A. Equivariance Proof

In this section we prove that our model is translation equivariant on $\mathbf{x}$ for any translation vector $g \in \mathbb{R}^n$ and it is rotation and reflection equivariant on $\mathbf{x}$ for any orthogonal matrix $Q \in \mathbb{R}^{n \times n}$. More formally, we will prove the model satisfies:

$$Q\mathbf{x}^{l+1} + g, \mathbf{h}^{l+1} = \mathrm{EGCL}(Q\mathbf{x}^l + g, \mathbf{h}^l)$$

We will analyze how a translation and rotation of the input coordinates propagates through our model. We start assuming $\mathbf{h}^0$ is invariant to $\mathrm{E}(n)$ transformations on $\mathbf{x}$, in other words, we do not encode any information about the absolute position or orientation of $\mathbf{x}^0$ into $\mathbf{h}^0$. Then, the output $\mathbf{m}_{ij}$ of Equation 3 will be invariant too since the distance between two particles is invariant to translations $\|\mathbf{x}_i^l + g - [\mathbf{x}_j^l + g]\|^2 = \|\mathbf{x}_i^l - \mathbf{x}_j^l\|^2$, and it is invariant to rotations and reflections $\|Q\mathbf{x}_i^l - Q\mathbf{x}_j^l\|^2 = (\mathbf{x}_i^l - \mathbf{x}_j^l)^\top Q^\top Q (\mathbf{x}_i^l - \mathbf{x}_j^l) = (\mathbf{x}_i^l - \mathbf{x}_j^l)^\top \mathbf{I} (\mathbf{x}_i^l - \mathbf{x}_j^l) = \|\mathbf{x}_i^l - \mathbf{x}_j^l\|^2$ such that the edge operation becomes invariant:

$$\mathbf{m}_{i,j} = \phi_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, \left\|Q\mathbf{x}_i^l + g - [Q\mathbf{x}_j^l + g]\right\|^2, a_{ij}\right) = \phi_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, \left\|\mathbf{x}_i^l - \mathbf{x}_j^l\right\|^2, a_{ij}\right)$$

The second equation of our model (eq. 4) that updates the coordinates $\mathbf{x}$ is $\mathrm{E}(n)$ equivariant. Following, we prove its equivariance by showing that an $\mathrm{E}(n)$ transformation of the input leads to the same transformation of the output. Notice $\mathbf{m}_{ij}$ is already invariant as proven above. We want to show:

$$Q\mathbf{x}_i^{l+1} + g = Q\mathbf{x}_i^l + g + C\sum_{j \neq i}\left(Q\mathbf{x}_i^l + g - [Q\mathbf{x}_j^l + g]\right)\phi_x\left(\mathbf{m}_{i,j}\right)$$

Derivation.

$$\begin{aligned} Q\mathbf{x}_i^l + g + C\sum_{j \neq i}\left(Q\mathbf{x}_i^l + g - Q\mathbf{x}_j^l - g\right)\phi_x\left(\mathbf{m}_{i,j}\right) &= Q\mathbf{x}_i^l + g + QC\sum_{j \neq i}\left(\mathbf{x}_i^l - \mathbf{x}_j^l\right)\phi_x\left(\mathbf{m}_{i,j}\right) \\ &= Q\left(\mathbf{x}_i^l + C\sum_{j \neq i}\left(\mathbf{x}_i^l - \mathbf{x}_j^l\right)\phi_x\left(\mathbf{m}_{i,j}\right)\right) + g \\ &= Q\mathbf{x}_i^{l+1} + g \end{aligned}$$

Therefore, we have proven that rotating and translating $\mathbf{x}^l$ results in the same rotation and translation on $\mathbf{x}^{l+1}$ at the output of Equation 4.

Furthermore equations 5 and 6 only depend on $\mathbf{m}_{ij}$ and $\mathbf{h}^l$ which as saw at the beginning of this proof, are $\mathrm{E}(n)$ invariant, therefore the output of Equation 6 $\mathbf{h}^{l+1}$ will be invariant too. Thus concluding that a transformation $Q\mathbf{x}^l + g$ on $\mathbf{x}^l$ will result in the same transformation on $\mathbf{x}^{l+1}$ while $\mathbf{h}^{l+1}$ will remain invariant to it such that $Q\mathbf{x}^{l+1} + g, \mathbf{h}^{l+1} = \mathrm{EGCL}(Q\mathbf{x}^l + g, \mathbf{h}^l)$ is satisfied.

B. Re-formulation for velocity type inputs

In this section we write down the EGNN transformation layer $\mathbf{h}^{l+1}, \mathbf{x}^{l+1}, \mathbf{v}^{l+1} = \mathrm{EGCL}[\mathbf{h}^l, \mathbf{x}^l, \mathbf{v}^l, \mathcal{E}]$ that can take in velocity input and output channels. We also prove it remains $\mathrm{E}(n)$ equivariant.

$$\begin{aligned} \mathbf{m}_{ij} &= \phi_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, \left\|\mathbf{x}_i^l - \mathbf{x}_j^l\right\|^2, a_{ij}\right) \\ \mathbf{v}_i^{l+1} &= \phi_v\left(\mathbf{h}_i^l\right)\mathbf{v}_i^l + C\sum_{j \neq i}\left(\mathbf{x}_i^l - \mathbf{x}_j^l\right)\phi_x\left(\mathbf{m}_{ij}\right) \\ \mathbf{x}_i^{l+1} &= \mathbf{x}_i^l + \mathbf{v}_i^{l+1} \\ \mathbf{m}_i &= \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij} \\ \mathbf{h}_i^{l+1} &= \phi_h\left(\mathbf{h}_i^l, \mathbf{m}_i\right) \end{aligned}$$

B.1. Equivariance proof for velocity type inputs

In this subsection we prove that the velocity types input formulation of our model is also $\mathrm{E}(n)$ equivariant on $\mathbf{x}$. More formally, for any translation vector $g \in \mathbb{R}^n$ and for any orthogonal matrix $Q \in \mathbb{R}^{n \times n}$, the model should satisfy:

$$\mathbf{h}^{l+1}, Q\mathbf{x}^{l+1} + g, Q\mathbf{v}^{l+1} = \mathrm{EGCL}[\mathbf{h}^l, Q\mathbf{x}^l + g, Q\mathbf{v}^l, \mathcal{E}]$$

In Appendix A we already proved the equivariance of our EGNN (Section 3) when not including vector type inputs. In its velocity type inputs variant we only replaced its coordinate updates (eq. 4) by Equation 7 that includes velocity. Since this is the only modification we will only prove that Equation 7 re-written below is equivariant.

$$\begin{aligned}
\mathbf{v}_i^{l+1} &= \phi_v\left(\mathbf{h}_i^l\right)\mathbf{v}_i^l + C\sum_{j\neq i}\left(\mathbf{x}_i^l - \mathbf{x}_j^l\right)\phi_x\left(\mathbf{m}_{ij}\right)\\
\mathbf{x}_i^{l+1} &= \mathbf{x}_i^l + \mathbf{v}_i^{l+1}
\end{aligned}$$

First, we prove the first line preserves equivariance, that is we want to show:

$$Q\mathbf{v}_i^{l+1} = \phi_v\left(\mathbf{h}_i^l\right)Q\mathbf{v}_i^l + C\sum_{j\neq i}\left(Q\mathbf{x}_i^l + g - [Q\mathbf{x}_j^l + g]\right)\phi_x\left(\mathbf{m}_{ij}\right)$$

Derivation.

$$\begin{aligned}
\phi_v\left(\mathbf{h}_i^l\right)Q\mathbf{v}_i^l + C\sum_{j\neq i}\left(Q\mathbf{x}_i^l + g - [Q\mathbf{x}_j^l + g]\right)\phi_x\left(\mathbf{m}_{ij}\right) &= Q\phi_v\left(\mathbf{h}_i^l\right)\mathbf{v}_i^l + QC\sum_{j\neq i}\left(\mathbf{x}_i^l - \mathbf{x}_j^l\right)\phi_x\left(\mathbf{m}_{ij}\right) && (10)\\
&= Q\left(\phi_v\left(\mathbf{h}_i^l\right)\mathbf{v}_i^l + C\sum_{j\neq i}\left(\mathbf{x}_i^l - \mathbf{x}_j^l\right)\phi_x\left(\mathbf{m}_{ij}\right)\right) && (11)\\
&= Q\mathbf{v}_i^{l+1} && (12)
\end{aligned}$$

Finally, it is straightforward to show the second equation is also equivariant, that is we want to show $Q\mathbf{x}_i^{l+1} + g = Q\mathbf{x}_i^l + g + Q\mathbf{v}_i^{l+1}$

Derivation.

$$\begin{aligned}
Q\mathbf{x}_i^l + g + Q\mathbf{v}_i^{l+1} &= Q\left(\mathbf{x}_i^l + \mathbf{v}_i^{l+1}\right) + g\\
&= Q\mathbf{x}_i^{l+1} + g
\end{aligned}$$

Concluding we showed that an $\mathrm{E}(n)$ transformation on the input set of points results in the same transformation on the output set of points such that $\mathbf{h}^{l+1}, Q\mathbf{x}^{l+1} + g, Q\mathbf{v}^{l+1} = \mathrm{EGCL}[\mathbf{h}^l, Q\mathbf{x}^l + g, Q\mathbf{v}^l, \mathcal{E}]$ is satisfied.

C. Implementation details

In this Appendix section we describe the implementation details of the experiments. First, we describe those parts of our model that are the same across all experiments. Our EGNN model from Section 3 contains the following three main learnable functions.

- **The edge function** ϕ_e (eq. 3) is a two layers MLP with two Swish non-linearities: Input $\rightarrow$ {LinearLayer() $\rightarrow$ Swish() $\rightarrow$ LinearLayer() $\rightarrow$ Swish() } $\rightarrow$ Output.
- **The coordinate function** ϕ_x (eq. 4) consists of a two layers MLP with one non-linearity: $\mathbf{m}_{ij}$ $\rightarrow$ {LinearLayer() $\rightarrow$ Swish() $\rightarrow$ LinearLayer() } $\rightarrow$ Output
- **The node function** ϕ_h (eq. 6) consists of a two layers MLP with one non-linearity and a residual connection: $[\mathbf{h}_i^l, \mathbf{m}_i]$ $\rightarrow$ {LinearLayer() $\rightarrow$ Swish() $\rightarrow$ LinearLayer() $\rightarrow$ Addition($\mathbf{h}_i^l$) } $\rightarrow$ $\mathbf{h}_i^{l+1}$

These functions are used in our EGNN across all experiments. Notice the GNN (eq. 2) also contains and edge operation and a node operation ϕ_e and ϕ_h respectively. We use the same functions described above for both the GNN and the EGNN such that comparisons are as fair as possible.

C.1. Implementation details for Dynamical Systems

Dataset

In the dynamical systems experiment we used a modification of the Charged Particle's N-body (N=5) system from (Kipf et al., 2018). Similarly to (Fuchs et al., 2020), we extended it from 2 to 3 dimensions customizing the original code from (https://github.com/ethanfetaya/NRI) and we removed the virtual boxes that bound the particle's positions. The sampled dataset consists of 3.000 training trajectories, 2.000 for validation and 2.000 for testing. Each trajectory has a duration of 1.000 timesteps. To move away from the transient phase, we actually generated trajectories of 5.000 time steps and sliced them from timestep 3.000 to timestep 4.000 (1.000 time steps into the future) such that the initial conditions are more realistic than the Gaussian Noise initialization from which they are initialized.

In our second experiment, we sweep from 100 to 50.000 training samples, for this we just created a new training partition following the same procedure as before but now generating 50.000 trajectories instead. The validation and test partition remain the same from last experiment.

Models

All models are composed of 4 layers, the details for each model are the following.

- **EGNN**: For the EGNN we use its variation that considers vector type inputs from Section 3.2. This variation adds the function ϕ_v to the model which is composed of two linear layers with one non-linearity: Input $\rightarrow$ {LinearLayer() $\rightarrow$ Swish() $\rightarrow$ LinearLayer() } $\rightarrow$ Output. Functions ϕ_e, ϕ_x and ϕ_h that define our EGNN are the same than for all experiments and are described at the beginning of this Appendix C.
- **GNN**: The GNN is also composed of 4 layers, its learnable functions edge operation ϕ_e and node operation ϕ_h from Equation 2 are exactly the same as ϕ_e and ϕ_h from our EGNN introduced in Appendix C. We chose the same functions for both models to ensure a fair comparison. In the GNN case, the initial position $\mathbf{p}^0$ and velocity $\mathbf{v}^0$ from the particles is passed through a linear layer and inputted into the GNN first layer $\mathbf{h}^0$. The particle's charges are inputted as edge attributes $a_{ij} = c_i c_j$. The output of the GNN $\mathbf{h}^L$ is passed through a two layers MLP that maps it to the estimated position.
- **Radial Field**: The Radial Field algorithm is described in the Related Work 4, its only parameters are contained in its edge operation $\phi_{\mathrm{rf}}()$ which in our case is a two layers MLP with two non linearities Input $\rightarrow$ {LinearLayer() $\rightarrow$ Swish() $\rightarrow$ LinearLayer() $\rightarrow$ Tanh } $\rightarrow$ Output. Notice we introduced a Tanh at the end of the MLP which fixes some instability issues that were causing this model to diverge in the dynamical system experiment. We also augmented the Radial Field algorithm with the vector type inputs modifications introduced in Section 3.2. In addition to the norms between pairs of points, $\phi_{\mathrm{rf}}()$ also takes as input the particle charges $c_i c_j$.
- **Tensor Field Network:** We used the Pytorch implementation from https://github.com/FabianFuchsML/se3-transformer-public. We swept over different hyper paramters, degree $\in \{2, 3, 4\}$, number of features $\in \{12, 24, 32, 64, 128\}$. We got the best performance in our dataset for degree 2 and number of features 32. We used the Relu activation layer instead of the Swish for this model since it provided better performance.
- **SE(3) Transformers:** For the SE(3)-Transformer we used code from https://github.com/FabianFuchsML/se3-transformer-public. Notice this implementation has only been validated in the QM9 dataset but it is the only available implementation of this model. We swept over different hyperparamters degree $\in \{1, 2, 3, 4\}$, number of features $\in$ 16, 32, 64 and divergence $\in \{1, 2\}$, along with the learning rate. We obtained the best performance for degree 3, number of features 64 and divergence 1. As in Tensor Field Networks we obtained better results by using the Relu activation layer instead of the Swish.

Other implementation details

In Table 2 all models were trained for 10.000 epochs, batch size 100, Adam optimizer, the learning rate was fixed and independently chosen for each model. All models are 4 layers deep and the number of training samples was set to 3.000.

C.2. Implementation details for Graph Autoneoders

Dataset

In this experiment we worked with Community Small (You et al., 2018) and Erdos&Renyi (Bollobás & Béla, 2001) generated datasets.

- Community Small: We used the original code from (You et al., 2018) (https://github.com/JiaxuanYou/graph-generation) to generate a Community Small dataset. We sampled 5.000 training graphs, 500 for validation and 500 for testing.
- Erdos&Renyi is one of the most famous graph generative algorithms. We used the "gnp_random_graph(M, p)" function from (https://networkx.org/) that generates random graphs when povided with the number of nodes M and the edge probability p following the Erdos&Renyi model. Again we generated 5.000 graphs for training, 500 for validation and 500 for testing. We set the edge probability (or sparsity value) to $p = 0.25$ and the number of nodes M ranging from 7 to 16 deterministically uniformly distributed. Notice that edges are generated stochastically with probability p, therefore, there is a chance that some nodes are left disconnected from the graph, "gnp_random_graph(M, p)" function discards these disconnected nodes such that even if we generate graphs setting parameters to $7 \leq M \leq 16$ and $p = 0.25$ the generated graphs may have less number of nodes.

Finally, in the graph autoencoding experiment we also overfitted in a small partition of 100 samples (Figure 5) for the Erdos&Renyi graphs described above. We reported results for different p values ranging from 0.1 to 0.9. For each p value we generated a partition of 100 graphs with initial number of nodes between $7 \leq M \leq 16$ using the Erdos&Renyi generative model.

Models

All models consist of 4 layers, 64 features for the hidden layers and the Swish activation function as a non linearity. The EGNN is defined as explained in Section 3 without any additional modules (i.e. no velocity type features or inferring edges). The functions ϕ_e, ϕ_x and ϕ_h are defined at the beginning of this Appendix C. The GNN (eq. 2) mimics the EGNN in terms that it uses the same ϕ_h and ϕ_e than the EGNN for its edge and node updates. The Noise-GNN is exactly the same as the GNN but inputting noise into the $\mathbf{h}_0$ features. Finally the Radial Field was defined in the Related Related work Section 4 which edge's operation ϕ_{rf} consists of a two layers MLP: Input $\rightarrow$ { Linear() $\rightarrow$ Swish() $\rightarrow$ Linear() } $\rightarrow$ Output.

Other implementation details

All experiments have been trained with learning rate 10^{-4}, batch size 1, Adam optimizer, weight decay 10^{-16}, 100 training epochs for the 5.000 samples sized datasets performing early stopping for the minimum Binary Cross Entropy loss in the validation partition. The overfitting experiments were trained for 10.000 epochs on the 100 samples subsets.

C.3. Implementation details for QM9

For QM9 (Ramakrishnan et al., 2014) we used the dataset partitions from (Anderson et al., 2019). We imported the dataloader from his code repository (https://github.com/risilab/cormorant) which includes his data-preprocessing. Additionally all properties have been normalized by substracting the mean and dividing by the Mean Absolute Deviation.

Our EGNN consists of 7 layers. Functions ϕ_e and ϕ_h are defined at the beginning of this Appendix C. Additionally, we use the module ϕ_{inf} presented in Section 3.3 that infers the edges . This function ϕ_{inf} is defined as a linear layer followed by a sigmoid: Input $\rightarrow$ {Linear() $\rightarrow$ sigmoid()} $\rightarrow$ Output. Finally, the output of our EGNN $\mathbf{h}^L$ is forwarded through a two layers MLP that acts node-wise, a sum pooling operation and another two layers MLP that maps the averaged embedding to the predicted property value, more formally: $\mathbf{h}^L \rightarrow$ {Linear() $\rightarrow$ Swish() $\rightarrow$ Linear() $\rightarrow$ Sum-Pooling() $\rightarrow$ Linear() $\rightarrow$ Swish() $\rightarrow$ Linear} $\rightarrow$ Property. The number of hidden features for all model hidden layers is 128.

We trained each property individually for a total of 1.000 epochs, we used Adam optimizer, batch size 96, weight decay 10^{-16}, and cosine decay for the learning rate starting at at a lr=$5 \cdot 10^{-4}$ except for the Homo, Lumo and Gap properties where its initial value was set to 10^{-3}.

D. Further experiments

D.1. Graph Autoencoder

In this section we present an extension of the Graph Autoencoder experiment 5.2. In Table 4 we report the approximation error of the reconstructed graphs as the embedding dimensionality n is reduced $n \in \{4, 6, 8\}$ in the Community Small

	Community Small						Erdos&Renyi					
	n=4		n=6		n=8		n=4		n=6		n=8	
	% Err.	F1	% Err.	F1	% Err.	F1	% Err.	F1	% Err.	F1	% Err.	F1
GNN	**1.45**	**0.977**	1.29	0.9800	1.29	0.980	7.92	0.844	5.22	0.894	4.62	0.907
Noise-GNN	1.94	0.970	0.44	0.9931	0.44	0.993	3.80	0.925	2.66	0.947	1.25	0.975
EGNN	2.19	0.966	**0.42**	**0.9934**	**0.06**	**0.999**	**3.09**	**0.939**	**0.58**	**0.988**	**0.11**	**0.998**

Table 4. Analysis of the % of wrong edges and F1 score for different n embedding sizes $\{2, 4, 8\}$ for the GNN, Noise-GNN and EGNN in Community Small and Erdos&Renyi datasets.

and Erdos&Renyi datasets for the GNN, Noise-GNN and EGNN models. For small embedding sizes ($n = 4$) all methods perform poorly, but as the embedding size grows our EGNN significantly outperforms the others.

E. Sometimes invariant features are all you need.

Perhaps surprisingly we find our EGNNs outperform other equivariant networks that consider higher-order representations. In this section we prove that when only positional information is given (i.e. no velocity-type features) then the geometry is completely defined by the invariant distance norms in-between points, without loss of relevant information. As a consequence, it is not necessary to consider higher-order representation types of the relative distances, not even the relative differences as vectors. To be precise, note that these invariant features still need to be *permutation* equivariant, they are only $\mathrm{E}(n)$ invariant.

To be specific, we want to show that for a collection of points $\{\mathbf{x}_i\}_{i=1}^{M}$ the norm of in-between distances $\ell_2(\mathbf{x}_i, \mathbf{x}_j)$ are a *unique* identifier of the geometry, where collections separated by an $\mathrm{E}(n)$ transformations are considered to be identical. We want to show *invariance* of the norms under $\mathrm{E}(n)$ transformations and *uniqueness*: two point collections are identical (up to $\mathrm{E}(n)$ transform) when they have the same distance norms.

Invariance. Let $\{\mathbf{x}_i\}$ be a collection of M points where $\mathbf{x}_i \in \mathbb{R}^n$ and the ℓ_2 distances are $\ell_2(\mathbf{x}_i, \mathbf{x}_j)$. We want to show that all $\ell_2(\mathbf{x}_i, \mathbf{x}_j)$ are unaffected by $\mathrm{E}(n)$ transformations.

Proof. Consider an arbitrary $\mathrm{E}(n)$ transformation $\mathbb{R}^n \to \mathbb{R}^n : \mathbf{x} \mapsto Q\mathbf{x} + t$ where Q is orthogonal and $t \in \mathbb{R}^n$ is a translation. Then for all i, j:

$$\begin{aligned}\ell_2(Q\mathbf{x}_i + t, Q\mathbf{x}_j + t) &= \sqrt{(Q\mathbf{x}_i + t - [Q\mathbf{x}_j + t])^T (Q\mathbf{x}_i + t - [Q\mathbf{x}_j + t])} = \sqrt{(Q\mathbf{x}_i - Q\mathbf{x}_j)^T (Q\mathbf{x}_i - Q\mathbf{x}_j)} \\ &= \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T Q^T Q (\mathbf{x}_i - \mathbf{x}_j)} = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j)} = \ell_2(\mathbf{x}_i, \mathbf{x}_j)\end{aligned}$$

This proves that the ℓ_2 distances are invariant under $\mathrm{E}(n)$ transforms.

Uniqueness. Let $\{\mathbf{x}_i\}$ and $\{\mathbf{y}_i\}$ be two collection of M points each where all in-between distance norms are identical, meaning $\ell_2(\mathbf{x}_i, \mathbf{x}_j) = \ell_2(\mathbf{y}_i, \mathbf{y}_j)$. We want to show that $\mathbf{x}_i = Q\mathbf{y}_i + t$ for some orthogonal Q and translation t, for all i.

Proof. Subtract $\mathbf{x}_0$ from all $\{\mathbf{x}_i\}$ and $\mathbf{y}_0$ from all $\{\mathbf{y}_i\}$, so $\tilde{\mathbf{x}}_i = \mathbf{x}_i - \mathbf{x}_0$ and $\tilde{\mathbf{y}}_i = \mathbf{y}_i - \mathbf{y}_0$. As proven above, since translation is an $\mathrm{E}(n)$ transformation the distance norms are unaffected and:

$$\ell_2(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j) = \ell_2(\mathbf{x}_i, \mathbf{x}_j) = \ell_2(\mathbf{y}_i, \mathbf{y}_j) = \ell_2(\tilde{\mathbf{y}}_i, \tilde{\mathbf{y}}_j).$$

So without loss of generality, we may assume that $\mathbf{x}_0 = \mathbf{y}_0 = \mathbf{0}$. As a direct consequence $||\mathbf{x}_i||_2 = ||\mathbf{y}_i||_2$. Now writing out the square:

$$\mathbf{x}_i^T\mathbf{x}_i - 2\mathbf{x}_i^T\mathbf{x}_j + \mathbf{x}_j^T\mathbf{x}_j = ||\mathbf{x}_i - \mathbf{x}_j||_2^2 = ||\mathbf{y}_i - \mathbf{y}_j||_2^2 = \mathbf{y}_i^T\mathbf{y}_i - 2\mathbf{y}_i^T\mathbf{y}_j + \mathbf{y}_j^T\mathbf{y}_j$$

And since $||\mathbf{x}_i||_2 = ||\mathbf{y}_i||_2$, it follows that $\mathbf{x}_i^T\mathbf{x}_j = \mathbf{y}_i^T\mathbf{y}_j$ or equivalently written as dot product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \langle \mathbf{y}_i, \mathbf{y}_j \rangle$. Notice that this already shows that angles between pairs of points are the same.

At this moment, it might already be intuïtive that the collections of points are indeed identical. To finalize the proof formally we will construct a linear map A for which we will show that (1) it maps every $\mathbf{x}_i$ to $\mathbf{y}_i$ and (2) that it is orthogonal. First note that from the angle equality it follows immediately that for every linear combination:

$$||\sum_i c_i\mathbf{x}_i||_2 = ||\sum_i c_i\mathbf{y}_i||_2 \quad (*).$$

Let V_x be the linear span of $\{\mathbf{x}_i\}$ (so V_x is the linear subspace of all linear combinations of $\{\mathbf{x}_i\}$). Let $\{\mathbf{x}_{i_j}\}_{j=1}^{d}$ be a basis of V_x, where $d \leq n$. Recall that one can define a linear map by choosing a basis, and then define for each basis vector where it maps to. Define a linear map A from V_x to V_y by the transformation from the basis $\mathbf{x}_{i_j}$ to $\mathbf{y}_{i_j}$ for $j = 1, ..., d$. Now pick any point $\mathbf{x}_i$ and write it in its basis $\mathbf{x}_i = \sum_j c_j\mathbf{x}_{i_j} \in V_x$. We want to show $A\mathbf{x}_i = \mathbf{y}_i$ or alternatively $||\mathbf{y}_i - A\mathbf{x}_i||_2 = 0$. Note that $A\mathbf{x}_i = A\sum_j c_j\mathbf{x}_{i_j} = \sum_j c_jA\mathbf{x}_{i_j} = \sum_j c_j\mathbf{y}_{i_j}$. Then:

$$\begin{aligned}||\mathbf{y}_i - \sum_j c_j\mathbf{y}_{i_j}||_2^2 &= \langle \mathbf{y}_i, \mathbf{y}_i \rangle - 2\langle \mathbf{y}_i, \sum_j c_i\mathbf{y}_{i_j} \rangle + \langle \sum_j c_i\mathbf{y}_{i_j}, \sum_j c_i\mathbf{y}_{i_j} \rangle \\ &\overset{(*)}{=} \langle \mathbf{x}_i, \mathbf{x}_i \rangle - 2\langle \mathbf{x}_i, \sum_j c_i\mathbf{x}_{i_j} \rangle + \langle \sum_j c_i\mathbf{x}_{i_j}, \sum_j c_i\mathbf{x}_{i_j} \rangle = \langle \mathbf{x}_i, \mathbf{x}_i \rangle - 2\langle \mathbf{x}_i, \mathbf{x}_i \rangle + \langle \mathbf{x}_i, \mathbf{x}_i \rangle = 0.\end{aligned}$$

Thus showing that $A\mathbf{x}_i = \mathbf{y}_i$ for all $i = 1, \ldots, M$, proving (1). Finally we want to show that A is orthogonal, when restricted to V_x. This follows since:

$$\langle A\mathbf{x}_{i_j}, A\mathbf{x}_{i_j} \rangle = \langle \mathbf{y}_{i_j}, \mathbf{y}_{i_j} \rangle = \langle \mathbf{x}_{i_j}, \mathbf{x}_{i_j} \rangle$$

for the basis elements $\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_d}$. This implies that A is orthogonal (at least when restricted to V_x). Finally A can be extended via an orthogonal complement of V_x to the whole space. This concludes the proof for (2) and shows that A is indeed orthogonal.