6.S966: Exam 2, Spring 2025

Do not tear exam booklet apart!

- This is a closed book exam. One page (8 1/2 in. by 11 in) of notes, front and back, are permitted. Calculators are not permitted.
- The total exam time is 1 hours and 50 minutes.
- The problems are not necessarily in any order of difficulty.
- Record all your answers in the places provided. If you run out of room for an answer, continue on a blank page and mark it clearly.
- If a question seems vague or under-specified to you, make an assumption, write it down, and solve the problem given your assumption.
- If you absolutely *have* to ask a question, come to the front.
- **Write your name on every piece of paper.**

Name: _____ MIT Email: _____

Question	Points	Score
1	50	
2	20	
3	30	
Total:	100	

Name: _____

4D Rotations and Cartesian Tensors

1. (50 points) In this problem, you will build intuition for rotations and Cartesian Tensors in 4D Euclidean space.

   (a) First, let's recap what we know about 3D rotations.

   i. How many distinct 2D coordinate planes can be formed from pairs of the standard basis directions in 3D (e.g., x, y, z)? Explain your reasoning.

   ii. In 3D, any nontrivial rotation occurs in a single 2D plane. A 3D rotation about an axis (e.g., the z-axis) is equivalent to a rotation in the plane orthogonal to that axis (e.g., the xy-plane). What type of object does a rotation axis transform like under the group $O(3)$?

   ☐ $\ell = 0$, $p = +1$ (Scalar)

   ☐ $\ell = 0$, $p = -1$ (Pseudoscalar)

   ☐ $\ell = 1$, $p = -1$ (Vector)

   ☐ $\ell = 1$, $p = +1$ (Pseudovector)

   ☐ $\ell = 2$, $p = +1$

   ☐ $\ell = 2$, $p = -1$

   Explain your reasoning:

Name: _____

(b) Rotations in 4D can be described using six generators, each corresponding to a rotation in one of six orthogonal 2D coordinate planes. *All six generators are provided at the end of the exam, before the "Work space" pages, but you do not need them to do this problem.* For example, the generator L_{xy} represents an infinitesimal rotation in the xy-plane. Two such generators are:

$$L_{xy} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad L_{zw} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Because xy and zw planes share no common axes, we say their axes are **disjoint**. This would still be true if we applied (the same) rotation to both of these planes.

i. Compute the commutator $[L_{xy}, L_{zw}]$. Please show your work.

ii. Based on your result above, what can you say about the relationship between rotations in orthogonal planes with **disjoint** (no shared) axes (e.g. xy and zw)?

Name: _____

(c) While in 3D Euclidean space (x, y, z), each nontrivial rotation occurs in a unique 2D plane, this is not the case in 4D Euclidean space (x, y, z, w). In 4D, a general rotation can be decomposed into simultaneous rotations in *at most two disjoint 2D planes* (with arbitrary orientation). However, even this decomposition may not be unique.

For example, the matrix $-\mathbb{I}_4$ can be formed by rotating by π in both the xy-plane and the zw-plane or by π in both the yw-plane and the xz-planes

$$-\mathbb{I}_4 = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} = \begin{bmatrix} \mathrm{c}(\pi) & -\mathrm{s}(\pi) & 0 & 0 \\ \mathrm{s}(\pi) & \mathrm{c}(\pi) & 0 & 0 \\ 0 & 0 & \mathrm{c}(\pi) & -\mathrm{s}(\pi) \\ 0 & 0 & \mathrm{s}(\pi) & \mathrm{c}(\pi) \end{bmatrix} = \begin{bmatrix} \mathrm{c}(\pi) & 0 & -\mathrm{s}(\pi) & 0 \\ 0 & \mathrm{c}(\pi) & 0 & -\mathrm{s}(\pi) \\ \mathrm{s}(\pi) & 0 & \mathrm{c}(\pi) & 0 \\ 0 & \mathrm{s}(\pi) & 0 & \mathrm{c}(\pi) \end{bmatrix}$$

where c and s are short for cosine and sine, respectively.

i. Give another pair of disjoint 2D planes whose simultaneous π-rotations also produce $-\mathbb{I}_4$.

ii. Let $R(\theta, P^+, \phi, P^-)$ denote a rotation by angle θ in one 2D plane P^+ and angle ϕ in a disjoint 2D plane P^-. The set of all such matrices (for arbitrary but fixed disjoint planes) forms which subgroup of $SO(4)$? *Hint: What kind of group structure arises when combining two commuting subgroups?*

- ☐ $SO(2)$
- ☐ $SO(3)$
- ☐ $SO(2) \times SO(2)$
- ☐ $SO(2) \times SO(3)$
- ☐ $SO(3) \times SO(3)$
- ☐ $SO(4)$

Explain your reasoning:

Name: _____

(d) An order two 3D Cartesian Tensor M_{ij} where $i, j \in \{x, y, z\}$ lives in the tensor product space of two 3D vectors, and under both $SO(3)$ and $O(3)$ decomposes into three irreps with the following dimensions:

$$3 \otimes 3 = 1 \oplus 3 \oplus 5.$$

Note, that these number indicate the number of **dimensions** per irrep, **not** the "L's" of the irreps.

i. What irrep of $SO(3)$ do 3D vectors (the 3 dimensional irrep on the left side of the equation) transform as?

ii. What irrep of $O(3)$ do 3D vectors (the 3 dimensional irrep on the left side of the equation) transform as?

iii. What irrep of $O(3)$ does the 1 dimensional irrep on the right side of the equation transform as?

iv. What irrep of $O(3)$ does the 3 dimensional irrep on the right side of the equation transform as?

v. What irrep of $O(3)$ does the 5 dimensional irrep on the right side of the equation transform as?

Name: _____

(e) An order two 4D Cartesian Tensor M_{ij} where $i, j \in \{x, y, z, w\}$ lives in the tensor product space of two 4D vectors.

Under $SO(4)$, this space decomposes into four irreps with the following dimensions:

$$4 \otimes 4 = 1 \oplus 3 \oplus 3 \oplus 9.$$

Under $O(4)$, this space decomposes into three irreps with the following dimensions:s

$$4 \otimes 4 = 1 \oplus 6 \oplus 9.$$

Again, these numbers indicate the number of dimensions, **not** indices of irreps.

i. Based on the difference between $SO(4)$ and $O(4)$ decomposition, argue whether $O(4)$ is a direct ($\times$) or semi-direct ($\rtimes$) product of $\mathbb{Z}_2$ and $SO(4)$.

(f) This tensor has both symmetric and antisymmetric parts. To compute the antisymmetric components, we use the following code.

```
import numpy as np
import scipy.linalg
from symm4ml import linalg, lie, grids

def so4_generators():
.....

# Project antisymmetric part of tensor product
tp = lie.tensor_product(so4_generators(), so4_generators())
perm_rep, sign_rep = grids.formula_to_perm_and_sign_group('ij=-ji')
basis, _ = grids.perm_and_sign_to_tensor_basis_and_proj(
    perm_rep, sign_rep, [4, 4]
)
antisym = np.einsum('ij,njk,kl->nil', basis, tp, basis.T)

# Decompose into irreps
ir_decomp = lie.decompose_rep_into_irreps(antisym)
print([ir.shape for ir in ir_decomp])
>> [(6, 3, 3), (6, 3, 3)]
```

i. Explain why we use `lie.tensor_product` instead of `rep.tensor_product` in the code above? What is different between these two implementations and why?

ii. Explain what the function `grids.formula_to_perm_and_sign_group` is doing in the code above.

iii. What does the variable `basis` represent in the code above and what is the significance of line 14?

iv. In the case of the order two 3D Cartesian tensor, the antisymmetric part of this tensor corresponds to the e.g. rotation generated by the cross-product between the two vectors used to form the tensor. Given this, what does the output of this code tell us about the irreps in the order two 4D Cartesian Tensor?

Name: _____

(g) The 6D space of antisymmetric 2-tensors (which span the space of planes in 4D) decomposes into two 3D irreducible representations under the action of $SO(4)$. These correspond to the *self-dual* and *anti-self-dual* subspaces.

This decomposition is defined by the following change of basis:

$$E_1^+ = \tfrac{1}{\sqrt{2}}(L_{xy} - L_{zw}) \qquad E_1^- = \tfrac{1}{\sqrt{2}}(L_{xy} + L_{zw})$$
$$E_2^+ = \tfrac{1}{\sqrt{2}}(L_{xz} + L_{yw}) \qquad E_2^- = \tfrac{1}{\sqrt{2}}(L_{xz} - L_{yw})$$
$$E_3^+ = \tfrac{1}{\sqrt{2}}(L_{xw} - L_{yz}) \qquad E_3^- = \tfrac{1}{\sqrt{2}}(L_{xw} + L_{yz})$$

Whereas the L_{ij} generators commute only with themselves $[L_{ij}, L_{ij}] = 0$ and with those that share no indices (i.e., rotations in disjoint orthogonal planes), all E^+ commute with all E^-. Moreover, the E^+ and E^- sets each satisfy the following commutation relations:

$$[E_i^\pm, E_j^\pm] = \epsilon_{ijk} E_k^\pm$$

where ϵ_{ijk} is the fully antisymmetric Levi-Civita symbol over the three indices that each span "directions" $1, 2, 3$.

i. What other Lie algebra that you've seen do these commutation relationships resemble?

ii. Given that the E^+ and E^- generators commute with one another and each obey the algebra above, what is the Lie algebra $\mathfrak{so}(4)$ isomorphic to (up to a Lie algebra isomorphism)? *Hint: Recall that E^+ and E^- commute. This will be analogous to how we combine groups that commute.*

Name: _____

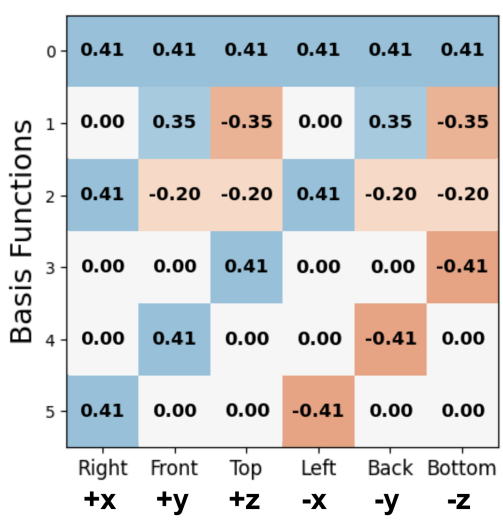
Convolutions on Cubes

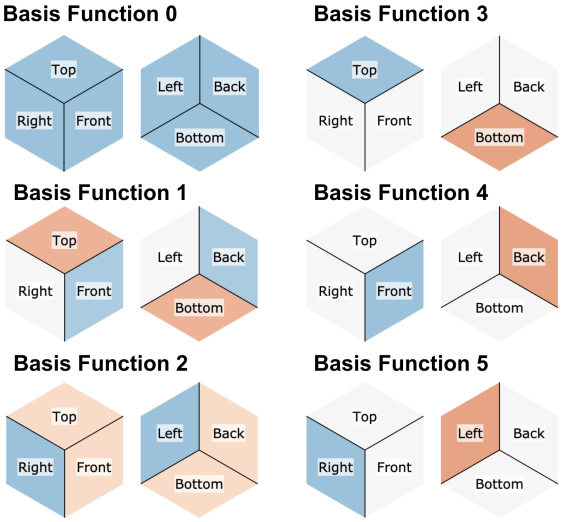
2. (20 points) In this problem, we will consider signals on the six faces of a cube, which is symmetric under the octahedral group O_h.

   (a) The group O_h has 48 elements. What is the dimension of its permutation representation acting on the cube's six faces? **Explain your reasoning.**

   (b) The permutation representation on the cube's faces decomposes into irreps of dimensions 1, 2, and 3. Using the output from `linalg.infer_change_of_basis`, we map face signals to the irrep vector space. In the left plot below, columns correspond to face centers $(+x, +y, +z, -x, -y, -z)$, and rows show the resulting irrep basis functions. The right side visualizes these on the cube, viewed along $(1, 1, 1)$ and $(-1, -1, -1)$. **Don't worry about interpreting the entire diagram yet – you'll analyze it step by step below.**

   At the end of the exam (before the "Work space" pages), you'll find a copy of this diagram and the O_h character table with labeled basis functions.





Name: _____

Using the figure and character table, answer the following:

i. What irrep does Basis Function 0 transform as? **Explain your reasoning.**

ii. Basis Functions 1 and 2 transform as E_g, proportional to $(2x^2 - y^2 - z^2,\ y^2 - z^2)$, which differs from the E_g basis in the character table: $(2z^2 - x^2 - y^2,\ x^2 - y^2)$. Why are both valid choices of E_g basis functions? **Explain your reasoning.**

iii. What irrep do the Basis Functions 3, 4 and 5 transform as? **Explain your reasoning.**

iv. Write Basis Functions 3, 4, and 5 as polynomials in (x, y, z). Clearly label which polynomial corresponds to each basis function.

(c) If we wanted to parameterize steerable filters based on a single copy of each basis function, how many free parameters would our steerable filter have? In other words, how many parameters are permitted in an equivariant linear map from `cube_perm_rep` to `cube_perm_rep`? **Explain your reasoning.**

(d) If we wanted to apply O_h group convolutions, we could promote signals on the cube to the $|O_h|$-dimensional regular representation.

Based on the shared irreps and their multiplicities, how many parameters are permitted in an equivariant linear map from `cube_perm_rep` to `octa_reg_rep`? Exclude contributions from irreps present in `octa_reg_rep` but absent in `cube_perm_rep`.

Name: _____

The Cubed Rule of Food

3. (30 points) The "Cubed Rule of Food" was proposed in 2017 by `@Phosphatide` on the social platform formerly known as Twitter to answer the controversial debate "Are hot dogs sandwiches?".

   The "Cubed Rule of Food" attempts to organize food by the location of structural starch (e.g. slices of bread). For example, a food item is classified as toast if it is supported by a single starch slab (e.g. pizza or nigiri sushi), a sandwich if it is surrounded by two parallel starch slabs (e.g. oreos), a taco if it is surrounded on three consecutive sides (e.g. a hot dog), and so on. Diagrams of each of these food types is listed below, with a number next to it indicating the number of sides of the cube covered in starch.

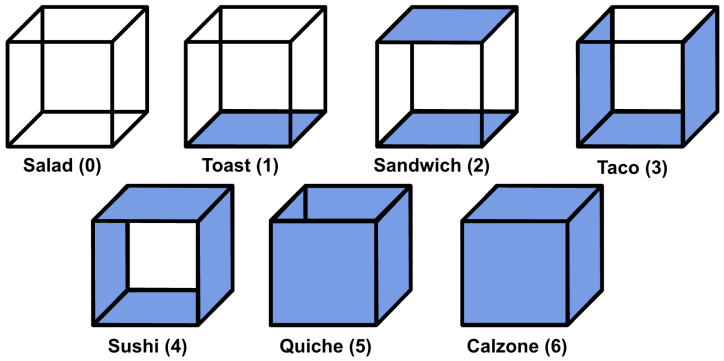


Figure 1: "Foods" classified by the Cubed Rule.

There are also several patterns that are not "allowed" by the "Cubed Rule of Food".

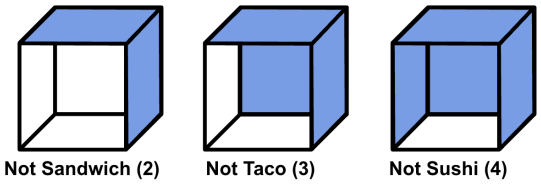


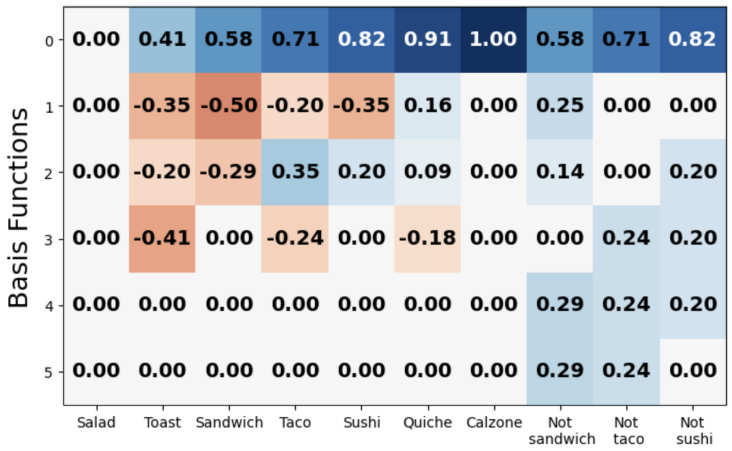
Figure 2: "Not foods" that are not covered by the Cubed Rule.

In this problem, you use the power-spectrum and linear layers to classify foods in any orientation using the "Cubed Rule of Food". Our inputs will be normalized binary signals on the faces of a cube (i.e. each face will have value $\frac{1}{\sqrt{N}}$ or 0 for N covered sides) and outputs will be classification labels over the food types.

Our arrays that store the signal will be length 6 vectors with faces given in the order `[Right, Back, Top, Left, Front, Bottom]`, matching the convention in the previous problem.

Name: _____

(a) First, we project our "Food" and "Not Food" examples onto the basis functions from the previous problem recover the following projections. Recall that Basis Function 0 tranforms as a 1D irrep (which we will call A), Basis Functions 1 and 2 transform as a 2D irrep (which we will call E), and Basis Functions 3, 4, and 5 transform as a 3D irrep (which we will call T). You do not need to have done Problem 2 to do this problem.



i. Are there any "foods" or "not foods" that transform as a single irrep?

ii. Which "foods" or "not foods" do not (at least partially) transform as E?

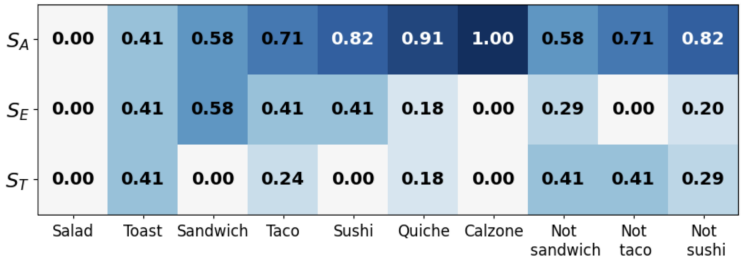
iii. Which "foods" or "not foods" do not (at least partially) transform as T?

Name: _____

(b) Now, we take the power spectrum of these projections to arrive at 3 scalars.

i. Why do we get 3 scalars for the power spectrum in this case? Explain your reasoning. You may find it helpful to draw an analogy to what we did for $SO(3)$, what did each component in the power spectrum "mean" in that case?

ii. The power spectra S is plotted below for each "Food" and "Not food".



How do the answers you gave in part (a) relate to the values of the power spectra?

iii. Suppose we rotated our signals on the cube by a random element of O_h. Would the plot of the projection onto Basis Functions change? Explain your reasoning.

iv. Would the plot of the power spectrum change? Explain your reasoning.

(c) Suppose we wanted to build a classifier that would correctly classify foods into appropriate categories and classify not foods as "Not foods".

i. Which "Foods" would we be able to correctly classify, simply by using only S_A?

ii. Describe a rule using the power spectrum that would allow you to distinguish a "Taco" from a "Not Taco".

iii. Describe a rule using the power spectrum that would allow you to distinguish a "Sandwich" from a "Not Sandwich".

iv. Based on your answers above, could you achieve 100% accuracy in this classification task just using the power spectrum.

Name: _____

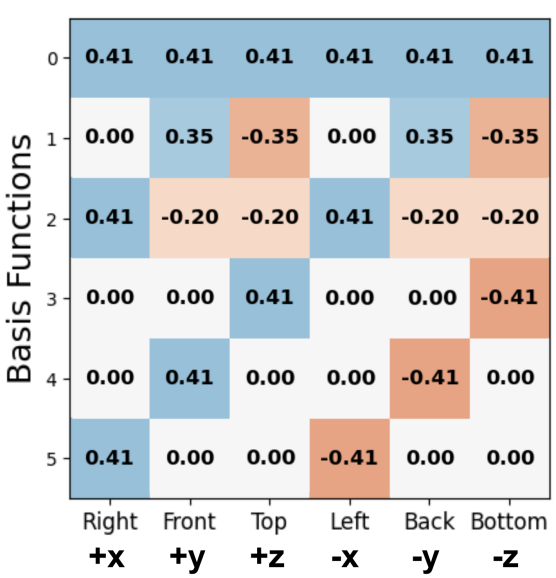
symm4ml Docstring listing

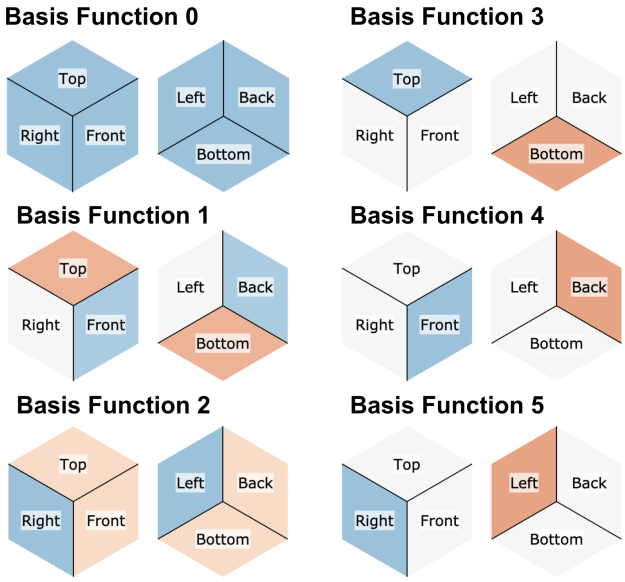
Modules listed in order: `lie, grids`

```
lie
----------------------
lie.decompose_rep_into_irreps:
Decomposes representation into irreducible representations.
    Input:
        X: np.array [lie_dim, d, d] - generators of a representation.
    Output:
        Ys: List[np.array] - list of generators of irreducible representations.

lie.tensor_product:
Tensor product of two representations of a Lie group.
    Input:
        X1: np.array [lie_dim, d1, d1] - generators of a representation.
        X2: np.array [lie_dim, d2, d2] - generators of a representation.
    Output:
        X: np.array [lie_dim, d1*d2, d1*d2] - tensor product of the
            representations.


grids
----------------------
grids.formula_to_perm_and_sign_group:
    Generate a permutation group from an einsum-style index formula.

    The function interprets a string formula in the style of einsum,
    where the left-hand side indicates the original index order and the
    right-hand side indicates the permuted order. A preceding '-' on an
    index indicates a sign change.

    Input:
        formula : (str) A string representing the index permutation.
            For example, 'ij=-ji' indicates that index 'i' maps to 'j'
            with a sign flip on the second term.

    Output:
        Tuple of np.array of matrix representation of permutation group
        generated by theprovided formula and np.array of sign group for
        the values. The vector space that the matrix acts on is in the
        same order as the letters of the left side of the formula.

    Notes:
        This function uses a helper functions `rep.direct_sum` and
        `groups.generate_group` to construct the full group from the list
```

```
        of generator matrices and signs.

    Examples:
    >>> group = formula_to_perm_and_sign_group('ij=-ji')
    >>> print(group)
    (array([[[1., 0.],
         [0., 1.]],

        [[0., 1.],
         [1., 0.]]]),
     array([ 1., -1.]))


grids.perm_and_sign_to_tensor_basis_and_proj:
    Obtain the tensor basis and projector from the permutation
    group representation.

    This function constructs the tensor representation of the
    permutation group acting on the tensor coordinates defined by dims,
    then sums over the group elements and applies Gram-Schmidt
    orthonormalization to obtain a convenient basis (and corresponding
    projection) for the tensor representation.

    Input:
        perm_rep: (ndarray) Array of shape (G, d, d) containing
            G permutation matrices representing the group.
        sign_rep: (ndarray) Array of shape (G) containing sign changes
            on values
        dims: (iterable of int) The dimensions of the tensor.
            Must satisfy len(dims) == d and np.prod(dims) equals the
            total number of tensor coordinates.
        tol: (float, optional) Tolerance for the Gram-Schmidt
            orthonormalization process. Default is 1e-8.

    Output:
        Two ndarrays representing the orthonormal tensor basis and
        projector obtained from the Gram-Schmidt process.

    Notes:
        - The function uses perm_to_tensor_rep to generate the
          permutation representation on the tensor coordinates.
        - The sum over the group elements is orthonormalized using
          linalg.gram_schmidt
```

Name: _____





O_h	E	$8C_3$	$6C_2$	$6C_4$	$3C_2=(C_4)^2$	i	$6S_4$	$8S_6$	$3\sigma_h$	$6\sigma_d$	linear functions, rotations	quadratic functions
A_{1g}	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-	$x^2+y^2+z^2$
A_{2g}	+1	+1	-1	-1	+1	+1	-1	+1	+1	-1	-	-
E_g	+2	-1	0	0	+2	+2	0	-1	+2	0	-	$(2z^2-x^2-y^2, x^2-y^2)$
T_{1g}	+3	0	-1	+1	-1	+3	+1	0	-1	-1	(R_x, R_y, R_z)	-
T_{2g}	+3	0	+1	-1	-1	+3	-1	0	-1	+1	-	(xz, yz, xy)
A_{1u}	+1	+1	+1	+1	+1	-1	-1	-1	-1	-1	-	-
A_{2u}	+1	+1	-1	-1	+1	-1	+1	-1	-1	+1	-	-
E_u	+2	-1	0	0	+2	-2	0	+1	-2	0	-	-
T_{1u}	+3	0	-1	+1	-1	-3	-1	0	+1	+1	(x, y, z)	-
T_{2u}	+3	0	+1	-1	-1	-3	+1	0	+1	-1	-	-

Name: _____

The six generators of $SO(4)$ can be written as:

$$L_1 = L_{(xy)} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad L_2 = L_{(xz)} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad L_3 = L_{(xw)} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix}$$

$$L_4 = L_{(yz)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad L_5 = L_{(yw)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} \quad L_6 = L_{(zw)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

where L_i enumerates the 6 generators and $L_{(jk)}$ specifies the plane of rotation.

Name: _____

Work space

Name: _____

Work space