(m_{ij})$$

where m_{ij} is the minor formed by removing i^{th} row and j^{th} column of M. Then $\det(M)$ is determined by the expansion:

$$\det(M) = \sum_{j=1}^{n} M_{ij} C_{ij} = M_{i1}C_{i1} + M_{i2}C_{i2} + \cdots + M_{in}C_{in}$$

Write a function my_rec_det(M), where the output is $\det(M)$ using recursion. The function should use Cramer's rule to compute the determinant, not Numpy's function.

2. Consider a circle with unit radius ($r = 1$). Use numpy.random.rand function to generate coordinates (x, y). Check if $x^2 + y^2$ lies within the circle. Run this again and again for user-specified large number of iterations (no. of iterations = 50000 or more) . Keep track of the number of times (x, y) lies within the circle. If total number of iterations is N and total number of times (x, y) are within circle is n_{inside}, then evaluate $(4\, n_{\text{inside}})/N$. Is this close to a special mathematical constant?

3. Write a function *my_is_orthogonal(v_1,v_2, tol)*, where v_1 and v_2 are column vectors of the same size and *tol* is a scalar value strictly larger than 0. The output should be 1 if the angle between v_1 and v_2 is within tol of $\pi/2$; that is, $|\pi/2 - \theta| < \text{tol}$, and 0 otherwise. You may assume that v_1 and v_2 are column vectors of the same size, and that *tol* is a positive scalar.

4. Find logical errors in the programs below, point them out in a markdown box, and then write a correct program that fixes the logical errors

Program 1

```
a = input("Input a number:")

b = input("Input another number:")

print("The highest number is:")

if a > b:

    print(a)

else:

    print(b)
```

Program 2 (Hint: test for all cases)

```
marks = int(input("Input a number between 0 and 100:"))

print("Your grade:")

if marks<50:
```

```
    print("U")
elif marks>50 and marks<69:
    print("Pass")
elif marks>70 and marks<89:
    print("Merit")
elif marks>90 and marks<100:
    print("Distinction")
else:
    print("Invalid␣Score!")
```

Program 3 (Hint: test for numbers 10, 5, 0, respectively)

```
def factorial(n):
f = 1
for i in range(1,n):
    f = f * i
    print(f)
return f
number = int(input("Input␣a␣number:␣"))
print(str(number) + "!␣=␣" + str(factorial(number)) )
```

5. Solve a system of equations $\mathrm{Hx} = b$ where $b = [1\,1\,1 \cdots 1]^T$ and H is a 20×20 Hilbert matrix. Now solve $\mathrm{Hx} = d$ where $d = [0.99\,0.99\,0.99 \cdots 0.99]^T$. Discuss the accuracy of the results. The $n \times n$ Hilbert matrix is defined as $H(i,j) = 1/(i+j-1)$ with $1 \leqslant i, j \leqslant n$.

6. Write a function which finds trace of a given matrix A. The trace of a matrix is given by the sum of its diagonal elements. Your function should check whether A is a square matrix.

7. Use

```
import numpy as np
T = -1*np.eye(10,10,k=-1)+np.eye(10,10)*2+-1*np.eye(10,10,k=1)
```

to create a tridiagonal matrix T. Write a program to factorize T into L and U.