# Quantum Double Models using SageMath

## 0.1 Simple Mathematics using SageMath

In [6]:
```
# Finding the gcd of two numbers
gcd?
```

In [7]:
```
gcd(25,30)
```

Out[7]:
```
5
```

In [8]:
```
factor(625)
```

Out[8]:
```
5^4
```

In [9]:
```
factor(2435)
```

Out[9]:
```
5 * 487
```

In [11]:
```
M = matrix([[1,2,4],[1,3,5],[5,4,2]]);M
```

Out[11]:
```
[1 2 4]
[1 3 5]
[5 4 2]
```

In [12]:
```
M.inverse()
```

Out[12]:
```
[   7/6     -1    1/6]
[-23/12    3/2   1/12]
[ 11/12   -1/2  -1/12]
```

In [13]:
```
M.eigenvalues()
```

Out[13]:
```
[-3.583940313126412?, 0.3631206519422502?, 9.22081966118417?]
```

---

## 0.2 Group Theory in SageMath

In [15]:
```
Z2 = SymmetricGroup(2); Z2
```

Out[15]:
```
Symmetric group of order 2! as a permutation group
```

In [16]:
```
S3 = SymmetricGroup(3); S3
```

Out[16]:
```
Symmetric group of order 3! as a permutation group
```

In [17]:
```
Z2.is_subgroup(S3)
```

Out[17]:
```
True
```

In [18]:
```
S3.is_cyclic()
```

Out[18]:
```
False
```

In [19]:
```
Z2.is_abelian()
```

Out[19]:
```
True
```

---

## 0.3 Excitations, Ribbon operators, Ground states in Quantum Double Models

**Defining different models - Quantum Double of Z2, S3, D4**

```
In [1]: QDM_Toric = SymmetricGroup(2)
        QDM_Toric
```

```
Out[1]: Symmetric group of order 2! as a permutation group
```

```
In [2]: QDM_S3 = SymmetricGroup(3)
        QDM_S3
```

```
Out[2]: Symmetric group of order 3! as a permutation group
```

```
In [3]: QDM_D4 = DihedralGroup(4)
        QDM_D4
```

```
Out[3]: Dihedral group of order 8 as a permutation group
```

---

**Developing the machinery to compute the number of excitations.**

1. **Computing the centralizers of the conjugacy class of the group.**

```
In [4]: def centralizer_conjugacy_class_QDM_generic(QDM_group):
            cent_QDM_group = []
            for conj_class in QDM_group.conjugacy_classes():
                centralizer = QDM_group.centralizer(conj_class.an_element())
                cent_QDM_group.append(centralizer)
            return cent_QDM_group
```

```
In [5]: cent_toric = centralizer_conjugacy_class_QDM_generic(QDM_Toric)
        cent_toric
```

```
Out[5]: [Subgroup of (Symmetric group of order 2! as a permutation group) generated by [(1,2)],
         Subgroup of (Symmetric group of order 2! as a permutation group) generated by [(1,2)]]
```

```
In [6]: cent_s3 = centralizer_conjugacy_class_QDM_generic(QDM_S3)
        cent_s3
```

```
Out[6]: [Subgroup of (Symmetric group of order 3! as a permutation group) generated by [(2,3), (1,3)],
         Subgroup of (Symmetric group of order 3! as a permutation group) generated by [(1,2)],
         Subgroup of (Symmetric group of order 3! as a permutation group) generated by [(1,2,3)]]
```

```
In [7]: cent_d4 = centralizer_conjugacy_class_QDM_generic(QDM_D4)
        cent_d4
```

```
Out[7]: [Subgroup of (Dihedral group of order 8 as a permutation group) generated by [(1,2,3,4), (1,4)(
         Subgroup of (Dihedral group of order 8 as a permutation group) generated by [(2,4), (1,3)(2,4)
         Subgroup of (Dihedral group of order 8 as a permutation group) generated by [(1,2)(3,4), (1,3)
         Subgroup of (Dihedral group of order 8 as a permutation group) generated by [(1,2,3,4), (1,3)(
         Subgroup of (Dihedral group of order 8 as a permutation group) generated by [(1,2,3,4), (1,4)(
```

**2. The character table gives the trace of irreducible representations (but the trace is used at a later stage).**

```
In [8]: def character_table_centralizers(centralizers_generic_group):
            char_table = []
            for subgroup in centralizers_generic_group:
                char_table.append(subgroup.character_table())
            return char_table
```

```
In [9]: cent_toric_centralizer_character = character_table_centralizers(cent_toric)
        cent_toric_centralizer_character
```

```
Out[9]: [
        [ 1 -1]  [ 1 -1]
        [ 1  1], [ 1  1]
        ]
```

```
In [10]: cent_s3_centralizer_character_table = character_table_centralizers(cent_s3)
         cent_s3_centralizer_character_table
```

```
Out[10]: [
         [ 1 -1  1]          [         1          1          1]
         [ 2  0 -1]  [ 1 -1] [         1      zeta3 -zeta3 - 1]
         [ 1  1  1], [ 1  1], [         1 -zeta3 - 1      zeta3]
         ]
```

```
In [11]: cent_d4_centralizer_character_table = character_table_centralizers(cent_d4)
         cent_d4_centralizer_character_table
```

```
Out[11]: [
         [ 1  1  1  1  1]
         [ 1 -1 -1  1  1]  [ 1  1  1  1]  [ 1  1  1  1]
         [ 1 -1  1 -1  1]  [ 1 -1 -1  1]  [ 1 -1 -1  1]
         [ 1  1 -1 -1  1]  [ 1 -1  1 -1]  [ 1 -1  1 -1]
         [ 2  0  0  0 -2], [ 1  1 -1 -1], [ 1  1 -1 -1],

                                          [ 1  1  1  1  1]
         [     1      1      1      1]  [ 1 -1 -1  1  1]
         [     1     -1      1     -1]  [ 1 -1  1 -1  1]
         [     1 -zeta4     -1  zeta4]  [ 1  1 -1 -1  1]
         [     1  zeta4     -1 -zeta4], [ 2  0  0  0 -2]
         ]
```

**3. Computing the number of excitations by counting the number of rows in the character table.**

```
In [12]: def excitations_count(QDM_group):
             count = 0
             generic_centralizer_character_table = character_table_centralizers(centralizer_conjugacy_cl
             for char_table in generic_centralizer_character_table:
                 count += char_table.nrows()
             return count
```

```
In [13]: QDM_toric_excitations = excitations_count(QDM_Toric)
         QDM_toric_excitations
```

```
Out[13]: 4
```

In [14]:
```
QDM_S3_excitations = excitations_count(QDM_S3)
QDM_S3_excitations
```

Out[14]: 8

In [15]:
```
QDM_D4_excitations = excitations_count(QDM_D4)
QDM_D4_excitations
```

Out[15]: 22

---

**Developing the machinery to compute the excitations that condense on a given boundary**

1. **Computing the character related to the irreducible representation of the group.**

In [16]:
```
def character_excitation(G, conjugacy_class, g, h):
    k_h = 0
    for g_1 in G:
        if h*g_1 == g_1*conjugacy_class.an_element():
            k_h = g_1
            break
    if g*h == h*g and k_h != 0:
        return k_h^-1*g*k_h
    else:
        return 0
```

2. **Computing the character related to a particular boundary.**

In [17]:
```
def character_subgroup(G, subgroup, g, h):
    sum = 0
    if h*g == g*h:
        for g_1 in G:
            if g_1*g*g_1^-1 in subgroup and g_1*h*g_1^-1 in subgroup:
                sum = sum + 1
    return sum/len(subgroup)
```

3. **Computing the inner product terms of the above characters.**

In [18]:
```
def inner_product_of_characters(QDM_group, subgroup, conjugacy_class):
    inner_product_terms = []
    for g in QDM_group:
        for h in QDM_group:
            if character_subgroup(QDM_group, subgroup, g, h) != 0 and character_excitation(QDM_
                inner_product_terms.append([character_subgroup(QDM_group, subgroup, g, h), cha
    return inner_product_terms
```

In [19]:
```
inner_product_of_characters(QDM_S3, QDM_S3.subgroups()[5], QDM_S3.conjugacy_classes()[0])
```

Out[19]: [[1, ()], [1, (1,2)], [1, (1,2,3)], [1, (1,3,2)], [1, (2,3)], [1, (1,3)]]

$1 * tr_{\pi_i}(e) + 1 * tr_{\pi_i}(1,2) + 1 * tr_{\pi_i}(1,2,3) + 1 * tr_{\pi_i}(1,3,2) + 1 * tr_{\pi_i}(2,3) + 1 * tr_{\pi_i}(1,3)$

From the character table for $S_3$, and labelling each excitation

$$\begin{array}{cccccccc} \{e\} & \{\tau\} & \{\sigma\} & & & & \\ 1 & -1 & 1 & -> & tr_{\pi_2} & -> & B \\ 2 & 0 & -1 & -> & tr_{\pi_3} & -> & C \\ 1 & 1 & 1 & -> & tr_{\pi_1} & -> & A \end{array}$$

Therefore $A$ condenses on the boundary as the inner product is greater than zero, the others go to zero.

```
In [20]: inner_product_of_characters(QDM_S3, QDM_S3.subgroups()[5], QDM_S3.conjugacy_classes()[1])
```

```
Out[20]: [[1, ()], [1, ()], [1, ()], [1, (1,2)], [1, (1,2)], [1, (1,2)]]
```

$3 * tr_{\pi_i}(e) + 3 * tr_{\pi_i}(1,2)$
From the character table for $Z_2$, and labelling each excitation
$1 \quad -1 \quad tr_{\pi_2} \quad -> \quad E$
$1 \quad 1 \quad tr_{\pi_1} \quad -> \quad D$
Therefore $D$ condenses on the boundary as the inner product is greater than zero, the others go to zero.

```
In [21]: inner_product_of_characters(QDM_S3, QDM_S3.subgroups()[5], QDM_S3.conjugacy_classes()[2])
```

```
Out[21]: [[1, ()], [1, ()], [1, (1,2,3)], [1, (1,3,2)], [1, (1,3,2)], [1, (1,2,3)]]
```

$2 * tr_{\pi_i}(e) + 2 * tr_{\pi_i}(1,2,3) + 2 * tr_{\pi_i}(1,3,2)$
From the character table for $Z_3$, and labelling each excitation
$1 \quad 1 \quad 1 \quad tr_{\pi_1} \quad -> \quad F$
$1 \quad zeta3 \quad -zeta3 - 1 \quad tr_{\pi_2} \quad -> \quad G$
$1 \quad -zeta3 - 1 \quad zeta3 \quad tr_{\pi_3} \quad -> \quad H$
Therefore $F$ condenses on the boundary as the inner product is greater than zero, the others go to zero.
Hence, for the subgroup $K = G$, the excitations $A, D, F$ condense on the boundary.

---

Similarly varying the boundaries (different subgroups) and using the inner product, the excitations which condense on the boundary can be determined.

```
In [22]: def boundary_condensates(QDM_group, QDM_subgroup):
             total_inner_product_terms = []
             for conj_class in QDM_group.conjugacy_classes():
                 total_inner_product_terms.append(inner_product_of_characters(QDM_group, QDM_subgroup, c
             return total_inner_product_terms
```

Boundary condensates for the boundary indexed by $\{e, \tau\}$

```
In [23]: boundary_condensates(QDM_S3, QDM_S3.subgroups()[1])
```

```
Out[23]: [[[3, ()], [1, (1,2)], [1, (2,3)], [1, (1,3)]],
          [[1, ()], [1, ()], [1, ()], [1, (1,2)], [1, (1,2)], [1, (1,2)]],
          []]
```

Observing the character table list, $A, C, D$ condense given the boundary is indexed by $\{e, \tau\}$

---

**Construction of the ribbon operators for lattice with boundary** Given that the boundary is given by the boundary (subgroup $K$), the ribbon operator with an excitation in the bulk and the condensate on the boundary is given by
$T^{(k,g)} = \Sigma_{l \in K} F^{(lkl^{-1}, gl^{-1})}$ where $k \in K, g \in G$
Fixing the subgroup $K = \{e, \tau\}$, $(\{e, (2,3)\}$ for example)

```
In [24]: K = QDM_S3.subgroups()[1];K
```

```
Out[24]: Subgroup of (Symmetric group of order 3! as a permutation group) generated by [(2,3)]
```

```
In [25]: def ribbon_operator_constructs(QDM_group, subgroup):
             ribbon_operator_terms = []
             for k in subgroup:
                 for g in QDM_group:
                     for l in subgroup:
                         ribbon_operator_terms.append([k,g,l*k*l^-1, l*g^-1])
             return ribbon_operator_terms
         ribbon_operator_constructs(QDM_S3, K)
```

```
Out[25]: [[(), (), (), ()],
          [(), (), (), (2,3)],
          [(), (1,2), (), (1,2)],
          [(), (1,2), (), (1,2,3)],
          [(), (1,2,3), (), (1,3,2)],
          [(), (1,2,3), (), (1,3)],
          [(), (1,3,2), (), (1,2,3)],
          [(), (1,3,2), (), (1,2)],
          [(), (2,3), (), (2,3)],
          [(), (2,3), (), ()],
          [(), (1,3), (), (1,3)],
          [(), (1,3), (), (1,3,2)],
          [(2,3), (), (2,3), ()],
          [(2,3), (), (2,3), (2,3)],
          [(2,3), (1,2), (2,3), (1,2)],
          [(2,3), (1,2), (2,3), (1,2,3)],
          [(2,3), (1,2,3), (2,3), (1,3,2)],
          [(2,3), (1,2,3), (2,3), (1,3)],
          [(2,3), (1,3,2), (2,3), (1,2,3)],
          [(2,3), (1,3,2), (2,3), (1,2)],
          [(2,3), (2,3), (2,3), (2,3)],
          [(2,3), (2,3), (2,3), ()],
          [(2,3), (1,3), (2,3), (1,3)],
          [(2,3), (1,3), (2,3), (1,3,2)]]
```

$$T^{(e,e)} = F^{(e,e)} + F^{(e,(2,3))},$$
$$T^{(e,(1,2))} = F^{(e,(1,2))} + F^{(e,(1,2,3))},$$
$$T^{(e,(1,2,3))} = F^{(e,1,3,2)} + F^{(e,(1,3))},$$
$$T^{((2,3),e)} = F^{((2,3),e)} + F^{((2,3),(2,3))},$$
$$T^{((2,3),(1,2))} = F^{((2,3),(1,2))} + F^{((2,3),(1,2,3))},$$
$$T^{((2,3),(1,2,3)} = F^{((2,3),(1,3,2))} + F^{((2,3),(1,3))},$$

Similarly for various boundaries, various ribbon operators connecting the bulk to the boundary can be generated. It is observed that for every boundary (every subgroup) there are 6 unique ribbon operators connecting the bulk to boundary in the case of $S_3$

---

**Ground states with respect to different `T` operators on a cylinder with a single lattice (implying boundary on both sides of the lattice)** The lattice looks in the following way :

```
−g1——g1−
    |
    |
    g2
```

|
|
$-g_3$——$g_3-$

Eigenstates of $\Pi\{\Sigma\ (vertex\ operators)\}\ (face\ operators)T$ are the ground states of the lattice with a ribbon operator. In the above lattice $g_1 and g_3$ are restricted to the subgroup (identified as boundary). There are three conditions to be satisfied, fixing the boundary to be $\{e, \tau\}$, due to the ribbon operators $g_2$ is restricted to $\{e, (2,3)\}$, due to the face operators the relationship between $g_1, g_2, g_3$ is as follows $g_3 g_2 g_1 g_2^{-1} = e$, and finally due to the vertex operators $g_1, g_2, g_3$ get mapped to $k_u g_1 k_u^{-1}, k_d g_2 k_u^{-1}, k_d g_3 k_d^{-1}$ respectively, where $k_u, k_d \in K$

```
In [26]: def ground_state_terms(g1, g2, g3, ku,  kd):
             return ku*g1*ku^-1, kd*g2*ku^-1, kd*g3*kd^-1
```

```
In [27]: def ground_state_sum(condition_set, subgroup):
             s = []
             for g2 in condition_set[1]:
                 for g3 in subgroup:
                     for g1 in subgroup:
                         if condition_set[0]*g3*g2*condition_set[0]*g1 == g2:
                             s.append((condition_set[0]*g1,g2,condition_set[0]*g3))
                             for i in subgroup:
                                 for j in subgroup:
                                     s.append([ground_state_terms(condition_set[0]*g1, g2, condition_se
             return s
```

Observing that $T^{(e,e)} = F^{(e,e)} + F^{(e,(2,3))}$ the condition set is that $g_2 \in \{e, (2,3)\}$ similarly to determine the other ground states the condition set is required

```
In [28]: ground_state_sum([QDM_S3[0],[QDM_S3[0], QDM_S3[4]]], QDM_S3.subgroups()[1])
```

```
Out[28]: [((), (), ()),
          [((), (), ())],
          [((), (2,3), ())],
          [((), (2,3), ())],
          [((), (), ())],
          ((2,3), (), (2,3)),
          [((2,3), (), (2,3))],
          [((2,3), (2,3), (2,3))],
          [((2,3), (2,3), (2,3))],
          [((2,3), (), (2,3))],
          ((), (2,3), ()),
          [((), (2,3), ())],
          [((), (), ())],
          [((), (), ())],
          [((), (2,3), ())],
          ((2,3), (2,3), (2,3)),
          [((2,3), (2,3), (2,3))],
          [((2,3), (), (2,3))],
          [((2,3), (), (2,3))],
          [((2,3), (2,3), (2,3))]]
```

This implies for the operator $T^{(e,e)}$ :

| $Possible\ initial\ configuration$ | $Ground\ state$ |
|---|---|
| $(e,e,e)$ | $2*(e,e,e) + 2*(e,(2,3),e)$ |
| $((2,3),e,(2,3))$ | $2*((2,3),e,(2,3)) + 2*((2,3),(2,3),(2,3))$ |
| $(e,(2,3),e)$ | $2*(e,e,e) + 2*(e,(2,3),e)$ |
| $((2,3),(2,3),(2,3))$ | $2*((2,3),e,(2,3)) + 2*((2,3),(2,3),(2,3))$ |

```
In [29]: ground_state_sum([QDM_S3[0],[QDM_S3[1], QDM_S3[2]]], QDM_S3.subgroups()[1])
```

```
Out[29]: [((), (1,2), ()),
          [((), (1,2), ())],
          [((), (1,2,3), ())],
          [((), (1,3,2), ())],
          [((), (1,3), ())],
          ((), (1,2,3), ()),
          [((), (1,2,3), ())],
          [((), (1,2), ())],
          [((), (1,3), ())],
          [((), (1,3,2), ())]]
```

This implies for the operator $T^{(e,(1,2))}$ :

| *Possible initial configuration* | *Ground state* |
|---|---|
| $(e,(1,2),e)$ | $(e,(1,2),e)+(e,(1,2,3),e)+(e,(1,3,2),e)+(e,(1,3),e)$ |
| $(e,(1,2,3),e)$ | $(e,(1,2),e)+(e,(1,2,3),e)+(e,(1,3,2),e)+(e,(1,3),e)$ |

```
In [30]: ground_state_sum([QDM_S3[0],[QDM_S3[3], QDM_S3[5]]], QDM_S3.subgroups()[1])
```

```
Out[30]: [((), (1,3,2), ()),
          [((), (1,3,2), ())],
          [((), (1,3), ())],
          [((), (1,2), ())],
          [((), (1,2,3), ())],
          ((), (1,3), ()),
          [((), (1,3), ())],
          [((), (1,3,2), ())],
          [((), (1,2,3), ())],
          [((), (1,2), ())]]
```

This implies for the operator $T^{(e,(1,2,3))}$ :

| *Possible initial configuration* | *Ground state* |
|---|---|
| $(e,(1,3),e)$ | $(e,(1,2),e)+(e,(1,2,3),e)+(e,(1,3,2),e)+(e,(1,3),e)$ |
| $(e,(1,3,2),e)$ | $(e,(1,2),e)+(e,(1,2,3),e)+(e,(1,3,2),e)+(e,(1,3),e)$ |

```
In [31]: ground_state_sum([QDM_S3[4],[QDM_S3[0], QDM_S3[4]]], QDM_S3.subgroups()[1])
```

```
Out[31]: [((2,3), (), (2,3)),
          [((2,3), (), (2,3))],
          [((2,3), (2,3), (2,3))],
          [((2,3), (2,3), (2,3))],
          [((2,3), (), (2,3))],
          ((), (), ()),
          [((), (), ())],
          [((), (2,3), ())],
          [((), (2,3), ())],
          [((), (), ())],
          ((2,3), (2,3), (2,3)),
          [((2,3), (2,3), (2,3))],
          [((2,3), (), (2,3))],
          [((2,3), (), (2,3))],
          [((2,3), (2,3), (2,3))],
          ((), (2,3), ()),
          [((), (2,3), ())],
          [((), (), ())],
          [((), (), ())],
          [((), (2,3), ())]]
```

This implies for the operator $T^{((2,3),e)}$ :

| *Possible initial configuration* | *Ground state* |
|---|---|
| $(e,e,e)$ | $2*(e,e,e)+2*(e,(2,3),e)$ |
| $((2,3),e,(2,3))$ | $2*((2,3),e,(2,3))+2*((2,3),(2,3),(2,3))$ |
| $(e,(2,3),e)$ | $2*(e,e,e)+2*(e,(2,3),e)$ |
| $((2,3),(2,3),(2,3))$ | $2*((2,3),e,(2,3))+2*((2,3),(2,3),(2,3))$ |

```
In [32]: ground_state_sum([QDM_S3[4],[QDM_S3[1], QDM_S3[2]]], QDM_S3.subgroups()[1])

Out[32]: [((), (1,2), ()),
          [((), (1,2), ())],
          [((), (1,2,3), ())],
          [((), (1,3,2), ())],
          [((), (1,3), ())],
          ((), (1,2,3), ()),
          [((), (1,2,3), ())],
          [((), (1,2), ())],
          [((), (1,3), ())],
          [((), (1,3,2), ())]]
```

This implies for the operator $T^{((2,3),(1,2))}$ :

| *Possible initial configuration* | *Ground state* |
|---|---|
| $(e,(1,2),e)$ | $(e,(1,2),e)+(e,(1,2,3),e)+(e,(1,3,2),e)+(e,(1,3),e)$ |
| $(e,(1,2,3),e)$ | $(e,(1,2),e)+(e,(1,2,3),e)+(e,(1,3,2),e)+(e,(1,3),e)$ |

```
In [33]: ground_state_sum([QDM_S3[4],[QDM_S3[3], QDM_S3[5]]], QDM_S3.subgroups()[1])

Out[33]: [((), (1,3,2), ()),
          [((), (1,3,2), ())],
          [((), (1,3), ())],
          [((), (1,2), ())],
          [((), (1,2,3), ())],
          ((), (1,3), ()),
          [((), (1,3), ())],
          [((), (1,3,2), ())],
          [((), (1,2,3), ())],
          [((), (1,2), ())]]
```

This implies for the operator $T^{((2,3),(1,2,3))}$ :

| *Possible initial configuration* | *Ground state* |
|---|---|
| $(e,(1,3,2),e)$ | $(e,(1,2),e)+(e,(1,2,3),e)+(e,(1,3,2),e)+(e,(1,3),e)$ |
| $(e,(1,3),e)$ | $(e,(1,2),e)+(e,(1,2,3),e)+(e,(1,3,2),e)+(e,(1,3),e)$ |

Therefore, there are 3 unique ground states for all possible configurations of ribbon operators with an excitation at one end and condensate at the other