# PERMUTATION EQUIVARIANT NEURAL NETWORKS FOR ANTISYMMETRIC TENSORS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Antisymmetric tensors change sign under index swaps, yet learning methods that exploit this structure remain largely unexplored. We characterise all linear permutation equivariant maps between antisymmetric power spaces of $\mathbb{R}^n$, showing that they are determined by a class of objects that we term antispherical bipartitions. As a consequence, we show that the dimension of the space of such maps is at most two, independently of the tensor orders and of $n$. We further introduce IO patterns, which replace large weight matrices with a representation whose storage is independent of $n$. This framework is well-suited to graph-structured data with signed or directed edges, where antisymmetry encodes the orientation of edges. We illustrate our approach on the Wiki-RfA dataset, showing that the characterised layers perform at least as well as general permutation equivariant maps at a lower parameter cost, and that IO patterns remain practical at sizes where the dense weight matrix exhausts available memory.

## 1 INTRODUCTION

Equivariance provides a principled framework for designing neural networks that respect symmetries in data. By constraining functions to commute with group actions, equivariant models encode structural priors, which can improve generalisation and sample efficiency. This idea was popularised in the context of group equivariant convolutional neural networks (Cohen & Welling, 2016; 2017) and has since been applied to more general group actions on a variety of structured data domains (Kondor & Trivedi, 2018; Kondor et al., 2018; Thomas et al., 2018; Weiler et al., 2018; Satorras et al., 2021). A central challenge in equivariant machine learning is the characterisation of linear equivariant maps between group representation spaces since they form the fundamental building blocks for developing new equivariant architectures (Ravanbakhsh et al., 2017; Zaheer et al., 2017; Maron et al., 2019; Finzi et al., 2021; Pearce-Crump, 2023; Godfrey et al., 2023; Sverdlov et al., 2025).

Despite these advances, relatively little attention has been given to equivariant learning on antisymmetric tensors. An antisymmetric tensor is a tensor whose components satisfy

$$T_{i_1,i_2,\dots,i_k} = \operatorname{sgn}(\pi) T_{i_{\pi(1)},i_{\pi(2)},\dots,i_{\pi(k)}} \quad \text{for all } \pi \in S_k \tag{1}$$

and vanish whenever any two indices coincide. They arise in graph-structured data with signed or directed edges, where antisymmetry encodes the orientation of edges. In this setting, learning should be permutation equivariant to reflect symmetries under node relabellings. However, existing permutation equivariant architectures (Zaheer et al., 2017; Maron et al., 2019; Godfrey et al., 2023; Pearce-Crump, 2024) are designed for general tensors and do not preserve antisymmetric structure.

In this work, we characterise all permutation equivariant linear maps between any two antisymmetric powers of $\mathbb{R}^n$. We show that these maps are determined by a new class of objects, which we term antispherical bipartitions. These objects naturally encode the interaction between antisymmetry and the permutation action on pairs of indices, where signs behave in a non-trivial way. We prove that the dimension of the space of such maps is at most two regardless of the tensor orders and of $n$, in contrast to the general tensor case (Maron et al., 2019), where the dimension grows like a Bell number in the tensor orders. We further introduce IO patterns, which avoid the explicit construction of large weight matrices by tracking, for each output index, the input indices that map to it. We apply our approach to the Wiki-RfA dataset, where our layers perform at least as well as general permutation equivariant maps while using fewer parameters. We also show that IO patterns scale to sizes at which the dense weight matrix exhausts available memory.



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

## 2 Related Work

**Equivariant Machine Learning on Antisymmetric Tensors** To the best of our knowledge, only a few recent publications have addressed this topic, but they focus only on equivariance with respect to Euclidean groups. Heilman et al. (2024) created $SO(3)$-equivariant models using spherical harmonics to predict components of antisymmetric tensors from crystal structures. Hodapp & Shapeev (2024) developed equivariant tensor networks to construct interatomic interaction models that are based on $SO(3)$-invariant tensor networks. By contrast, our work is the first to address permutation equivariance on antisymmetric tensors.

**Equivariant Machine Learning on Symmetric Tensors** Several works in equivariant machine learning have studied structured data in the symmetric tensor setting (Gao et al., 2022; Lou & Ganose, 2024; Wen et al., 2024; Heilman et al., 2024; Garanger et al., 2024). Pearce-Crump (2025) characterised permutation equivariant linear maps for symmetric tensors. The antisymmetric case, however, gives rise to a fundamentally different structure, due to the signs of permutations that must be considered in equation (4).

**Permutation Equivariant Neural Networks** These neural networks have been designed to learn from different types of data, focusing primarily on set-structured data (Ravanbakhsh et al., 2017; Qi et al., 2017; Zaheer et al., 2017; Hartford et al., 2018; Maron et al., 2020; Sverdlov et al., 2025) and graph- and hypergraph-structured data (Maron et al., 2019; Thiede et al., 2020; Finzi et al., 2021; Morris et al., 2022; Godfrey et al., 2023; Puny et al., 2023; Pearce-Crump, 2024; Pearce-Crump & Knottenbelt, 2024b). In particular, Pan & Kondor (2022) and Pearce-Crump & Knottenbelt (2024a) have investigated how to organise the computations that are involved in these networks to improve their performance. Our work contributes to the literature by providing the first characterisation of linear permutation equivariant maps in the antisymmetric tensor setting.

## 3 Antisymmetric Powers of $\mathbb{R}^n$

We present the fundamental definitions that are needed to study linear permutation equivariant functions on antisymmetric tensors. We let $[n] \coloneqq \{1, \dots, n\}$ throughout. The key idea is that condition (1) implies that antisymmetric tensors are uniquely determined by coefficients that are indexed by strictly increasing index tuples.

Define the set $\Lambda[n]^k$ to consist of all $k$-length tuples ($k \geq 0$) of the form

$$I = (i_1, i_2, \dots, i_k) \text{ such that } 1 \leq i_1 < i_2 < \dots < i_k \leq n. \tag{2}$$

We see that, for $k \leq n$, the cardinality of $\Lambda[n]^k$ is $\binom{n}{k}$, otherwise it is $0$.

Consequently, we define the ***k*th antisymmetric power of** $\mathbb{R}^n$, $\Lambda^k(\mathbb{R}^n)$, to be the vector space with basis $\{e_I : I \in \Lambda[n]^k\}$. We will write $e_{i_1,\dots,i_k}$ in place of $e_{(i_1,\dots,i_k)}$ for notational convenience. We equip $\Lambda[n]^k$ with the lexicographic order $\preceq$: for distinct $I, J \in \Lambda[n]^k$, let $l$ be the first index such that $i_l \neq j_l$. Then $I \preceq J$ if and only if $i_l < j_l$. This induces a canonical ordering of the basis, giving a **standard basis**. With this ordering, we identify $\Lambda^k(\mathbb{R}^n)$ with $\mathbb{R}^{\binom{n}{k}}$, so that linear maps from $\Lambda^k(\mathbb{R}^n)$ to $\Lambda^l(\mathbb{R}^n)$ can be represented as matrices (see Technical Appendix, Section A for more details).

With this setup, we obtain a natural left action of the symmetric group $S_n$ on $\Lambda[n]^k$ by applying a permutation $\sigma$ entrywise to a tuple $I$ and then, critically, reordering the resulting tuple into increasing order. That is, there exists a unique reordering permutation $\pi_{\sigma(I),k} \in S_k$ such that $\sigma(I)\pi_{\sigma(I),k} \in \Lambda[n]^k$. We let $S_k$ act on tuples on the right, to distinguish it from the action of $S_n$ on the left.

*Example* 3.1. Let $n = 5$ and $k = 3$. Consider $I = (1, 3, 5)$ in $\Lambda[5]^3$, and suppose that we apply $\sigma = (145)(23)$ in $S_5$ to it. We obtain $\sigma(I) = (4, 2, 1)$ which we reorder to $\sigma(I)\pi_{\sigma(I),k} = (1, 2, 4)$ (by applying the reordering permutation $\pi_{\sigma(I),k} = (13)$ in $S_3$) to give another element of $\Lambda[5]^3$. Note, in particular, that $\sigma(I)$ by itself is not an element of $\Lambda[5]^3$.

We extend the action of $S_n$ on the set $\Lambda[n]^k$ to a representation on $\Lambda^k(\mathbb{R}^n)$ by defining it on the basis elements and extending linearly. The basis element $e_I$ is sent to the basis element corresponding to the reordered tuple $\sigma(I)\pi_{\sigma(I),k}$, multiplied by the sign of the reordering permutation. Denoting the representation by $\rho_k^\Lambda$, we therefore have that $\rho_k^\Lambda(\sigma) : e_I \mapsto \operatorname{sgn}(\pi_{\sigma(I),k}) e_{\sigma(I)\pi_{\sigma(I),k}}$.

*Example* 3.2. Following on from Example 3.1, we see that $\rho_3^{\Lambda}((145)(23))$ maps $e_{1,3,5}$ to $-e_{1,2,4}$ since the reordering permutation in $S_3$, $(13)$, has sign $-1$.

We now turn to our goal of characterising the linear permutation equivariant maps between antisymmetric power spaces of $\mathbb{R}^n$. They are defined as follows.

**Definition 3.3.** A linear map $\phi : \Lambda^k(\mathbb{R}^n) \to \Lambda^l(\mathbb{R}^n)$ is said to be **permutation equivariant** if, for all $\sigma \in S_n$ and $v \in \Lambda^k(\mathbb{R}^n)$,

$$\phi(\rho_k^{\Lambda}(\sigma)[v]) = \rho_l^{\Lambda}(\sigma)[\phi(v)] \tag{3}$$

Given the standard bases of $\Lambda^k(\mathbb{R}^n)$ and $\Lambda^l(\mathbb{R}^n)$, any such map $\phi$ is represented by a weight matrix $W$. (3) therefore becomes the matrix constraint $W\rho_k^{\Lambda}(\sigma) = \rho_l^{\Lambda}(\sigma)W$ for all $\sigma \in S_n$. Hence the problem reduces to characterising the vector space of weight matrices satisfying this constraint, which we denote by $\mathrm{Hom}_{S_n}(\Lambda^k(\mathbb{R}^n), \Lambda^l(\mathbb{R}^n))$. As a result, it suffices to construct a basis for this space.

## 4 Basis Construction for Equivariant Weight Matrices

We obtain a basis of matrices for $\mathrm{Hom}_{S_n}(\Lambda^k(\mathbb{R}^n), \Lambda^l(\mathbb{R}^n))$ in three steps: 1) we obtain a condition that a matrix needs to satisfy in order to be $S_n$-equivariant, and rewrite it as a constraint in terms of the orbits of the action of $S_n$ on the Cartesian product set $\Lambda[n]^l \times \Lambda[n]^k$; 2) we represent each orbit by a diagram; and 3) we use these diagrams to obtain the basis matrices themselves. We begin **Step 1** with the following:

**Proposition 4.1.** *A matrix $f = (f_{I,J})$ lies in $\mathrm{Hom}_{S_n}(\Lambda^k(\mathbb{R}^n), \Lambda^l(\mathbb{R}^n))$ if and only if*

$$\boxed{f_{I,J} = \mathrm{sgn}(\pi_{\sigma(I),l})\,\mathrm{sgn}(\pi_{\sigma(J),k}) f_{\sigma(I)\pi_{\sigma(I),l},\sigma(J)\pi_{\sigma(J),k}}} \tag{4}$$

*for all $\sigma \in S_n$, $I \in \Lambda[n]^l$ and $J \in \Lambda[n]^k$, where $\pi_{\sigma(I),l}$ and $\pi_{\sigma(J),k}$ are the unique reordering permutations in $S_l$ and $S_k$ that restore the entries of $\sigma(I)$ and $\sigma(J)$ into increasing order.*

Equation (4) reduces the problem of constructing basis elements of $\mathrm{Hom}_{S_n}(\Lambda^k(\mathbb{R}^n), \Lambda^l(\mathbb{R}^n))$ to studying the action of $S_n$ on the Cartesian product set $\Lambda[n]^l \times \Lambda[n]^k$. Writing the elements of this set as $\binom{I}{J}$, where $I \in \Lambda[n]^l$ and $J \in \Lambda[n]^k$, we define the action as

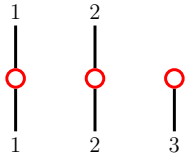
$$\sigma \cdot \binom{I}{J} := \binom{\sigma(I)\pi_{\sigma(I),l}}{\sigma(J)\pi_{\sigma(J),k}} \tag{5}$$

**Step 2: Orbit-to-Diagram Correspondence**

This step gives a diagrammatic model for the $S_n$ orbits on $\Lambda[n]^l \times \Lambda[n]^k$. The resulting diagrams form the key intermediate structure behind the construction of the basis. For each orbit, we first construct an associated equivalence class of diagrams. Let $\binom{I}{J}$ be a class representative of some orbit. We form a diagram for $\binom{I}{J}$ in the following way:

> We place the entries of $I$ in one row and the entries of $J$ in another row below it. For each value $x \in [n]$, if $x$ does not appear in either $I$ or $J$, we proceed to the next value in $[n]$. Otherwise, we insert a central red node in between the two rows. If $x$ appears in $I$, then we draw a line from its position in the top row down to the red node. If $x$ appears in $J$, then we draw a line from its position in the bottom row up to the red node.

*Example* 4.2. Suppose that $l = 2$, $k = 3$, and $n = 5$. Consider the $S_5$ orbit of $\Lambda[5]^2 \times \Lambda[5]^3$ that contains the element $\binom{1,2}{1,2,3}$. Applying the construction in the green-coloured box, we obtain

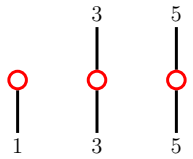



(6)

Removing the labels defines the diagram that is associated to $\binom{1,2}{1,2,3}$. In general, for such a diagram, we call the black lines **wires** and we call each individual central red node together with the wires that are attached to it a **spider**.

Suppose that we change the class representative of the orbit: if we choose $\binom{\sigma(I)\pi_{\sigma(I),l}}{\sigma(J)\pi_{\sigma(J),k}}$ instead and apply the same construction, then the spiders that appear in the diagram for this class representative (viewed without their labels) will be the same as those for $\binom{I}{J}$, just potentially in a different order. Hence, by our construction, each orbit of the $S_n$ action on $\Lambda[n]^l \times \Lambda[n]^k$ determines a well-defined equivalence class of diagrams.

*Example* 4.3. Continuing on from Example 4.2, we see that $\binom{3,5}{1,3,5}$ lies in the same orbit as $\binom{1,2}{1,2,3}$, since it can be obtained by applying the permutation $\sigma = (13)(25)$ to $\binom{1,2}{1,2,3}$ followed by reordering the entries of each row into increasing order. Applying the procedure in the green-coloured box to $\binom{3,5}{1,3,5}$, we obtain the diagram




(7)

After removing the labels, this is the same diagram as (6), except the spiders are in a different order.

However, we can strengthen this correspondence by first introducing the following structure:

**Definition 4.4.** A $(k, l)$**–bipartition** $\beta$ is a multiset

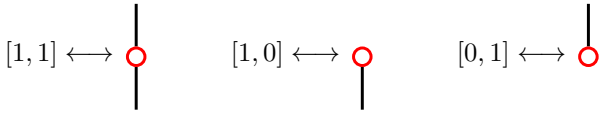
$$\{[x_1, y_1], [x_2, y_2], \dots, [x_t, y_t]\} \tag{8}$$

of some $t$ pairs $[x_i, y_i]$ such that $x_i, y_i \geq 0$ for all $i$, not both zero, $\sum_{i=1}^{t} x_i = k$, and $\sum_{i=1}^{t} y_i = l$. We call the individual pairs **blocks**. Bipartitions were first introduced in pure mathematics by MacMahon (1896).

We now define a restricted class of bipartitions that is central to our construction. To the best of our knowledge, this class has not appeared in the literature before.

**Definition 4.5.** We define a $(k, l)$**–antispherical bipartition** to be a $(k, l)$–bipartition in which every block is one of $[1, 1]$, $[1, 0]$, or $[0, 1]$.

Each $(k, l)$–antispherical bipartition encodes a diagram that is obtained by replacing blocks according to




(9)

where the first entry of each block counts wires to the bottom row and the second entry counts wires to the top row, so that the diagram has $k$ wires to the bottom row and $l$ wires to the top row in total.

Hence, by the above construction, each orbit of the $S_n$ action on $\Lambda[n]^l \times \Lambda[n]^k$ determines a unique $(k, l)$–antispherical bipartition, since all of the diagrams in its associated equivalence class are encodings of the same $(k, l)$–antispherical bipartition (they have the same block structure).

Consequently, each $(k, l)$–antispherical bipartition admits a canonical representative diagram:

**Definition 4.6.** The **normal form** of a $(k, l)$–antispherical bipartition is the diagram that is obtained by representing all of the blocks as spiders, arranged from left to right with all of the $[1, 1]$ blocks appearing first, then all of the $[1, 0]$ blocks, and then finally all of the $[0, 1]$ blocks.

Hence we associate to each orbit the normal form of a $(k, l)$–antispherical bipartition. In fact, by the construction given in the green-coloured box, the bipartition can have at most $n$ blocks.

*Example* 4.7. Continuing on from Examples 4.2 and 4.3, we see that the $(3, 2)$–antispherical bipartition that corresponds to the orbit containing both $\binom{1,2}{1,2,3}$ and $\binom{3,5}{1,3,5}$ is $\{[1, 1], [1, 1], [1, 0]\}$. The corresponding $(3, 2)$–antispherical bipartition in normal form is given in (6) (without the labels).

Having shown that each orbit of the $S_n$ action on $\Lambda[n]^l \times \Lambda[n]^k$ determines the normal form of a $(k, l)$–antispherical bipartition having at most $n$ blocks, we now show that every $(k, l)$–antispherical bipartition having at most $n$ blocks arises from a unique orbit.

Representing each $(k, l)$–antispherical bipartition $\beta$ having some $0 \leq t \leq n$ blocks in normal form, we obtain its unique orbit as follows. We call this the **diagram-to-orbit procedure for** $\beta$.

1. For each $t$-length tuple having distinct elements in $[n]$, label the central red nodes of the spiders in the normal form from left to right with the entries of the tuple. We call each such diagram a **labelled normal form diagram**.
2. Permute the labelled spiders so that their central node labels are arranged in increasing order from left to right. We call such a diagram a **reordered diagram**.
3. Then assign to each endpoint of a wire the label of the central red node to which the wire is attached, and form the element $\binom{P}{Q}$, where $P$ is the top row and $Q$ is the bottom row of the diagram. We call this operation **label propagation** and the element $\binom{P}{Q}$ a **propagated pair**.
4. The set of all such $\binom{P}{Q}$ obtained in this way is precisely the $S_n$ orbit.

We summarise the result of **Step 2** with the following proposition.

**Proposition 4.8.** *The $S_n$ orbits on $\Lambda[n]^l \times \Lambda[n]^k$ correspond bijectively with the normal form of the $(k,l)$–antispherical bipartitions having at most $n$ blocks.*

*Remark* 4.9. In the Technical Appendix, Section C, we provide a procedure for how to generate all $(k,l)$–antispherical bipartitions, and hence determine their total number.

*Remark* 4.10. We work with $t$-length tuples having distinct elements in $[n]$ instead of individual elements of $S_n$ because the set of all such tuples is equal to the set

$$\{\sigma \cdot (1,2,\dots,t) \coloneqq (\sigma(1),\sigma(2),\dots,\sigma(t)) : \sigma \in S_n\} \tag{10}$$

that is, there are $(n-t)!$ elements in $S_n$ that correspond to the same $t$-length tuple, and so we use $t$-length tuples to avoid overgenerating the elements of the $S_n$ orbit.

**Step 3: Diagrams-to-Basis Matrices**

Putting together the results of Step 1 (equation (4) and the orbit description) and Step 2 (Proposition 4.8), we see that the problem of constructing the basis matrices of $\mathrm{Hom}_{S_n}(\Lambda^k(\mathbb{R}^n), \Lambda^l(\mathbb{R}^n))$ is reduced to studying the normal form of $(k,l)$–antispherical bipartitions having at most $n$ blocks.

Fix a $(k,l)$–antispherical bipartition $\beta$ having some $0 \leq t \leq n$ blocks. From its normal form, we need to show how to construct a matrix $f = (f_{I,J})$ that is associated to it.

Looking back at equation (4), it suggests the following key points:

- We need a pair of tuples, say $\binom{I}{J}$, to be a **reference pair** from which we can calculate all $\binom{\sigma(I)\pi_{\sigma(I),l}}{\sigma(J)\pi_{\sigma(J),k}}$, and
- In applying any $\sigma \in S_n$ to the reference pair $\binom{I}{J}$, we need to track the product of the signs of the reordering permutations $\pi_{\sigma(I),l}$, $\pi_{\sigma(J),k}$ that are used to reorder $\sigma(I)$ and $\sigma(J)$.

For the first key point, we can choose the propagated pair coming from $(1,2,\dots,t)$ in the diagram-to-orbit procedure for $\beta$ to be our reference pair $\binom{I}{J}$. By Remark 4.10, this procedure will then generate all $\binom{\sigma(I)\pi_{\sigma(I),l}}{\sigma(J)\pi_{\sigma(J),k}}$. Hence, to address the second key point, we can modify the procedure to attach signs to diagrams, so that we can use these diagrams together with their signs as a tool to implicitly track the product of the signs of reordering permutations. We do this as follows.

For each labelled normal form diagram, we set the sign to $+$. This is because, by Remark 4.10, the $t$-length tuple that labels this diagram is equal to some $(\sigma(1),\sigma(2),\dots,\sigma(t))$. Hence, if we were to perform label propagation at this stage, we would have $\binom{\sigma(I)}{\sigma(J)}$, which shows that no reordering permutations have been applied at this point. It follows that each labelled normal form diagram must carry a consistent sign, and so we choose $+$.

We still proceed to permute the labelled spiders so that each labelled normal form diagram becomes a reordered diagram. However, as we do this, we now update the sign as follows: as adjacent pairs of spiders are swapped, we track the swaps that are made in the endpoints of the top and bottom rows. If exactly one swap occurs in the endpoints of either the top or bottom row, then we flip the sign; otherwise, we leave it unchanged. We call this **the sign rule.**

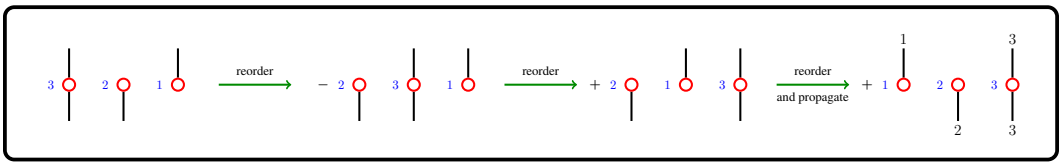



Figure 1: We show how to calculate the sign of the reordered diagram that comes from the normal form of the $(2,2)$–antispherical bipartition $\{[1,1],[1,0],[0,1]\}$ whose spiders are labelled with the 3-length tuple $(3,2,1)$. We calculate the signs either by tracking the swaps that we make to the endpoints of the top and bottom rows or by following the rules that are given in Figure 2.

This is because, by swapping adjacent pairs of spiders until we obtain the reordered diagram, we are, in effect, building the reordering permutations for each row as a product of transpositions and identity elements. Each swap contributes either a transposition in one row and an identity in the other row, transpositions in both rows, or identities in both rows. The sign rule is recording the cumulative effect of these contributions, with the result that the sign that is associated to the reordered diagram corresponds to the product of the signs of the reordering permutations.

We can immediately improve the implementation of the sign rule: since we track sign changes as we swap adjacent pairs of spiders, we can view the sign changes as a set of six rules on adjacent pairs of spiders as we swap their order. Reducing these swaps to a set of rules means that we do not have to track swaps of endpoints on individual rows. We have provided these rules in Figure 2.

*Example* 4.11. We give an example of the sign rule in Figure 1: we show how to calculate the sign of the reordered diagram that comes from the normal form of the $(2,2)$–antispherical bipartition $\{[1,1],[1,0],[0,1]\}$ whose spiders are labelled with the 3-length tuple $(3,2,1)$.

At this point, for each $t$-length tuple having distinct elements in $[n]$, we have a reordered diagram with a sign attached to it. We call each such pair a **signed diagram**.

**Definition 4.12.** We define the matrix $f_\beta = ((f_\beta)_{I,J})$ that is associated to $\beta$ as follows. For each reordered diagram $D$ that appears in the set of signed diagrams, let $\binom{P}{Q}$ be its associated propagated pair, and consider the signs that are attached to those signed diagrams whose reordered diagram is $D$. If both signs occur, then $(f_\beta)_{P,Q} = 0$. If only the positive sign occurs, then $(f_\beta)_{P,Q} = +1$. If only the negative sign occurs, then $(f_\beta)_{P,Q} = -1$. All remaining entries of $f_\beta$ are defined to be $0$.

Although we have obtained a matrix from each $(k,l)$–antispherical bipartition having at most $n$ blocks, we have deliberately avoided saying that each such antispherical bipartition generates a basis matrix: indeed, certain classes only generate the zero matrix.

**Lemma 4.13.** *Any $(k,l)$–antispherical bipartition having at most $n$ blocks that has more than one $[1,0]$ block or more than one $[0,1]$ block maps to the zero matrix.*

However, as stated in the following theorem, the remaining $(k,l)$–antispherical bipartitions having at most $n$ blocks do give the basis matrices. (The proof shows that, for these $(k,l)$–antispherical bipartitions, each reordered diagram that arises in the construction appears with a unique sign.)

**Theorem 4.14.** *The basis matrices $f_\beta$ of $\mathrm{Hom}_{S_n}(\Lambda^k(\mathbb{R}^n), \Lambda^l(\mathbb{R}^n))$ are in bijective correspondence with the set of $(k,l)$–antispherical bipartitions $\beta$ having at most $n$ blocks such that each antispherical bipartition has at most one $[1,0]$ block and at most one $[0,1]$ block.*

**Corollary 4.15.** *The $S_n$-equivariant weight matrix from $\Lambda^k(\mathbb{R}^n)$ to $\Lambda^l(\mathbb{R}^n)$ is $W = \sum_\beta w_\beta f_\beta$ for weights $w_\beta \in \mathbb{R}$ and basis matrices $f_\beta$, where the sum is over all $(k,l)$–antispherical bipartitions $\beta$ having at most $n$ blocks with at most one $[1,0]$ block and at most one $[0,1]$ block.*

**Corollary 4.16.** *The dimension of $\mathrm{Hom}_{S_n}(\Lambda^k(\mathbb{R}^n), \Lambda^l(\mathbb{R}^n))$ is two if $0 < k = l < n$, one if $k = l = 0$, if $k = l = n$, or if $|k-l| = 1$ and $n \geq \max(k,l)$, and zero otherwise. In particular, it is at most two, independently of $k$, $l$ and $n$.*

*Remark* 4.17. Corollary 4.16 contrasts with the general tensor case, where the dimension grows like the Bell number $B(k+l)$ (Maron et al., 2019); in particular, Deep Sets (Zaheer et al., 2017) is the case where $k = l = 1$ (see Example E.2). In the Technical Appendix, Section H, we use Corollary 4.16 to relate the basis maps to the linear operators of classical exterior algebra.

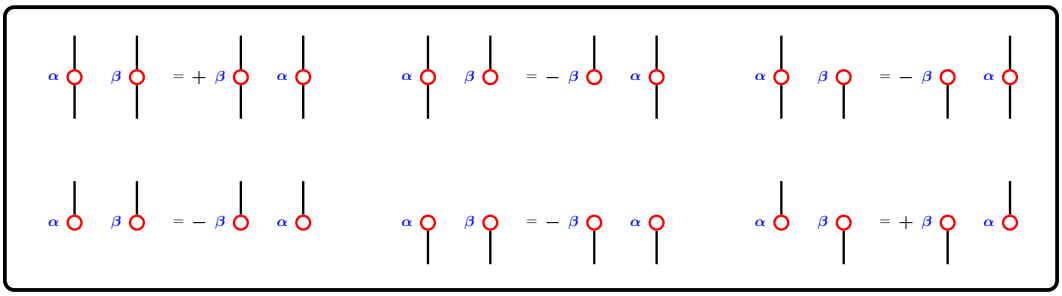
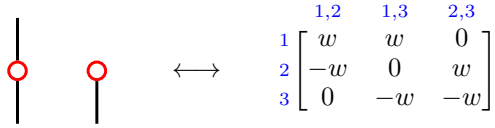


Figure 2: This set of six rules describes the sign changes that occur when we swap the order of an adjacent pair of labelled spiders. A swap of adjacent spiders that induces exactly one swap of endpoints in either the top or bottom row changes the sign, whereas a swap that induces either zero swaps or one swap in both rows leaves the sign unchanged.

*Example* 4.18. We show how to obtain the $S_3$-equivariant weight matrix from $\Lambda^2(\mathbb{R}^3)$ to $\Lambda^1(\mathbb{R}^3)$. We need to consider the normal form of the $(2,1)$–antispherical bipartitions having at most $n=3$ blocks, with at most one $[1,0]$ block and at most one $[0,1]$ block. There is only one such diagram, which corresponds to a basis matrix in $\mathrm{Hom}_{S_3}(\Lambda^2(\mathbb{R}^3), \Lambda^1(\mathbb{R}^3))$ of size $3\times 3$. The calculations to obtain this matrix are given in Example E.1. Hence, by Corollary 4.15, for a weight $w\in\mathbb{R}$, the $S_3$-equivariant weight matrix from $\Lambda^2(\mathbb{R}^3)$ to $\Lambda^1(\mathbb{R}^3)$ is of the form



$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} \begin{array}{ccc} 1{,}2 & 1{,}3 & 2{,}3 \end{array} \\ \begin{bmatrix} w & w & 0 \\ -w & 0 & w \\ 0 & -w & -w \end{bmatrix} \end{array} \tag{11}$$

*Remark* 4.19. We have focused solely on understanding the possible $S_n$-equivariant weight matrices from $\Lambda^k(\mathbb{R}^n)$ to $\Lambda^l(\mathbb{R}^n)$ because it is clear that the pointwise non-linearities must be odd in order to preserve the antisymmetric structure.

## 5 Memory-Efficient Implementation

Since antisymmetric tensors are typically represented as elements of $(\mathbb{R}^n)^{\otimes k}$, we would like the weight matrices that we have characterised to act directly on tensors in this form. In principle, this can be achieved by converting the weight matrices from $\Lambda^k(\mathbb{R}^n)$ to $\Lambda^l(\mathbb{R}^n)$ to be matrices from $(\mathbb{R}^n)^{\otimes k}$ to $(\mathbb{R}^n)^{\otimes l}$. However, as $n$, $k$ and $l$ increase, storing these matrices in memory (and applying them) becomes impractical.

Instead, we introduce some new notation that describes how to map an antisymmetric tensor $T\in(\mathbb{R}^n)^{\otimes k}$ to an antisymmetric tensor $f_\beta(T)\in(\mathbb{R}^n)^{\otimes l}$, where we regard $f_\beta$ as a basis matrix in its converted form. This notation will allow us to perform the mapping without us having to explicitly realise $f_\beta$ as an $n^l\times n^k$ matrix. The key idea is that we track, for each fixed index $I$ of the output $f_\beta(T)$, all of the possible input indices $J$ that map to this output, without constructing the full matrix.

**Definition 5.1.** Let $\beta$ be a $(k,l)$–antispherical bipartition having at most one $[1,0]$ block and at most one $[0,1]$ block, and let $n$ be greater than or equal to the number of blocks in $\beta$. We define the **IO pattern** for $\beta$ as follows: considering the normal form of $\beta$, let $I=(i_1,i_2,\dots,i_l)$ be a tuple of pairwise distinct formal symbols. We perform the following steps:

1. Label the spiders with wires to the top row by the entries of $I$, in all possible ways such that the labels on the $[1,1]$ spiders are increasing in the order of the entries of $I$ from left to right. Each such choice produces a new labelled diagram.
2. If the $[1,0]$ spider appears in $\beta$, label it with $j$ in each diagram that is obtained in Step 1.
3. If the $[0,1]$ spider appears in $\beta$, for each diagram obtained in Step 2, swap it into the position such that if we performed label propagation, the top row tuple would be $I$, updating signs according to the rules in Figure 2.

4. If a $[1,0]$ spider appears in $\beta$, for each diagram obtained in Step 3, generate new diagrams by moving the $[1,0]$ spider leftwards through adjacent swaps with the ordered $[1,1]$ spiders, updating signs according to the rules in Figure 2, until there are no $[1,1]$ spiders to its left.

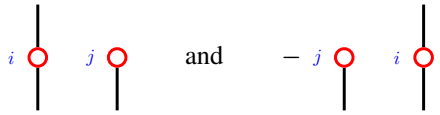
5. If there is more than one $[1,1]$ spider, for each diagram obtained in Step 4, fix the top-row wires and permute only the bottom-row wires of the $[1,1]$ spiders, generating a new diagram for each permutation. Flip the sign in the generated diagram if the permutation is odd.

6. For each diagram obtained in Step 5, take its sign and attach it to its propagated pair $\binom{I}{J}$.

7. The IO pattern consists of an arrow $\leftarrow$ with $I$ on the left-hand side and, on the right-hand side, the signed linear combination of all tuples $J$ obtained in Step 6, where, if the $[1,0]$ spider appears in $\beta$, the index $j$ appearing in $J$ is summed over all values in $[n] \setminus \{i_1, \dots, i_l\}$. $I = (i_1, \dots, i_l)$ is interpreted as distinct elements of $[n]$ at this stage of the construction.

The IO pattern for $\beta$ encodes the desired map $T \mapsto f_\beta(T)$, when we replace $I$ by $f_\beta(T)_I$ and $J$ by $T_J$. This is because, for each fixed output index $I$, the construction first identifies all input index configurations that are consistent with the ordering of the entries of $I$ (together with an additional index $j$ if a $[1,0]$ block is present), and then applies antisymmetrisation to each input configuration, namely by performing a signed sum over all permutations of each input configuration, where each term is weighted by the product of the sign of the diagram before antisymmetrisation and the sign of the permutation.

*Example* 5.2. Continuing on from Example 4.18, we consider once again the normal form of the $(2,1)$–antispherical bipartition that was given in (11). Note that it has at most one $[1,0]$ block and at most one $[0,1]$ block. Let $n \geq 2$ and the output tuple be $I = (i)$. We see that Step 1 gives us the left-hand diagram in (12), and by the end of Step 4, we have




(12)

At the end of Step 6, we have $\binom{i}{i,j}$ with sign $+$ and $\binom{i}{j,i}$ with sign $-$. Hence, with Step 7, we obtain the IO pattern corresponding to this antispherical bipartition, namely

$$i \leftarrow \sum_{j:j\neq i}^{n} [i, j-j, i] \tag{13}$$

For $n = 3$, one can check this against the basis matrix that is given in Example D.1. From this, we get that $f_\beta(T)_i = \sum_{j:j\neq i}^{3} [T_{i,j} - T_{j,i}] = 2\sum_{j:j\neq i}^{3} T_{i,j} = 2\sum_{j=1}^{3} T_{i,j}$ which describes in full the transformation on an input antisymmetric tensor $T \in (\mathbb{R}^3)^{\otimes 2}$.

*Remark* 5.3. Example 5.2 illustrates the effectiveness of IO patterns. A single IO pattern corresponding to a $(k,l)$–antispherical bipartition having at most one $[1,0]$ block and at most one $[0,1]$ block determines the map $T \in (\mathbb{R}^n)^{\otimes k} \mapsto f_\beta(T) \in (\mathbb{R}^n)^{\otimes l}$ for every $n$ greater than or equal to the number of blocks in $\beta$. If we could only work with matrices instead, then we would have to realise a separate matrix for each value of $n$. Hence, a single IO pattern encodes a uniform family of tensor maps across all such dimensions.

*Remark* 5.4. We provide further examples of how to calculate IO patterns for various $(k,l)$–antispherical bipartitions in the Technical Appendix, Section E, give pseudocode for constructing and applying an IO pattern in Section F, and provide a reference list of IO patterns for commonly appearing $(k,l)$–antispherical bipartitions in Section G.

**Complexity.** Since a full layer is a weighted sum of at most two basis maps by Corollary 4.16, the storage of the map itself using IO patterns is $O(1)$ in $n$ and its compute is $O(n^{l+1})$, against $O(n^{k+l})$ for both using an explicit weight matrix. We give full details in the Technical Appendix, Section I, and compare the two approaches empirically in Section J.

## 6 Numerical Experiments

Our goal in this section is to demonstrate that the characterised maps can be trained and behave sensibly on real-world data, rather than to compete with state-of-the-art applied models. We provide full implementation details in the Technical Appendix, Section L.

We considered the Wiki-RfA dataset (West et al., 2014), where nodes are Wikipedia members and directed edges are votes, which are positive for support and negative for opposition. We removed neutral votes to retain only directional interactions. The task was to predict a candidate's real adminship outcome, giving $3{,}494$ roughly balanced labelled candidates. For each candidate we constructed a $10$ node subgraph consisting of the candidate together with nine of its neighbours in the underlying undirected graph, drawn uniformly at random, and took its induced antisymmetric adjacency matrix as input. If a candidate had fewer than nine neighbours, we sampled the remaining nodes uniformly from the graph. Our models consisted of six layers of permutation equivariant maps between antisymmetric tensors with an invariant pooled readout, in single- and four-channel versions. We replaced the permutation equivariant maps between antisymmetric tensors with general permutation equivariant maps (Maron et al., 2019) to give two controlled baselines. We also compared against a signed message-passing GNN and an unconstrained MLP, training every model over three independent $60/20/20$ splits with five seeds each.

| Model | Accuracy | AUC | Parameters | Time/step |
|---|---|---|---|---|
| Antisymmetric, $4$-channel (ours) | $0.760{\pm}0.008$ | $0.852{\pm}0.013$ | $\mathbf{77}$ | $34$ ms |
| General equivariant, $4$-channel | $0.751{\pm}0.008$ | $0.836{\pm}0.012$ | $177$ | $50$ ms |
| Antisymmetric, single-channel (ours) | $0.738{\pm}0.011$ | $0.831{\pm}0.019$ | $122$ | $13$ ms |
| General equivariant, single-channel | $0.728{\pm}0.007$ | $0.817{\pm}0.006$ | $147$ | $25$ ms |
| Signed message-passing GNN | $\mathbf{0.790}{\pm}0.010$ | $\mathbf{0.881}{\pm}0.010$ | $9{,}697$ | $13$ ms |
| MLP | $0.786{\pm}0.012$ | $0.861{\pm}0.014$ | $10{,}689$ | $4$ ms |

Table 1: Adminship outcome prediction on Wiki-RfA. Accuracy and AUC are reported as mean $\pm$ standard deviation over three splits, where each split's score is the average over five random seeds. Training was full-batch, so Time/step is the cost of one gradient step.

In the controlled comparisons only the family of maps differed, with our models performing comparably or better with fewer parameters (AUC $0.852$ against $0.836$, at $77$ parameters against $177$; and $0.831$ against $0.817$, at $122$ against $147$), indicating that the characterised layers are doing useful work. Our models were also faster to train in both cases, at $34$ ms against $50$ ms per step in the four-channel comparison and $13$ ms against $25$ ms in the single-channel one. Our four-channel network reached an AUC of $0.852$, against $0.861$ for the MLP and $0.881$ for the signed GNN. Although both of these baselines lead on accuracy and AUC, each uses roughly $10{,}000$ parameters, whereas the characterised maps give rise to a compact, permutation invariant architecture that comes within $0.03$ of them on either metric and uses only $77$ parameters.

## 7 Conclusion

In this paper, we provided the first complete characterisation of linear permutation equivariant maps between antisymmetric tensor spaces, showing that they are determined by antispherical bipartitions and that the space of such maps has dimension at most two. To implement these maps efficiently, we introduced IO patterns, which avoid the explicit materialisation of large weight matrices, and reduce the storage of a layer to $O(1)$ in $n$ and the cost of applying it to $O(n^{l+1})$. We validated our approach on graph-structured data with directed edges, where our layers performed at least as well as general permutation equivariant maps at a lower parameter cost, and showed that IO patterns remain practical at sizes for which the dense weight matrix exhausted the available memory. We believe that the idea behind IO patterns could be useful for other equivariant architectures, and hope that this work encourages further exploration of antisymmetric tensor structure in machine learning.

### AI USE STATEMENT

In this work, we used generative AI tools to implement the experiments, including the preprocessing of data. We have not used generative AI tools to develop theoretical models or conceptual frameworks, to formulate mathematical claims, or to interpret results. However, we have used generative AI tools to create and edit software code, and to check for spelling and grammatical errors that might have appeared in the paper. We verified that the LLM-generated implementation of the models was correct using tests that we implemented ourselves using the `pytest` testing framework. We take full responsibility for the final content of this work, including text, claims or artifacts that were produced with the aid of generative AI.

### ETHICS STATEMENT

This work is primarily theoretical and involves no human subjects or new data collection. Our experiments use the publicly available Wiki-RfA dataset, as distributed by the SNAP repository (Section M). Although it records votes by named Wikipedia users, we used the dataset only to form aggregate graph inputs, and made no attempt to identify or profile individuals.

### REPRODUCIBILITY STATEMENT

We give complete proofs of our results in the Technical Appendix, Sections B, C and H. Pseudocode for IO patterns and a reference list of patterns with closed forms are given in Sections F and G respectively. We describe the data processing, architectures, parameter counts and the training protocol for the experiments of Section 6 in Section L. and the hardware and measurement setup for the scaling benchmark in Section J. All experiments use fixed random seeds. We also provide anonymised code in the supplementary material.

## REFERENCES

Taco Cohen and Max Welling. Group Equivariant Convolutional Networks. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 2990–2999, New York, USA, 20–22 Jun 2016. PMLR.

Taco Cohen and Max Welling. Steerable CNNs. In *International Conference on Learning Representations*, 2017.

Marc Finzi, Max Welling, and Andrew Gordon Wilson. A Practical Method for Constructing Equivariant Multilayer Perceptrons for Arbitrary Matrix Groups. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 3318–3328. PMLR, 18–24 Jul 2021.

Liyao Gao, Yifan Du, Hongshan Li, and Guang Lin. RotEqNet: Rotation-Equivariant Network for Fluid Systems with Symmetric High-Order Tensors. *Journal of Computational Physics*, 461: 111205, 2022.

Kévin Garanger, Julie Kraus, and Julian J. Rimoli. Symmetry-enforcing neural networks with applications to constitutive modeling. *Extreme Mechanics Letters*, 71:102188, 2024.

Charles Godfrey, Michael G. Rawson, Davis Brown, and Henry Kvinge. Fast computation of permutation equivariant layers with the partition algebra. In *ICLR 2023 Workshop on Physics for Machine Learning*, 2023.

Jason S. Hartford, Devon R. Graham, Kevin Leyton-Brown, and Siamak Ravanbakhsh. Deep Models of Interactions Across Sets. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1914–1923. PMLR, 2018.

Alex Heilman, Claire Schlesinger, and Qimin Yan. Equivariant Graph Neural Networks for Prediction of Tensor Material Properties of Crystals, 2024. `arXiv:2406.03563`.

M. Hodapp and A. Shapeev. Equivariant tensor network potentials. *Machine Learning: Science and Technology*, 5:035075, 2024.

Risi Kondor and Shubhendu Trivedi. On the Generalization of Equivariance and Convolution in Neural Networks to the Action of Compact Groups. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2747–2755. PMLR, 10–15 Jul 2018.

Risi Kondor, Zhen Lin, and Shubhendu Trivedi. Clebsch–Gordan Nets: a Fully Fourier Space Spherical Convolutional Neural Network. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Yuchen Lou and Alex M. Ganose. Discovery of highly anisotropic dielectric crystals with equivariant graph neural networks, 2024. arXiv:2405.07915.

Percy Alexander MacMahon. XVI. Memoir on the Theory of the Partition of Numbers. Part I. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 187:619–673, 1896.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and Equivariant Graph Networks. In *International Conference on Learning Representations*, 2019.

Haggai Maron, Or Litany, Gal Chechik, and Ethan Fetaya. On Learning Sets of Symmetric Elements. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6734–6744. PMLR, 13–18 Jul 2020.

Christopher Morris, Gaurav Rattan, Sandra Kiefer, and Siamak Ravanbakhsh. SpeqNets: Sparsity-aware Permutation-equivariant graph networks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 16017–16042, 17–23 Jul 2022.

Horace Pan and Risi Kondor. Permutation Equivariant Layers for Higher Order Interactions. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 5987–6001. PMLR, 28–30 Mar 2022.

Edward Pearce-Crump. Brauer's Group Equivariant Neural Networks. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 27461–27482. PMLR, 23–29 Jul 2023.

Edward Pearce-Crump. Connecting Permutation Equivariant Neural Networks and Partition Diagrams. In *Proceedings of the 27th European Conference on Artificial Intelligence*, volume 392, pp. 1511–1518. IOS Press Ebooks, 2024.

Edward Pearce-Crump. Permutation Equivariant Neural Networks for Symmetric Tensors. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48595–48634. PMLR, 13–19 Jul 2025.

Edward Pearce-Crump and William J. Knottenbelt. A Diagrammatic Approach to Improve Computational Efficiency in Group Equivariant Neural Networks. 2024a. arXiv:2412.10837.

Edward Pearce-Crump and William J. Knottenbelt. Graph Automorphism Group Equivariant Neural Networks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 40051–40077. PMLR, 21–27 Jul 2024b.

Omri Puny, Derek Lim, Bobak Kiani, Haggai Maron, and Yaron Lipman. Equivariant Polynomials for Graph Neural Networks. In *International Conference on Machine Learning*, pp. 28191–28222. PMLR, 2023.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 652–660, 2017.

Siamak Ravanbakhsh, Jeff Schneider, and Barnabás Póczos. Equivariance Through Parameter-Sharing. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 2892–2901, 06–11 Aug 2017.

Víctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) Equivariant Graph Neural Networks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9323–9332. PMLR, 18–24 Jul 2021.

Yonatan Sverdlov, Ido Springer, and Nadav Dym. Revisiting Multi-Permutation Equivariance through the Lens of Irreducible Representations. In *International Conference on Learning Representations*, 2025.

Erik Henning Thiede, Truong Son Hy, and Risi Kondor. The general theory of permutation equivarant neural networks and higher order graph variational encoders, 2020. arXiv:2004.03990.

Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds, 2018. arXiv:1802.08219.

Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Mingjian Wen, Matthew K. Horton, Jason M. Munro, Patrick Huck, and Kristin A. Persson. An equivariant graph neural network for the elasticity tensors of all seven crystal systems. *Digital Discovery*, 3(5):869–882, 2024.

Robert West, Hristo S Paskov, Jure Leskovec, and Christopher Potts. Exploiting Social Network Structure for Person-to-Person Sentiment Analysis. *Transactions of the Association for Computational Linguistics*, 2:297–310, October 2014.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R. Salakhutdinov, and Alexander J. Smola. Deep Sets. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

# A Background on Antisymmetric Powers of $\mathbb{R}^n$

In this section, we provide additional intuition for the construction of $\Lambda^k(\mathbb{R}^n)$ that was introduced in Section 3 by developing a running example that illustrates the concepts from that section.

We started by saying that antisymmetric tensors are uniquely determined by coefficients that are indexed by strictly increasing index tuples. To see this, suppose that we have the following antisymmetric tensor in $(\mathbb{R}^3)^{\otimes 2}$ (i.e. $n = 3$ and $k = 2$), expressed in the standard basis:

$$T = a\, e_1 \otimes e_2 - b\, e_1 \otimes e_3 + c\, e_2 \otimes e_3 - a\, e_2 \otimes e_1 + b\, e_3 \otimes e_1 - c\, e_3 \otimes e_2. \tag{14}$$

This tensor can be rewritten using antisymmetry by grouping together terms that correspond to permutations of each index tuple:

$$T = a\,(e_1 \otimes e_2 - e_2 \otimes e_1) - b\,(e_1 \otimes e_3 - e_3 \otimes e_1) + c\,(e_2 \otimes e_3 - e_3 \otimes e_2). \tag{15}$$

Hence, $T$ is completely determined by the coefficients that are associated with the strictly increasing index tuples $(1,2)$, $(1,3)$, and $(2,3)$. In the antisymmetric power space $\Lambda^2(\mathbb{R}^3)$, this is written in condensed form as

$$T_c = a\, e_{1,2} - b\, e_{1,3} + c\, e_{2,3}. \tag{16}$$

where $(1,2), (1,3), (2,3)$ are the elements of $\Lambda[3]^2$ that, in this order, index the standard basis (since they are presented in lexicographical order).

Conversely, any element of $\Lambda^2(\mathbb{R}^3)$ of the form (16) uniquely determines an antisymmetric tensor in $(\mathbb{R}^3)^{\otimes 2}$ by expanding each basis element as the antisymmetrisation over all permutations in $S_2$:

$$e_{i_1,i_2} \;\mapsto\; \sum_{\pi \in S_2} \operatorname{sgn}(\pi)\, e_{i_{\pi(1)}} \otimes e_{i_{\pi(2)}} = e_{i_1} \otimes e_{i_2} - e_{i_2} \otimes e_{i_1} \tag{17}$$

for all $(i_1, i_2) \in \Lambda[3]^2$. This recovers the full antisymmetric tensor (15) in $(\mathbb{R}^3)^{\otimes 2}$. In general, one could easily formalise this to establish a one-to-one correspondence between antisymmetric tensors and coefficient data that is indexed by strictly increasing index tuples.

We now illustrate how linear maps between antisymmetric power spaces are represented as matrices. We continue with $\Lambda^2(\mathbb{R}^3)$, and consider a linear map $\phi : \Lambda^2(\mathbb{R}^3) \to \Lambda^1(\mathbb{R}^3)$.

Recall that the standard basis of $\Lambda^2(\mathbb{R}^3)$ is $\{e_{1,2}, e_{1,3}, e_{2,3}\}$ and the standard basis of $\Lambda^1(\mathbb{R}^3)$ is $\{e_1, e_2, e_3\}$.

Hence any element $T_c \in \Lambda^2(\mathbb{R}^3)$ is identified with a vector $v_c \in \mathbb{R}^3$ via

$$T_c = a\, e_{1,2} + b\, e_{1,3} + c\, e_{2,3} \;\longleftrightarrow\; v_c = \begin{matrix}1,2\\1,3\\2,3\end{matrix}\!\begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{18}$$

Similarly, elements of $\Lambda^1(\mathbb{R}^3)$ are identified with $\mathbb{R}^3$ in the standard way.

Hence, $\phi$ is represented by a matrix $W \in \mathbb{R}^{3\times 3}$ such that the identification of $\phi(T_c)$ is $W v_c$.

Explicitly, writing

$$W = \begin{matrix}\\1\\2\\3\end{matrix}\!\begin{matrix}\begin{matrix}1,2 & 1,3 & 2,3\end{matrix}\\\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}\end{matrix} \qquad v_c = \begin{matrix}1,2\\1,3\\2,3\end{matrix}\!\begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{19}$$

we obtain

$$W v_c = \begin{matrix}1\\2\\3\end{matrix}\!\begin{bmatrix} w_{11}a + w_{12}b + w_{13}c \\ w_{21}a + w_{22}b + w_{23}c \\ w_{31}a + w_{32}b + w_{33}c \end{bmatrix} \tag{20}$$

In this way, in general, linear maps from $\Lambda^k(\mathbb{R}^n)$ to $\Lambda^l(\mathbb{R}^n)$ are identified with matrices acting on coefficient vectors via standard matrix multiplication, with respect to the standard bases. In particular,

the matrices $E_{I,J}$, which are defined to have a 1 in the $(I, J)$–entry and zeros elsewhere, form a natural basis for this space. For example, if $n = 3$, $k = 2$ and $l = 1$, then the matrix $E_{(2),(1,3)}$ is

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1,2 & 1,3 & 2,3 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{array} \tag{21}$$

Finally, we illustrate the matrices $\rho_k^\Lambda(\sigma)$ that are induced by the action of $S_n$ on $\Lambda^k(\mathbb{R}^n)$.

Continuing with the case $n = 3$ and $k = 2$, consider the permutation $\sigma = (123) \in S_3$. We compute its action on the basis elements:

$$\begin{aligned} \sigma \cdot e_{1,2} &= e_{2,3}, \\ \sigma \cdot e_{1,3} &= -e_{1,2}, \\ \sigma \cdot e_{2,3} &= -e_{1,3}. \end{aligned}$$

Here, the sign in the third line arises because the tuple $\sigma(2, 3) = (3, 1)$ must be reordered to the tuple $(1, 3)$ via the transposition $(12) \in S_2$. Similar reasoning applies for the sign in the second line.

Hence, with respect to the standard basis $\{e_{1,2}, e_{1,3}, e_{2,3}\}$, the action of $\sigma$ is represented by the matrix

$$\rho_2^\Lambda((123)) = \begin{array}{c} \\ 1,2 \\ 1,3 \\ 2,3 \end{array} \begin{array}{c} \begin{array}{ccc} 1,2 & 1,3 & 2,3 \end{array} \\ \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix} \end{array} \tag{22}$$

In general, each $\rho_k^\Lambda(\sigma)$ acts on coefficient vectors by permuting coordinates and introducing signs according to the reordering permutations.

## B Supplementary Proofs

*Proof of Proposition 4.1.* With respect to the standard bases of $\Lambda^k(\mathbb{R}^n)$ and $\Lambda^l(\mathbb{R}^n)$, linear maps from $\Lambda^k(\mathbb{R}^n)$ to $\Lambda^l(\mathbb{R}^n)$ are represented by matrices. Hence we can express these maps in the basis of matrix units $E_{I,J}$ that were defined in Section A.

Now let $e_P \in \Lambda^k(\mathbb{R}^n)$ be a standard basis element. We have that

$$E_{I,J} e_P = \delta_{J,P} e_I \tag{23}$$

where $\delta_{J,P} = 1$ if $J = P$ and 0 otherwise.

Hence, for any linear map $f : \Lambda^k(\mathbb{R}^n) \to \Lambda^l(\mathbb{R}^n)$, expressing $f$ in the basis of matrix units as

$$f = \sum_{I \in \Lambda[n]^l} \sum_{J \in \Lambda[n]^k} f_{I,J} E_{I,J} \tag{24}$$

we get that

$$f(e_P) = \sum_{I \in \Lambda[n]^l} f_{I,P} e_I \tag{25}$$

By Definition 3.3, $f$ is an $S_n$-equivariant linear map if and only if, for all $\sigma \in S_n$ and standard basis elements $e_J \in \Lambda^k(\mathbb{R}^n)$,

$$f(\rho_k^\Lambda(\sigma)[e_J]) = \rho_l^\Lambda(\sigma)[f(e_J)] \tag{26}$$

Consequently, (26) holds if and only if

$$\sum_{I \in \Lambda[n]^l} \operatorname{sgn}(\pi_{\sigma(J),k}) f_{I,\sigma(J)\pi_{\sigma(J),k}} e_I = \sum_{I \in \Lambda[n]^l} f_{I,J} \operatorname{sgn}(\pi_{\sigma(I),l}) e_{\sigma(I)\pi_{\sigma(I),l}} \tag{27}$$

for some $\pi_{\sigma(J),k} \in S_k$ and $\pi_{\sigma(I),l} \in S_l$ that restores the entries of $\sigma(J)$ and $\sigma(I)$ into increasing order.

Hence, (27) holds if and only if

$$\operatorname{sgn}(\pi_{\sigma(J),k}) f_{\sigma(I)\pi_{\sigma(I),l},\sigma(J)\pi_{\sigma(J),k}} = f_{I,J} \operatorname{sgn}(\pi_{\sigma(I),l}) \tag{28}$$

for all $\sigma \in S_n$, $I \in \Lambda[n]^l$ and $J \in \Lambda[n]^k$.

Since $\operatorname{sgn}(\pi_{\sigma(I),l})$ equals $\pm 1$, we see that (28) is equivalent to

$$f_{I,J} = \operatorname{sgn}(\pi_{\sigma(I),l}) \operatorname{sgn}(\pi_{\sigma(J),k}) f_{\sigma(I)\pi_{\sigma(I),l},\sigma(J)\pi_{\sigma(J),k}} \tag{29}$$

for all $\sigma \in S_n$, $I \in \Lambda[n]^l$ and $J \in \Lambda[n]^k$. □

*Proof of Lemma 4.13.* Suppose that we are given the normal form of a $(k, l)$–antispherical bipartition $\beta$ having some $t \in [n]$ blocks, where $x \geq 2$ of the blocks are $[1, 0]$. Fix a set of $x$ distinct labels from $[n]$ for them, and fix the labels of the other $t - x$ spiders in the diagram completely. Then there are $x!$ possible tuples with which to label the $[1, 0]$ spiders in the diagram (for each fixed choice of labels for the other spiders).

However, these $x!$ tuples are in bijective correspondence with the permutations of $S_x$, and so once we reorder the $[1, 0]$ spiders in each of the $x!$ diagrams to be in increasing order, these $x!$ diagrams will all be the same, except half of them will come with positive sign and half will come with negative sign, by the rules in Figure 2. Hence, once we reorder the spiders entirely in each of the $x!$ diagrams, we will have $x!$ copies of the same reordered diagram, with half coming with positive sign and half coming with negative sign. As a result of how we construct the matrix that corresponds to $\beta$, the resulting $(P, Q)$–entry (for the propagated pair $\binom{P}{Q}$ that corresponds to the reordered diagram) in the matrix will be zero.

Since this is true for all possible sets of distinct labels for the $[1, 0]$ spiders and all possible labels for the other spiders, i.e. for all $t$-length tuples with distinct elements in $[n]$, this implies that every $(P, Q)$–entry (for every $\binom{P}{Q}$ that is obtained in this process) in the matrix is zero. Since all of the other entries in the matrix are already defined to be zero (by how we assign values to entries), this implies that $\beta$ maps to the zero matrix.

The proof is exactly the same for $(k, l)$–antispherical bipartitions $\beta$ having some $t \in [n]$ blocks, where $x \geq 2$ of the blocks are $[0, 1]$. □

*Proof of Theorem 4.14.* Equation (4) tells us that every element of $\operatorname{Hom}_{S_n}(\Lambda^k(\mathbb{R}^n), \Lambda^l(\mathbb{R}^n))$ is determined on each orbit of the action of $S_n$ on the set $\Lambda[n]^l \times \Lambda[n]^k$ by its value at a single pair. Proposition 4.8 states that these orbits correspond bijectively with the $(k, l)$–antispherical bipartitions having at most $n$ blocks, and Definition 4.12 associates a matrix $f_\beta$ to each such bipartition $\beta$. Combining these statements, we see that every element of $\operatorname{Hom}_{S_n}(\Lambda^k(\mathbb{R}^n), \Lambda^l(\mathbb{R}^n))$ is a linear combination of the matrices $f_\beta$, each of which lies in $\operatorname{Hom}_{S_n}(\Lambda^k(\mathbb{R}^n), \Lambda^l(\mathbb{R}^n))$ by construction.

Furthermore, Lemma 4.13 showed that any such antispherical bipartition that has more than one $[1, 0]$ block or more than one $[0, 1]$ block generates the zero matrix. Hence we are left with considering the set of $(k, l)$–antispherical bipartitions having at most $n$ blocks such that each antispherical bipartition has at most one $[1, 0]$ block and at most one $[0, 1]$ block.

For each bipartition $\beta$ in the remaining set, $f_\beta$ can be non-zero only at the indices in its corresponding orbit. Since distinct orbits are disjoint, we just need to show that these matrices are in fact non-zero in order to obtain linear independence and therefore prove our result.

We can split these into four separate cases. (Note that for certain values of $k$, $l$, and $n$, some of these cases do not appear, but we can approach this with a level of generality).

**Case 1: there are no $[1, 0]$ blocks and no $[0, 1]$ blocks.**

In this case, $k = l$, since the only blocks that appear are $[1, 1]$ blocks. Hence there are $k$ blocks in the antispherical bipartition. Hence we consider all possible $k$-length tuples with distinct elements from $[n]$. We can split this set of tuples into separate classes, where $(a_1, a_2, \ldots, a_k)$ and $(b_1, b_2, \ldots, b_k)$ are in the same class if and only if the elements in the tuples are the same up to some permutation. For each such class, the diagrams that come from labelling the normal form with the tuples in the class all generate the same $\binom{P}{Q}$ element once the spiders are reordered, and all with the same sign,

since, by Figure 2, swapping two $[1,1]$ spiders does not change their sign. Hence, for each such class, the $(P,Q)$–entry of the corresponding matrix is $+1$. Hence the matrix is non-zero in this case.

**Case 2: there is one $[1,0]$ block and no $[0,1]$ blocks.**

In this case, $k = l + 1$, since we must have $l$ $[1,1]$ blocks and one $[1,0]$ block. Hence there are $k$ blocks in the antispherical bipartition. As before, we can consider all possible $k$-length tuples with distinct elements from $[n]$. Once again, we split this set of tuples into separate classes, except this time $(a_1, a_2, \dots, a_k)$ and $(b_1, b_2, \dots, b_k)$ are in the same class if and only if $a_k = b_k$ and the remaining $k-1$ elements in each tuple are the same up to some permutation. From here the proof is exactly the same as in Case 1: for each class, we obtain the set of diagrams that come from labelling the normal form with the tuples in the class. For each diagram, we can reorder the $[1,1]$ spiders in each diagram to be in increasing order without changing the sign, giving us the same diagram overall (repeated many times). Hence once we swap the $[1,0]$ spider into position, changing sign as appropriate, all of these diagrams generate the same $\binom{P}{Q}$ element, all with the same sign. Hence, as before, the matrix is non-zero in this case.

**Case 3: there is no $[1,0]$ block and one $[0,1]$ block.**

This is exactly the same as Case 2, except now $l = k + 1$ and there are $k$ $[1,1]$ blocks and one $[0,1]$ block. Replacing $k$ by $l$ in Case 2 and $[1,0]$ by $[0,1]$ and following exactly the same argument gives the result. Consequently, the resulting matrix is non-zero in this case.

**Case 4: there is one $[1,0]$ block and one $[0,1]$ block.**

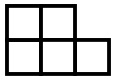
The final case is also very similar to the previous ones. Here we must have $k = l$, since there are $k-1$ $(= l-1)$ $[1,1]$ blocks, one $[1,0]$ block and one $[0,1]$ block. Hence there are now $k+1$ blocks in the antispherical bipartition. As before, we can consider all possible $(k+1)$-length tuples with distinct elements from $[n]$. Once again, we split this set of tuples into separate classes, except this time $(a_1, a_2, \dots, a_k, a_{k+1})$ and $(b_1, b_2, \dots, b_k, b_{k+1})$ are in the same class if and only if $a_k = b_k$, $a_{k+1} = b_{k+1}$ and the remaining $k-1$ elements in each tuple are the same up to some permutation. We now follow the same argument as before: for each class, we obtain the set of diagrams that come from labelling the normal form with the tuples in the class. For each diagram, we can reorder the $[1,1]$ spiders in each diagram to be in increasing order without changing the sign, giving us the same diagram overall (repeated many times). Hence once we swap the $[1,0]$ spider and the $[0,1]$ spider into position, changing sign as appropriate, all of these diagrams generate the same $\binom{P}{Q}$ element, all with the same sign. Hence, as before, the matrix is non-zero in this case. □

## C Algorithm to Generate all $(k,l)$–Antispherical Bipartitions

In this section, we give an explicit procedure for generating all $(k,l)$–antispherical bipartitions, and hence determine their number. We do this by constructing the normal form of each one as follows.

Since the total number of wires in the normal form of a $(k,l)$–antispherical bipartition is $k+l$, we represent these wires using a tableau with $l$ boxes in the first row and $k$ boxes in the second row. We then assign colours to the boxes according to the rules stated below so as to generate all possible $(k,l)$–antispherical bipartitions.

For example, if $l = 2$ and $k = 3$, then the tableau is of the form



(30)

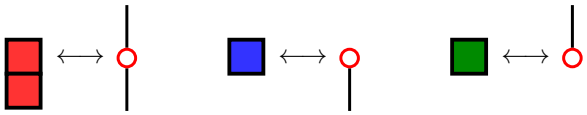
We only allow three colours, red, blue, and green, that are subject to the following rules:

- if a box in the first row is coloured in red, then there must be a box directly below it that is also coloured in red,
- if a box in the second row is coloured in red, then there must be a box directly above it that is also coloured in red,
- if a box is coloured in green, then it can only appear in the first row, and
- if a box is coloured in blue, then it can only appear in the second row.

These rules allow us to define a correspondence between coloured boxes and spiders:

- each column consisting of two red boxes corresponds to a $[1, 1]$ spider,
- each blue box corresponds to a $[1, 0]$ spider, and
- each green box corresponds to a $[0, 1]$ spider,

i.e.



(31)

We impose one additional rule for colouring in boxes: blue and green boxes cannot appear to the left of any red box. Under the above correspondence between coloured boxes and spiders, this ensures that a fully coloured tableau corresponds to the normal form of a $(k, l)$–antispherical bipartition.

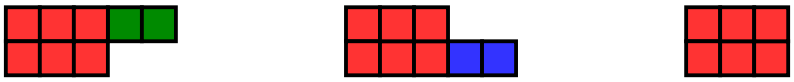
From here, we are able to construct the algorithm.

First, we take the tableau and initially colour it as follows, depending on the values of $k$ and $l$:

- if $k > l$, then we colour the last $k - l$ boxes in the second row with blue, and the remaining boxes with red,
- if $k < l$, then we colour the last $l - k$ boxes in the first row with green, and the remaining boxes with red,
- otherwise, we colour all boxes with red.

Note that the tableau is empty when $k = l = 0$, in which case there is nothing to colour.

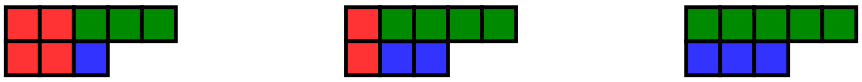
For example, for different values of $k$ and $l$, we would have the following initial tableaux:



(32)

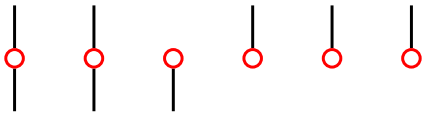
Then, starting from the right, we change the colours of each red column by colouring the box in the first row with green and the box in the second row with blue, creating new coloured tableaux as we move towards the left, until we form a tableau that has only blue and green colours.

For example, for the initial tableau on the left in (32), this process generates the following tableaux:



(33)

By the above correspondence between coloured boxes and spiders, we see that each one produces the normal form of a $(k, l)$–antispherical bipartition. For example, we see that the left most diagram in (33) corresponds to the normal form of the $(3, 5)$–antispherical bipartition



(34)

It is clear that this algorithm generates all possible $(k, l)$–antispherical bipartitions. In fact, we know their total number.

**Lemma C.1.** *The number of $(k, l)$–antispherical bipartitions is $1 + \min(k, l)$.*

*Proof.* This is immediate from the construction. We initially generate a tableau with $\min(k, l)$ columns, all coloured red, and depending on the values of $k$ and $l$, potentially an additional right tail of boxes coloured in either green or blue. From there we generate new tableaux by changing each of the $\min(k, l)$ columns in this tableau, one by one, from two red colours to a column consisting of a green and blue coloured cell. Since each tableau corresponds to a $(k, l)$–antispherical bipartition, there are $1 + \min(k, l)$ such antispherical bipartitions. □

We would also like to generate the subset of $(k, l)$–antispherical bipartitions having at most $n$ blocks with at most one $[1, 0]$ block and at most one $[0, 1]$ block. With the latter restriction, the algorithm given above shows that:

- if $k$ and $l$ differ by 2 or more, then there are no such antispherical bipartitions;
- if $k$ and $l$ differ by 1, then there is only one such antispherical bipartition so long as $n \geq \max(k, l)$, otherwise there are none; and
- if $k$ and $l$ are equal, then:
  - there is one such antispherical bipartition if $0 = k(= l)$;
  - there are two such antispherical bipartitions if $0 < k\ (= l) < n$;
  - there is one such antispherical bipartition if $n = k\ (= l)$;
  - there are no such antispherical bipartitions if $n < k\ (= l)$.

Since these $(k, l)$–antispherical bipartitions correspond bijectively to the basis matrices of $\mathrm{Hom}_{S_n}(\Lambda^k(\mathbb{R}^n), \Lambda^l(\mathbb{R}^n))$, by Theorem 4.14, the above characterisation immediately gives us its dimension, proving Corollary 4.16.

## D Expanding Matrices into Tensor Power Form

In Section 5, we said that although one can convert matrices from $\Lambda^k(\mathbb{R}^n)$ to $\Lambda^l(\mathbb{R}^n)$ into matrices from $(\mathbb{R}^n)^{\otimes k}$ to $(\mathbb{R}^n)^{\otimes l}$, it is impractical to store these matrices in memory as $n$, $k$ and $l$ increase. In any case, we show in this section how to perform this conversion.

Suppose that we have a matrix from $\Lambda^k(\mathbb{R}^n)$ to $\Lambda^l(\mathbb{R}^n)$. We convert each column into a map from a basis element of $\Lambda^k(\mathbb{R}^n)$ to a linear combination of basis elements in $\Lambda^l(\mathbb{R}^n)$. Then, for each such map, we replace the basis element on the left-hand side by its antisymmetrised expansion in $(\mathbb{R}^n)^{\otimes k}$ via

$$e_{i_1,i_2,\dots,i_k} \mapsto \sum_{\pi \in S_k} \mathrm{sgn}(\pi)\, e_{i_{\pi(1)}} \otimes e_{i_{\pi(2)}} \otimes \cdots \otimes e_{i_{\pi(k)}} \tag{35}$$

and similarly replace each basis element on the right-hand side by its antisymmetrised expansion in $(\mathbb{R}^n)^{\otimes l}$. We then reinterpret the resulting expressions as defining a matrix from $(\mathbb{R}^n)^{\otimes k}$ to $(\mathbb{R}^n)^{\otimes l}$.

*Example* D.1. Returning to Example 4.18, we saw that the $S_3$-equivariant weight matrix from $\Lambda^2(\mathbb{R}^3)$ to $\Lambda^1(\mathbb{R}^3)$ is

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1,2 & 1,3 & 2,3 \end{array} \\ \begin{bmatrix} w & w & 0 \\ -w & 0 & w \\ 0 & -w & -w \end{bmatrix} \end{array} \tag{36}$$

We now convert it into the $S_3$-equivariant weight matrix from $(\mathbb{R}^3)^{\otimes 2}$ to $\mathbb{R}^3$. The three columns become maps

$$e_{1,2} \mapsto we_1 - we_2 \qquad e_{1,3} \mapsto we_1 - we_3 \qquad e_{2,3} \mapsto we_2 - we_3 \tag{37}$$

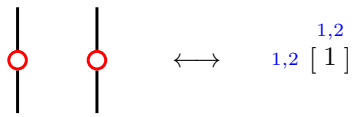
Replacing each basis element by its antisymmetrised expansion (35), we get

$$e_1 \otimes e_2 - e_2 \otimes e_1 \mapsto we_1 - we_2 \qquad e_1 \otimes e_3 - e_3 \otimes e_1 \mapsto we_1 - we_3 \qquad e_2 \otimes e_3 - e_3 \otimes e_2 \mapsto we_2 - we_3 \tag{38}$$

Hence the corresponding $S_3$-equivariant weight matrix from $(\mathbb{R}^3)^{\otimes 2}$ to $\mathbb{R}^3$ is

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1,1 & 1,2 & 1,3 & 2,1 & 2,2 & 2,3 & 3,1 & 3,2 & 3,3 \end{array} \\ \begin{bmatrix} 0 & w & w & -w & 0 & 0 & -w & 0 & 0 \\ 0 & -w & 0 & w & 0 & w & 0 & -w & 0 \\ 0 & 0 & -w & 0 & 0 & -w & w & w & 0 \end{bmatrix} \end{array} \tag{39}$$

*Example* D.2. One can show the $S_2$-equivariant matrix from $\Lambda^2(\mathbb{R}^2)$ to $\Lambda^2(\mathbb{R}^2)$ corresponding to the $(2, 2)$–antispherical bipartition $\{[1, 1], [1, 1]\}$ is




(40)

The column is $e_{1,2} \mapsto e_{1,2}$, which becomes

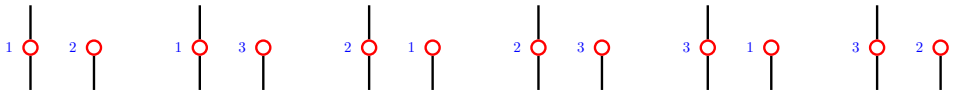
$$e_1 \otimes e_2 - e_2 \otimes e_1 \mapsto e_1 \otimes e_2 - e_2 \otimes e_1 \tag{41}$$

after replacing each basis element by its antisymmetrised expansion (35). Hence, the corresponding $S_2$-equivariant matrix from $(\mathbb{R}^2)^{\otimes 2}$ to $(\mathbb{R}^2)^{\otimes 2}$ is

$$\begin{array}{c} \\ 1,1 \\ 1,2 \\ 2,1 \\ 2,2 \end{array} \begin{array}{c} \begin{array}{cccc} 1,1 & 1,2 & 2,1 & 2,2 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \tag{42}$$

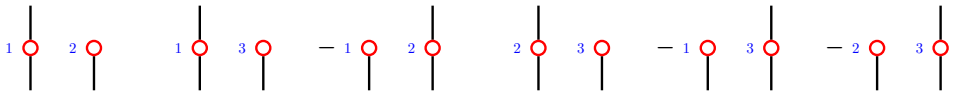
## E Additional Examples

*Example* E.1 (Calculations for Example 4.18). We first need to consider all possible 2-length tuples with elements in $[3]$, not allowing for repetitions amongst the elements, and use those to label the central red nodes of the normal form of the $(2,1)$–antispherical bipartition $\{[1,1],[1,0]\}$. They give six labelled diagrams, namely:




(43)

Next, we reorder the spiders in each diagram so that they are in increasing order from left to right, tracking any sign changes as appropriate.




(44)

We propagate the block labels to the end of each wire to produce the element $\binom{P}{Q}$ with accompanying sign:

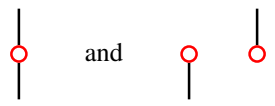
$$\binom{1}{1,2} : + \quad \binom{1}{1,3} : + \quad \binom{2}{1,2} : - \quad \binom{2}{2,3} : + \quad \binom{3}{1,3} : - \quad \binom{3}{2,3} : - \tag{45}$$

Hence we see that the basis matrix that corresponds to this antispherical bipartition is

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1,2 & 1,3 & 2,3 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & -1 \end{bmatrix} \end{array} \tag{46}$$

*Example* E.2 (Deep Sets $S_n$-Equivariant Weight Matrix from $\mathbb{R}^n$ to $\mathbb{R}^n$). To find the Deep Sets $S_n$-equivariant weight matrix from $\mathbb{R}^n$ to $\mathbb{R}^n$ that first appeared in Zaheer et al. (2017), we need to find a basis of $\mathrm{Hom}_{S_n}(\Lambda^1(\mathbb{R}^n), \Lambda^1(\mathbb{R}^n))$, since $\Lambda^1(\mathbb{R}^n) = \mathbb{R}^n$.

We need to consider the normal form of all $(1,1)$–antispherical bipartitions having at most $n$ blocks with at most one $[1,0]$ block and at most one $[0,1]$ block. Assuming that $n > 1$, they are

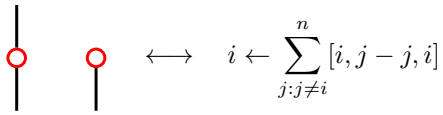



(47)

For the first diagram in (47), the only valid 1-length tuples are the elements of $[n]$. Hence, the matrix that corresponds to this $(1,1)$–antispherical bipartition is the $n \times n$ identity matrix. For the second diagram in (47), all possible 2-length tuples with distinct elements from $[n]$ are valid. Since we can swap the spiders around in this diagram without changing sign, by the rules that are given in Figure 2, this means that the matrix that corresponds to this antispherical bipartition is the $n \times n$ matrix that is 0 on the diagonal and 1 elsewhere. Hence, the $(i,j)$-entry of the $S_n$-equivariant weight matrix from $\mathbb{R}^n$ to $\mathbb{R}^n$ is $w_1$ if $i = j$ and $w_2$ if $i \neq j$, for weights $w_1, w_2 \in \mathbb{R}$, which is precisely the Deep Sets characterisation.

*Example* E.3. In Remark 5.3, we observed that once we obtain the IO pattern for a $(k, l)$–antispherical bipartition having at most one $[1, 0]$ block and at most one $[0, 1]$ block, we can use it to generate the corresponding basis matrix for any $n$ that is greater than or equal to the number of blocks in the antispherical bipartition.

For example, in Example 5.2, we saw that the IO pattern corresponding to the $(2, 1)$–antispherical bipartition $\{[1, 1], [1, 0]\}$, for $n \geq 2$, is



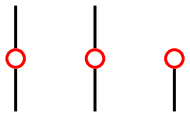
$$\longleftrightarrow \quad i \leftarrow \sum_{j:j\neq i}^{n} [i, j - j, i] \tag{48}$$

Since this is the only $(2, 1)$–antispherical bipartition that generates a basis element, we can immediately construct, for example, the $S_4$-equivariant weight matrix from $(\mathbb{R}^4)^{\otimes 2}$ to $\mathbb{R}^4$ from the IO pattern: it is

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\!\!\begin{array}{c} \begin{array}{cccccccccccccccc} 1{,}1 & 1{,}2 & 1{,}3 & 1{,}4 & 2{,}1 & 2{,}2 & 2{,}3 & 2{,}4 & 3{,}1 & 3{,}2 & 3{,}3 & 3{,}4 & 4{,}1 & 4{,}2 & 4{,}3 & 4{,}4 \end{array} \\ \begin{bmatrix} 0 & w & w & w & -w & 0 & 0 & 0 & -w & 0 & 0 & 0 & -w & 0 & 0 & 0 \\ 0 & -w & 0 & 0 & w & 0 & w & w & 0 & -w & 0 & 0 & 0 & -w & 0 & 0 \\ 0 & 0 & -w & 0 & 0 & 0 & -w & 0 & w & w & 0 & w & 0 & 0 & -w & 0 \\ 0 & 0 & 0 & -w & 0 & 0 & 0 & -w & 0 & 0 & 0 & -w & w & w & w & 0 \end{bmatrix} \end{array} \tag{49}$$

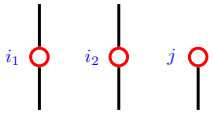
This shows that, instead of repeating the diagrammatic labelling procedure for each $n$, we can compute a single IO pattern and recover all of the corresponding matrices directly from it.

*Example* E.4. By following the steps given in Definition 5.1, we show how to calculate the IO pattern for the $(3, 2)$–antispherical bipartition $\beta$ whose normal form is



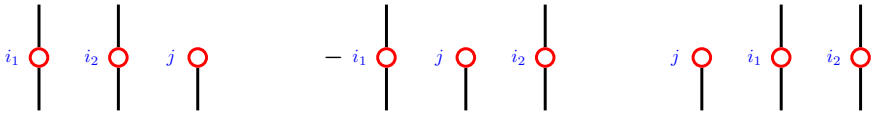
(50)

Since there is only one way to label this diagram with $i_1, i_2$ in Step 1, and given that the $[1, 0]$ spider appears in $\beta$, we have, after Step 2, the following labelled diagram:



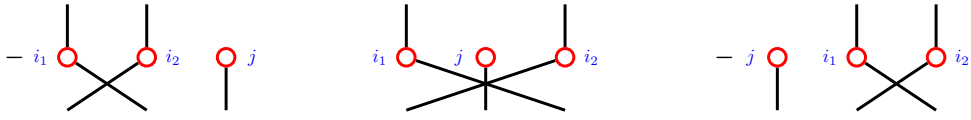

(51)

Step 3 does not apply as there is no $[0, 1]$ spider. Hence, after Step 4, having moved the $[1, 0]$ spider leftwards, we have the following three diagrams:




(52)

In Step 5, we generate three further diagrams, one from each of the diagrams in (52), by permuting the bottom-row wires of the $[1, 1]$ spiders in each diagram and flipping the sign in each (since the permutation that permutes the wires is odd):




(53)

By propagating the block labels to the end of each wire in each of the diagrams in (52) and (53), we obtain, in Step 6, the elements $\binom{P}{Q}$ with accompanying sign:

$$\binom{i_1, i_2}{i_1, i_2, j} : + \quad \binom{i_1, i_2}{i_1, j, i_2} : - \quad \binom{i_1, i_2}{j, i_1, i_2} : + \tag{54}$$

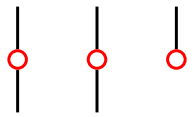
$$\binom{i_1, i_2}{i_2, i_1, j} : - \quad \binom{i_1, i_2}{i_2, j, i_1} : + \quad \binom{i_1, i_2}{j, i_2, i_1} : - \tag{55}$$

Hence, in Step 7, we obtain the IO pattern that corresponds to $\beta$:

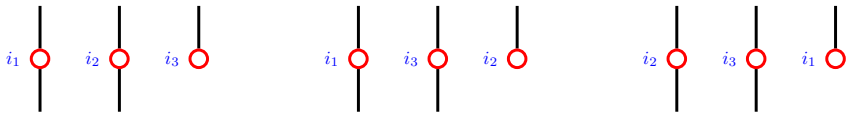
$$i_1, i_2 \leftarrow \sum_{j:j\neq i_1,i_2}^{n} [i_1, i_2, j - i_1, j, i_2 + j, i_1, i_2 - i_2, i_1, j + i_2, j, i_1 - j, i_2, i_1] \tag{56}$$

where the sum over $j$ appears on the right-hand side because $j$ does not appear on the left hand side.

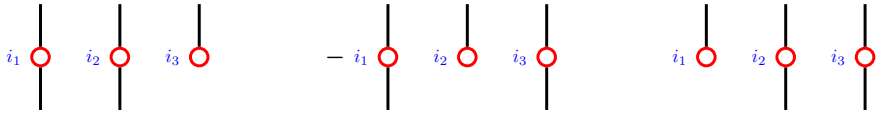
*Example* E.5. We now show how to calculate the IO pattern for the $(2, 3)$–antispherical bipartition $\beta$ whose normal form is

 (57)

After Step 1, we obtain the following three diagrams:

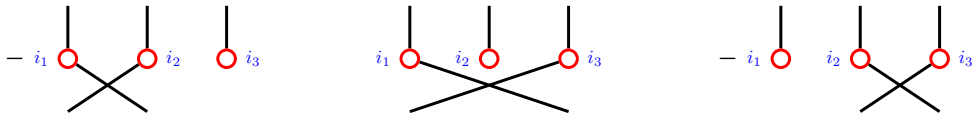
 (58)

Step 2 does not apply because $[1, 0]$ does not appear in $\beta$. Hence, after Step 3, these three diagrams become

 (59)

so that the output tuple, if we were to propagate the block labels, would be $I = (i_1, i_2, i_3)$.

As Step 4 does not apply, we now apply Step 5 to each of these diagrams to get

 (60)

Consequently, in Step 6, from (59) and (60), we obtain:

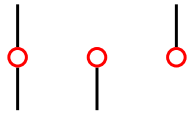
$$\binom{i_1, i_2, i_3}{i_1, i_2} : + \quad \binom{i_1, i_2, i_3}{i_1, i_3} : - \quad \binom{i_1, i_2, i_3}{i_2, i_3} : + \tag{61}$$

$$\binom{i_1, i_2, i_3}{i_2, i_1} : - \quad \binom{i_1, i_2, i_3}{i_3, i_1} : + \quad \binom{i_1, i_2, i_3}{i_3, i_2} : - \tag{62}$$

Hence, in Step 7, we obtain the IO pattern that corresponds to $\beta$:

$$i_1, i_2, i_3 \leftarrow i_1, i_2 - i_2, i_1 + i_3, i_1 - i_1, i_3 + i_2, i_3 - i_3, i_2 \tag{63}$$

*Example* E.6. Finally, we show how to calculate the IO pattern for the $(2, 2)$–antispherical bipartition $\beta$ whose normal form is

 (64)

After Step 1, we obtain the following two diagrams:

 (65)

which become, after Step 2:

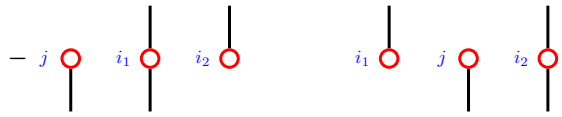
 (66)

We apply Step 3 to the diagrams in (66) to get




(67)

In Step 4, we move the $[1,0]$ spider to the left of the $[1,1]$ spider in each diagram to obtain:




(68)

Since there is only one $[1,1]$ spider, Step 5 is trivial. Hence, in Step 6, from (67) and (68), we obtain:

$$\binom{i_1,i_2}{i_1,j}:+ \qquad \binom{i_1,i_2}{i_2,j}:- \qquad \binom{i_1,i_2}{j,i_1}:- \qquad \binom{i_1,i_2}{j,i_2}:+ \tag{69}$$

Consequently, in Step 7, we obtain the IO pattern that corresponds to $\beta$:

$$i_1,i_2 \leftarrow \sum_{j:j\neq i_1,i_2}^{n} [i_1, j-j, i_1-i_2, j+j, i_2] \tag{70}$$

## F Pseudocode for IO Patterns

Definition 5.1 describes, diagrammatically, how an IO pattern is constructed from a $(k,l)$–antispherical bipartition $\beta$ having at most one $[1,0]$ block and at most one $[0,1]$ block, in terms of labelled spiders and the sign rules of Figure 2. In this section we give the same construction procedurally in the form of pseudocode, together with a procedure for applying an IO pattern to an antisymmetric tensor. This makes the cost of the method explicit and shows that no weight matrix is formed at any stage.

For the purposes of these algorithms we record an IO pattern as a pair $(\mathrm{sum}, \mathcal{T})$. Here $\mathrm{sum} \in \{\mathrm{true}, \mathrm{false}\}$ records whether $\beta$ contains a $[1,0]$ block, and $\mathcal{T}$ is a set of **terms**, one for each signed tuple that appears on the right hand side of the IO pattern. Each term in $\mathcal{T}$ is a pair $(\varepsilon, s)$ such that $\varepsilon \in \{+1,-1\}$ is a sign and $s = (s_1,\dots,s_k)$ is a $k$-tuple of **formal symbols**. If sum is true, then exactly one entry of $s$ is the formal symbol $j$, otherwise no entry is. The remaining entries are distinct formal symbols from $\{i_1,\dots,i_l\}$, namely those that label the $[1,1]$ blocks of $\beta$. In each term, the symbol that labels the $[0,1]$ block in that labelling is omitted from $s$, since such a block has no wire to the bottom row. Crucially, $\mathcal{T}$ depends only on $\beta$, hence it is *independent of* $n$. This makes the storage cost of an IO pattern $O(1)$ in $n$.

For example, the IO pattern $i \leftarrow \sum_{j:j\neq i}[\,i, j-j, i\,]$ of Example 5.2 is recorded as $\mathrm{sum} = \mathrm{true}$ and $\mathcal{T} = \{(+1,(i,j)),\ (-1,(j,i))\}$, writing $i$ for $i_1$ as $l = 1$.

Now fix a tuple $I = (i_1,\dots,i_l)$ of pairwise distinct elements of $[n]$. For each term $(\varepsilon, s) \in \mathcal{T}$, substituting the entries of $I$ into $s$, together with a value $j \in [n] \setminus \{i_1,\dots,i_l\}$ when sum is true, gives a tuple of indices of $T$, which we write as $s[I,j]$ in that case and as $s[I]$ otherwise. Having fixed these values, the IO pattern determines the entry $f_\beta(T)_I$ as

$$f_\beta(T)_I \;=\; \begin{cases} \displaystyle\sum_{j\in[n]\setminus\{i_1,\dots,i_l\}} \ \sum_{(\varepsilon,s)\in\mathcal{T}} \varepsilon\, T_{s[I,j]} & \text{if sum} = \text{true}, \\ \displaystyle\sum_{(\varepsilon,s)\in\mathcal{T}} \varepsilon\, T_{s[I]} & \text{otherwise.} \end{cases} \tag{71}$$

We now present two algorithms. Algorithm 1 builds the IO pattern $(\mathrm{sum}, \mathcal{T})$ of a $(k,l)$–antispherical bipartition $\beta$ having at most one $[1,0]$ block and at most one $[0,1]$ block, following the steps of Definition 5.1. Since the building of an IO pattern depends only on $\beta$, it needs to be performed only once for each such bipartition. Algorithm 2 then applies the resulting IO pattern to an antisymmetric tensor. This is the computation that is performed whenever the corresponding basis map is evaluated for some choice of $n$. By Corollary 4.16 a layer is a weighted sum of at most two basis maps, so a layer is realised by applying at most two IO patterns and taking the corresponding weighted sum. All dependence on $n$ is confined to the second stage, and no weight matrix is formed in either stage.

**Algorithm 1** COMPILEIOPATTERN($\beta$) — build the IO pattern of a $(k, l)$–antispherical bipartition $\beta$ having at most one $[1, 0]$ block and at most one $[0, 1]$ block. This is Definition 5.1 written procedurally, and is performed once for each bipartition.

**Require:** A $(k, l)$–antispherical bipartition $\beta$ in normal form, having at most one $[1, 0]$ block and at most one $[0, 1]$ block; pairwise distinct formal symbols $i_1, \ldots, i_l$, and a further distinct formal symbol $j$ if $\beta$ has a $[1, 0]$ block.
**Ensure:** The IO pattern $(\mathrm{sum}, \mathcal{T})$ for $\beta$.

$m \leftarrow$ number of $[1, 1]$ blocks of $\beta$
$\mathrm{sum} \leftarrow$ **true** if $\beta$ has a $[1, 0]$ block, else **false**
$b \leftarrow 1$ if $\beta$ has a $[0, 1]$ block, else $0$
$\mathcal{T} \leftarrow \emptyset$
**for all** subsets $C = \{c_1 < \cdots < c_m\} \subseteq \{1, \ldots, l\}$ **do**
    **if** $b = 1$ **then**
        $\{d\} \leftarrow \{1, \ldots, l\} \setminus C$
        $\varepsilon_0 \leftarrow (-1)^{l-d}$
    **else**
        $\varepsilon_0 \leftarrow +1$
    **end if**
    $R \leftarrow \{0, 1, \ldots, m\}$ if sum, else $\{m\}$
    **for all** $r \in R$ **do**
        $\varepsilon_1 \leftarrow \varepsilon_0 \cdot (-1)^{m-r}$
        **for all** $\tau \in S_m$ **do**
            $\varepsilon \leftarrow \varepsilon_1 \cdot \mathrm{sgn}(\tau)$
            $s \leftarrow (i_{c_{\tau(1)}}, \ldots, i_{c_{\tau(m)}})$, with $j$ inserted in position $r + 1$ if sum
            add $(\varepsilon, s)$ to $\mathcal{T}$
        **end for**
    **end for**
**end for**
**return** $(\mathrm{sum}, \mathcal{T})$

**How Algorithm 1 realises Definition 5.1.** We match the algorithm to the steps of Definition 5.1 in turn. Since $\beta$ is in normal form, its $m$ blocks of type $[1, 1]$ come first, followed by the $[1, 0]$ block and then the $[0, 1]$ block. Hence $l = m + b$, where $b = 1$ if $\beta$ has a $[0, 1]$ block and $b = 0$ otherwise. Similarly, $k = m + 1$ if sum is true and $k = m$ otherwise. Throughout, $C$ and $d$ record positions in the tuple of formal symbols $I = (i_1, \ldots, i_l)$ rather than index values. Consequently, requiring that the top row reads $i_1, \ldots, i_l$ in order is the same as requiring that the positions read $1, \ldots, l$ in order. This is what makes the construction independent of $n$.

*The loop over subsets $C$* is Step 1 of Definition 5.1. The spiders with a wire to the top row are the $m$ spiders of type $[1, 1]$ together with the $b$ spiders of type $[0, 1]$, so there are $l$ of them, matching the length of $I$. Choosing $C$ fixes which positions of $I$ label the $[1, 1]$ spiders, and the remaining position $d$ labels the $[0, 1]$ spider if it exists. Because Step 1 needs the labels of the $[1, 1]$ spiders to increase from left to right, only the subset matters and not its order, giving $\binom{l}{m}$ choices. Step 2, which labels the $[1, 0]$ spider with $j$, needs no separate line, since $j$ is supplied as an input and is inserted into $s$ when the term is formed.

*The sign $\varepsilon_0$* is Step 3. The $[0, 1]$ spider is rightmost in normal form, so its top-row wire begins at position $l$ and must move to position $d$, passing $l - d$ of the other top-row wires, each of which belongs to a $[1, 1]$ spider. Each such swap flips the sign by Figure 2. If a $[1, 0]$ spider is also present then the $[0, 1]$ spider passes it as well, but that swap leaves the sign unchanged, since the two spiders have wires in different rows.

*The loop over $r$* is Step 4. When sum is true, the $[1, 0]$ spider exists and begins to the right of all $m$ spiders of type $[1, 1]$. It then moves leftwards, generating one diagram for each resting position; writing $r$ for the number of $[1, 1]$ spiders remaining to its left, it has passed $m-r$ of them, contributing $(-1)^{m-r}$ to the sign. When sum is false there is no such spider and the loop runs once with no sign change.

**Algorithm 2** APPLYIOPATTERN($(\mathrm{sum}, \mathcal{T})$, $T$, $n$) — apply the IO pattern of a $(k, l)$–antispherical bipartition $\beta$ having at most one $[1, 0]$ block and at most one $[0, 1]$ block to an antisymmetric tensor $T \in (\mathbb{R}^n)^{\otimes k}$, returning $f_\beta(T) \in (\mathbb{R}^n)^{\otimes l}$. No weight matrix is ever formed.

**Require:** IO pattern $(\mathrm{sum}, \mathcal{T})$ for $\beta$; antisymmetric $T \in (\mathbb{R}^n)^{\otimes k}$; dimension $n \geq$ number of blocks of $\beta$.
**Ensure:** $U = f_\beta(T) \in (\mathbb{R}^n)^{\otimes l}$.

```
1: U ← 0 ∈ (ℝ^n)^{⊗l}
2: for all I = (i_1, ..., i_l) ∈ [n]^l with pairwise distinct entries do
3:     u ← 0
4:     for all (ε, s) ∈ T do
5:         if sum then
6:             for j ∈ [n] \ {i_1, ..., i_l} do
7:                 u ← u + ε · T_{s[I,j]}
8:             end for
9:         else
10:            u ← u + ε · T_{s[I]}
11:        end if
12:    end for
13:    U_I ← u
14: end for
15: return U
```

*The loop over $\tau$* is Step 5, permuting only the bottom-row wires of the $[1, 1]$ spiders and flipping the sign for odd permutations.

*Forming $s$* is Step 6. The tuple $s$ is the bottom row of the propagated pair $\binom{I}{J}$: the $[1, 1]$ spiders contribute their labels, which are permuted by $\tau$, and the $[1, 0]$ spider contributes $j$ at the position it reached. The $[0, 1]$ spider has no wire to the bottom row, so its label $i_d$ does not appear. Hence $s$ uses only $m$ of the $l$ symbols $i_1, \ldots, i_l$, together with $j$ when sum is true, and so it has length $k$, as required.

**How Algorithm 2 applies an IO pattern.** Algorithm 2 is a direct evaluation of (71): the outer loop fixes an output index $I$, and the inner loops accumulate the signed sum over the terms of $\mathcal{T}$ and, when sum is true, they accumulate over the values of $j$. Note that the outer loop ranges over all $I \in [n]^l$ with pairwise distinct entries rather than only over increasing tuples. Since $f_\beta$ maps into $\Lambda^l(\mathbb{R}^n)$, its output is antisymmetric by construction; the outer loop simply writes that output across all orderings of the indices, so that $U$ is produced directly as a tensor in $(\mathbb{R}^n)^{\otimes l}$ and is a valid input to a subsequent layer. In particular, the entries of $U$ at repeated indices are zero.

**Cost.** Counting the iterations of Algorithm 1 gives the size of $\mathcal{T}$ exactly: letting $t = |\mathcal{T}|$, we have that $t = \binom{l}{m}(m+1)\, m! = \binom{l}{m}(m+1)!$ when sum is true and $t = \binom{l}{m}\, m!$ otherwise, so in particular $t$ does not depend on $n$. The work done by Algorithm 2 follows from its loop structure. The outer loop runs over $O(n^l)$ output indices, the loop within it runs over the $t$ terms of $\mathcal{T}$, and, when sum is true, the innermost loop runs over the $O(n)$ values of $j$. Multiplying these gives $O(t \cdot n^{l+1})$ work when sum is true and $O(t \cdot n^l)$ otherwise. Section I gives the resulting complexity of a layer and compares it with the cost of realising an explicit weight matrix.

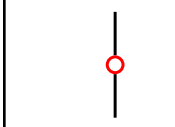
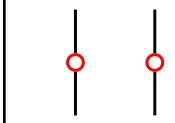
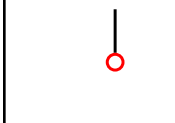
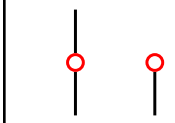
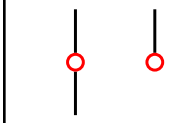
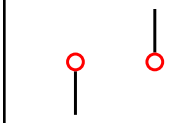
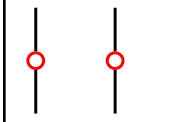
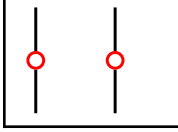
**Vectorised realisation.** Algorithm 2 is written with explicit loops for clarity only. In practice the terms of $\mathcal{T}$ are evaluated by vectorised tensor operations rather than by looping over indices. The closed form can often be simplified further if one uses the antisymmetry of $T$. For the antispherical bipartition of Example 5.2, the pattern $i \leftarrow \sum_{j:j\neq i}[\, i, j-j, i\,]$ reduces to $f_\beta(T)_i = 2\sum_{j=1}^{n} T_{i,j}$, which in code is simply `2 * T.sum(dim=-1)`. We give simplified closed forms alongside each of the IO patterns that are listed in Section G.

## G REFERENCE LIST: IO PATTERNS FOR $(k, l)$–ANTISPHERICAL BIPARTITIONS

We provide some IO patterns below for commonly appearing $(k, l)$–antispherical bipartitions that have at most one $[1, 0]$ block and at most one $[0, 1]$ block. Note that the IO pattern only appears for $n$ greater than or equal to the number of spiders in the corresponding antispherical bipartition.

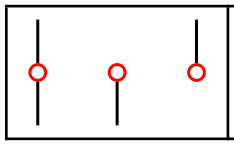
For each IO pattern, we also give the simplified closed form that is obtained by using the antisymmetry of the input tensor $T$. Throughout, $T$ is the input tensor and $f_\beta(T)$ is the output tensor. Note that when $k = 0$ the input is a scalar, since $\Lambda^0(\mathbb{R}^n) = \mathbb{R}$, and when $l = 0$ so is the output.

Table 2: IO patterns and closed forms for commonly appearing $(k, l)$–antispherical bipartitions.

| $(k, l)$– Antispherical Bipartition | IO Pattern | Closed form for antisymmetric $T$ |
|---|---|---|
|  | $i \leftarrow i$ | $f_\beta(T)_i = T_i$ |
|  | $i, j \leftarrow i, j - j, i$ | $f_\beta(T)_{i,j} = 2T_{i,j}$ |
|  | $i \leftarrow 1$ | $f_\beta(T)_i = T$ |
|  | $1 \leftarrow \sum_{j=1}^n j$ | $f_\beta(T) = \sum_{j=1}^n T_j$ |
|  | $i \leftarrow \sum_{j:j\neq i}^n [i, j - j, i]$ | $f_\beta(T)_i = 2\sum_{j=1}^n T_{i,j}$ |
|  | $i, j \leftarrow i - j$ | $f_\beta(T)_{i,j} = T_i - T_j$ |
|  | $i \leftarrow \sum_{j:j\neq i}^n j$ | $f_\beta(T)_i = \left(\sum_{j=1}^n T_j\right) - T_i$ |
|  | $i, j \leftarrow \sum_{k:k\neq i,j}^n [i, j, k - i, k, j + k, i, j - j, i, k + j, k, i - k, j, i]$ | $f_\beta(T)_{i,j} = 6\sum_{k=1}^n T_{i,j,k}$ |
|  | $i, j, k \leftarrow i, j-j, i+k, i-i, k+j, k-k, j$ | $f_\beta(T)_{i,j,k} = 2(T_{i,j} + T_{j,k} + T_{k,i})$ |

Continued on next page

Table 2: IO patterns and closed forms for commonly appearing $(k, l)$–antispherical bipartitions. (Continued)

| | | |
|---|---|---|
|  | $i, j \leftarrow \sum_{k:k\neq i,j}^{n} [i, k-k, i-j, k+k, j]$ | $f_\beta(T)_{i,j} = 2\sum_{k=1}^{n}(T_{i,k} - T_{j,k}) - 4T_{i,j}$ |

## H Relationship to Classical Exterior Algebra

The spaces $\Lambda^k(\mathbb{R}^n)$ are the antisymmetric powers of classical exterior algebra, which come with a standard family of linear operators. For completeness, it is worth understanding how the maps that we have characterised in Theorem 4.14 and Corollary 4.16 relate to those operators.

We do this in two parts. We first show that our basis maps can be written in terms of the classical operators, and then explain why the resulting correspondence is a consequence of the classification results rather than an alternative route to them.

Throughout this section we take $0 \leq k \leq n$, since $\Lambda^k(\mathbb{R}^n) = 0$ otherwise, and we write $e_I = e_{i_1} \wedge \cdots \wedge e_{i_k}$ for $I \in \Lambda[n]^k$. This is consistent with Section 3, since these wedge products form the standard basis of the exterior power space. We also abbreviate $\rho_k^\Lambda(\sigma)x$, where $\rho_k^\Lambda$ is the representation of $S_n$ on $\Lambda^k(\mathbb{R}^n)$ that is defined in Section 3, to $\sigma x$, for any $k$.

**Two families of operators.** Fix a vector $u = \sum_{i\in[n]} u_i e_i$ in $\mathbb{R}^n$. There are two standard ways to build a linear map between antisymmetric powers out of $u$: one that raises the degree by inserting an index, and one that lowers it by deleting an index.

The *exterior product with* $u$ is the map $u \wedge (\cdot) : \Lambda^k(\mathbb{R}^n) \to \Lambda^{k+1}(\mathbb{R}^n)$ that sends $x$ to $u \wedge x$. On a basis element it acts as follows. If $i \notin I$, we have that

$$e_i \wedge e_I \;=\; (-1)^{|\{j\,:\,i_j<i\}|}\, e_{I\cup\{i\}} \tag{72}$$

which is obtained by moving $e_i$ to the right past each index of $I$ that is smaller than $i$, one transposition at a time, where $I \cup \{i\}$ denotes the tuple in $\Lambda[n]^{k+1}$ that is obtained by adding $i$ to $I$ and rewriting it in increasing order. If $i \in I$ instead, then $e_i \wedge e_I = 0$, since $e_i \wedge e_i = 0$. Consequently, expanding $u$ in the standard basis and using linearity in the first argument gives

$$u \wedge e_I \;=\; \sum_{i\in[n]\setminus I} u_i\,(-1)^{|\{j\,:\,i_j<i\}|}\, e_{I\cup\{i\}} \tag{73}$$

For example, when $n = 3$ and $I = (2)$,

$$u \wedge e_2 \;=\; u_1\,(-1)^0\, e_{1,2} \;+\; u_3\,(-1)^1\, e_{2,3} \;=\; u_1 e_{1,2} - u_3 e_{2,3} \tag{74}$$

where the sign in the second term arises because 3 must pass the entry 2.

For $k \geq 1$, the *interior product with* $u$, also called contraction, is the map $\iota_u : \Lambda^k(\mathbb{R}^n) \to \Lambda^{k-1}(\mathbb{R}^n)$ that deletes one factor at a time: for any $v_1, \ldots, v_k \in \mathbb{R}^n$,

$$\iota_u(v_1 \wedge \cdots \wedge v_k) \;=\; \sum_{j=1}^{k} (-1)^{j-1} \langle u, v_j\rangle\, v_1 \wedge \cdots \wedge \widehat{v_j} \wedge \cdots \wedge v_k \tag{75}$$

where $\widehat{v_j}$ denotes omission of the $j$th factor. On a basis element this gives

$$\iota_u\, e_I \;=\; \sum_{j=1}^{k} (-1)^{j-1}\, u_{i_j}\, e_{I\setminus\{i_j\}} \tag{76}$$

since $\langle u, e_{i_j}\rangle = u_{i_j}$, where $I \setminus \{i_j\}$ is the tuple that is obtained by deleting the $j$th entry.

Note in particular that each term removes one index, with the coefficient $u_{i_j}$ weighting that choice, and the sign recording the position of the deleted entry. For example,

$$\iota_u\, e_{1,2,3} \;=\; u_1 e_{2,3} - u_2 e_{1,3} + u_3 e_{1,2}. \tag{77}$$

Moreover, note that when $k = 1$, the sum in (76) has a single term and reduces to $\iota_u\, e_i = u_i$, and so for $v \in \mathbb{R}^n$

$$\iota_u v \;=\; \iota_u\Big(\sum_{i\in[n]} v_i e_i\Big) \;=\; \sum_{i\in[n]} v_i\, \iota_u(e_i) \;=\; \sum_{i\in[n]} v_i u_i \;=\; \langle u, v\rangle \tag{78}$$

Together with the identity $\mathrm{id} : \Lambda^k(\mathbb{R}^n) \to \Lambda^k(\mathbb{R}^n)$, we compare our basis maps against these operators.

**Only one choice of $u$ is $S_n$-equivariant.** We see that (73) and (76) give a whole family of operators, one for each $u \in \mathbb{R}^n$. Almost none of them are $S_n$-equivariant, and identifying which are is the first step in obtaining the desired relationship.

Recall from Section 3 the representation $\rho_k^\Lambda$ of $S_n$ on $\Lambda^k(\mathbb{R}^n)$, which relabels the indices of basis elements and reorders them, and multiplies the result by the sign of the reordering permutation. That sign is exactly the one that is incurred by reordering the factors of a wedge product: interchanging two factors changes the sign, and so the sign of the reordering permutation is the parity of the transpositions that are needed to sort the relabelled indices into increasing order. Hence

$$\sigma\, e_I \;=\; (\sigma e_{i_1}) \wedge \cdots \wedge (\sigma e_{i_k}) \tag{79}$$

that is, $\sigma$ acts on each factor of a wedge product separately. In particular, $\sigma(e_i \wedge e_I) = (\sigma e_i) \wedge (\sigma e_I)$, and so extending by bilinearity in $u$ and $x$ gives

$$\sigma(u \wedge x) \;=\; (\sigma u) \wedge (\sigma x) \qquad \text{for all } u \in \mathbb{R}^n,\ x \in \Lambda^k(\mathbb{R}^n) \tag{80}$$

Now suppose that $u \wedge (\cdot)$ is equivariant, that is, $\sigma(u \wedge x) = u \wedge (\sigma x)$ for all $\sigma$ and $x$. Comparing this with (80) gives $(\sigma u) \wedge (\sigma x) = u \wedge (\sigma x)$, and since $\sigma x$ ranges over all of $\Lambda^k(\mathbb{R}^n)$ as $x$ does, this implies that

$$(\sigma u - u) \wedge y = 0 \qquad \text{for every } y \in \Lambda^k(\mathbb{R}^n) \tag{81}$$

Now $(\sigma u - u) \wedge y$ lies in $\Lambda^{k+1}(\mathbb{R}^n)$ which is non-zero so long as $k < n$. In that case wedging by a non-zero vector is not the zero map, so by (81) we must have that $\sigma u - u = 0$, that is, $\sigma u = u$.

The contraction behaves in a similar way. Permutations preserve the inner product, so $\langle \sigma u, \sigma e_{i_j}\rangle = \langle u, e_{i_j}\rangle = u_{i_j}$. Applying (75) to $\sigma e_I = (\sigma e_{i_1}) \wedge \cdots \wedge (\sigma e_{i_k})$ and using the same factor-wise property one degree lower gives $\iota_{\sigma u}(\sigma e_I) = \sum_j (-1)^{j-1} u_{i_j}\, \sigma e_{I\setminus\{i_j\}} = \sigma(\iota_u e_I)$, and extending linearly in $x$ gives

$$\sigma(\iota_u x) \;=\; \iota_{\sigma u}(\sigma x) \qquad \text{for all } x \in \Lambda^k(\mathbb{R}^n) \tag{82}$$

If $\iota_u$ is equivariant then $\sigma(\iota_u x) = \iota_u(\sigma x)$ for all $x \in \Lambda^k(\mathbb{R}^n)$, hence by (82), $\iota_{\sigma u}(\sigma x) = \iota_u(\sigma x)$ for all such $x$. Since $\sigma x$ ranges over all of $\Lambda^k(\mathbb{R}^n)$ as $x$ does, we must have $\iota_{\sigma u} = \iota_u$ as maps. From (76) we see that $\iota_v$ is linear in $v$, hence $\iota_{\sigma u} - \iota_u = \iota_{\sigma u - u}$, and so $\iota_{\sigma u - u}$ is the zero map. Writing $w = \sigma u - u$, if $w_i \neq 0$ for some $i$ then, choosing $I \in \Lambda[n]^k$ that contains $i$, say $i = i_p$ (which is possible since $1 \le k \le n$), the coefficient of $e_{I\setminus\{i_p\}}$ in $\iota_w e_I = \sum_{q=1}^k (-1)^{q-1} w_{i_q} e_{I\setminus\{i_q\}}$ is $(-1)^{p-1} w_i \neq 0$, since the tuples $I \setminus \{i_q\}$ are pairwise distinct. Hence $w = 0$, giving $\sigma u = u$.

So an equivariant operator of either kind requires a vector $u$ that is fixed by every permutation in $S_n$, and the only such vectors are the constant ones. (The exception to this statement is the case $k = n$ for the exterior product, where the target space is zero. The operator is then the zero map, and is therefore equivariant for every $u$. But since the zero map cannot be a basis element, this case can be set aside.) Conversely, writing $\mathbf{1} = \sum_{i\in[n]} e_i$, we have $\sigma\mathbf{1} = \mathbf{1}$ for every $\sigma \in S_n$, since $\sigma$ merely permutes the $e_i$. Setting $u = \mathbf{1}$ in (80) and (82) therefore gives $\sigma(\mathbf{1} \wedge x) = \mathbf{1} \wedge (\sigma x)$ and $\sigma(\iota_{\mathbf{1}} x) = \iota_{\mathbf{1}}(\sigma x)$, so both operators are indeed equivariant. We conclude that the equivariant members of these two families are exactly the scalar multiples of

$$\mathbf{1} \wedge e_I = \sum_{i\notin I} (-1)^{|\{j\,:\,i_j<i\}|} e_{I\cup\{i\}} \qquad \iota_{\mathbf{1}}\, e_I = \sum_{j=1}^{k} (-1)^{j-1} e_{I\setminus\{i_j\}} \tag{83}$$

which is obtained by setting every $u_i = 1$ in (73) and (76).

**Splitting $\Lambda^k(\mathbb{R}^n)$ into two pieces.** To see which combinations of $\mathrm{id}$, $\mathbf{1} \wedge (\cdot)$ and $\iota_{\mathbf{1}}$ are non-zero and independent, it helps to decompose the space they act on. Let

$$V \;=\; \Big\{ x \in \mathbb{R}^n \;:\; \textstyle\sum_i x_i = 0 \Big\} \tag{84}$$

be the *standard representation*, which has dimension $n-1$. Since $\mathbf{1}$ does not lie in $V$, we get that $\mathbb{R}^n = \mathrm{span}(\mathbf{1}) \oplus V$. Moreover, both summands are preserved by every permutation in $S_n$. Taking exterior powers of this splitting separates $\Lambda^k(\mathbb{R}^n)$ into the part that is built entirely from $V$ and the part that carries a factor of $\mathbf{1}$ as an isomorphism of representations of $S_n$:

$$\Lambda^k(\mathbb{R}^n) \;\cong\; \Lambda^k(V) \;\oplus\; \big(\mathbf{1} \wedge \Lambda^{k-1}(V)\big) \tag{85}$$

Because $\dim V = n-1$, the space $\Lambda^j(V)$ is non-zero exactly when $0 \le j \le n-1$. Hence the first summand of (85) is non-zero precisely when $k < n$, and the second is non-zero precisely when $k > 0$.

The two non-identity operators interact with (85) in a simple way. For the first summand, since $\iota_{\mathbf{1}} v = \langle \mathbf{1}, v \rangle$ by (78) and then $\langle \mathbf{1}, v \rangle = 0$ for every $v \in V$ by (84), we see that $\iota_{\mathbf{1}}$ vanishes on $\Lambda^k(V)$ by (75). For the second summand, for $w \in \Lambda^{k-1}(V)$, by (75)

$$\iota_{\mathbf{1}}(\mathbf{1} \wedge w) \;=\; \langle \mathbf{1}, \mathbf{1} \rangle\, w \;-\; \mathbf{1} \wedge \iota_{\mathbf{1}} w \;=\; n\, w \tag{86}$$

where the second term is zero because $\iota_{\mathbf{1}}$ vanishes on $\Lambda^{k-1}(V)$.

Applying $\mathbf{1} \wedge (\cdot)$ to the result gives $n(\mathbf{1} \wedge w)$, so the composite $\mathbf{1} \wedge \iota_{\mathbf{1}}$ (apply $\iota_{\mathbf{1}}$ then $\mathbf{1} \wedge (\cdot)$) on $\Lambda^k(\mathbb{R}^n)$ satisfies

$$\mathbf{1} \wedge \iota_{\mathbf{1}} \;=\; n \cdot \big(\text{projection onto } \mathbf{1} \wedge \Lambda^{k-1}(V)\big). \tag{87}$$

In other words, $\mathbf{1} \wedge \iota_{\mathbf{1}}$ is, up to the factor $n$, the projection picking out the second summand of (85), while $\mathrm{id}$ is the sum of the projections onto both. These two are linearly independent exactly when both summands are non-zero, that is, when $0 < k < n$. At $k = 0$ the second summand is absent and $\mathbf{1} \wedge \iota_{\mathbf{1}} = 0$; at $k = n$ the first is absent and (87) gives $\mathbf{1} \wedge \iota_{\mathbf{1}} = n \cdot \mathrm{id}$. In either case only $\mathrm{id}$ survives.

The same decomposition determines the behaviour of the degree-changing operators. The map $\iota_{\mathbf{1}}$ sends $\mathbf{1} \wedge \Lambda^{k-1}(V)$ onto $\Lambda^{k-1}(V)$ and annihilates $\Lambda^k(V)$, so it is non-zero precisely when $1 \le k \le n$. The map $\mathbf{1} \wedge (\cdot)$ is injective on $\Lambda^k(V)$ and annihilates the other summand, so it is non-zero precisely when $n \ge k+1$.

**Reading off the basis.** Corollary 4.16 gives the dimension of $\mathrm{Hom}_{S_n}(\Lambda^k(\mathbb{R}^n), \Lambda^l(\mathbb{R}^n))$ in every case. Matching those dimensions against the operators identified above results in the following dictionary:

- if $l = k-1$ and $n \ge k$, then $\{\iota_{\mathbf{1}}\}$ is a basis;
- if $l = k+1$ and $n \ge l$, then $\{\mathbf{1} \wedge (\cdot)\}$ is a basis;
- if $0 < k = l < n$, then $\{\mathrm{id},\ \mathbf{1} \wedge \iota_{\mathbf{1}}\}$ is a basis;
- if $k = l = 0$ or $k = l = n$, then $\{\mathrm{id}\}$ is a basis;
- in all remaining cases the space is zero.

In each line, the preceding paragraphs supply operators that are non-zero, or independent, and Corollary 4.16 supplies the fact that there is no room for anything further.

**Why the classification is needed.** The argument given in this section establishes that certain equivariant maps exist and are independent; it cannot establish that no others do. A candidate list of operators is known to be complete only once one knows $\dim \mathrm{Hom}_{S_n}(\Lambda^k(\mathbb{R}^n), \Lambda^l(\mathbb{R}^n))$ for every triple $(k, l, n)$, which is supplied by Corollary 4.16 and used at every step above. The dictionary is therefore a consequence of the classification rather than a substitute for it.

# I Memory and Computational Complexity Analysis

We expand here on the complexity paragraph that appeared in Section 5.

The activations themselves, $O(n^k)$ for the input and $O(n^l)$ for the output, are unavoidable and are identical under either approach, so we compare only the cost of a layer map. Since a full layer between antisymmetric tensors is a weighted sum of at most two basis maps by Corollary 4.16, it is sufficient to consider the storage and compute complexity of a single basis map.

For storage complexity, realising a basis map as an explicit weight matrix requires $O(n^{k+l})$ entries, whereas an IO pattern stores a fixed set of signed index tuples whose size is independent of $n$, giving $O(1)$ storage in $n$.

For computational complexity, applying the map through its IO pattern costs $O(n^{l+1})$, against $O(n^{k+l})$ for the dense matrix–vector product: indeed, there are $O(n^l)$ output indices and, by Theorem 4.14, a nonzero basis map has at most one $[1, 0]$ block, hence there is a sum over at most one free index by Definition 5.1.

## J Scaling Benchmark for IO Patterns

We empirically compare the time and memory that is used by IO patterns against the dense realisation of the same map. The benchmark applies a $2 \to 2$ permutation equivariant map between antisymmetric tensors on an NVIDIA GeForce RTX 3080 with 10 GB of memory, in single precision, with a batch size of 4. Each IO time is the mean over 50 applications, preceded by per-shape warm-up iterations so that the comparison across values of $n$ is fair. "IO mem" is the peak memory that is used by our implementation, which is dominated by the activations themselves. "Matrix mem" is the measured peak memory that is required to allocate the corresponding $n^2 \times n^2$ weight matrix, which does not depend on the batch size.

| $n$ | IO time (ms) | IO mem (MB) | Matrix mem (MB) |
|---|---|---|---|
| 10 | 0.14 | 0.01 | 0.04 |
| 30 | 0.14 | 0.09 | 3.24 |
| 100 | 0.14 | 0.96 | 400.56 |
| 200 | 0.14 | 3.84 | 6400.51 |
| 210 | 0.14 | 4.24 | 7779.24 |
| 220 | 0.19 | 4.65 | 9370.24 |
| 230 | 0.16 | 5.09 | — |
| 240 | 0.17 | 5.53 | — |
| 300 | 0.14 | 8.65 | — |
| 1000 | 0.44 | 100.68 | — |
| 2000 | 1.69 | 390.10 | — |
| 5000 | 10.40 | 2403.42 | — |
| 7500 | 23.35 | 5400.12 | — |

Table 3: Time and memory that is needed to apply a $2 \to 2$ permutation equivariant map between antisymmetric tensors using an IO pattern versus the measured peak memory that is required to allocate the corresponding dense weight matrix. Entries marked with "—" exceeded the available GPU memory.

Because the IO pattern is executed without allocating any temporary tensor that is larger than the input and output tensors, it applies the map in $O(n^2)$ working memory, against the $O(n^4)$ memory that is needed to represent the map densely. Applying the IO patterns took between $0.14$ ms and $23.4$ ms across this range. The timings are flat up to roughly $n = 300$ because at those sizes the cost is dominated by the overhead of launching kernels rather than by the computation itself. The timings only begin to reflect the size of $n$ once the work is large enough to saturate the GPU. The memory used by the IO pattern grows at the predicted rate: between $n = 1000$ and $n = 2000$ it increases by a factor of $3.9$, between $n = 2000$ and $n = 5000$ by a factor of $6.2$, and between $n = 5000$ and $n = 7500$ by a factor of $2.25$, against the values of $4$, $6.25$ and $2.25$ that the $O(n^2)$ scaling predicts. The dense $n^2 \times n^2$ weight matrix exhausted the available GPU memory at $n = 230$, whereas the IO pattern only required $5$ MB for this value of $n$. At $n = 220$, the last size that fit on the GPU, the dense matrix already required $9.4$ GB against $4.7$ MB for the IO pattern. At $n = 7500$ the dense matrix would require roughly $12.7$ PB ($7500^4$ entries in single precision), against $5.4$ GB for the

IO pattern. These measurements confirm the memory scaling that is predicted by the complexity analysis of Section I.

## K LIMITATIONS

This work is primarily theoretical, providing a complete characterisation of linear permutation equivariant functions between antisymmetric tensors. Although the empirical evaluation is illustrative rather than exhaustive, these results may inform future developments in neural network architectures for data having antisymmetric tensor structure.

## L IMPLEMENTATION DETAILS

We describe the architecture and training details that were used for the experiments. In implementing the linear permutation equivariant functions on antisymmetric tensors, we used batch vectorised operations that correspond naturally to the IO patterns for each of the basis transformations. The models were implemented in PyTorch 2.2.2 and trained on a remote server with an NVIDIA GeForce RTX 3080 GPU. All of the results are fully reproducible using fixed random seeds.

**Data.** We used the Wiki-RfA dataset (West et al., 2014), where nodes are Wikipedia members and directed edges are votes, which are positive for support and negative for opposition. The dataset contains $198{,}275$ votes over $3{,}494$ labelled candidates, of whom $1{,}683$ ($48\%$) were promoted. We removed neutral votes to retain only directional interactions. For each candidate we built a $10$ node subgraph consisting of the candidate together with nine of its neighbours in the underlying undirected graph, drawn uniformly at random. If a candidate had fewer than nine neighbours, then the remaining nodes were sampled uniformly from the graph. We took the induced antisymmetric adjacency matrix as the input, and the candidate's actual adminship outcome as the label. It was possible for neighbourhoods of distinct candidates to overlap, but because the labels are attached only to the central candidate, no label information crossed a split.

**Architectures.** Our single-channel model consisted of six layers of permutation equivariant maps between antisymmetric tensors, combining transformations between tensor orders of type $2 \to 1$, $2 \to 2$, $1 \to 2$ and $1 \to 1$, interleaved with $\tanh$ activations and with residual connections at the order-two stages, followed by an invariant pooled readout. Recall from Remark 4.19 that we can use a $\tanh$ activation because it is odd, hence it preserves the antisymmetric structure. The pooled readout applied five permutation invariant statistics to the order-one output and passed them through a two-layer network of width $16$. The four-channel model used four copies of that stack, followed by a linear layer for channel mixing and a linear layer for the readout. The general equivariant models were identical, and used the same readout, except that every one of our maps was replaced by the corresponding general permutation equivariant map of Maron et al. (2019) so that only the family of maps differed. The signed message-passing GNN had three layers of width $32$ and aggregated separately over positive and negative edges. The MLP operated on the flattened adjacency matrix. The resulting parameter counts were $122$ (antisymmetric single-channel), $77$ (antisymmetric four-channel), $147$ and $177$ for the corresponding general equivariant models, and $9{,}697$ and $10{,}689$ for the signed GNN and the MLP, respectively. The difference between the single- and four-channel totals is due to the readout: in the single-channel models the readout accounted for $113$ of the parameters, leaving a backbone of $9$, whereas the four-channel models used a linear readout of $21$ parameters together with $20$ for channel mixing and four copies of the backbone.

**Training and evaluation.** All models were trained full-batch with Adam using a binary cross-entropy loss, so one epoch corresponded to a single gradient step. We used three independent $60/20/20$ train/validation/test splits and five seeds per split. For each model and each learning rate in $\{0.3, 0.1, 0.01, 0.001\}$ we trained five models for $2{,}000$ epochs and averaged their best validation losses, selecting the learning rate with the lowest average. Our models selected a learning rate of $0.1$ on every split, the signed GNN selected $0.01$, and the MLP selected $0.001$; the general equivariant models selected $0.01$ on all but one split, where the single-channel model selected $0.1$. Each model was then evaluated on the test set at the epoch at which it attained its lowest validation loss. The MLP reached that minimum after roughly $37$ epochs on average and overfitted thereafter, whereas

our models reached it after roughly $1{,}380$ (single-channel) and $1{,}280$ (four-channel) epochs and the general equivariant models after roughly $1{,}194$ (single-channel) and $1{,}506$ (four-channel) epochs. Every model therefore reached its minimum inside the epoch budget, so none was undertrained. We report accuracy and AUC as mean $\pm$ one standard deviation across the three splits, where each split's score is the average over its five seeds. The test set was used only for this final measurement and was never used for model or hyperparameter selection.

## M Dataset and Licensing Details

We have used the Wikipedia Requests for Adminship (Wiki-RfA) dataset which is accessed via the SNAP repository (`https://snap.stanford.edu/data/wiki-RfA.html`). The dataset is derived from Wikipedia administrative voting data. We have used the version provided by SNAP without modification. The SNAP repository is distributed under the BSD licence, which permits free use for academic and commercial purposes with attribution.