Symmetry analysis for the free energy of superconductors

Fanzheng Meng

I. SYMMETRY ANALYSIS

To understand the phenomenology of the high-temperature superconductor, we express the corresponding free energy F as a function of the superconducting order parameters using group theory. We start from the **Ginzburg-Landau functional**

$$f[\Delta] = \alpha_{ij}\Delta_i^*\Delta_j + \beta_{ijkl}\Delta_i^*\Delta_j^*\Delta_k\Delta_l + K_{ijkl}\partial_i\Delta_j^*\partial_k\Delta_l. \tag{S1}$$

Equation (S1) is invariant under the symmetry group:

$$G = G_0 \times U(1) \times T. \tag{S2}$$

Choosing particular representation, Δ transforms according to $\Delta \to R_{ij}(g_0)\exp(i\phi)\Delta_j$. Then the first term in Eq. S1 transforms according to

$$\alpha_{ij}R_{ik}(g)R_{jl}^*(g)\Delta_k^*\Delta_l = R_{ki}^\dagger(g)\alpha_{ij}R_{jl}(g)\Delta_k^*\Delta_l.$$

To be invariant, we should have

$$R_{ki}^\dagger(g)\alpha_{ij}R_{jl}(g) = \alpha_{kl},$$

or in another form

$$R(g)\alpha = \alpha R(g).$$

Schur's Lemma tells us that under an irreducible representation, α is proportional to a constant matrix such that $\alpha_{ij} = \alpha\delta_{ij}$. Under a reducible representation by choosing the appropriate basis, α_{ij} can be block-diagonalized. When we consider the temperature dependence of the matrix α_{ij}. At high temperature, the α_{ij} is positive definite and thus Δ_i should be all zero(only in this way, the free energy is lowest).As temperature turns lower, the α_{ij} cease to be definite.(consider α_1 firstly become nonpositive).Then we can write our $\Delta_i = \sum_{j=1}^{g}\eta_j x_i^{(j)}$, where $x^{(i)}$ is the eigenfunction of the α_1 (corresponding to a irreducible representation), and g is the dimensionality of the representation.Note that we neglect the components from other irreducible representation eigenfuctions.Also we can consider other irreducible representaions and conduct the same process above. That's to say we can only consider " a group thery problem".We denote the η_i as out order parameters and rewrite our equation S1:

$$f[\eta] = \alpha(T - T_c)\eta_i^*\eta_i + \beta_{ijkl}\eta_i^*\eta_j^*\eta_k\eta_l + K_{ijkl}\partial_i\eta_j^*\partial_k\eta_l + \cdot \tag{S3}$$

We should consider further the basis transformation:

$$\Delta_i = \sum_{j=1}^{g}\eta_j x_i^{(j)} \tag{S4}$$

My understanding is that:
We sould treat η_i as our new basis(order parameter) in place of Δ_i such that:

$$\eta_i = \sum_j (x_i^{(j)})^{-1}\Delta_j \tag{S5}$$

That's good because in such a basis, our function will be in a more convenient form as we can see(the first term has only $\eta_i^*\eta_i$).I will further consider this question later.
Our next question is how could we reduce the second and the third term to a conveninet form.Let's consider how we reduce the first term to insire us.
$\alpha_{ij}\Delta_i^*\Delta_j \quad \xrightarrow{\eta_i=\sum_j (x_i^{(j)})^{-1}\Delta_j} \quad \alpha(T-T_c)\eta_i^*\eta_i,$ $\quad \alpha(T-T_c)$ is a constant and $\eta_i^*\eta_i$ must be invariant under transformation(fantastic).So how we get the invariant $\eta_i^*\eta_i$. The answer is by using linear combination of the older Δ_i and Δ_i^*.Or in other words by using linear combination of $\Delta_i^*\Delta_j$,we get $\eta_i^*\eta_i$.
Δ_i transform under $R_{ik}(g)$, which correspond to a particular representation.Different representation will induce different transformation matrx(different dimensionality , different elements).
$\Delta_i^*\Delta_j$ transform under the direct product of representations $\Gamma^*\otimes\Gamma = \Gamma_1 \oplus \cdots$, which can be rewrite as a direct sum of irreducible representations.Here Γ_1 is the identity representation.Of couse, we can't hope there always be a Γ_1 in the direct sum, it depends. But only when some Γ_1 appear, there can be some linear combinations that are invariant under transformations.
To get the appropriate linear combination, we should get the **C-G coefficients**.
For a direct product of irreducible representations:

$$\Gamma^{(\mu)} \otimes \Gamma^{(\nu)} = \sum_\sigma a_\sigma \Gamma^{(\sigma)} \tag{S6}$$

if the basis of $\Gamma^{(\mu)}$ and $\Gamma^{(\nu)}$ is $|\mu, j\rangle$ and $|\nu, k\rangle$ then the basis of $\Gamma^{(\mu)}\otimes\Gamma^{(\nu)}$ is $|\mu, j\rangle\,|\nu, k\rangle = |\mu, \nu; j, k\rangle$ and the basis of the right hand side is $|\sigma, s; l\rangle$, where σ denote the irreducible representation, s denote the particular representation of multiple terms in the sum.Then we have:

$$|\sigma s; l\rangle = \sum_{jk} \begin{pmatrix} h & v & | & r & s. \\ j & k & | & l & \end{pmatrix} |\mu, \nu; j, k\rangle \tag{S7}$$

To write (3) in a easy form we can still use group theory method.
For quartic term, it transforms under operators in G_0 according to the direct product representation:

$$\Gamma^*\otimes\Gamma^*\otimes\Gamma\otimes\Gamma = \sum_i n_i\Gamma_i \tag{S8}$$

And the independent quartic order parameter terms can be split out from the decomposition of the reducible direct product representation.

A. Tetragonal symmetry

Take section (2.3.2. Tetragonal symmetry)[2] as an example.
the product like $\eta_i\eta_j$ transform according to :

$$E_g \otimes E_g = A_{1g} \oplus A_{2g} \oplus B_{1g} \oplus B_{2g} \tag{S9}$$

1. *Orthogonal method*

By **orthogonality method**

$$\sum_s \begin{pmatrix} \mu\nu & \mid & \sigma s \\ ik & \mid & m \end{pmatrix} \begin{pmatrix} \sigma s & \mid & \mu\nu \\ n & \mid & jl \end{pmatrix} = \frac{d_\sigma}{g} \sum_{R\in\mathcal{G}} \Gamma^{(\mu)}_{ij}(R)\Gamma^{(\nu)}_{kl}(R)\Gamma^{(\sigma)*}_{mn}(R). \tag{S10}$$

we determine the CGCs with $m = n, i = j$ and $k = l$:

$$\left|\begin{pmatrix} \mu\nu & \mid & \sigma s \\ ik & \mid & m \end{pmatrix}\right|^2 = \frac{d_\sigma}{g} \sum_{R\in\mathcal{G}} \Gamma^{(\mu)}_{ii}(R)\Gamma^{(\nu)}_{kk}(R)\Gamma^{(\sigma)*}_{mm}(R).$$

Hew give the representation of E_g.

$$^{(5)}\Gamma(E) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad ^{(5)}\Gamma(C_{2z}) = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}, \quad ^{(5)}\Gamma(C_{4z}) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \quad ^{(5)}\Gamma(C_{4z}^{-1}) = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix},$$

$$^{(5)}\Gamma(C_{2x}) = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \quad ^{(5)}\Gamma(C_{2y}) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad ^{(5)}\Gamma(C_{2xy}) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, {}^{(5)}\Gamma(C_{2\bar{x}\bar{y}}) = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}.$$

$$^{(5)}\Gamma(i) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad ^{(5)}\Gamma(iC_{2z}) = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}, \quad ^{(5)}\Gamma(iC_{4z}) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \quad ^{(5)}\Gamma(iC_{4z}^{-1}) = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix},$$

$$^{(5)}\Gamma(iC_{2x}) = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \quad ^{(5)}\Gamma(iC_{2y}) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad ^{(5)}\Gamma(iC_{2xy}) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, {}^{(5)}\Gamma(iC_{2\bar{x}\bar{y}}) = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}.$$

Then with the **Orthogonal Method**, we can get:

$$\left(\begin{array}{c|c} 55 & 1 \\ ik & 1 \end{array}\right) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \left(\begin{array}{c|c} 55 & 2 \\ ik & 1 \end{array}\right) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$
$$\left(\begin{array}{c|c} 55 & 3 \\ ik & 1 \end{array}\right) = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \quad \left(\begin{array}{c|c} 55 & 4 \\ ik & 1 \end{array}\right) = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Thus the basis are

$$\eta_1\eta_1 + \eta_2\eta_2 \qquad \eta_1\eta_1 - \eta_2\eta_2 \qquad \eta_1\eta_2 + \eta_2\eta_1 \qquad \eta_1\eta_2 - \eta_2\eta_1$$

And the quartic term is $(E_g \otimes E_g)^* \otimes (E_g \otimes E_g)$ such that:

$$(A_{1g} \oplus A_{2g} \oplus B_{1g} \oplus B_{2g})^* \otimes (A_{1g} \oplus A_{2g} \oplus B_{1g} \oplus B_{2g}) = 4A_{1g} \oplus \cdots \tag{S11}$$

Note that

$$A_{1g}\otimes A_{1g}=A_{1g} \qquad A_{2g}\otimes A_{2g}=A_{1g}$$
$$B_{1g}\otimes B_{1g}=A_{1g} \qquad B_{2g}\otimes B_{2g}=A_{1g}$$

Then the four invariants are:

$$|\eta_1\eta_1+\eta_2\eta_2|^2, \qquad |\eta_1\eta_1-\eta_2\eta_2|^2, \qquad |\eta_1\eta_2+\eta_2\eta_1|^2 \qquad |\eta_1\eta_2-\eta_2\eta_1|^2=0$$

2. ***Projection Method***

There is another view of point to see this question.
The basis of representation E_g is η_1 and η_2,which transform as the matrices of E_g. The basis of representation $E_g\otimes E_g^*$ is $\{\eta_1\eta_1^*,\eta_1\eta_2^*,\eta_2\eta_1^*,\eta_2\eta_2^*\}$, which transform like the direct product matrices of E_g and E_g^*.The projection operator method is used to extract the basis functions of $A_{1g},A_{2g},B_{1g},B_{2g}$ as a combination of the basis of $E_g\otimes E_g^*$.
We apply the projection operator:

$$^{(\sigma)}P_{ii}=\frac{d_\sigma}{g}\sum_{R\in G}{}^{(\sigma)}\Gamma_{ii}(R^{-1})\hat{\mathbf{R}}_{outer} \tag{S12}$$

which in this case is a 4×4 matrix.The R_{outer} is the outer product of the matrix representation of E_g and E_g^*.Then we apply the matrix to a arbitrary 4-dimensional colomn vector and then get the appropriate colomn vector that show the correct combination of the four basis.(Note that we take $\{\eta_1\eta_1^*,\eta_1\eta_2^*,\eta_2\eta_1^*,\eta_2\eta_2^*\}$ as our basis).
I will show you more explicitly below:
I write a python code to do the trivial computation:

```python
import sympy as sp
import numpy as np
# Define all 16 symmetry operations for E_g
symmetry_operations_Eg = {
     "E": np.array([[1, 0], [0, 1]]),
    "C2z": np.array([[-1, 0], [0, -1]]),
    "C4z": np.array([[0, 1], [-1, 0]]),
    "C4z_inv": np.array([[0, -1], [1, 0]]),
    "C2x": np.array([[-1, 0], [0, 1]]),
    "C2y": np.array([[1, 0], [0, -1]]),
    "C2xy": np.array([[0, 1], [1, 0]]),
    "C2x_y": np.array([[0, -1], [-1, 0]]),
    "i": np.array([[1, 0], [0, 1]]),
    "iC2z": np.array([[-1, 0], [0, -1]]),
    "iC4z": np.array([[0, 1], [-1, 0]]),
    "iC4z_inv": np.array([[0, -1], [1, 0]]),
    "iC2x": np.array([[-1, 0], [0, 1]]),
    "iC2y": np.array([[1, 0], [0, -1]]),
    "iC2xy": np.array([[0, 1], [1, 0]]),
    "iC2x_y": np.array([[0, -1], [-1, 0]]),
}

character_table = {
    "A1g": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
    "A2g": [1, 1, 1, 1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1],
    "B1g": [1, 1, -1, -1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1, -1, -1],
    "B2g": [1,1]+[-1]*4+[1]*4+[-1]*4+[1,1],
    "A1u": [1]*8+[-1]*8,
    "A2u": [1]*4+[-1]*8+[1]*4,
    "B1u": [1,1,-1,-1,1,1]+[-1]*4+[1]*2+[-1]*2+[1]*2,
    "B2u": [1,1]+[-1]*4+[1,1]+[-1]*2+[1]*4+[-1]*2
}

# Group order
order_of_group = 16

# Function to calculate the projection matrix
def calculate_projection_matrix(irrep, character_table, symmetry_operations_1, symmetry_operations_2):
    projection_matrix = sp.zeros(4, 4)
    for i, op in enumerate(symmetry_operations_1.keys()):
        char = character_table[irrep][i]
        T1 = symmetry_operations_1[op]
        T2 = symmetry_operations_2[op]
        direct_product = sp.Matrix([
            [T1[0, 0] * T2[0, 0], T1[0, 0] * T2[0, 1], T1[0, 1] * T2[0, 0], T1[0, 1] * T2[0, 1]],
            [T1[0, 0] * T2[1, 0], T1[0, 0] * T2[1, 1], T1[0, 1] * T2[1, 0], T1[0, 1] * T2[1, 1]],
            [T1[1, 0] * T2[0, 0], T1[1, 0] * T2[0, 1], T1[1, 1] * T2[0, 0], T1[1, 1] * T2[0, 1]],
            [T1[1, 0] * T2[1, 0], T1[1, 0] * T2[1, 1], T1[1, 1] * T2[1, 0], T1[1, 1] * T2[1, 1]],
    ])
        projection_matrix += char * direct_product
    projection_matrix /= order_of_group
    return projection_matrix


projection_matrix_A1g = calculate_projection_matrix("A1g", character_table, symmetry_operations_Eg, symmetry_operations_Eg)
projection_matrix_A2g = calculate_projection_matrix("A2g", character_table, symmetry_operations_Eg, symmetry_operations_Eg)
```

```
projection_matrix_B1g = calculate_projection_matrix("B1g", character_table,
    symmetry_operations_Eg, symmetry_operations_Eg)
projection_matrix_B2g = calculate_projection_matrix("B2g", character_table,
    symmetry_operations_Eg, symmetry_operations_Eg)
print("Projection Matrix for A1g,A2g,B1g,B2g:")
# 2* is just for normalization
sp.pprint(2*projection_matrix_A1g)
print('\n')
sp.pprint(2*projection_matrix_A2g)
print('\n')
sp.pprint(2*projection_matrix_B1g)
print('\n')
sp.pprint(2*projection_matrix_B2g)
```

the output of the program is:

Projection Matrix for A1g , A2g, B1g, B2g:

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Thus the basis are

$$\eta_1\eta_1 + \eta_2\eta_2 \qquad \eta_1\eta_1 - \eta_2\eta_2 \qquad \eta_1\eta_2 + \eta_2\eta_1 \qquad \eta_1\eta_2 - \eta_2\eta_1$$

Subsequent analysis is the same as in the Orthogonal Method.

3. gradient terms

Note that ∂_i transforms like $E_u \oplus A_{2\mu}$ because the basis of E_u and $A_{2\mu}$ is (x, y) and z.Thus $\partial_i \eta_j$ transforms like $(E_\mu \oplus A_{2\mu}) \otimes E_g$.

$$(E_\mu \oplus A_{2\mu}) \otimes E_g = A_{1\mu} \oplus A_{2\mu} \oplus B_{1\mu} \oplus B_{2\mu} \oplus E_\mu \tag{S13}$$

Thus $\partial_i \eta_j \partial_k \eta_l^*$ transform like:

$$(A_{1\mu} \oplus A_{2\mu} \oplus B_{1\mu} \oplus B_{2\mu} \oplus E_\mu) \otimes (A_{1\mu} \oplus A_{2\mu} \oplus B_{1\mu} \oplus B_{2\mu} \oplus E_\mu)^* = (4+1)A_{1g} \oplus \cdots \tag{S14}$$

The first four A_{1g} are derived from:

$$A_{1u} \otimes A_{1u} = A_{2u} \otimes A_{2u} = B_{1u} \otimes B_{1u} = B_{2u} \otimes B_{2u} = A_{1g} \tag{S15}$$

the fifth A_{1g} derived from:

$$E_u \otimes E_u = A_{1g} \oplus A_{2g} \oplus B_{1g} \oplus B_{2g} \tag{S16}$$

We start from equation S13. First we note that:

$$A_{2u} \otimes E_g = E_u$$

Then the basis of E_u is $(\partial_z \eta_1, \partial_z \eta_2)$.Given S16 and the projection matrix of A_{1g} (I will give the prgram calculate this matrx later) we can get the first invariant combination: $|\partial_z \eta_1|^2$+$|\partial_z \eta_2|^2$.
In case you want to know, the projection matrix of A_{1g} is:

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

In equation S13, we have:

$$E_u \otimes E_g = A_{1\mu} \oplus A_{2\mu} \oplus B_{1\mu} \oplus B_{2\mu} \tag{S17}$$

The basis of $E_u \otimes E_g$ is $(\partial_x \eta_1, \partial_x \eta_2, \partial_y \eta_1, \partial_y \eta_2)$. We calculate the projection matrix of $A_{1u}, A_{2u}, B_{1u}, B_{2u}$ and get the basis are separately:

$$\partial_x \eta_2 - \partial_y \eta_1 \qquad \partial_x \eta_1 + \partial_y \eta_2 \tag{S18}$$

$$\partial_x \eta_2 + \partial_y \eta_1 \qquad \partial_x \eta_1 - \partial_y \eta_2 \tag{S19}$$

Thus the invariant components are:

$$|\partial_x \eta_2 - \partial_y \eta_1|^2 \qquad |\partial_x \eta_1 + \partial_y \eta_2|^2 \tag{S20}$$

$$|\partial_x \eta_2 + \partial_y \eta_1|^2 \qquad |\partial_x \eta_1 - \partial_y \eta_2|^2 \tag{S21}$$

$$|\partial_x\eta_2 - \partial_y\eta_1|^2 = |\partial_x\eta_2|^2+|\partial_y\eta_1|^2-(\partial_x\eta_2\partial_y\eta_1^* + c.c) \tag{S22}$$

$$|\partial_x\eta_1 + \partial_y\eta_2|^2 = |\partial_x\eta_1|^2+|\partial_y\eta_2|^2+(\partial_x\eta_1\partial_y\eta_2^* + c.c) \tag{S23}$$

$$|\partial_x\eta_2 + \partial_y\eta_1|^2 = |\partial_x\eta_2|^2+|\partial_Y\eta_1|^2+(\partial_x\eta_2\partial_y\eta_1^* + c.c) \tag{S24}$$

$$|\partial_x\eta_1 - \partial_y\eta_2|^2 = |\partial_x\eta_1|^2+|\partial_y\eta_2|^2-(\partial_x\eta_1\partial_y\eta_2^* + c.c) \tag{S25}$$

We choose the linear combination of the four invariant components:

$$|\partial_x\eta_2|^2+|\partial_y\eta_1|^2 \qquad |\partial_x\eta_1|^2+|\partial_y\eta_2|^2 \tag{S26}$$

$$(\partial_x\eta_2\partial_y\eta_1^* + c.c) \qquad (\partial_x\eta_1\partial_y\eta_2^* + c.c) \tag{S27}$$

Finally the **Ginzburg-Landau** free energy is therefore of the following form:

$$f[\eta_1,\eta_2] = \alpha(T-T_c)\eta_i^*\eta_i + \beta_1(\eta_i\eta_i)^2 + \beta_2|\eta_i\eta_i|^2+\beta_3(|\eta_1|^4+|\eta_2|^4) \tag{S28}$$

$$+\frac{\hbar^2}{2m_1'}(|\partial_x\eta_1|^2+|\partial_y\eta_2|^2) + \frac{\hbar^2}{2m_1''}(|\partial_y\eta_1|^2+|\partial_x\eta_2|^2) \tag{S29}$$

$$+\frac{\hbar^2}{2m_2}(|\partial_x\eta_1|^2+|\partial_z\eta_2|^2) \tag{S30}$$

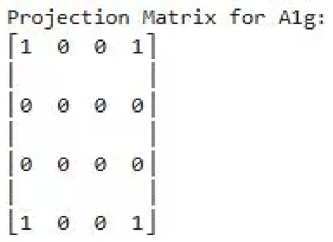
$$+\frac{\hbar^2}{2m_3'}(\partial_x\eta_2^*\partial_y\eta_1 + c.c) + \frac{\hbar^2}{2m_3''}(\partial_x\eta_1^*\partial_y\eta_2 + c.c) \tag{S31}$$

The program that calculates the projection matrix:

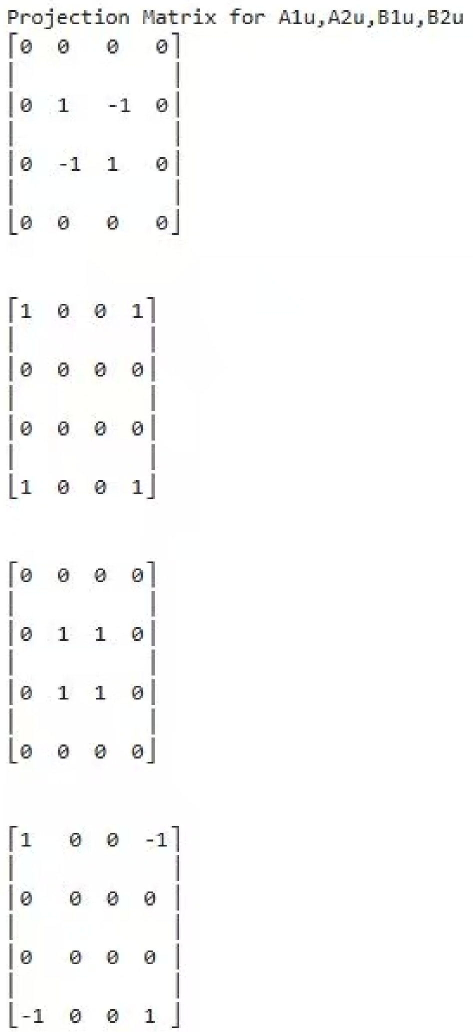
```
import sympy as sp
import numpy as np
# Define all 16 symmetry operations for E_g and E_u
symmetry_operations_Eg = {
     "E": np.array([[1, 0], [0, 1]]),
    "C2z": np.array([[-1, 0], [0, -1]]),
    "C4z": np.array([[0, 1], [-1, 0]]),
    "C4z_inv": np.array([[0, -1], [1, 0]]),
    "C2x": np.array([[-1, 0], [0, 1]]),
    "C2y": np.array([[1, 0], [0, -1]]),
    "C2xy": np.array([[0, 1], [1, 0]]),
    "C2x_y": np.array([[0, -1], [-1, 0]]),
    "i": np.array([[1, 0], [0, 1]]),
    "iC2z": np.array([[-1, 0], [0, -1]]),
    "iC4z": np.array([[0, 1], [-1, 0]]),
    "iC4z_inv": np.array([[0, -1], [1, 0]]),
    "iC2x": np.array([[-1, 0], [0, 1]]),
    "iC2y": np.array([[1, 0], [0, -1]]),
    "iC2xy": np.array([[0, 1], [1, 0]]),
    "iC2x_y": np.array([[0, -1], [-1, 0]]),
}

symmetry_operations_Eu = {
   "E": np.array([[1, 0], [0, 1]]),
    "C2z": np.array([[-1, 0], [0, -1]]),
    "C4z": np.array([[0, 1], [-1, 0]]),
    "C4z_inv": np.array([[0, -1], [1, 0]]),
    "C2x": np.array([[1, 0], [0, -1]]),
    "C2y": np.array([[-1, 0], [0, 1]]),
    "C2xy": np.array([[0, -1], [-1, 0]]),
    "C2x_y": np.array([[0, 1], [1, 0]]),
    "i": np.array([[-1, 0], [0, -1]]),
    "iC2z": np.array([[1, 0], [0, 1]]),
    "iC4z": np.array([[0, -1], [1, 0]]),
    "iC4z_inv": np.array([[0, 1], [-1, 0]]),
    "iC2x": np.array([[-1, 0], [0, 1]]),
    "iC2y": np.array([[1, 0], [0, -1]]),
    "iC2xy": np.array([[0, 1], [1, 0]]),
    "iC2x_y": np.array([[0, -1], [-1, 0]]),
}


character_table = {
    "A1g": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
    "A2g": [1, 1, 1, 1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1],
    "B1g": [1, 1, -1, -1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1, -1, -1],
    "B2g": [1,1]+[-1]*4+[1]*4+[-1]*4+[1,1],
    "A1u": [1]*8+[-1]*8,
    "A2u": [1]*4+[-1]*8+[1]*4,
    "B1u": [1,1,-1,-1,1,1]+[-1]*4+[1]*2+[-1]*2+[1]*2,
    "B2u": [1,1]+[-1]*4+[1,1]+[-1]*2+[1]*4+[-1]*2
}


# Group order
order_of_group = 16

# Function to calculate the projection matrix
def calculate_projection_matrix(irrep, character_table, symmetry_operations_1,
    symmetry_operations_2):
    projection_matrix = sp.zeros(4, 4)
    for i, op in enumerate(symmetry_operations_1.keys()):
        char = character_table[irrep][i]
        T1 = symmetry_operations_1[op]
        T2 = symmetry_operations_2[op]
        direct_product = sp.Matrix([
            [T1[0, 0] * T2[0, 0], T1[0, 0] * T2[0, 1], T1[0, 1] * T2[0, 0], T1[0, 1]
                 * T2[0, 1]],
            [T1[0, 0] * T2[1, 0], T1[0, 0] * T2[1, 1], T1[0, 1] * T2[1, 0], T1[0, 1]
```

```
                * T2[1, 1]],
            [T1[1, 0] * T2[0, 0], T1[1, 0] * T2[0, 1], T1[1, 1] * T2[0, 0], T1[1, 1]
                * T2[0, 1]],
            [T1[1, 0] * T2[1, 0], T1[1, 0] * T2[1, 1], T1[1, 1] * T2[1, 0], T1[1, 1]
                * T2[1, 1]],
        ])
        projection_matrix += char * direct_product
    projection_matrix /= order_of_group
    return projection_matrix


projection_matrix_A1g = calculate_projection_matrix("A1g", character_table,
    symmetry_operations_Eu, symmetry_operations_Eu)
print("Projection Matrix for A1g:")
#2* just for normalization
sp.pprint(2*projection_matrix_A1g)
print('\n')
projection_matrix_A1u = calculate_projection_matrix("A1u", character_table,
    symmetry_operations_Eu, symmetry_operations_Eg)
projection_matrix_A2u = calculate_projection_matrix("A2u", character_table,
    symmetry_operations_Eu, symmetry_operations_Eg)
projection_matrix_B1u = calculate_projection_matrix("B1u", character_table,
    symmetry_operations_Eu, symmetry_operations_Eg)
projection_matrix_B2u = calculate_projection_matrix("B2u", character_table,
    symmetry_operations_Eu, symmetry_operations_Eg)
print("Projection Matrix for A1u,A2u,B1u,B2u")
sp.pprint(2*projection_matrix_A1u)
print('\n')
sp.pprint(2*projection_matrix_A2u)
print('\n')
sp.pprint(2*projection_matrix_B1u)
print('\n')
sp.pprint(2*projection_matrix_B2u)
print('\n')
```

The outcome is as below:



```
Projection Matrix for A1g:
⎡1  0  0  1⎤
⎢          ⎥
⎢0  0  0  0⎥
⎢          ⎥
⎢0  0  0  0⎥
⎢          ⎥
⎣1  0  0  1⎦
```



```
Projection Matrix for A1u,A2u,B1u,B2u
⎡0  0   0   0⎤
⎢            ⎥
⎢0  1   -1  0⎥
⎢            ⎥
⎢0  -1  1   0⎥
⎢            ⎥
⎣0  0   0   0⎦

⎡1  0  0  1⎤
⎢          ⎥
⎢0  0  0  0⎥
⎢          ⎥
⎢0  0  0  0⎥
⎢          ⎥
⎣1  0  0  1⎦

⎡0  0  0  0⎤
⎢          ⎥
⎢0  1  1  0⎥
⎢          ⎥
⎢0  1  1  0⎥
⎢          ⎥
⎣0  0  0  0⎦

⎡1   0  0  -1⎤
⎢            ⎥
⎢0   0  0  0 ⎥
⎢            ⎥
⎢0   0  0  0 ⎥
⎢            ⎥
⎣-1  0  0  1 ⎦
```

II. TRIPLET STATES(WEAK SPIN-ORBIT) OF ORTHORHOMBIC AND TETRAGONAL SYMMETRY

Once we get the singlet states, it's easy to generalize singlet states to triplet states as below:

A. Orthorhombic symmetry

The singlet state of Orthorhombic symmetry is $\eta\eta\eta^*\eta^*$.The triplet states take the form $\eta_u\eta_u\eta_v^*\eta_v^*$ and $\eta_u\eta_v\eta_u^*\eta_v^*$.The gradient term takes the form $\partial_i\eta_u\partial_j\eta_u^*$.

B. Tetragonal symmetry

For singlet term, we have:

$$\eta_i^*\eta_i\eta_j^*\eta_j \qquad \eta_i\eta_i\eta_j^*\eta_j^* \qquad \epsilon_{ijk}\eta_i\eta_i\eta_j^*\eta_j^* \qquad \epsilon_{ijk}\eta_i\eta_i^*\eta_j\eta_j^* \tag{S32}$$

where the second and third term should be understood as $\epsilon_{ij3}\cdot\cdots$.
The triplet states take the form:

$$\eta_{ui}^*\eta_{ui}\eta_{vj}^*\eta_{vj} \qquad \eta_{ui}\eta_{ui}\eta_{vj}^*\eta_{vj}^* \qquad \epsilon_{ijk}\eta_{ui}\eta_{ui}\eta_{vj}^*\eta_{vj}^* \qquad \epsilon_{ijk}\eta_{ui}^*\eta_{ui}\eta_{vj}^*\eta_{vj} \tag{S33}$$

$$\eta_{ui}^*\eta_{vi}\eta_{vj}^*\eta_{uj} \qquad \eta_{ui}\eta_{vi}\eta_{uj}^*\eta_{vj}^* \qquad \epsilon_{ijk}\eta_{ui}\eta_{vi}\eta_{uj}^*\eta_{vj}^* \qquad \epsilon_{ijk}\eta_{ui}^*\eta_{vi}\eta_{vj}^*\eta_{uj} \tag{S34}$$

Note that the forth term is zero.

Appendix A

1. Schur's Lemma(copied from Hassani's text book)

Lemma A.1 (Schur's lemma) Let $T : G \to GL(\mathcal{V})$ and $T' : G \to GL(\mathcal{V}')$ be irreducible representations of G. If $A \in \mathcal{L}(\mathcal{V}, \mathcal{V}')$ is such that

$$AT_g = T'_g A \quad \forall g \in G, \tag{S1}$$

then either A is an isomorphism (i.e., T is equivalent to T'), or $A = 0$.

Proof Let $|a\rangle \in \ker A$. Then

$$AT_g|a\rangle = T'_g A \underbrace{|a\rangle}_{=0} = 0 \quad \Longrightarrow \quad T_g|a\rangle \in \ker A \quad \forall g \in G.$$

It follows that $\ker A$, a subspace of $\mathcal{V}$, is invariant under T. Irreducibility of T implies that either $\ker A = \mathcal{V}$, or $\ker A = 0$. The first case asserts that A is the zero linear transformation; the second case implies that A is injective.

Similarly, let $|b\rangle \in A(\mathcal{V})$. Then $|b\rangle = A|x\rangle$ for some $|x\rangle \in \mathcal{V}$:

$$T'_g|b\rangle = T'_g A|x\rangle = AT_g|x\rangle \in A(\mathcal{V}) \quad \forall g \in G.$$

It follows that $A(\mathcal{V})$, a subspace of $\mathcal{V}'$, is invariant under T'. Irreducibility of T' implies that either $A(\mathcal{V}) = 0$, or $A(\mathcal{V}) = \mathcal{V}'$. The first case is consistent with the first conclusion drawn above: $\ker A = \mathcal{V}$. The second case asserts that A is surjective. Combining the two results, we conclude that A is either the zero operator or an isomorphism.

Lemma A.2 Let $T : G \to GL(\mathcal{V})$ be an irreducible representation of G. If $A \in \mathcal{L}(\mathcal{V})$ is such that $AT_g = T_g A$ for all $g \in G$, then $A = \lambda \mathbf{1}$.

Proof Replacing $\mathcal{V}'$ with $\mathcal{V}$ in Lemma A.1, we conclude that $A = 0$ or A is an isomorphism of $\mathcal{V}$. In the first case, $\lambda = 0$. In the second case, A must have a nonzero eigenvalue λ and at least one eigenvector. It follows that the operator $A - \lambda\mathbf{1}$ commutes with all T_g's and it is not an isomorphism (*why not?*). Therefore, it must be the zero operator. □

2. Orthogonal methods for C-G coefficients

First, we have:

$$^{(\mu\nu)}\mathcal{M}^{-1}\,^{(\mu\otimes\nu)}\Gamma^{(\mu\nu)}\mathcal{M} = \begin{bmatrix} ^{(\sigma)}\Gamma^{1} & & & \\ & \ddots & & \\ & & ^{(\sigma)}\Gamma^{\langle\mu\nu|\sigma\rangle} & \\ & & & \ddots \end{bmatrix} \tag{S2}$$

Where ${}^{(\mu\nu)}M$ is the unitary matrix that reduces ${}^{(\mu\otimes\nu)}\Gamma$ (the direct product of irreducible representation ${}^{(\mu)}\Gamma$ and ${}^{(\nu)}\Gamma$) into block-diagonalized form.The $\langle\mu\nu|\sigma\rangle$ is the number of constitute ${}^{(\sigma)}\Gamma$ of ${}^{(\mu\otimes\nu)}\Gamma$, which can be easily got from analysis of character table.
Thus we have:

$$ {}^{(\mu\otimes\nu)}\Gamma = {}^{(\mu\nu)} M \sum_{\sigma}^{\otimes} \langle\mu\nu|\sigma\rangle^{(\sigma)}\, \Gamma^{(\mu\nu)} M^{-1} \tag{S3} $$

or more explicitly

$$ \begin{aligned} {}^{(\mu\otimes\nu)}\Gamma_{ik,jl}(R) &= {}^{(\mu)}\,\Gamma_{ij}(R)^{(\nu)}\Gamma_{kl}(R) \\ &= \sum_{\sigma srt} \left(\begin{array}{c|c} \mu\nu & \sigma s \\ ik & r \end{array}\right)^{(\sigma)} \Gamma_{rt}(R) \left(\begin{array}{c|c} \sigma s & \mu\nu \\ t & jl \end{array}\right) \end{aligned} $$

Multiplying by ${}^{(\sigma)}\Gamma^{*}_{mn}(R)$ and summing over $R \in G$, we obtain

$$ \sum_{s} \left(\begin{array}{c|c} \mu\nu & \sigma s \\ ik & m \end{array}\right) \left(\begin{array}{c|c} \sigma s & \mu\nu \\ n & jl \end{array}\right) = \frac{d_\sigma}{g} \sum_{R\in G} {}^{(\mu)}\Gamma_{ij}(R)^{(\nu)}\Gamma_{kl}(R)^{(\sigma)}\Gamma^{*}_{mn}(R) \tag{S4} $$

[1] M. El-Batanouny and F. Wooten. *Symmetry and Condensed Matter Physics.* Cambridge University Press, 2008.

[2] Annett, James F. "Symmetry of the order parameter for high-temperature superconductivity." Advances in Physics 39.2 (1990): 83-126.

[3] Rykhlinskaya, Katja, and Stephan Fritzsche. "Generation of Clebsch–Gordan coefficients for the point and double groups." Computer physics communications 174.11 (2006): 903-913.