Single Layer Winding Scheme (Code)

In [1]:

```
# import libraries
import math

# define inputs
number_of_phases=3
number_of_slots = int(input("Enter the number of slots: "))
number_of_poles = int(input("Enter the number of poles: "))
```

```
Enter the number of slots: 36
Enter the number of poles: 6
```

In [2]:

```
# define a function to check the existence of single layer winding scheme.
def single_layer_checkPossiblity(number_of_slots,number_of_poles):
    number_of_slots = int(number_of_slots)
    number_of_poles = int(number_of_poles)
    flag = 0
    import math

    # define a factor which helps in existance
    gcd1 =  math.gcd(number_of_slots, number_of_poles)
    factor = number_of_slots/(3*gcd1)


    # define total number of coils
    number_of_coils = float(number_of_slots/2)

    # coils per pole
    coils_per_pole = float(number_of_slots/(2*number_of_poles))

    # coils per phase
    coils_per_phase = float(number_of_slots/(2*3))

    # define motor periodicity
    motor_periodicity = float(math.gcd(number_of_slots,number_of_poles//2))

    # define number of spokes
    number_of_spokes = float(number_of_slots/motor_periodicity)

    if (number_of_poles%2 != 0 or factor.is_integer()== False or number_of_slots % 3
        number_of_coils.is_integer()==False or coils_per_phase.is_integer()==False o
        number_of_spokes.is_integer()==False or motor_periodicity.is_integer() == Fa
        flag = 1
    return flag



if single_layer_checkPossiblity(number_of_slots,number_of_poles)==0:
    print('Winding is possible')
else:
    print('Winding is not feasible')
```

```
Winding is possible
```

In [3]:

```
# define total number of coils
number_of_coils = number_of_slots/2

# coils per pole
coils_per_pole = number_of_slots/(2*number_of_poles)
```

```
# coils per phase
coils_per_phase = number_of_slots/(2*3)

# define coil span
coil_span = number_of_slots//number_of_poles

# define motor periodicity or number of rotation
motor_periodicity = math.gcd(number_of_slots,number_of_poles//2)

# define phase group
phase_group = number_of_slots/(4*number_of_phases)

# define number of spokes
number_of_spokes = number_of_slots/motor_periodicity
```

In [4]:

```
# define coil numbers
coil_number = [i for i in range(1,(number_of_slots+2)//2)]

# define coil pitch in both electrical and mechanical degrees
coil_pitch_mech = 360/number_of_coils
coil_pitch_elec = (number_of_poles/2)*coil_pitch_mech


# now define coil angles in both mechanical and electrial degrees
coil_angle_mech = [n*coil_pitch_mech for n in range(len(coil_number))]
coil_angle_elec = [n*coil_pitch_elec for n in range(len(coil_number))]
```

In [6]:

```
# make slot angle list named as theta
theta = coil_angle_elec

# make a suitable list for periodicity handling
list1 = []
iter_ = 1
for i in range(int(number_of_slots/number_of_spokes)):
    temp = []
    for j in range(int(number_of_spokes)):
        temp.append(iter_)
        iter_ += 1
    list1.append(temp)


# handling of odd length
arr = []
for ele in list1:
    arr.append(ele[:len(ele)//2])
    arr.append(ele[len(ele)//2:])

# Now again assign arr to the list1 which is in the required form.
list1 = arr
```

In [7]:

```
# convert slot angle between -180 to +180 degrees
for i in range(0,int(number_of_coils)):
    theta[i] = ((theta[i]+180)%360)-180

# round-off theta to nearest integer
for i in range(len(theta)):
    theta[i] = math.ceil(theta[i])

# define slotin and slotout with the help of list1 we got in previous step
```

```
slotin = [x for i in range(len(list1)) for x in list1[i] if i%2==0]
slotout = [x for i in range(len(list1)) for x in list1[i] if i%2==1]
```

In [8]:

```
# initialize a list that should contains positive angle anf for negative angle add 3
thetai = [x+360 if x<0 else x for x in theta ]

# Now sort the positive relative slot angles
theta1 = sorted(thetai)

# Final step to select the phases with same logic used in double layer winding
# Phase A selection
slotin1 = []
slotout1 = []
set1=  [False] * int(number_of_coils)
for i in range(len(theta1)):
    for j in range(int(number_of_coils)):
        if(len(slotin1)== int(number_of_coils)//3):
            break
        else:
            if thetai[j]== theta1[i]:
                if set1[j] ==False:
                    slotin1.append(slotin[j])

                    slotout1.append(slotout[j])
                    set1[j]=True

# Phase B selection
slotin2 = []
slotout2 = []

for i in range(len(theta1)):
    for j in range(int(number_of_coils)):
        if(len(slotin2)== int(number_of_coils)//3):
            break
        else:
            if thetai[j]== theta1[i]:
                if set1[j] ==False:
                    slotin2.append(slotin[j])
                    slotout2.append(slotout[j])
                    set1[j]=True


# Phase C selection
slotin3 = []
slotout3 = []

for i in range(len(theta1)):
    for j in range(int(number_of_coils)):
        if(len(slotin3)== int(number_of_coils)//3):
            break
        else:
            if thetai[j]== theta1[i]:
                if set1[j] ==False:
                    slotin3.append(slotin[j])

                    slotout3.append(slotout[j])
                    set1[j]=True
```

In [9]:

```
print('motor periodicity :',motor_periodicity)
print('number of spokes :', number_of_spokes)
print('coil pitch in mechanical degrees',coil_pitch_mech)
print('coil pitch in electrical degrees',coil_pitch_elec)
```

```python
print('coil span :',coil_span)
print('coils per phase :',coils_per_phase)
print('coil per pole',coils_per_pole)
print('phase group :', phase_group)
print('coil_number: ',coil_number)
print('theta: ',theta)
print('slotin: ',slotin)
print('slotout: ',slotout)
print('Phase A In :',slotin1)
print('Phase A Out :',slotout1)
print('Phase B In :',slotin2)
print('Phase B out :',slotout2)
print('Phase C In :',slotin3)
print('Phase C Out :',slotout3)
```

```
motor periodicity : 3
number of spokes : 12.0
coil pitch in mechanical degrees 20.0
coil pitch in electrical degrees 60.0
coil span : 6
coils per phase : 6.0
coil per pole 3.0
phase group : 3.0
coil_number:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
theta:  [0, 60, 120, -180, -120, -60, 0, 60, 120, -180, -120, -60, 0, 60, 120, -180,
-120, -60]
slotin:  [1, 2, 3, 4, 5, 6, 13, 14, 15, 16, 17, 18, 25, 26, 27, 28, 29, 30]
slotout:  [7, 8, 9, 10, 11, 12, 19, 20, 21, 22, 23, 24, 31, 32, 33, 34, 35, 36]
Phase A In : [1, 13, 25, 2, 14, 26]
Phase A Out : [7, 19, 31, 8, 20, 32]
Phase B In : [3, 15, 27, 4, 16, 28]
Phase B out : [9, 21, 33, 10, 22, 34]
Phase C In : [5, 17, 29, 6, 18, 30]
Phase C Out : [11, 23, 35, 12, 24, 36]
```