SE(3)-Equivariant Diffusion Graph Nets: Synthesizing Flow Fields by Denoising Invariant Latents on Graphs

Anonymous Authors[1]

Abstract

We introduce SE(3)-equivariant diffusion graph nets (SE3-DGNs) for generating physical fields on graphs. SE3-DGNs integrate a SE(3)-equivariant variational graph autoencoder (VGAE) and a diffusion graph net (DGN) to produce high-quality, SE(3)-equivariant flow fields. The S-VGAE learns an invariant latent space that abstracts directional information, and the DGN is trained on this latent space. Equivariant inference requires minimal additional computation, needing only a single evaluation of the edge encoder and node decoder. Demonstrated on laminar vortex-shedding under out-of-distribution Reynolds numbers and fluid domain parameters, SE3-DGNs showed superior sample quality compared to baseline DGNs and latent DGNs. SE3-DGNs can efficiently generate fully-developed flow fields to use as initial conditions for numerical solvers, bypassing the need for simulating transition regimes.

1. Introduction

1.1. Background & Related Work

Fluid dynamics simulation, typically by numerically solving partial differential equations (PDEs), is essential in many scientific and engineering fields (Karniadakis & Sherwin, 2013). However, the high computational cost can limit its practical application and exploration of large parameter spaces. Recently, deep-learning approaches have shown promise in creating surrogate models for fluid dynamics simulations (Lino et al., 2023).

While convolutional neural networks (CNNs) constrain solutions to regular Cartesian grids, graph neural networks (GNNs) have been used to learn simulations of fluids employing Lagrangian discretisations (Ummenhofer et al., 2019; Sanchez-Gonzalez et al., 2020; Prantl et al., 2022) and unstructured meshes (Pfaff et al., 2021). In this last case, the fluid domain is discretised into fixed nodes forming a graph, and the information is processed through learned message passing (MP) (Battaglia et al., 2018) across the graph, and possibly across multiple coarsenings of it (Lino et al., 2022; Fortunato et al., 2022). Unsteady simulations can be obtained by iteratively evaluating models with past predictions.

Most PDEs possess several symmetries, with translation invariance and rotation equivariance being among the most common (Pope & Pope, 2000). The general approach for achieving rotation equivariance is through training with data augmented by rotations (Brandstetter et al., 2022). However, strictly enforcing rotation equivariance is gaining importance for improving generalisation accuracy (Ling et al., 2016; Wang et al., 2021; Siddani et al., 2021; Lino et al., 2022; Gao et al., 2024). In GNNs, rotation equivariance is achieved either through feature engineering (Klicpera et al., 2020) or by using tensor-valued features and tensor products (Thomas et al., 2018; Weiler et al., 2018). While the latter approach is computationally intensive, it avoids adding extra graph elements like angles (Lino et al., 2022) and cells (Gao et al., 2024), and it works for 3D problems without modifications. To manage the high computational cost, low-order tensor features are typically used (Brandstetter et al., 2021), and models with equivariant layers only at the beginning and at the end have been proposed (Shankar et al., 2023). Inspired by this, we employ steerable features and steerable MP (Brandstetter et al., 2021) in an autoencoder and train the backbone model in the learnt invariant latent space.

Recently, denoising diffusion probabilistic models (DDPMs) (Ho et al., 2020) have been applied to turbulent flow (Shu et al., 2023), uncertainty estimation of ReynoldS-averaged simulations (Liu & Thuerey, 2024), and improving the stability of long-term simulations (Kohl et al., 2023; Lippe et al., 2024). More related to our work, Lienen et al. (2024) proposed using a DDPM to learn the manifold of all possible turbulent flow states without relying on any initial flow state. Nonetheless, all these studies considered CNN-based and non-equivariant DDPMs. Hoogeboom et al. (2022)

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

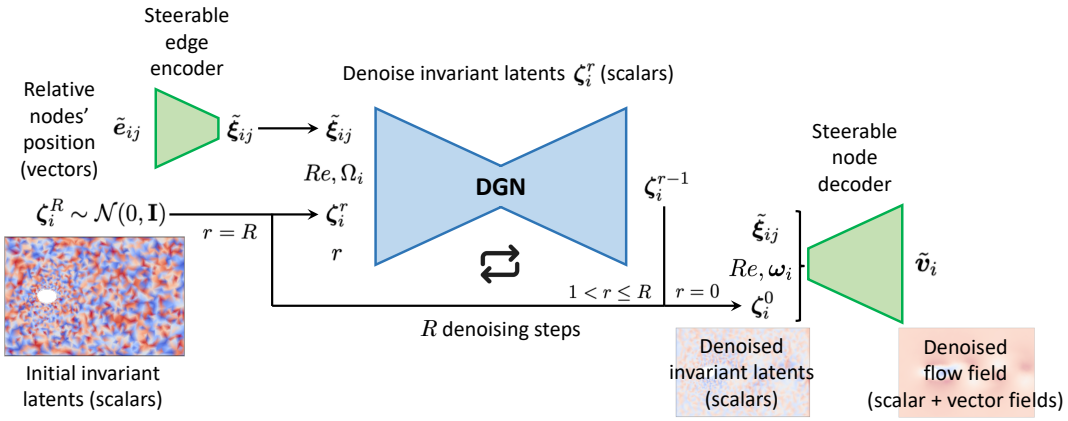

Figure 1. Sampling with a SE3-DGN from Gaussian noise in a SE(3)-invariant latent space. The relative position between adjacent nodes, $\tilde{\boldsymbol{e}}_{ij}$, is encoded into steerable latent edge-features, $\tilde{\boldsymbol{\xi}}_{ij}$. Scalar-valued node-features, $\boldsymbol{\zeta}_i^R$, are sampled from a isotropic Gaussian distribution and iteratively denoised by a Diffusion Graph Net (DGN). The DGN is also fed with $\tilde{\boldsymbol{\xi}}_{ij}$, the diffusion step r, the Reynolds number, Re, and one-hot vectors indicating the type of each node, $\boldsymbol{\omega}_i$. The denoised node-features, $\boldsymbol{\zeta}_i^0$, are decoded back to a physical space of steerable features, $\tilde{\boldsymbol{v}}_i$.

used a GNN-based rotation-equivariant DDPM to synthesise molecules by inferring atom types and coordinates, but their model misses directional information by using node distances instead of relative position vectors.

1.2. Contributions

In this work, we apply the DDPM framework to GNNs. Given the domain geometry and the Reynolds number (Re), these models generate flow fields by iteratively denoising node features sampled from a Gaussian distribution. Utilising graphs allows adjusting the resolution to increase node count near immersed bodies where gradients are larger. This approach enables the direct generation of unsteady flow snapshots without simulating the transition regime, preventing error accumulation and reducing runtimes.

We introduce a variational graph autoencoder (VGAE) (Kipf & Welling, 2016) that learns a latent space where node features are SE(3)-invariant. We propose training a GNN-based DDPM on this latent space. The trained model is SE(3)-equivariant, achieving rotation equivariance with minimal additional computational cost by evaluating an encoder and decoder only once versus multiple denoising steps of the DDPM. We demonstrate this by learning the distribution of pressure and velocity fields for laminar vortex-shedding behind 2D elliptical cylinders.

2. SE(3)-Equivariant Diffusion Graph Nets

We propose a SE(3)-equivariant DDPM for flow fields defined on directed graphs $\mathcal{G} := (\mathcal{V}, \mathcal{E})$. We refer to it as *SE(3)-Equivariant Diffusion Graph Net* (SE3-DGN). The set of nodes $\mathcal{V}$ discretises the fluid domain, and the set of edges $\mathcal{E}$ connects each node to its k-nearest neighbours. Each node i is assigned node features $\boldsymbol{v}_i$, and each edge (i, j) is assigned edge features $\boldsymbol{e}_{i,j}$. SE3-DGNs denoise $V = \{\boldsymbol{v}_i\}$ given $E = \{\boldsymbol{e}_{i,j}\}$ and conditioned on the geometry of the fluid domain and the Reynolds number.

2.1. Diffusion Graph Nets

Diffusion Models DDPMs can learn to synthesise a sample $\boldsymbol{z}^0$ from a data distribution $q(\boldsymbol{z}^0)$ by progressive denoising a sample $\boldsymbol{z}^R$ drawn from an isotropic Gaussian distribution (Song & Ermon, 2020; Ho et al., 2020). This denoising process mirrors the inverse of a fixed Markov chain of length R, known as the *forward process*, and consists of R denoising steps. The forward (or *diffusion*) process produces variables $\boldsymbol{z}^1$ through $\boldsymbol{z}^R$ by sequentially injecting them with Gaussian noise, $q(\boldsymbol{z}^r|\boldsymbol{z}^{r-1}) := \mathcal{N}(\boldsymbol{z}^r; \sqrt{1-\beta_t}\boldsymbol{z}^{r-1}, \beta_r \mathbf{I})$. It is possible to shortcut in the forward process and directly sample $\boldsymbol{z}^r$, at any diffusion step r, according to

$$\boldsymbol{z}^r = \sqrt{\bar{\alpha}_r}\boldsymbol{z}^0 + \sqrt{1-\bar{\alpha}_r}\boldsymbol{\epsilon} \tag{1}$$

where $\alpha_r := 1 - \beta_r$, $\bar{\alpha}_r := \prod_{s=1}^{r} \alpha_s$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$ (Ho et al., 2020). If R is large enough and the β-schedule is selected properly, the forward process leads to $\boldsymbol{z}^R$ being (nearly) a Gaussian distribution. Typical values for R are around thousands, and linear and cosine β-schedules are the most common (Nichol & Dhariwal, 2021).

Given the data distribution $q(\boldsymbol{z}^0)$, the transitions of the reverse (or *denoising*) process $q(\boldsymbol{z}^{r-1}|\boldsymbol{z}^r)$ can be approximated using a neural network parametrised by θ,

$$p_\theta(\boldsymbol{z}^{r-1}|\boldsymbol{z}^r) := \mathcal{N}\left(\boldsymbol{z}^r; \boldsymbol{\mu}_\theta(\boldsymbol{z}^r, r), \boldsymbol{\Sigma}_\theta(\boldsymbol{z}^r, r)\right). \tag{2}$$

Although the mean $\boldsymbol{\mu}_\theta(\boldsymbol{z}^r, r)$ could be parametrised in several ways, Ho et al. (2020) found best to first predict the noise $\boldsymbol{\epsilon}$ with the neural network and then compute $\boldsymbol{\mu}_\theta(\boldsymbol{z}^r, r)$ according to

$$\boldsymbol{\mu}_\theta(\boldsymbol{z}^r, r) = \frac{1}{\sqrt{\alpha_r}}\left(\boldsymbol{z}^r - \frac{\beta_r}{\sqrt{1-\bar{\alpha}_r}}\boldsymbol{\epsilon}_\theta(\boldsymbol{z}^r, r)\right). \quad (3)$$

Additionally, following Nichol & Dhariwal (2021), we parameterise $\boldsymbol{\Sigma}_\theta(\boldsymbol{z}^r, r)$ as an interpolation between the lower, $\tilde{\beta}_r$, and upper, β_r, bounds on the reverse process entropy:

$$\boldsymbol{\Sigma}_\theta(\boldsymbol{z}^r, r) = e^{\mathbf{v}_\theta(\boldsymbol{z}^r, r)\log\beta_r + (1-\mathbf{v}_\theta(\boldsymbol{z}^r, r))\log\tilde{\beta}_r}, \quad (4)$$

where $\mathbf{v}_\theta(\boldsymbol{z}^r, r)$ is predicted by the neural network.

To train a network to approximate $\boldsymbol{\epsilon}_\theta(\boldsymbol{z}^r, r)$ and $\boldsymbol{v}_\theta(\boldsymbol{z}^r, r)$, we minimise the loss function $\mathcal{L} = \mathcal{L}_{\text{simple}} + \lambda_{\text{vlb}}\mathcal{L}_{\text{vlb}}$, where $\mathcal{L}_{\text{simple}}$ and $\mathcal{L}_{\text{vlb}}$ are defined as

$$\mathcal{L}_{\text{simple}} = E_{r,\boldsymbol{z}^0,\boldsymbol{\epsilon}}\left[||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{z}^r, r)||^2\right], \quad (5)$$

$$\mathcal{L}_{\text{vlb}} = -\log p_\theta(\boldsymbol{z}^0|\boldsymbol{z}^1) + \sum_2^T D_{KL}\left(q(\boldsymbol{z}^{t-1}|\boldsymbol{z}^t, \boldsymbol{z}^0) \,||\, p_\theta(\boldsymbol{z}^{t-1}|\boldsymbol{z}^t)\right). \quad (6)$$

The term $\mathcal{L}_{\text{vlb}}$ represents the variational lower bound. Its first term is the negative log-likelihood of a Gaussian distribution, while the remaining terms are the Kullback-Leibler (KL) divergence between two Gaussians. The gradients of $\boldsymbol{\epsilon}_\theta(\boldsymbol{z}^r, r)$ are only backpropagated through $\mathcal{L}_{\text{simple}}$, whereas the $\mathcal{L}_{\text{vlb}}$ term is used to optimise $\boldsymbol{v}_\theta(\boldsymbol{z}^r, r)$. This training strategy was proposed by Nichol & Dhariwal (2021) and has also been proven successful in subsequent work (Dhariwal & Nichol, 2021). To reduce gradient noise, we employ importance sampling and dynamically weight each loss term based on the loss history (Nichol & Dhariwal, 2021).

Denosing Graph Node Features The DDPM framework primarily focuses on learning the data distribution $q(\boldsymbol{z}^0)$ by optimising the parameters θ of the neural network, without addressing the network architecture itself. Although CNNs, particularly modern U-Nets, have been the preferred architectures in these studies and subsequent research, GNNs also offer a viable option for denoising node and/or edge features defined on graphs (Hoogeboom et al., 2022; Vignac et al., 2023).

In this work, we employ an MP-based GNN to learn synthesising F flow fields encoded as node features, $V^0 \in \mathbb{R}^{|\mathcal{V}|\times F}$, on a directed graph, $\mathcal{G} := (\mathcal{V}, \mathcal{E})$, by denoising its node features from $V^R \sim \mathcal{N}(0, \mathbf{I})$ and through all the intermediate features V^r. We denote this network as the *Diffusion Graph Net* (DGN). In our setting, the diffusion process adds noise only to the node features, leaving the edge features and graph structure unaltered. According to expression (1), at diffusion step r, the noisy node features are

$$V^r = \sqrt{\bar{\alpha}_r}V^0 + (1-\bar{\alpha}_r)\boldsymbol{\epsilon}. \quad (7)$$

The DGN returns as output node features the noise $\boldsymbol{\epsilon}_\theta \in \mathbb{R}^{|\mathcal{V}|}$ and the variance $\mathbf{v}_\theta \in \mathbb{R}^{|\mathcal{V}|}$ as a function of the graph $\mathcal{G}$, the input edge-features $E \in \mathbb{R}^{|\mathcal{E}|\times D}$ and the input node features V^r. To ensure translation invariance, we set the input features of each directed edge from node i to node j to its nodes' relative position, i.e., $\boldsymbol{e}_{ij} = \boldsymbol{x}_j - \boldsymbol{x}_i$.

Instead of sampling flow fields from $q(V^0)$ unconditionally, we condition these on the geometry of the fluid domain, Ω, and the Reynolds number, Re. The geometry Ω is encoded by assigning to each node i a one-hot vector, $\boldsymbol{\omega}_i$, indicating if the node is inside the fluid domain, on an inlet, or on a solid boundary (Pfaff et al., 2021; Lino et al., 2022). In this scenario, the conditional distribution $q(V^0 \mid \Omega, Re)$ is learnt by a DGN such that

$$[\boldsymbol{\epsilon}_\theta, \boldsymbol{v}_\theta] \leftarrow \text{DGN}_\theta(\mathcal{G}, V^r, E, \Omega, Re, r). \quad (8)$$

The architecture of the DGN adheres to the encoder-propagator-decoder structure introduced by Sanchez-Gonzalez et al. (2020). We employ the multi-scale propagator from (Lino et al., 2022), which applies MP across multiple scales in a U-Net fashion, albeit with certain modifications. Specifically, we replace the higher-to-lower resolution MP – as defined in equation (13) – with a weighted interpolation technique proposed by (Qi et al., 2017). Similarly, the lower-to-higher resolution MP – equation (14) – is substituted with a nearest-neighbour interpolation followed by a node-wise linear layer and the addition of a weighted skip connection from the encoder branch. This architectural choice was made based on the U-Net's ability to remove high- and low-frequency noise (Si et al., 2023). The adjustments made ensure no directional dependency.

In the DGN, the node encoder applies a linear layer to the concatenation of the noisy input features $\boldsymbol{v}_i^r$ with the conditional features $[\boldsymbol{\omega}_j, Re]$ of each node. Similarly, the edge encoder applies a linear layer to the input edge features $\boldsymbol{e}_{ij}$. Both encoders yield an F_h-dimensional feature vector for each node or edge. The diffusion step is encoded using a sinusoidal position embedding (Vaswani et al., 2017; Ho et al., 2020), followed by a linear layer producing an F_{emb}-dimensional embedding vector denoted as $\boldsymbol{r}_{\text{emb}}$. Before being fed into the propagator network, the encoded node features are concatenated with $\boldsymbol{r}_{\text{emb}}$ and projected back to an F_h-dimensional space.

The MP layers in the DGN's propagator follow the general framework described in Battaglia et al. (2016) and Battaglia et al. (2018). The edge- and node-update functions are modelled by MLPs with one hidden layer with F_h neurons and F_h output features. These MLPs are preceded by layer

normalisation (Ba et al., 2016). Besides, these MP layers have been modified to also account for $\boldsymbol{r}_{\text{emb}}$. Specifically, before the edge and node updates (equations (10) and (12) respectively), $\boldsymbol{r}_{\text{emb}}$ is projected to an F_h-dimensional space and added to the node features (equation (9)):

$$\boldsymbol{v}_j \leftarrow \boldsymbol{v}_j + W_{\text{emb}}\boldsymbol{r}_{\text{emb}} + b_{\text{emb}}, \tag{9}$$

$$\boldsymbol{e}_{ij} \leftarrow W_e\boldsymbol{e}_{ij} + \text{MLP}^e\left(\text{LN}\left([\boldsymbol{e}_{ij}|\boldsymbol{v}_i|\boldsymbol{v}_j]\right)\right), \tag{10}$$

$$\bar{\boldsymbol{e}}_j \leftarrow \frac{1}{|\mathcal{N}_j^-|}\sum_{i\in\mathcal{N}_j^-}\boldsymbol{e}_{ij}, \tag{11}$$

$$\boldsymbol{v}_j \leftarrow W_v\boldsymbol{v}_j + \text{MLP}^v\left(\text{LN}\left([\bar{\boldsymbol{e}}_j|\boldsymbol{v}_j]\right)\right). \tag{12}$$

2.2. SE(3)-Equivariant Message Passing

A function f is considered equivariant with respect to a group $\mathcal{T}$, whose transformations are parametrised by g, if $T_g \circ f = f \circ T'_g$ for all g, where T_g and T'_g denote transformations on the input and output domain of f, respectively. The SE(3) group, also known as the special Euclidean group, represents the set of rigid body transformations in three-dimensional Euclidean space. SE(3) transformations combine translation and rotation transformations. Rotations alone form the special orthogonal group, commonly denoted as the SO(3) group. While SE3-DGNs guarantee translation equivariance by taking relative nodes' position as input, ensuring SO(3) equivariance is not straightforward. This is achieved by applying the diffusion and denoising transitions in a rotation-invariant latent space leaned by an autoencoder, which leverages steerable feature vectors and steerable MLPs (Weiler et al., 2018; Brandstetter et al., 2021).

A steerable feature vector consists of the concatenation (equivalent to the direct sum) of irreducible representations. We denote steerable feature vectors with a tilde, e.g., a vector $\tilde{\boldsymbol{v}}$ is steerable. Each irreducible representation, abbreviated as *irrep*, has a given rotation order $l \in \mathbb{W}$ and dimension $2l+1$. The number of irreps in $\tilde{\boldsymbol{v}}$ with the same rotation order is known as the multiplicity of such rotation order. SO(3) elements can be represented by Wigner D-matrices, parametrised by $g \in \mathbb{R}^3$. An l-th degree Wigner D-matrix has dimensions $(2l+1)\times(2l+1)$, and it can rotate order-l irreps via matrix-vector multiplication (Gilmore, 2008). A steerable feature vector consisting of irreps of order l_0 to l_n can be rotated by multiplying it with a block diagonal matrix containing the matrices $D^{(l_0)}$ to $D^{(l_n)}$ along its diagonal – again this is equivalent to their direct sum (Weiler et al., 2018; Brandstetter et al., 2021).

Steerable MLPs (S-MLP) (Brandstetter et al., 2021) are SO(3) equivariant, i.e.,

$$\text{S-MLP}_{\tilde{\boldsymbol{a}}}\left(D(g)\,\tilde{\boldsymbol{v}}\right) = D'(g)\,\text{S-MLP}_{\tilde{\boldsymbol{a}}}\left(\tilde{\boldsymbol{v}}\right), \tag{13}$$

where $D = \bigoplus_{l=l_0}^{l_n} D^{(l)}$ and $D' = \bigoplus_{l=l'_0}^{l'_n} D'^{(l)}$, and $\tilde{\boldsymbol{a}}$ is a conditioning steerable vector, which typically contains geometrical information. Similar to conventional MLPs, S-MLPs interleave linear mappings with non-linearities. Specifically, the steerable linear mappings return the weighted sum of the Clebsch-Gordan (CG) tensor product between the irreps of the input steerable features, $\tilde{\boldsymbol{v}}$, and the irreps of $\tilde{\boldsymbol{a}}$. The m-th component of the order-l irrep resulting from the CG tensor product between an irrep of order l_1, u, and an irrep of order l_2, a, is given by

$$(\boldsymbol{u}\otimes\boldsymbol{a})_m^{(l)} = \sum_{m_1=-l_1}^{l_1}\sum_{m_2=-l_2}^{l_2} C_{l_1,m_1,l_2,m_2}^{l,m} u_{m_1} a_{m_2}, \tag{14}$$

where $C_{l_1,m_1,l_2,m_2}^{l,m}$ are known as the CG coefficients. These are zero except for l between $|l_1-l_2|$ and (l_1+l_2), inclusive. For instance, for $l_1=1$ and $l_2=1$, the CG tensor product yields an irrep of order $l=0$ and another irrep of order $l=1$. These correspond to the dot and cross product between two vectors, respectively. In the steerable linear mappings, order-l irreps resulting from the CG tensor product between all the irreps of $\tilde{\boldsymbol{v}}$ and all the irreps of $\tilde{\boldsymbol{a}}$ are combined via a learnable linear layer,

$$\tilde{\boldsymbol{v}}_m^{(l)} \leftarrow \sum_{(l_1),(l_2)} W_m^{(l)}\left(\tilde{\boldsymbol{v}}^{(l_1)}\otimes\tilde{\boldsymbol{a}}^{(l_2)}\right)_m^{(l)}, \tag{15}$$

for all l and $-l \le m \le l$ (Thomas et al., 2018). Here, $\boldsymbol{v}^{(l_1)}$ and $\tilde{\boldsymbol{a}}^{(l_2)}$ denote an order-l_1 irrep of $\boldsymbol{v}$ and an order-l_2 irrep of $\tilde{\boldsymbol{a}}$, respectively; and $W_m^{(l)}$ is the matrix of learnable weights for the m-th component of the order-l irreps. Its number of rows equals the desired output multiplicity for rotation order l, and the number of columns equals its input multiplicity. Although the maximum order of the irreps in the steerable output vector is equal to the sum of the maximum irreps of $\tilde{\boldsymbol{v}}$ and $\tilde{\boldsymbol{a}}$, in practice, the output irreps with an order larger than a given threshold are omitted to limit the computational cost (Thomas et al., 2018; Weiler et al., 2018). Following Brandstetter et al. (2021), we also add a learnable bias to the order-0 output irreps, and employ the gate activation function from Weiler et al. (2018) as the non-linearities.

Since layer normalisation is a common operation in MP layers (Sanchez-Gonzalez et al., 2020; Pfaff et al., 2021), we found it necessary to define a SE(3)-equivariant layer normalisation operation for steerable features and refer to it as steerable layer normalisation (S-LN). This scales the norm of each irrep according to

$$\text{S-LN}(\tilde{\boldsymbol{v}}) := \bigoplus_{l=l_0}^{l_n} c_l\,\frac{\tilde{\boldsymbol{v}}^{(l)}}{||\tilde{\boldsymbol{v}}^{(l)}||_2}, \tag{16}$$

where c_l is the normalised norm of the l-th irrep, i.e., the l-th element of $\text{LN}([||\tilde{\boldsymbol{v}}^{l_0}||_2,\ldots,||\tilde{\boldsymbol{v}}^{l_n}||_2])$.

We construct steerable message-passing (S-MP) layers utilising S-MLPs, preceded by S-LN, for the edge- and node-update functions. Specifically, the edge update, edge aggregation and node update steps are, in this order:

$$\tilde{\boldsymbol{e}}_{ij} \leftarrow \tilde{\boldsymbol{e}}_{ij} + \text{S-MLP}^{e}_{\tilde{\boldsymbol{a}}_{ij}}\left(\text{S-LN}\left([\tilde{\boldsymbol{e}}_{ij}, \tilde{\boldsymbol{v}}_i, \tilde{\boldsymbol{v}}_j]\right)\right), \tag{17}$$

$$\tilde{\boldsymbol{e}}_j \leftarrow \frac{1}{|\mathcal{N}_j^-|} \sum_{i \in \mathcal{N}_j^-} \tilde{\boldsymbol{e}}_{ij}, \tag{18}$$

$$\tilde{\boldsymbol{v}}_j \leftarrow \text{S-MLP}^{v}_{\tilde{\boldsymbol{a}}_j}\left(\text{S-LN}\left([\tilde{\boldsymbol{e}}_j, \tilde{\boldsymbol{v}}_j]\right)\right). \tag{19}$$

The edge-update S-ML, denoted by $\text{S-MLP}^{e}_{\tilde{\boldsymbol{a}}_{ij}}$, is conditioned on the spherical harmonic embedding of the relative position between nodes:

$$\tilde{\boldsymbol{a}}_{ij} = \bigoplus_{l=0}^{l_n} Y^{(l)}\left(\boldsymbol{x}_j - \boldsymbol{x}_i\right), \tag{20}$$

where $Y^{(l)}$ represents the l-th-degree spherical harmonic. Similarly, the node-update S-MLP, represented by $\text{S-MLP}^{v}_{\tilde{\boldsymbol{a}}_j}$, is conditioned on the neighbourhood-wise mean of such embedding, $\tilde{\boldsymbol{a}}_j = \sum_{\mathcal{N}_j^-} \tilde{\boldsymbol{a}}_{ij}/|\mathcal{N}_j^-|$. It is noteworthy that since all operations in equations (17) to (19) are SE(3) equivariant, the S-MP layer as a whole is also equivariant. For the same reason, a GNN consisting of stacked S-MP layers is SE(3)-equivariant too.

This form of MP is akin to the one presented in Brandstetter et al. (2021) concerning the utilisation of S-MLPs. However, it diverges in its utilisation of the computed messages to also update the edge features (equation (17)) and the inclusion of layer normalisation, trends prevalent in recent literature (Pfaff et al., 2021; Lino et al., 2022). Preliminary tests have demonstrated improvements with these additions.

2.3. Departure to a SE(3)-Invariant Latent Space

Latent diffusion models (LDMs) (Rombach et al., 2022) consist of a variational autoencoder (VAE) and a DDPM. The VAE is first trained to learn a latent space that is perceptually equivalent to the data space but has a lower dimensionality. The DDPM is then trained in this latent space following standard techniques (Ho et al., 2020; Nichol & Dhariwal, 2021). In this setting, the VAE's task is capturing high-frequency information from real images, while the DDPM is responsible for understanding the images' semantic information. This approach reduces the cost of training and evaluating diffusion models. Although with a different objective, we draw inspiration from LDMs for designing SE3-DGNs. A SE3-DGN consists of a steerable VGAE (S-VGAE) and an invariant DGN (I-DGN): the S-VGAE learns a latent space that represents the geometrical and physical information of the graph data in a rotation-invariant manner, and the DGN learns to denoise physical fields expressed in that latent space. Combining both components results in a time-efficient rotation-equivariant diffusion model, with which rotated versions of the physical domain result in the synthesis of equally rotated vector fields and invariant scalar fields (Figure 1).

S-VGAEs Although we refer to S-VGAEs as autoencoders, their latent space does not have reduced dimensionality in terms of the node or edge features nor a reduced graph size. In this case, the data compression refers to encoding the directional information, given as input steerable features, into latent scalar-valued features. As illustrated in Figure 2, the S-VGAE consists of an edge encoder, a node encoder, and a node decoder. All their input, hidden, and output features are steerable vectors.

The edge encoder takes as input edge features the relative position between adjacent nodes, $\tilde{\boldsymbol{e}}_{ij}$, and as input node features its neighbourhood-wise mean, $\sum_{\mathcal{N}_j^-} \tilde{\boldsymbol{e}}_{ij}/|\mathcal{N}_j^-|$. A steerable linear mapping is first applied to the edge inputs to reach the desired multiplicity for each rotation order of the hidden features. Analogously, another steerable linear mapping is applied to the node inputs. Then, several S-MP layers are applied sequentially to those hidden node and edge features. Finally, a steerable linear mapping transforms the resulting edge features to meet the target multiplicity for each rotation order, and SE(3)-equivariant batch normalisation (Weiler et al., 2018) (i.e., feature-wise normalisation) is applied to the outputs. We will denote the latent edge features of each edge (i, j) by $\tilde{\boldsymbol{\xi}}_{ij}$.

The node encoder follows the same architecture as the edge encoder. It takes the latents $\tilde{\boldsymbol{\xi}}_{ij}$ as input edge features and the steerable features, $\tilde{\boldsymbol{v}}_i$, as input node features. The feature $\tilde{\boldsymbol{v}}_i$ corresponds to the direct sum of the physical fields, expressed as irreps, at node i. After all the S-MP layers are applied, a steerable linear mapping maps the resulting node features to a $4L$-dimensional steerable vector of only scalars ($l = 0$ irreps). Then, after activation, a (standard) linear layer transforms these scalar-valued features into vectors $\boldsymbol{\mu}_i \in \mathbb{R}^L$ and $\boldsymbol{\sigma}_i \in \mathbb{R}^L$, which define a normal distribution $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i)$ for the L latent features of each node i. These latents are normalised via momentum batch normalisation. We will denote the latent node features of node i by $\boldsymbol{\zeta}_i^0$. Importantly, since they are order-0 irreps, they are invariant under rotations of $\tilde{\boldsymbol{e}}_{ij}$ and $\tilde{\boldsymbol{v}}_i$ (they rotate with $D = \bigoplus_{l_0}^{l_L} 1 = \mathbf{I}_L$).

The node decoder also takes the latents $\tilde{\boldsymbol{\xi}}_{ij}$ as input edge features, and it is conditioned on the Reynolds number and the geometry encoding $\boldsymbol{\omega}_i$ of each node i. It decodes the invariant latents $\boldsymbol{\zeta}_i^0$ back to the physical space. Its architecture is asymmetric to that of the node encoder, but first, the condition $[Re, \boldsymbol{\omega}_i]$ is embedded according to

$$\tilde{\boldsymbol{v}}_i \leftarrow \text{SELU}\left(W_\zeta\, \boldsymbol{\zeta}_i^0 + W_c\, [Re, \boldsymbol{\omega}_i]^\top + b_c\right), \tag{21}$$

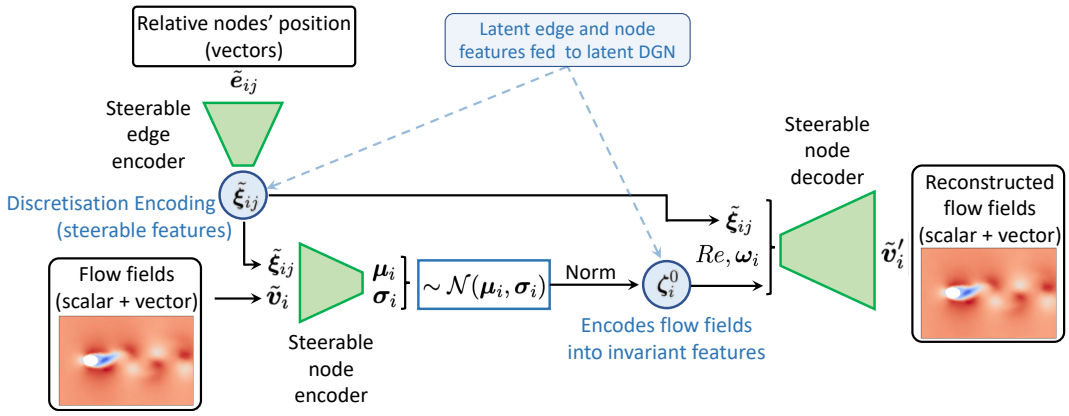

Figure 2. Steerable VGAEs (S-VGAEs). The relative position between adjacent nodes, $\tilde{e}_{ij}$, is encoded into steerable latent edge-features, $\tilde{\xi}_{ij}$, via a SE(3)-equivariant graph encoder. The flow fields, $\tilde{v}_i$, are encoded, together with $\tilde{\xi}_{ij}$, into scalar-valued latent node-features, ζ_i^0, via a SE(3)-invariant variational graph encoder. The flow fields are recovered via a SE(3)-equivariant graph decoder conditioned on the Reynolds number and each node's type. The latent features $\tilde{\xi}_{ij}$ are used as edge inputs to a DGN, which is trained to denoise ζ_i^0 after this has been diffused.

where SELU is the scaled exponential linear unit (Klambauer et al., 2017).

We train S-VGAEs following a common procedure for training VAEs. Specifically, we consider a loss function which minimises the reconstruction loss of $\tilde{v}_i$ and regularises the latents ζ_i^0 to a standard normal (Kingma & Welling, 2014).

I-DGN With our trained S-VGAE, we can now access an invariant latent space where directional information is abstracted. An I-DGN takes as input the edge features $\tilde{\Xi} := \{\tilde{\xi}_{ij} \mid (i,j) \in E\}$ and the noisy node features $Z^r := \{\zeta_i^r \mid i \in \mathcal{V}\}$, which are diffused in the latent space to diffusion step r according to equation (1), i.e.,

$$Z^r = \sqrt{\bar{\alpha}_r} Z^0 + (1 - \bar{\alpha}_r)\mathbf{I}\epsilon. \tag{22}$$

An I-DGN, such that

$$[\epsilon_\theta, \mathbf{v}_\theta] \leftarrow \text{I-DGN}_\theta(\mathcal{G}, Z^r, \tilde{\Xi}, \Omega, Re, r), \tag{23}$$

can be trained as described in Section 2.1 to learn the ϵ_θ and $\mathbf{v}_\theta$ parametrising the transitions of the reverse process. The I-DGN's architecture is the same as for the DGN described in Section 2.1, but the edge encoder, which takes the steerable latents $\tilde{\xi}_{ij}$ as input, is replaced by a steerable linear mapping which returns F_h scalar-valued features for each edge.

As depicted in Figure 1, at inference time, the edge encoder is evaluated only once to obtain $\tilde{\xi}_{ij}$. Then, ζ_i^R is sampled from an isotropic Gaussian distribution and iteratively denoised by the I-DGN. Although the diffusion model is trained with R diffusion steps, in practice, it is also possible to employ only a subset of those diffusion steps (Song & Ermon, 2020). Finally, the denoised node-features, ζ_i^0, are decoded back to the physical space of steerable features, obtaining a sample $\tilde{v}_i$ from the learnt data distribution. The added cost we pay for SE(3)-equivariant flow synthesis is the evaluation of the edge encoder and node decoder, once each. This is significantly smaller than the denoising cost.

Instead of decomposing the generative model into a S-VGAE and an I-DGN, it is also possible to achieve SE(3)-equivariance by employing a single DGN that follows the same architecture as the I-DGN but replaces the (non-steerable) MP layers with S-MP layers. Nevertheless, the high cost of S-MP makes the training and inference prohibitively expensive unless very small graphs are considered.

3. Flow Field Synthesis from Invariant Latents

While SE3-DGNs can address 3D domains, we demonstrate them on 2D flows in this work, as they offer lower computational demands and suitable datasets are scarce for 3D flows. For training and testing the flow synthesis, we employed a low-resolution version of the datasets from Lino et al. (2022), which contain simulations of the incompressible Navier-Stokes equations for the flow around an ellipse parallel to the free-stream flow, with varying Re, minor-axis size (b), and domain height (H). Each simulation consists of 101 time-steps of fully developed flow under a laminar vortex-shedding regime. The test datasets allowed us to check the extrapolation to unseen values of Re, b, and H. Additionally, one of the test datasets introduces rotations of 1 to 10 degrees to the ellipses. The parameters of each dataset are listed in Table 1. We focus on generating the pressure (p) and velocity fields ($\boldsymbol{u}$) jointly, i.e., $\tilde{\boldsymbol{v}}_i = p_i \oplus \boldsymbol{u}_i$.

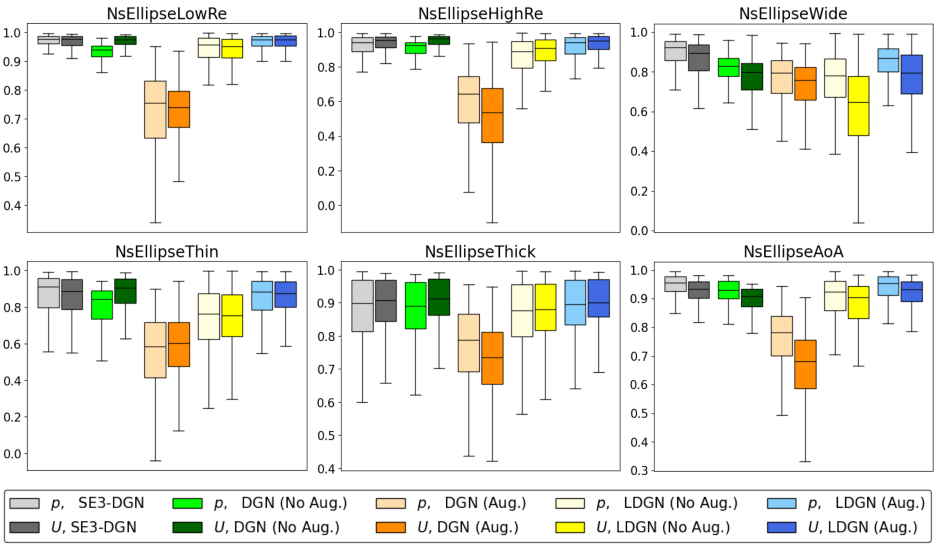

Figure 3. Coefficient of determination (R^2) of the pressure field (p) and the magnitude of the velocity field (U) for each of the tested models across the test datasets. Twenty simulation conditions were considered from each dataset (with evenly distributed Re) and 100 inferences were run for each simulation, with 50 denoising steps each.

We trained a S-VGAE with four S-MP layers in the edge encoder, node encoder, and node decoder. Each S-MLP in the S-VGAE has one hidden layer with steerable features consisting of 32 order-0 irreps and 32 order-1 irreps. The edge latents $\tilde{\boldsymbol{\xi}}_{ij}$ also consist of 32 order-0 irreps and 32 order-1 irreps, while the node latents $\boldsymbol{\zeta}_i^0$ consist of four scalars. In total, the S-VGAE has 792k learnable parameters. The KL term was weighted by 10^{-6}.

Next, we trained an I-DGN on the latent space of the S-VGAE. The diffusion process consists of $R = 1000$ steps, and the β-schedule is linear (Ho et al., 2020). The loss weight for the variational lower bound was set to $\lambda_{\text{vlb}} = 0.001$. The I-DGN has five resolution levels and two MP layers before and after each pooling/unpooling layer. Each MLP has one hidden layer with $F_h = 128$ neurons. The total number of learnable parameters is 4.176M.

During the training of the S-VGAE and the I-DGN, the learning rate was initially set to 10^{-4} and decreased by a factor of 10 after 5 and 50 epochs, respectively, without improvement. The training was stopped when the learning rate reached 10^{-8}. Together, these S-VGAE and I-DGN form a s3-DGN.

We conducted comparative analyses with non-equivariant models, including DGNs trained with and without rotation-augmented data, and latent DGNs (LDGNs) whose VGAEs were first trained with and without rotation-augmented data. The data was augmented by applying a rotation at an angle randomly sampled from 0 to 360 degrees to each data sample from the dataset during training. These non-equivariant models maintain the same architecture as their equivariant counterparts but utilise non-equivariant versions of the steerable layers. Like the SE3-DGN, the tested LDGNs comprise a VGAE and a DGN. While the VGAEs do not perform any data compression, these LDGNs serve as a relevant baseline for evaluating the SE3-DGN's performance. All models have approximately 4.9 million parameters in total. Considering this, adjustments were made to the width of both the VGAEs' and the DGNs' MLPs.

Applying rotations to the data during training can help models approximately learn the rotation equivariance between inputs and outputs. However, this does not impose any constraints on the hidden features of the trained models (Cohen & Welling, 2016). As a result, the latent space of the VGAE is not SE(3)-equivariant, even if trained with rotated data, and consequently, an LDGN trained on that latent space is also not equivariant. In contrast, a (non-latent) DGN trained with rotated data can learn to approximate rotation equivariance, as the data distribution is directly learned in the physical space where rotations are applied. Figure 6 illustrates the SE(3)-equivariance error of each model for velocity fields synthesised in a fluid domain used for training and rotated by $\gamma \in [0, 360)$ degrees. These results indicate that while the DGN trained with rotated data does learn some degree of rotation equivariance, it still falls short of guaranteeing this property even on the training data.

Diffusion models sample a flow field at a time point $t = t^*$, which cannot be directly controlled. Since we are dealing with periodic laminar flow, the time point $t = t^* + kT$,

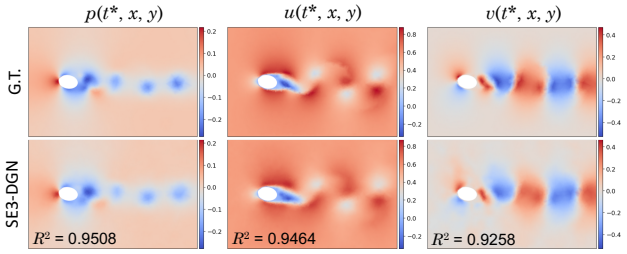

Figure 4. Pressure field (p), and horizontal (u) and vertical (v) components of the velocity fields synthesised by the SE3-DGN (bottom row) and their corresponding ground truth in the dataset `NsEllipseAoA` (top row). The ellipse has an angle of attack of 10 degrees and $Re = 800$.

where T is the period and $k \in \mathbb{N}$, falls within the range of the time points included in the ground truth simulations in the datasets. To evaluate the accuracy of the samples, we compute the coefficient of determination (R^2) between a sampled flow field and all the flow fields available in the dataset for the same domain and Re. We then select the highest R^2 value, which occurs close to $t = t^* + T$. As an example, Figure 1 shows the fields generated by the SE3-DGN (bottom row) and their corresponding ground truth in the dataset `NsEllipseAoA` (top row). In this example, the ellipse has an angle of attack of 10 degrees and $Re = 800$, illustrating the good generalisation of the SE3-DGN to non-zero angles of attack.

We compared the performance of each model across the testing datasets. Figure 3 presents the distribution of the R^2 values for the pressure field and the magnitude of the velocity field (U) for each model and dataset. For each domain and Re, inferences were run 100 times with 50 denoising steps each. The SE3-DGN outperforms the non-rotation-equivariant models. However, it is closely followed by the LDGN whose VGAE was trained with rotation-augmented data. This close performance can be attributed to two factors: (i) the VGAE assists the DGN in focusing on perceptually relevant flow features while managing the high-frequency content, and (ii) the rotation-augmented data makes the VGAE more robust to variations in the latent space, leading to higher quality decoded flows. Conversely, the DGN trained with rotation-augmented data exhibits the worst performance by a significant margin. This poor performance is likely due to insufficient learnable parameters to handle the increased training data, as the input depends on the diffusion step r (Dhariwal & Nichol, 2021) and also the rotation angle.

Figure 5 displays samples of the horizontal component of the velocity field synthesised by each of the tested models for various domains and Re from the `NsEllipseAoA` dataset. Overall, the SE3-GNN demonstrates higher sample quality. This figure also highlights the importance of the VGAE decoder, whether steerable or not, in removing high-frequency noise left after diffusion denoising, since the baseline DGNs' samples exhibit noticeable high-frequency noise, whereas the samples from the latent models do not.

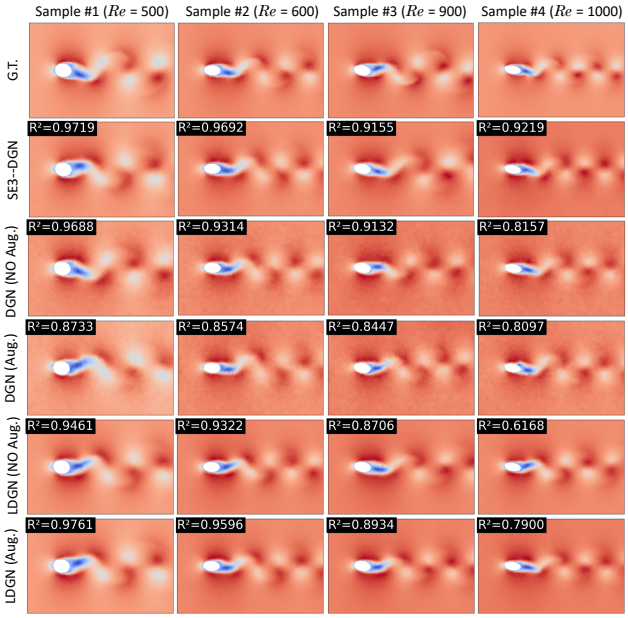

Figure 5. Samples of the horizontal component of the velocity field synthesised by each of the tested models from geometries and Re in the dataset `NsEllipseAoA`. Overall, SE3-GNN shows higher sample quality. The LDGN whose VGAE was trained with rotated-augmented data shows only slightly worse performance.

4. Conclusion

This work introduced SE3-DGNs, efficient SE(3)-equivariant diffusion models for field generation on graphs. SE3-DGNs combine a S-VGAE, which abstracts directional information and learns an invariant latent space, with a DGN, which generates physical fields in that latent space. We demonstrated this method by learning the distribution of pressure and velocity fields for laminar vortex-shedding behind 2D elliptical cylinders. We found that, under new Reynolds numbers and domain geometries, a s3-DGN can generate high-quality fields with lower residual noise than conventional DGNs while ensuring SE(3)-equivariance. Its non-equivariant counterpart, using a VGAE trained with rotated data, also achieved similar generalisation performance due to a robust decoder that filters high-frequency noise. Overall, SE3-DGNs and LDGNs present a promising approach for generating fully-developed initial conditions for transient numerical solvers.

References

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Battaglia, P. W., Pascanu, R., Lai, M., Rezende, D., and Kavukcuoglu, K. Interaction networks for learning about objects, relations and physics. *Advances in Neural Information Processing Systems*, 37, 2016.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv:1806.01261*, 2018.

Brandstetter, J., Hesselink, R., van der Pol, E., Bekkers, E. J., and Welling, M. Geometric and physical quantities improve e(3) equivariant message passing. In *International Conference on Learning Representations*, 2021.

Brandstetter, J., Welling, M., and Worrall, D. E. Lie point symmetry data augmentation for neural pde solvers. In *International Conference on Machine Learning*, pp. 2241–2256. PMLR, 2022.

Cohen, T. and Welling, M. Group equivariant convolutional networks. In *International conference on machine learning*, pp. 2990–2999. PMLR, 2016.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Fortunato, M., Pfaff, T., Wirnsberger, P., Pritzel, A., and Battaglia, P. Multiscale MeshGraphNets. In *ICML 2022 Workshop on AI for Science*, 2022.

Gao, R., Deo, I. K., and Jaiman, R. K. A finite element-inspired hypergraph neural network: Application to fluid dynamics simulations. *Journal of Computational Physics*, pp. 112866, 2024.

Gilmore, R. *Lie groups, physics, and geometry: an introduction for physicists, engineers and chemists*. Cambridge University Press, 2008.

Guillard, H. Node-nested multi-grid method with Delaunay coarsening. Technical report, INRIA, 1993.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Hoogeboom, E., Satorras, V. G., Vignac, C., and Welling, M. Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning*, pp. 8867–8887. PMLR, 2022.

Karniadakis, G. and Sherwin, S. *Spectral/hp element methods for computational fluid dynamics*. Oxford University Press, 2013.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. Self-normalizing neural networks. *Advances in Neural Information Processing Systems*, 30, 2017.

Klicpera, J., Groß, J., and Günnemann, S. Directional message passing for molecular graphs. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

Kohl, G., Chen, L.-W., and Thuerey, N. Turbulent flow simulation using autoregressive conditional diffusion models. *arXiv preprint arXiv:2309.01745*, 2023.

Lienen, M., Lüdke, D., Hansen-Palmus, J., and Günnemann, S. From zero to turbulence: Generative modeling for 3d flow simulation. In *The Twelfth International Conference on Learning Representations*, 2024.

Ling, J., Jones, R., and Templeton, J. Machine learning strategies for systems with invariance properties. *Journal of Computational Physics*, 318:22–35, 2016.

Lino, M. Nsellipse datasets from "multi-scale rotation-equivariant graph neural networks for unsteady eulerian fluid dynamics" (1.0.0) [data set], 2023. URL https://doi.org/10.5281/zenodo.7892171.

Lino, M., Fotiadis, S., Bharath, A. A., and Cantwell, C. D. Multi-scale rotation-equivariant graph neural networks for unsteady eulerian fluid dynamics. *Physics of Fluids*, 34(8), 2022.

Lino, M., Fotiadis, S., Bharath, A. A., and Cantwell, C. D. Current and emerging deep-learning methods for the simulation of fluid dynamics. *Proceedings of the Royal Society A*, 479(2275):20230058, 2023.

Lippe, P., Veeling, B., Perdikaris, P., Turner, R., and Brandstetter, J. Pde-refiner: Achieving accurate long rollouts with neural pde solvers. *Advances in Neural Information Processing Systems*, 36, 2024.

Liu, Q. and Thuerey, N. Uncertainty-aware surrogate models for airfoil flow simulations with denoising diffusion probabilistic models. *AIAA Journal*, pp. 1–22, 2024.

Nichol, A. Q. and Dhariwal, P. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PMLR, 2021.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. Learning mesh-based simulation with graph networks. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

Pope, S. B. and Pope, S. B. *Turbulent flows*. Cambridge University Press, 2000.

Prantl, L., Ummenhofer, B., Koltun, V., and Thuerey, N. Guaranteed conservation of momentum for learning particle-based fluid dynamics. *Advances in Neural Information Processing Systems*, 35, 2022.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30, 2017.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Shankar, V., Barwey, S., Kolter, Z., Maulik, R., and Viswanathan, V. Importance of equivariant and invariant symmetries for fluid flow modeling. *arXiv preprint arXiv:2307.05486*, 2023.

Shu, D., Li, Z., and Farimani, A. B. A physics-informed diffusion model for high-fidelity flow field reconstruction. *Journal of Computational Physics*, 478:111972, 2023.

Si, C., Huang, Z., Jiang, Y., and Liu, Z. Freeu: Free lunch in diffusion u-net. *arXiv preprint arXiv:2309.11497*, 2023.

Siddani, B., Balachandar, S., and Fang, R. Rotational and reflectional equivariant convolutional neural network for data-limited applications: Multiphase flow demonstration. *Physics of Fluids*, 33(10):103323, 2021.

Song, Y. and Ermon, S. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448, 2020.

Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., and Riley, P. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

Ummenhofer, B., Prantl, L., Thuerey, N., and Koltun, V. Lagrangian fluid simulation with continuous convolutions. In *In Proceedings of the 7th International Conference on Learning Representations*, 2019.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Vignac, C., Krawczuk, I., Siraudin, A., Wang, B., Cevher, V., and Frossard, P. Digress: Discrete denoising diffusion for graph generation. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

Wang, R., Walters, R., and Yu, R. Incorporating symmetry into deep dynamics models for improved generalization. In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. S. 3D steerable CNNs: Learning rotationally equivariant features in volumetric data. *Advances in Neural Information Processing Systems*, 31, 2018.

A. Incompressible flow data

For training and testing the flow synthesis, we employed a low-resolution version of the datasets from Lino et al. (2022) and Lino (2023), which contain simulations of the incompressible Navier-Stokes equations for the flow around an elliptical cylinder with varying Reynolds number (Re), minor-axis size (b), and domain height (H). Each simulation consists of 101 time-steps of fully developed flow under a laminar vortex-shedding regime. The resolution was reduced by a factor of approximately two via Guillard's coarsening (Guillard, 1993).

These datasets are listed in Table 1. The first dataset was used to train the models, while the remaining datasets were used to test generalization to out-of-distribution flow parameters.

Table 1. Training and testing datasets (Lino et al., 2022; Lino, 2023). AoA stands for *angle of attack*, and A_{element} refers to the element-size parameter in *Gmsh* meshes.

Dataset	Re	b	H	AoA (deg)	A_{element}	#Simulations	Purpose
`NSEllipse`	500-1000	0.5-0.8	5-6	0	0.10-0.16	5000	Training
`NSEllipseLowRe`	300-500	0.5-0.8	5-6	0	0.10-0.16	500	Testing
`NSEllipseHighRe`	1000-1200	0.5-0.8	5-6	0	0.10-0.16	500	Testing
`NSEllipseThin`	500-1000	0.4-0.5	5-6	0	0.10-0.16	500	Testing
`NSEllipseThick`	500-1000	0.8-1.0	5-6	0	0.10-0.16	500	Testing
`NSEllipseNarrow`	500-1000	0.5-0.8	4-5	0	0.10-0.16	500	Testing
`NSEllipseWide`	500-1000	0.5-0.8	6-7	0	0.10-0.16	500	Testing
`NsEllipseAoA`	500-1000	0.5-0.8	5.5	0-10	0.12	240	Testing

B. SE(3)-Equivariance error

Figure 6 illustrates the SE(3)-equivariance error of each tested model for velocity fields synthesised in a fluid domain used for training and rotated by $\gamma \in [0, 360)$ degrees. This error was calculated as the node-wise mean error of $|f(D^{(1)}(\gamma)\,\tilde{\boldsymbol{e}}_{ij}) - D^{(1)}(\gamma)\,f\,(\tilde{\boldsymbol{e}}_{ij})\,|$, where f represents the entire denoising process from Gaussian noise. These results indicate that while the DGN trained with rotated data does learn some degree of rotation equivariance, it still falls short of guaranteeing this property even on the training data.

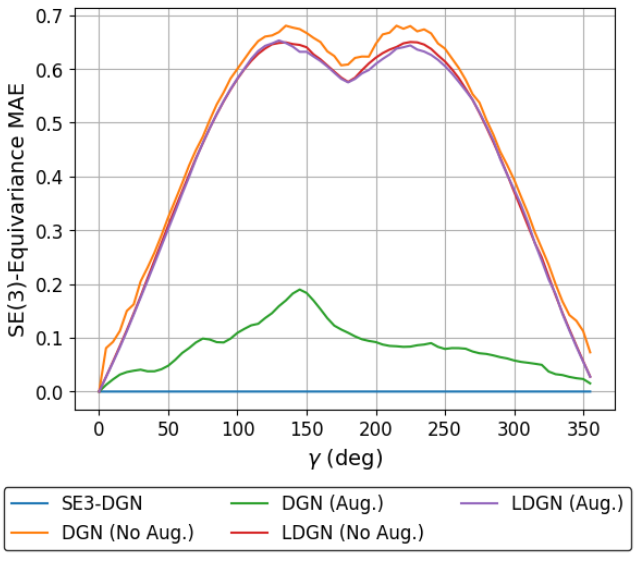

Figure 6. For each tested model, SE(3)-equivariance error of velocity fields generated from a training domain rotated an angle $\gamma \in [0, 360)$ degrees.