Groups - 2nd Semester

June 2, 2021

1. To show that $[1, -1, i, -i]$ is a group under multiplication

[1]:
```python
from numpy import *
G = array([1, 1j, -1j])
n = len(G)

switch = 0                  # Assuming that G does not satisfies the laws.
switch1 = 0                 # Contradiction: Assuming that G is not a group.

# Closure Law
# To prove a*b is in G for all the elements in G
for i in arange(n):
    for j in arange(n):
        prod = G[i]*G[j]    # product a*b in G
        if prod in G:       # To check if the product is in G
            switch = 1      # If Law is satisfied
        else:
            switch = 0      # If law not satisfied
            break

if switch == 1:
    print("G is closed under multiplication\n");
    switch1 = 1             # If closure law satisfied, our assumption might be␣
↪wrong, so we switch this to 1
else:
    print("G is not closed under multiplication.\nG is not a Group.")
    switch1 = 0             # If closure law is not satisfied, then G won't be␣
↪a group.


# Assocoative Law
if switch1 == 0:
    quit()
else:
    for i in arange(n):
        for j in arange(n):
            for k in arange(n):
```

```
                if G[i]*(G[i]*G[k]) == (G[i]*G[j])*G[k]:  #To check if a*(b*c)␣
↪= (a*b)*c
                    switch = 1
                else:
                    switch = 0
                    break

    if switch == 1:
        print("G is associative under multiplication\n")
        switch1 = 1             # If associative law satisfied, our assumption␣
↪might be wrong, so we switch this to 1
    else:
        print("G is not associative under multiplication.\nG is not a Group.")
        switch1 = 0             # If associative law is not satisfied, then G␣
↪won't be a group.


# Identity Element
if switch1 == 0:
    quit()
else:
    for i in arange(n):
        if all(G*G[i] == G):    # G multiplied with each element of G. G*a=G =>␣
↪a = e.
            e1 = G[i]
            print(f'e1 = {G[i]} is an unique identity element\n')
            switch1 = 1
            break
        else:
            print(f"{G[i]} is not the identity element.\n")
            switch1 = 0


# Inverse element
if switch1 == 0:
    quit()
else:
    inv = zeros(n, dtype=complex) # A dummy array of zeros to store the inverse␣
↪values.
    for i in arange(n):
        inv[i] = e1/G[i]        # For each elements of G, find the e/that␣
↪element. i.e., inv = identity/G (or inv = e/a)
        if inv[i] in G:
            print(f'{inv[i]} is the inverse of {G[i]}');
            switch1 = 1
        else:
```

```
            print(f"No inverse element exists \n");
            switch1 = 0
            break

    print("\n")
    if switch1 == 1:
        print("G is a group under multiplication.")
    else:
        print("G is not a group");
```

```
G is not closed under multiplication.
G is not a Group.
```

2. To construct a Cayley Table for $(Z_4, +_4)$.

[2]:
```
# Cayley Table
from numpy import *
n = int(input("Enter the number of elements in Z : ")); # Input number of␣
↪elements.
Z = arange(n)         # Creates Z with the help of n
Z1 = zeros((n, n))    # Create a dummy matrix of nxn to store the values after␣
↪operations. Table of all zeros.

for i in arange(n):
    for j in arange(n):
        Z1[i, j] = mod(Z[i] + Z[j], n)  #Mod(a*b, 4).Stores the values of each␣
↪operation in Z1 for each i and j respectively.

print(Z1)      #To print the matrix after the oprations. This will be the Cayley␣
↪Table.
```

```
Enter the number of elements in Z : 4
[[0. 1. 2. 3.]
 [1. 2. 3. 0.]
 [2. 3. 0. 1.]
 [3. 0. 1. 2.]]
```

3. To construct a Cayley Table for (Z_4, x_4).

[1]:
```
# Cayley Table
from numpy import *
n = int(input("Enter the number of elements in Z : ")); # Input number of␣
↪elements.
Z = arange(n)         # Creates Z with the help of n
Z1 = zeros((n, n))    # Create a dummy matrix of nxn to store the values after␣
↪operations. Table of all zeros.

for i in arange(n):
    for j in arange(n):
```

```
        Z1[i, j] = mod(Z[i] * Z[j], n)  #Mod(a*b, 4).Stores the values of each␣
↪operation in Z1 for each i and j respectively.

print(Z1)     #To print the matrix after the oprations. This will be the Cayley␣
↪Table.
```

```
Enter the number of elements in Z : 4
[[0. 0. 0. 0.]
 [0. 1. 2. 3.]
 [0. 2. 0. 2.]
 [0. 3. 2. 1.]]
```

4. To create Cayley table for (G, x_{10}) where $G = 2, 4, 6, 8$

[2]:
```
from numpy import *
n = int(input("Enter the value of n: "))     # The mod value for the operator
G = arange(2, n, 2)                           # To start the values of set G␣
↪from 2 with incerment of 2.
print(f'\nThe given set is {G}\n')
m = len(G)                                    # To find the total number of␣
↪elements in G
G1 = zeros((m, m))                            # Dummy zero matrix to store the␣
↪values after the operation.

for i in range(m):
    for j in range(m):
        G1[i, j] = mod(G[i] * G[j], n)        # Operation, mod(a*b, n) and to␣
↪store the respective values.

print(f'The Cayley Table for given set G = {G} under multiplication mod {n} is :
↪ \n {G1}')
```

```
Enter the value of n: 10

The given set is [2 4 6 8]

The Cayley Table for given set G = [2 4 6 8] under multiplication mod 10 is :
 [[4. 8. 2. 6.]
 [8. 6. 4. 2.]
 [2. 4. 6. 8.]
 [6. 2. 8. 4.]]
```

ARAVIND N