University of Oxford



SigmaFlow: Riemannian Flow Matching for Generative Protein-Ligand Docking

by

Julian Herbert Müller

St-Hugh's College

A dissertation submitted in partial fulfilment of the degree of Master of Science in Statistical Science.

Department of Statistics, 24-29 St Giles,
Oxford, OX1 3LB

September 2026

This is my own work (except where otherwise indicated)

Candidate: Julian Herbert Müller

Signed:Julian Müller

Date:14.09.2026

Abstract

Molecular docking predicts the bound geometry of a ligand in a protein binding site. SigmaDock, a state-of-the-art re-docking model, uses score-based Riemannian diffusion over rigid fragments in SE(3) and selects one pose from multiple generated candidates [21]. In our reproduction, a single candidate succeeds in only 24.1% of cases, while ranking 40 candidates raises success to 64.5%. This motivates a quality-quantity trade-off: at fixed inference compute, can slightly weaker individual samples be offset by generating more candidates?

SigmaDock itself offers little room for this exchange, since reducing inference to 5 integration steps lowers success to 0.6% per sample and 6.5% after ranking. We therefore replace diffusion with Riemannian flow matching, using geodesic paths with time-independent conditional velocities. The resulting models, SigmaFlow-NE and SigmaFlow-TR[1], differ only in their translational and rotational readout. Both remain effective at 5 integration steps. With a fixed budget of one thousand network evaluations, 200 5-step samples achieve Top-1 success rates of 72.3% and 70.4% on PoseBusters, and 87.1% and 92.9% on Astex, compared with 68.1% and 65.2%, and 81.2% and 82.4%, respectively, for 40 25-step samples. Reallocating inference compute towards more low-step samples therefore improves performance and significantly outperforms our reproduced SigmaDock baseline.

The gain is associated with a broader generated distribution. SigmaFlow produces more structurally distinct candidate poses and places less mass in its dominant structural mode, allowing additional sampling to remain useful for longer. The generated distribution also contains uncertainty information: the fraction of candidates in the largest structural cluster predicts final success with ROC-AUC 0.74-0.79 across the tested configurations. Finally, a cluster oracle that identifies a structural mode containing a successful pose improves Top-1 by 10-13 percentage points and nearly exhausts the full pose oracle. The remaining recoverable ranking error therefore lies primarily in selecting between competing binding modes rather than ranking poses within the correct mode.

[1]Code: https://github.com/jmuelleo/SigmaFlow

Acknowledgements

I would first like to thank Professor Garrett Morris for supervising this thesis and for his guidance, support and encouragement throughout the project. His feedback, ideas and willingness to discuss the many questions that arose during the development of this work were invaluable.

I am grateful to the Friedrich Naumann Foundation for Freedom for supporting me throughout my MSc, both through the Hans-Dietrich-Genscher Scholarship for one-year Master's programmes in England and through its general scholarship programme. This support made my year at Oxford possible and allowed me to focus fully on my studies and research.

I would also like to acknowledge the use of the University of Oxford Advanced Research Computing (ARC) facility in carrying out this work. https://doi.org/10.5281/zenodo.22558

On a personal level, I would like to thank my grandmother, who cared for me like a mother and who, from the time I was very young, spent countless hours reading to me and finding newspaper articles about subjects she thought might capture my interest. She did more than anyone to awaken my curiosity about the world and the desire to understand it. Without her, I would never have made it to Oxford. I hope that she rests in peace and that, in some way, she is able to see this milestone in my life, for which she bears so much responsibility.

I would also like to thank my mother for all of her support throughout this year. Her extraordinary work ethic, perseverance and diligence have always impressed me and continue to inspire me.

Finally, I would like to thank my fiancée. I am especially grateful for her patience and for everything she has put up with during this year, including my decision to spend my birthday working on this thesis rather than seeing her for the first time in three months. I hope she knows how much I love her and how deeply I appreciate everything she has done for me.

Finally, I would like to thank God for the gift of life, for the opportunity to experience this journey, and for allowing me to work on things that I genuinely love and find fascinating.

Declaration of AI Use

In creating this work, I have used AI tools for the following tasks:

Spelling and grammar: *I used Claude Code (Claude Opus 5) and ChatGPT (GPT-5.6 Sol) to correct spelling, grammar and punctuation throughout the thesis.*

Improvements to writing: *I used ChatGPT and Claude Code to suggest revisions to the clarity, concision and structure of text, including shortening passages to meet the word limit and redrafting individual sentences and passages. The scientific arguments, interpretations and conclusions were developed by me, and all AI-assisted text was critically reviewed and revised multiple times by me.*

Coding: *I wrote the model implementation myself and used Claude Code and ChatGPT to review it, to debug errors during development, and to explain unfamiliar library behaviour.*

Plotting, formatting etc.: *I used Claude Code to help generate the figures and tables of this thesis, including the result plots, the PyMOL scenes used for the pose visualisations, and the interactive web pages linked in the text. I also used it for LaTeX formatting and for the layout and labelling of figures.*

Repository organisation: *I used Claude Code to assist with organising and presenting the accompanying GitHub repository, including its directory structure, documentation, README files, and general formatting.*

Contents

1 Introduction **1**

1.1 Docking as a generative problem 1

1.2 Research question and scope 2

2 Rigid-Fragment Docking and the SigmaDock Baseline **3**

2.1 Rigid fragments and the generative state 3

2.2 SigmaDock on translations: why learn a score? 5

2.3 Extending SigmaDock from translations to rotations 7

2.3.1 From Gaussian noise to rotational diffusion 10

2.4 SigmaDock on the full product space 13

2.5 The inherited backbone and Newton-Euler readout 14

2.6 Sampling, ranking and evaluation 16

3 From Scores to Velocities: Riemannian Flow Matching **18**

3.1 Flow matching in Euclidean space 18

3.2 Conditional paths and the marginal vector field 19

3.3 Learning the field by conditional flow matching 20

3.4 From straight lines to rotational geodesics 21

3.5 Body frame and spatial frame 24

3.6 The SigmaFlow probability path and objective 27

4 SigmaFlow-NE: The Minimal Conversion **29**

4.1 Design principle and what actually changed 29

4.2 The vector field readout 29

5 SigmaFlow-TR: Translation and Rotation Specific Heads **31**

5.1 Motivation 31

5.2 Construction 31

6 Results **33**

6.1 Experimental basis . . . 33
6.1.1 The compute budget . . . 33
6.1.2 Performance estimation . . . 34
6.2 Integration steps and what a candidate costs . . . 34
6.3 PoseBusters . . . 36
6.4 The Astex Diverse Set . . . 37
6.5 Performance over fragment count . . . 38
6.6 Position against published methods . . . 39

7 Generated distribution **42**
7.1 Training trajectories . . . 42
7.2 Effective poses . . . 43
7.3 Distributional concentration as uncertainty . . . 45
7.4 Where docking failures occur . . . 46
7.5 The cluster oracle . . . 46
7.6 Implications for selection and uncertainty . . . 47

8 Discussion **49**
8.1 What the comparison establishes . . . 49
8.2 Limitations . . . 50
8.3 Future work . . . 50
8.3.1 Matching the published training protocol . . . 50
8.3.2 Learning to select the structural mode . . . 50
8.3.3 Informative source distributions . . . 50
8.3.4 Likelihood-based pose confidence . . . 51
8.3.5 Rectified Flows . . . 51

9 Conclusion **52**

List of Figures

2.1 Fragment decomposition of a ligand 4
2.2 Rigid fragments per ligand in both benchmarks 5
2.3 Geodesics, exponential and logarithmic map 8
2.4 The SigmaDock architecture 14

3.1 Spatial and body-frame coordinates 25
3.2 Conditional probability paths on the two factors 28

5.1 Conceptual comparison of SigmaFlow-TR and SigmaFlow-NE 32

6.1 Generated fragment trajectories 35
6.2 Top-1 after ranking against the number of drawn poses 38
6.3 Top-1 after ranking against the number of ligand fragments 39
6.4 Performance against Published Methods on PB308 40
6.5 Performance against Published Methods on AX85 41

7.1 Metric trajectories over training snapshots 43
7.2 Clustered pose distributions for one complex 44

List of Tables

4.1 Changes from SigmaDock to SigmaFlow 29

6.1 The 72h Training Runs 33
6.2 Performance on PB308 36
6.3 Performance on AX85 37

7.1 Generated Distribution Summary 45
7.2 Cluster Oracle vs. Full Oracle 46

Chapter 1

Introduction

1.1 Docking as a generative problem

Molecular docking predicts where a small molecule, the *ligand*, binds to a protein and in what orientation and internal conformation. Classical docking treats this as optimisation, searching the pose space for the minimum of an approximate binding score. This is difficult because the score is only a surrogate for the underlying interactions, conformational search is costly, and a ranked list of poses gives no probability distribution from which uncertainty can be read.

A generative model instead learns a distribution over plausible binding configurations conditioned on the protein, ligand and binding pocket, so that docking becomes sampling from

$$p_{\text{data}} = p(\text{pose} \mid \text{protein}, \text{ligand}, \text{binding pocket}), \tag{1.1}$$

with different samples representing alternative hypotheses. DiffDock demonstrated diffusion-based generative docking at scale and motivated models adapted to the geometry of molecular pose space [6].

For a rigid fragment, a pose consists of a translation and a rotation, hence an element of the special Euclidean group SE(3). Decomposing a flexible ligand into N rigid fragments leads to $\text{SE}(3)^N$. SigmaDock combines this representation with an equivariant network and Riemannian diffusion [21]. Its final prediction, however, is selected from multiple generated candidates. Performance therefore depends on both the quality of each draw and the set of alternatives available to the ranker.

This creates a computational trade-off. At fixed network evaluations, compute can be spent on more integration steps per candidate or on more candidates. The latter is useful only if low-step sampling remains reliable. Flow matching is attractive because its probability path can be chosen directly, allowing us to prescribe simple transport paths that should remain easy to integrate with few steps [9, 13]. In protein backbone generation, replacing SE(3) diffusion [26] with flow matching has already enabled effective sampling with substantially fewer integration

steps [25].

SigmaFlow tests the corresponding hypothesis for docking. It replaces SigmaDock's diffusion dynamics with Riemannian flow matching while retaining its fragment representation, data pipeline and, as far as possible, its architecture. If useful samples survive coarser integration, the saved network evaluations can instead enlarge the candidate pool and potentially improve the final ranked prediction at fixed compute.

1.2 Research question and scope

Flow matching learns a velocity field directly by regression, and its Riemannian extension carries the construction to manifolds [3, 13]. Unlike diffusion, it gives direct control over the transport path. For docking this leads to the central question of the thesis:

> *Diffusion-based docking allocates substantial inference compute to the quality of each individual sample, while final performance depends critically on selection among multiple candidates. This thesis asks whether Riemannian flow matching enables a different allocation through shorter integration and larger candidate sets.*

The design principle is *minimal intervention*. SigmaFlow-NE replaces SigmaDock's diffusion process with Riemannian flow matching while retaining its Newton-Euler readout, isolating the change in generative mechanism. SigmaFlow-TR then replaces that shared readout with separate translation and rotation heads, testing an alternative architectural choice.

The remainder of this thesis is organised as follows. Chapter 2 defines the rigid-fragment state on $\mathrm{SE}(3)^N$, introduces only the rotational geometry needed for SigmaDock and develops its score-based diffusion baseline. Chapter 3 builds conditional flow matching first in Euclidean space and then on $\mathrm{SO}(3)$, leading to SigmaFlow's paths and training objective. These two chapters are intentionally self-contained, their aim is to make the models readable without prior background in differential geometry, diffusion or flow matching, while avoiding machinery that is not used later. Chapters 4 and 5 give the two implementations. Chapter 6 studies integration cost, candidate count and docking performance, Chapter 7 analyses the generated distributions and ranking failures, and Chapter 8 discusses the conclusions and limitations.

The experimental comparison is controlled rather than a full reproduction of SigmaDock's published training. All three models use the same leaner training regime and are evaluated on PoseBusters version 2 [2] and the Astex Diverse Set [8].

Chapter 2

Rigid-Fragment Docking and the SigmaDock Baseline

Chapter 1 posed docking as conditional generation on a fragment state space. SigmaDock represents each rigid fragment by one translation and one rotation. The translational factor behaves like an ordinary Euclidean diffusion model, while rotations require additional geometry.

This chapter develops the model in that order. Translation first introduces noising, scores and reverse generation without differential geometry. The rotational section then adds only the objects needed later: relative rotations, geodesic motion, uniform rotational noise and rotational scores. Finally both factors are combined with the inherited EquiformerV2 backbone and Newton-Euler readout. The aim is a self-contained route to the mathematics used by SigmaDock and SigmaFlow rather than a general treatment of Lie groups or Riemannian geometry.

2.1 Rigid fragments and the generative state

Let a ligand contain N_a atoms. A pose assigns a position in $\mathbb{R}^3$ to each atom, but generating all $3N_a$ coordinates directly ignores substantial chemical structure. Bond lengths and bond angles fluctuate comparatively little, whereas much of a ligand's conformational flexibility comes from rotation about torsional bonds. SigmaDock exploits this by decomposing the ligand into N rigid fragments and treating the internal geometry of each fragment as fixed at a precomputed conformer [21]. Figure 2.1 illustrates the resulting representation. Once a fragment is treated as rigid, placing it requires only two choices: its centre $p \in \mathbb{R}^3$ and its orientation $R \in \mathrm{SO}(3)$. If $\tilde{x}_1, \ldots, \tilde{x}_{n_F}$ denote the fixed atom positions in a coordinate frame attached to fragment F, their global positions are

$$x_i = R\tilde{x}_i + p. \tag{2.1}$$

The pair (R, p) is an element of the special Euclidean group

$$\mathrm{SE}(3) = \{(R, p) : R \in \mathrm{SO}(3),\ p \in \mathbb{R}^3\}, \qquad (R_2, p_2) \circ (R_1, p_1) = (R_2R_1, R_2p_1 + p_2). \tag{2.2}$$



(a) **(b)**

Figure 2.1: Ligand 82V from PDB entry 7USH. Left: 26 heavy atoms, hence 78 Cartesian coordinates. Right: severing two bonds (dashed) yields three rigid fragments, so $D = 18$.

For generation it is useful to think of the state more simply as one translation and one rotation per fragment,

$$z = ((p_1, R_1), \ldots, (p_N, R_N)), \qquad \mathcal{M} = \mathrm{SE}(3)^N \cong (\mathbb{R}^3)^N \times \mathrm{SO}(3)^N, \qquad \dim \mathcal{M} = 6N. \quad (2.3)$$

Torsional flexibility then appears through the relative placement of neighbouring fragments rather than explicit torsion variables.

This introduces more formal degrees of freedom than a torsional model but simplifies the geometry of learning. Changing one torsion moves every atom downstream and therefore induces coupled, non-local Cartesian displacements. Disjoint rigid fragments instead admit a factorised product structure, formalised by SigmaDock as follows:

Theorem 2.1 (21, Theorem 1)**.** *For standard molecular topologies, torsional models define nonlinear mappings from torsion angles to Cartesian coordinates, producing highly entangled, non-product induced measures. In contrast, disjoint rigid fragments yield a factorised product of Haar measures on* $\mathrm{SE}(3)^N$.

Two ingredients limit the cost of this representation. SigmaDock stochastically merges compatible neighbouring fragments, and cross-fragment distance features preserve bond-length and bond-angle information across severed bonds without fixing the corresponding dihedral [21]. Thus the fragments are independent coordinates of the generative state but not chemically independent network inputs. Figure 2.2 visualises the empirical distribution of fragment counts on the evaluation sets and gives a median of four fragments, corresponding to 24 generated degrees of freedom. The main geometric difficulty is therefore not dimension alone, but that three of the six degrees of freedom per fragment are rotational and lie on $\mathrm{SO}(3)$.

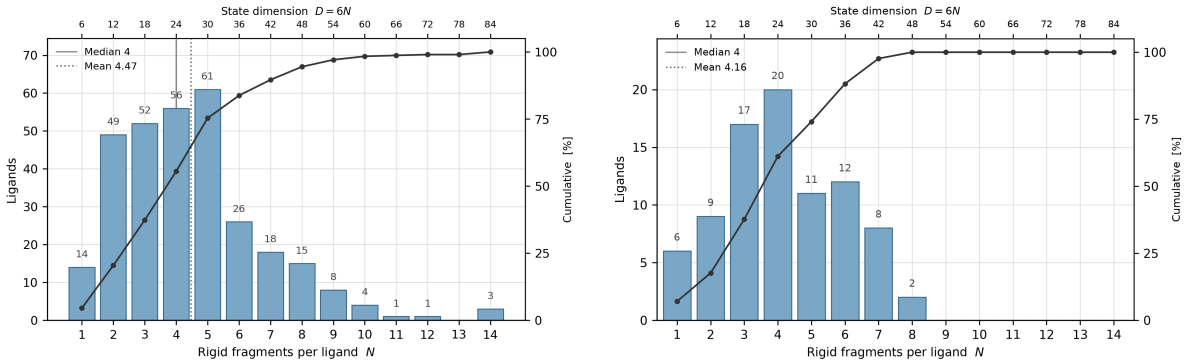


Figure 2.2: Number of rigid fragments each ligand is cut into, on PoseBusters (left, $n = 308$) and Astex (right, $n = 85$), with the resulting state dimension $D = 6N$ on the upper axis and the cumulative share on the right. Both panels share the same horizontal range: Astex contains no ligand beyond eight fragments, where PoseBusters reaches fourteen.

Conventions: Translations are measured relative to the pocket centroid. Each fragment is centred at its atom centroid and inherits axes from the input conformer, so $R_F = I$ retains that conformer's orientation. These axes are only a coordinate convention, replacing $\tilde{x}_F$ by $R_0\tilde{x}_F$ and R_F by $R_F R_0^{-1}$ leaves all Cartesian atom positions unchanged. This gauge freedom matters for the readout in Section 2.5. Finally, although SE(3) is a semidirect product, SigmaDock uses a direct-product geometry and noises translations and rotations independently. SigmaFlow inherits this choice.

2.2 SigmaDock on translations: why learn a score?

Before treating rotations, consider only the fragment centres. Let P_t denote their random state at time t, with a realisation $p_t = (p_{t,1}, \ldots, p_{t,N}) \in \mathbb{R}^{N\times 3}$. On this factor, SigmaDock is an ordinary score-based diffusion model: it gradually corrupts a crystallographic pose with noise and learns how to reverse this process.

For translation, SigmaDock uses a variance-preserving process. Conditional on clean centres $P_0 = p_0$, the state at time t has the Gaussian form

$$p_t = \alpha_t p_0 + \sigma_t \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \qquad \sigma_t^2 = 1 - \alpha_t^2, \tag{2.4}$$

where

$$\alpha_t = \exp\left(-\frac{1}{2}\int_0^t \beta_s \, \mathrm{d}s\right), \qquad \beta_t = \beta_{\min} + t(\beta_{\max} - \beta_{\min}), \qquad \beta_{\min} = 0.1, \quad \beta_{\max} = 20. \tag{2.5}$$

As α_t decreases, less of the crystallographic position remains and more Gaussian noise is present [21].

The same noising process can equivalently be described dynamically by

$$\mathrm{d}P_t = -\frac{1}{2}\beta_t P_t \,\mathrm{d}t + \sqrt{\beta_t}\,\mathrm{d}W_t. \tag{2.6}$$

The first term contracts the current state, while the Brownian increment $\mathrm{d}W_t$ injects Gaussian noise. It can be shown that this SDE has exactly the transition law in (2.4) [21, 24].

Generation requires reversing this process. Write q_t for the density of P_t. The reverse dynamics can be obtained from the known forward dynamics if one also knows the *score*

$$s_t(p) = \nabla_p \log q_t(p). \tag{2.7}$$

The following theorem makes precise why SigmaDock needs to learn this quantity.

Theorem 2.2 (Reverse-time SDE: Adapted from 24, Sections 3.1-3.2)**.** *Suppose the forward process has the Euclidean form*

$$\mathrm{d}P_t = f(P_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}W_t \tag{2.8}$$

and has density q_t. Under the regularity conditions required for time reversal, the same family of marginal densities is traversed backwards from T to 0 by

$$\mathrm{d}P_t = \Big[f(P_t, t) - g(t)^2 \nabla_p \log q_t(P_t)\Big]\,\mathrm{d}t + g(t)\,\mathrm{d}\overline{W}_t, \qquad \mathrm{d}t < 0. \tag{2.9}$$

For (2.6),

$$f(p, t) = -\frac{1}{2}\beta_t p, \qquad g(t) = \sqrt{\beta_t}.$$

Both are fixed by the forward process. The only unknown term in the reverse drift is therefore $\nabla_p \log q_t(p)$. Learning this score at every noise level specifies the reverse dynamics and allows a numerical SDE solver to transport samples from noise back towards the data distribution.

The marginal density q_t cannot be evaluated directly. The transition conditional on a clean training example is known, however. From (2.4),

$$q_{t|0}(p_t \mid p_0) = \mathcal{N}\Big(p_t; \alpha_t p_0, \sigma_t^2 I\Big), \tag{2.10}$$

so its score is obtained by differentiating the Gaussian log-density with respect to the noisy state p_t,

$$\nabla_{p_t} \log q_{t|0}(p_t \mid p_0) = -\frac{p_t - \alpha_t p_0}{\sigma_t^2} = -\frac{\epsilon}{\sigma_t}. \tag{2.11}$$

This conditional score is available exactly from the sampled training example and noise. Denoising score matching shows that regressing against it recovers the inaccessible marginal score.

Theorem 2.3 (Denoising Score Matching: Adapted from 24, Section 3.3)**.** *For any positive*

weighting function $w(t)$, the minimiser of

$$\mathbb{E}_{t\sim\mathcal{U}[0,T],\, P_0\sim p_{\text{data}},\, P_t\sim q_{t|0}(\cdot|P_0)}\left[w(t)\left\|s_\theta^p(P_t,t)-\nabla_{P_t}\log q_{t|0}(P_t\mid P_0)\right\|^2\right] \tag{2.12}$$

is the marginal score $s_t(p)=\nabla_p \log q_t(p)$, up to sets of measure zero, provided the relevant expectations exist and the model class can represent the optimum.

Together, Theorems 2.2 and 2.3 explain the learning problem: the network is trained against an accessible conditional score and thereby learns the marginal score needed to construct the reverse drift.

For numerical conditioning, SigmaDock does not expose the raw scale $1/\sigma_t$ to its translational network output. If ΔT_F denotes the translation-like quantity produced by its rigid-body readout, it forms

$$s_\theta^p=\lambda_{\mathbb{R}^3}(t)\,\Delta T_F, \qquad \lambda_{\mathbb{R}^3}(t)=\frac{1}{\sigma_t}. \tag{2.13}$$

Thus the readout can remain on a comparable scale across noise levels while the known factor $\lambda_{\mathbb{R}^3}(t)$ restores the score scale.

For complete rigid fragments this construction must also act on rotations. The score-matching principle is unchanged, but the geometry and noise process must respect SO(3). The presentation below follows Solà et al. [23], with rotational diffusion as in De Bortoli et al. [7], Yim et al. [26] and Prat et al. [21].

2.3 Extending SigmaDock from translations to rotations

A fragment orientation is represented by a rotation matrix

$$\mathrm{SO}(3)=\{R\in\mathbb{R}^{3\times3}: R^\top R=I,\ \det R=1\}. \tag{2.14}$$

Unlike $\mathbb{R}^3$, this set is not closed under addition or scalar multiplication. For example, $(1-t)R_a+tR_b$ generally leaves SO(3), as would adding Gaussian noise independently to the nine matrix entries. Euclidean operations therefore need rotational counterparts.

Although SO(3) is curved globally, it is linear locally. The instantaneous velocities of smooth curves through a rotation R form its three-dimensional tangent space $T_R\mathrm{SO}(3)$. At the identity I, this space consists exactly of the skew-symmetric matrices,

$$\mathfrak{so}(3)=T_I\mathrm{SO}(3)=\{A\in\mathbb{R}^{3\times3}: A^\top=-A\}, \tag{2.15}$$

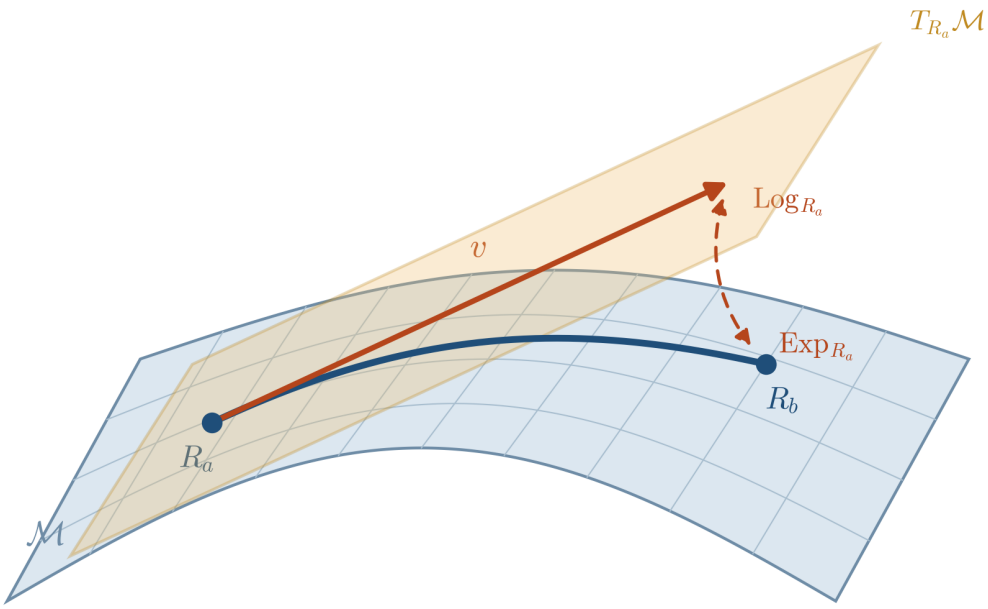


Figure 2.3: Schematic illustration of the exponential and logarithmic maps. The shaded surface represents a curved manifold rather than SO(3) itself. The solid curve is the shortest geodesic from R_a to R_b, $v = \mathrm{Log}_{R_a}(R_b)$ is its tangent displacement at R_a, and $R_b = \mathrm{Exp}_{R_a}(v)$.

called the Lie algebra of SO(3). Each element can be identified with a vector $\omega \in \mathbb{R}^3$ through the *hat map*

$$\widehat{\omega} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}, \qquad \omega \in \mathbb{R}^3, \tag{2.16}$$

with inverse $(\cdot)^\vee$ [23].

Moving between the manifold and a tangent space. The tangent space provides a linear representation of local motion, while the rotations themselves remain on the manifold. The exponential and logarithmic maps connect these two descriptions. On a Riemannian manifold $\mathcal{M}$,

$$\mathrm{Exp}_{R_a} : T_{R_a}\mathcal{M} \longrightarrow \mathcal{M} \tag{2.17}$$

maps a tangent vector v to the point reached by following the corresponding geodesic from R_a. Conversely,

$$\mathrm{Log}_{R_a} : \mathcal{M} \longrightarrow T_{R_a}\mathcal{M} \tag{2.18}$$

maps a point R_b to the tangent vector at R_a pointing along the shortest geodesic towards it. Where this geodesic is unique,

$$R_b = \mathrm{Exp}_{R_a}(\mathrm{Log}_{R_a}(R_b)). \tag{2.19}$$

Figure 2.3 visualises this. The Euclidean operation $x_b - x_a$ produces a displacement in the same ambient vector space. On a manifold, subtraction is replaced by $\mathrm{Log}_{R_a}(R_b)$, which represents

the displacement in the local tangent space $T_{R_a}\mathcal{M}$. The map Exp_{R_a} then returns such a tangent displacement to the manifold [23].

The rotational substitutes for subtraction and addition. For Euclidean points, the displacement from x_a to x_b satisfies

$$x_a + (x_b - x_a) = x_b. \tag{2.20}$$

For rotations let ΔR denote the relative rotation that carries R_a to R_b:

$$R_a \Delta R = R_b \quad \Longrightarrow \quad \Delta R = R_a^\top R_b, \tag{2.21}$$

where $R_a^{-1} = R_a^\top$. Thus $R_a^\top R_b$ is the relative rotation expressed in the coordinate convention obtained by right-composing an update with R_a, that is multiplying R_a by it recovers R_b.

Applying the logarithmic map at the identity moves this relative rotation into the linear tangent space,

$$\Omega = \mathrm{Log}\Big(R_a^\top R_b\Big) \in T_I\mathrm{SO}(3) = \mathfrak{so}(3). \tag{2.22}$$

The matrix Ω gives the direction and magnitude of the shortest rotational motion from R_a to R_b, expressed at the identity. Hence

$$x_b - x_a \qquad \longleftrightarrow \qquad \mathrm{Log}(R_a^\top R_b) \tag{2.23}$$

is the correspondence between Euclidean and rotational displacement.

The inverse operation follows immediately:

$$\begin{aligned} \mathrm{Exp}(\Omega) &= R_a^\top R_b, \\ R_a \,\mathrm{Exp}(\Omega) &= R_b, \end{aligned} \tag{2.24}$$

or

$$R_b = R_a \,\mathrm{Exp}\Big(\mathrm{Log}(R_a^\top R_b)\Big). \tag{2.25}$$

This is the rotational counterpart of $x_b = x_a + (x_b - x_a)$.

On SO(3), the exponential map at the identity is the matrix exponential. For $\widehat{\omega} \in \mathfrak{so}(3)$, Rodrigues' formula gives

$$\mathrm{Exp}(\widehat{\omega}) = I + \frac{\sin\theta}{\theta}\widehat{\omega} + \frac{1-\cos\theta}{\theta^2}\widehat{\omega}^2, \qquad \theta = \|\omega\|, \tag{2.26}$$

while for $0 < \theta < \pi$ the logarithmic map is

$$\mathrm{Log}(R) = \frac{\theta}{2\sin\theta}(R - R^\top), \qquad \theta = \arccos\!\left(\frac{\mathrm{tr}\,R - 1}{2}\right). \tag{2.27}$$

Thus Log converts a finite relative rotation into an element of $T_I\mathrm{SO}(3) = \mathfrak{so}(3)$, and Exp converts it back into a valid relative rotation [21]. The corresponding tangent displacement at a general base point R_a is obtained by left translation,

$$\mathrm{Log}_{R_a}(R_b) = R_a \,\mathrm{Log}(R_a^\top R_b) \in T_{R_a}\mathrm{SO}(3). \tag{2.28}$$

This identity will connect in Chapter 3 the general Riemannian flow-matching notation $\mathrm{Log}_x(y)$ directly to the matrix expression used on SO(3) and will be explained there in more detail.

Under the standard bi-invariant metric, tangent vectors $RA, RB \in T_R\mathrm{SO}(3)$ with $A, B \in \mathfrak{so}(3)$ have inner product

$$\langle RA, RB\rangle_R = \frac{1}{2}\,\mathrm{tr}(A^\top B). \tag{2.29}$$

Accordingly, define the induced norm on the Lie algebra by

$$\|A\|_{\mathfrak{so}(3)} = \sqrt{\frac{1}{2}\,\mathrm{tr}(A^\top A)} = \frac{1}{\sqrt{2}}\|A\|_F, \qquad A \in \mathfrak{so}(3). \tag{2.30}$$

The corresponding norm of a tangent vector is

$$\|RA\|_R = \|A\|_{\mathfrak{so}(3)}. \tag{2.31}$$

For $A = \widehat{\omega}$, this convention gives $\|A\|_{\mathfrak{so}(3)} = \|\omega\|_2$. Thus the geodesic distance between two rotations is

$$d_{\mathrm{SO}(3)}(R_a, R_b) = \left\|\mathrm{Log}(R_a^\top R_b)\right\|_{\mathfrak{so}(3)} = \theta \in [0, \pi], \tag{2.32}$$

the angle of the smallest rotation connecting the two orientations [21, 23, 26].

These operations provide the geometry needed to compare and move between rotations. SigmaDock additionally requires a rotational counterpart of the Gaussian forward diffusion, a process that progressively randomises an orientation without leaving SO(3).

2.3.1 From Gaussian noise to rotational diffusion

The translational construction of Section 2.2 followed three steps. A forward diffusion progressively corrupted the clean state, reversing that diffusion required the score of its noisy marginal distribution, and denoising score matching made this score learnable from known conditional transition densities. SigmaDock applies the same principle to orientations. The difference is geometric: both the forward noising process and the score must now respect SO(3).

For translations, Brownian motion produces Gaussian transition kernels and the forward process approaches a Gaussian reference distribution. On SO(3), the corresponding forward process is Brownian motion on the rotation group. Starting from a clean orientation R_0, it produces a random orientation R_t whose conditional transition density is the heat kernel on SO(3),

commonly called the isotropic Gaussian

$$q_{t|0}(R_t \mid R_0) = \mathrm{IGSO}(3)(R_t; R_0, \sigma_t^2). \tag{2.33}$$

The parameter σ_t controls how far the orientation has diffused from R_0. Because the process is intrinsic to the rotation group, every intermediate state remains a valid rotation [11, 26].

The limiting distribution of this forward process is the normalised Haar measure $\mathcal{U}_{\mathrm{SO}(3)}$. It is invariant under multiplication by any fixed rotation,

$$\mathcal{U}_{\mathrm{SO}(3)}(QA) = \mathcal{U}_{\mathrm{SO}(3)}(AQ) = \mathcal{U}_{\mathrm{SO}(3)}(A), \qquad Q \in \mathrm{SO}(3), \tag{2.34}$$

so no orientation or coordinate frame is preferred [5]. The Haar measure therefore represents a uniform orientation distribution.

The intermediate transition densities describe how this information is lost. By isotropy, the density of R_t conditional on R_0 cannot depend on a preferred rotational axis. It therefore depends only on their geodesic separation under the metric introduced above,

$$\omega_t = d_{\mathrm{SO}(3)}(R_0, R_t) = \left\| \mathrm{Log}(R_0^\top R_t) \right\|_{\mathfrak{so}(3)}. \tag{2.35}$$

Thus $\omega_t \in [0, \pi]$ is the angle of the relative rotation $R_0^\top R_t$. Relative to Haar measure, the corresponding heat kernel has the spectral expansion

$$f_{\sigma_t}(\omega_t) = \sum_{\ell=0}^{\infty} (2\ell+1) \frac{\sin\left((\ell + \frac{1}{2})\omega_t\right)}{\sin(\omega_t/2)} \exp\left(-\tfrac{1}{2}\ell(\ell+1)\sigma_t^2\right). \tag{2.36}$$

For small σ_t, this density is concentrated near R_0. As σ_t increases, it spreads over the group, and in the limit $\sigma_t \to \infty$ all terms with $\ell \geq 1$ vanish so that $f_{\sigma_t} \to 1$, the Haar-uniform density [21]. The forward rotational process can therefore be summarised as

$$R_0 \quad \xrightarrow{\text{Brownian motion on SO(3)}} \quad R_t \sim \mathrm{IGSO}(3)(R_0, \sigma_t^2) \quad \xrightarrow[\sigma_t \to \infty]{} \quad \mathcal{U}_{\mathrm{SO}(3)}. \tag{2.37}$$

SigmaDock uses the rotational noise schedule

$$\sigma_t = \log\left((1-t)e^{\sigma_{\min}} + te^{\sigma_{\max}}\right), \qquad \sigma_{\min} = 0.1, \qquad \sigma_{\max} = 1.5. \tag{2.38}$$

As t increases, the conditional kernels become progressively broader. Because the implemented schedule terminates at the finite value $\sigma_{\max} = 1.5$ rather than at $\sigma_t \to \infty$, SigmaDock does not reach exact Haar noise.

Generation now requires reversing this corruption process. The Euclidean reverse-time result of Theorem 2.2 has a direct Riemannian analogue. The time reversal of a diffusion on a Riemannian manifold is again a diffusion, and its reverse drift depends on the Riemannian score of the noisy marginal distribution.

Theorem 2.4 (Time-reversed diffusion: Adapted from 7, Theorem 3.1)**.** *Let $(X_t)_{t\in[0,T]}$ be a diffusion on a Riemannian manifold $\mathcal{M}$ satisfying*

$$\mathrm{d}X_t = b(X_t)\,\mathrm{d}t + \mathrm{d}B_t^{\mathcal{M}}, \tag{2.39}$$

where $B_t^{\mathcal{M}}$ is Brownian motion on $\mathcal{M}$. Let $\mathbb{P}$ denote the path distribution of $(X_t)_{t\in[0,T]}$ and $\mathbb{Q}$ that of $B_t^{\mathcal{M}}$. Suppose that $\mathrm{KL}(\mathbb{P}\|\mathbb{Q}) < \infty$ and that each marginal distribution of X_t admits a smooth positive density q_t with respect to the Riemannian volume measure. Then the time-reversed process $\overleftarrow{X}_t = X_{T-t}$ is again a diffusion and satisfies

$$\mathrm{d}\overleftarrow{X}_t = \left[-b(\overleftarrow{X}_t) + \nabla \log q_{T-t}(\overleftarrow{X}_t)\right]\mathrm{d}t + \mathrm{d}\overleftarrow{B}_t^{\mathcal{M}}, \tag{2.40}$$

where ∇ denotes the Riemannian gradient.

For rotational diffusion, the unknown quantity required to reverse the forward process is therefore the score of the noisy rotational marginal,

$$s_t(R) = \nabla_R \log q_t(R) \in T_R\mathrm{SO}(3). \tag{2.41}$$

As in the Euclidean case, this marginal score cannot be evaluated directly. The forward process does, however, provide the conditional transition density. For a clean orientation R_0 and its noisy version R_t,

$$q_{t|0}(R_t \mid R_0) = f_{\sigma_t}(\omega_t), \qquad \omega_t = \left\|\mathrm{Log}(R_0^\top R_t)\right\|_{\mathfrak{so}(3)}. \tag{2.42}$$

Since this IGSO(3) heat kernel is known, its conditional score

$$s_{t|0}(R_t \mid R_0) = \nabla_{R_t} \log q_{t|0}(R_t \mid R_0) = \nabla_{R_t} \log f_{\sigma_t}(\omega_t) \tag{2.43}$$

is available as a training target. Riemannian denoising score matching provides the link between this tractable conditional score and the unknown marginal score required by the reverse diffusion.

Theorem 2.5 (Riemannian denoising score matching: Adapted from 7, Section 3.4)**.** *For any positive weighting function $w(t)$, the minimiser of*

$$\mathbb{E}_{t\sim\mathcal{U}[0,T],\, R_0\sim p_{\mathrm{data}},\, R_t\sim q_{t|0}(\cdot|R_0)}\left[w(t)\left\|s_\theta^R(R_t,t) - \nabla_{R_t}\log q_{t|0}(R_t \mid R_0)\right\|_{R_t}^2\right] \tag{2.44}$$

is the marginal Riemannian score

$$s_t(R) = \nabla_R \log q_t(R),$$

up to sets of measure zero, provided the relevant expectations exist and the model class can represent the optimum.

The norm $\|\cdot\|_{R_t}$ in (2.44) is the tangent-space norm induced by the same bi-invariant metric.

In particular, for a tangent vector $R_t A \in T_{R_t}\mathrm{SO}(3)$,

$$\|R_t A\|_{R_t} = \|A\|_{\mathfrak{so}(3)}. \tag{2.45}$$

Thus the geodesic distance, the Riemannian gradient and the score-matching loss all use the same underlying geometry.

Training therefore never evaluates q_t itself. It samples R_t from the known IGSO(3) kernel around R_0 and regresses on its conditional score. Unlike the Gaussian case, the heat kernel and its derivative use an infinite spectral series, so SigmaDock truncates and tabulates them before training. These tables provide the rotational targets and noise-dependent score scaling.

2.4 SigmaDock on the full product space

The translational and rotational components can now be combined. Let

$$Z_t = (P_t, R_t) \in \left(\mathbb{R}^3\right)^N \times \mathrm{SO}(3)^N \tag{2.46}$$

denote the random positions and orientations of all fragments, with realisation

$$z_t = (p_t, R_t). \tag{2.47}$$

In SigmaDock, time runs from data to noise: $t = 0$ is the crystallographic pose, and increasing t progressively randomises both components.

The forward process factorises across the two geometric components:

$$\begin{aligned} &\text{translation:} && \text{Gaussian diffusion on } \mathbb{R}^{3N} && \longrightarrow\ \mathcal{N}(0, I_{3N}), \\ &\text{rotation:} && \text{Brownian diffusion on } \mathrm{SO}(3)^N && \longrightarrow\ \mathcal{U}_{\mathrm{SO}(3)}^{\otimes N}. \end{aligned} \tag{2.48}$$

In the infinite-noise limit, the stationary reference distribution is

$$q_\infty(z) = \mathcal{N}(0, I_{3N}) \otimes \mathcal{U}_{\mathrm{SO}(3)}^{\otimes N}. \tag{2.49}$$

The theoretical stationary law q_∞ in (2.49) must be distinguished from the terminal law used by the released SigmaDock implementation. Its rotational schedule ends at $\sigma_{\max} = 1.5$, so the forward kernel at $t = 1$ is finite-noise IGSO(3) rather than exact Haar. The released `sample_ref` routine samples from this $t = 1$ kernel, and the SigmaDock runs reproduced in this thesis therefore start from that finite-noise rotational reference.

This completes the theoretical diffusion construction that SigmaFlow will modify. SigmaDock defines a forward path from data towards a simple reference distribution and learns the score required to reverse it. SigmaFlow instead specifies a transport path directly and learns the velocity field that generates it.

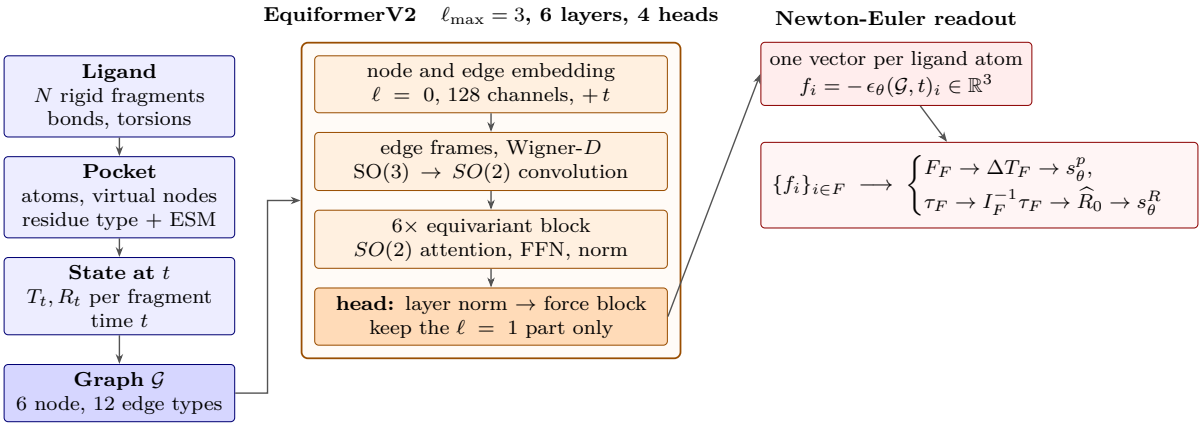


Figure 2.4: Left: ligand, pocket and the diffused state at time t form a heterogeneous graph. Centre: EquiformerV2 propagates features of degree $\ell \leq 3$. Right: the output block yields one equivariant $\ell = 1$ vector per ligand atom, which the Newton-Euler readout converts into translational and rotational quantities for each rigid fragment.

2.5 The inherited backbone and Newton-Euler readout

The model that learns to predict the score is an EquiformerV2 [12], augmented by SigmaDock with virtual nodes and edges and structural-role-dependent molecular featurisation [21]. SigmaFlow inherits this backbone, so the discussion here focuses on its output and rigid-body readout. A global rotation of the protein-ligand complex should rotate a predicted geometric update by the same amount. EquiformerV2 enforces this through steerable features transforming under irreducible representations of SO(3), with $\ell = 0$ corresponding to scalars and $\ell = 1$ to vectors. The output block retains one $\ell = 1$ feature per ligand atom,

$$f_i \in \mathbb{R}^3, \qquad i = 1, \ldots, N_a. \tag{2.50}$$

These vectors are not fragment scores but an intermediate equivariant field interpreted through rigid-body mechanics.

This construction also avoids dependence on arbitrary local fragment axes. Rather than predicting a rotation coordinate directly, SigmaDock adapts the equivariant Newton-Euler head of Jin et al. [10], which constructs rigid-body updates independently of the arbitrary choice of local fragment coordinates [21, Theorem 2].

For a fragment F with global atom coordinates r_i and centroid c_F, the field is reduced to a total pseudo-force and pseudo-torque,

$$F_F = \sum_{i \in F} f_i, \qquad \tau_F = \sum_{i \in F} (r_i - c_F) \times f_i. \tag{2.51}$$

Translation therefore uses the zeroth spatial moment of the field, while rotation uses its first moment about the fragment centre. Since

$$\sum_{i \in F} (r_i - c_F) = 0, \tag{2.52}$$

a constant field can produce net force without torque, while a varying field whose vectors sum to zero can produce torque without net force.

Assigning unit mass to every atom gives $M_F = |F|$ and inertia tensor

$$I_F = \sum_{i \in F} \left[\|r_i - c_F\|_2^2 I_3 - (r_i - c_F)(r_i - c_F)^\top \right], \tag{2.53}$$

with small eigenvalues clamped before inversion. The rigid-body increments are

$$\Delta T_F = \frac{F_F}{M_F}, \qquad \omega_F = I_F^{-1} \tau_F. \tag{2.54}$$

Thus ΔT_F is the mean atom-wise vector, while ω_F also depends on fragment geometry through I_F^{-1}. Since r_i, τ_F and I_F are expressed in global coordinates, ω_F is a *spatial-frame* angular quantity. This distinction becomes important for SigmaFlow in Section 3.5.

The translational output can be converted directly into the score through (2.13).The rotational case requires an additional step because ω_F is not itself the rotational score required by the reverse SDE. Instead, SigmaDock interprets it as a denoising prediction from which an estimate of the clean orientation is constructed,

$$\widehat{R}_0 = \mathrm{Exp}\big(\lambda_{\mathrm{SO}(3)}(t)\widehat{\omega}_F\big) R_t, \tag{2.55}$$

where $\lambda_{\mathrm{SO}(3)}(t)$ is the per-coordinate root-mean-square magnitude of the true conditional rotational score at the current noise level, tabulated once before training. It is the rotational counterpart of $\lambda_{\mathbb{R}^3}(t) = 1/\sigma_t$. The exponential map converts the scaled angular prediction into a valid rotation, while left multiplication applies it as a spatial-frame update to the current orientation R_t.

The estimated clean orientation $\widehat{R}_0$ must then be converted into the score required by the reverse diffusion. This is possible because the conditional IGSO(3) transition density is known. Its density depends on R_t only through the geodesic distance from the estimated clean orientation,

$$\rho = \left\| \mathrm{Log}\big(\widehat{R}_0^\top R_t\big) \right\|_{\mathfrak{so}(3)}. \tag{2.56}$$

Hence the conditional log-density has the form $\log f_{\sigma_t}(\rho)$. Applying the chain rule gives

$$\begin{aligned} \nabla_{R_t} \log f_{\sigma_t}(\rho) &= \frac{\partial}{\partial \rho} \log f_{\sigma_t}(\rho)\, \nabla_{R_t} \rho, \\ \nabla_{R_t} \rho &= R_t \frac{\mathrm{Log}(\widehat{R}_0^\top R_t)}{\rho}. \end{aligned} \tag{2.57}$$

Combining the two therefore yields the rotational score prediction

$$s_\theta^R = R_t \frac{\mathrm{Log}(\widehat{R}_0^\top R_t)}{\rho} \frac{\partial}{\partial \rho} \log f_{\sigma_t}(\rho), \qquad \rho = \left\| \mathrm{Log}(\widehat{R}_0^\top R_t) \right\|_{\mathfrak{so}(3)}. \tag{2.58}$$

Hence the rotational readout has two conceptually distinct stages. The learned Newton-Euler head produces ω_F and thereby an estimate $\widehat{R}_0$ of the clean orientation. The known geometry of the IGSO(3) diffusion then converts the displacement from $\widehat{R}_0$ to R_t into the rotational score required by the reverse SDE. This distinction is important for SigmaFlow, which predicts a velocity directly and therefore does not require this score-conversion step.

The resulting SigmaDock objective combines the translational and rotational denoising score-matching losses across all fragments,

$$\begin{aligned} \mathcal{L}_{\text{SigmaDock}} = \mathbb{E}_{t, Z_0, Z_t} \Bigg[\frac{1}{N} \sum_{F=1}^{N} \Bigg(& w_p \lambda_{\mathbb{R}^3}(t)^{-2} \left\| s_\theta^p(Z_t, t) - \nabla_{P_t} \log q_{t|0}(P_t \mid P_0) \right\|_2^2 \\ & + w_R \lambda_{\mathrm{SO}(3)}(t)^{-2} \left\| s_\theta^R(Z_t, t) - \nabla_{R_t} \log q_{t|0}(R_t \mid R_0) \right\|_F^2 \Bigg) \Bigg], \end{aligned} \tag{2.59}$$

where

$$t \sim \mathcal{U}[0,1],\ Z_0 = (P_0, R_0) \sim p_{\text{data}},\ Z_t = (P_t, R_t) \sim q_{t|0}(\cdot \mid Z_0),\ w_p = 2,\ w_R = 0.5. \tag{2.60}$$

The time distribution, factor weights and averaging over fragments are retained by SigmaFlow and therefore cannot explain performance differences between the two mechanisms.

2.6 Sampling, ranking and evaluation

The generative mechanism produces candidate poses, but selecting and evaluating them are different problems. SigmaDock does not train a confidence model. Instead, independently generated samples are ranked by a fixed heuristic. Let $b_i \in \mathbb{R}$ be the Vinardo pseudo binding energy [22] evaluated by GNINA [18], and let $p_i \in [0,1]$ be the fraction of 5 selected PoseBusters stereochemical checks passed by sample i: bond lengths, bond angles, tetrahedral chirality, internal steric clash and minimum distance to the protein. SigmaDock ranks by

$$s_i = -\, b_i p_i^\beta, \qquad \beta = 4, \tag{2.61}$$

with larger s_i preferred [21, App. F.2].

Accuracy. An atom-indexed RMSD would penalise chemically identical relabellings, such as interchanging symmetric atoms in a phenyl ring. The reported displacement is therefore minimised over graph automorphisms,

$$\mathrm{RMSD}(\hat{x}, x) = \min_{\pi \in \mathrm{Aut}(G)} \sqrt{\frac{1}{N_a} \sum_{i=1}^{N_a} \|\hat{x}_{\pi(i)} - x_i\|_2^2}. \tag{2.62}$$

This is the symmetry-corrected definition used by SigmaDock [19, 21].

Physical plausibility. PoseBusters is run in its `redock` configuration. Its checks cover intrinsic ligand geometry, including bond lengths, bond angles, ring planarity, stereochemistry and internal clashes, together with receptor-facing checks such as protein distance and volume overlap [2]. The main success criterion is

$$\mathrm{RMSD} < 2\,\text{Å} \qquad \text{and} \qquad \text{PoseBusters-valid.} \tag{2.63}$$

This conjunction matters because a pose can be close to the crystallographic coordinates while being chemically impossible.

Conformer source. The fixed local fragment geometry can be extracted from the bound crystal structure or from a newly generated conformer. Every principal result in this thesis uses the generated conformer. This is the realistic setting for an unseen ligand, extracting fragment orientations from the crystallographic pose would provide information that is unavailable at deployment.

Benchmarks and reporting. The main test sets are PoseBusters version 2 (PB308) and the Astex Diverse Set (AX85), matching the benchmarks reported by SigmaDock [21]. PB308 contains 308 complexes and provides a temporal split relative to the PDBBind training data. AX85 contains 85 curated docking complexes [8].

For K generated poses per complex, three quantities are distinguished. The *per-draw* rate estimates the probability that a single generated pose satisfies (2.63), *Top-1* evaluates the pose selected by a specified ranking rule and the *oracle* rate records whether at least one of the K generated poses succeeds. The oracle therefore measures the quality of the generated set independently of ranking, while Top-1 measures generation and selection together.

Chapter 3

From Scores to Velocities: Riemannian Flow Matching

SigmaDock and SigmaFlow act on the same fragment state space but learn different objects. SigmaDock predicts a *score field*, the local gradient of a noisy log-density. SigmaFlow predicts a *velocity field*, which moves an intermediate state under deterministic transport from a simple source to the data distribution.

The distinction is not merely stochastic versus deterministic dynamics. Scores and velocities can parameterise the same probability path, SigmaFlow changes the path itself. This chapter develops flow matching in Euclidean space and then extends it to rotations using only the geometry from Section 2.3. The aim is to make explicit which theoretical changes SigmaFlow introduces and why they may permit coarser numerical integration. The following presentation follows Holderrieth and Erives [9].

3.1 Flow matching in Euclidean space

A flow model generates data by transporting a random initial point along an ordinary differential equation. Let

$$X_0 \sim p_{\text{init}} \tag{3.1}$$

be drawn from an easily sampled source distribution and let a time-dependent vector field u_t define

$$\frac{\mathrm{d}}{\mathrm{d}t} X_t = u_t(X_t). \tag{3.2}$$

For any fixed initial state the ODE is deterministic, randomness enters only through X_0. The generative objective is to learn a field whose endpoint satisfies

$$X_0 \sim p_{\text{init}} \quad \Longrightarrow \quad X_1 \sim p_{\text{data}}. \tag{3.3}$$

A vector field can be read literally as a velocity map. At every time t and location x, $u_t(x)$ says in which direction a particle at x should move and how quickly. If the learned field is correct,

repeatedly following these local velocities transforms the simple source distribution into the data distribution.

Time convention. This direction is opposite to SigmaDock's *forward* diffusion. SigmaDock assigns $t = 0$ to the clean crystallographic pose and increases t towards noise. Throughout flow matching in this thesis,

$$t = 0 : \text{ source}, \qquad t = 1 : \text{ data}. \tag{3.4}$$

3.2 Conditional paths and the marginal vector field

Flow matching begins not with the field itself but with a *probability path*. For a training example $z \sim p_{\text{data}}$, choose a family of conditional densities $p_t(\cdot \mid z)$ satisfying

$$p_0(\cdot \mid z) = p_{\text{init}}, \qquad p_1(\cdot \mid z) = \delta_z. \tag{3.5}$$

Conditional on one particular target z, the path therefore evolves from the same source distribution to a point mass on that example.

For each data point $z \in \mathbb{R}^d$, let $u_t^{\text{target}}(\cdot \mid z)$ denote a *conditional vector field*. This is any vector field whose ODE generates the conditional probability path $p_t(\cdot \mid z)$, i.e. such that

$$X_0 \sim p_{\text{init}}, \qquad \frac{\mathrm{d}}{\mathrm{d}t} X_t = u_t^{\text{target}}(X_t \mid z) \quad \Longrightarrow \quad X_t \sim p_t(\cdot \mid z), \qquad 0 \le t \le 1. \tag{3.6}$$

Such a conditional vector field can often be obtained analytically. The key result is that these conditional paths and fields are sufficient to construct the marginal vector field required to sample from p_{data}.

Theorem 3.1 (Marginalisation trick: 9, Theorem 9)**.** *Let $u_t^{\text{target}}(x \mid z)$ be a conditional vector field generating the conditional probability path $p_t(\cdot \mid z)$. Define the marginal vector field by*

$$u_t^{\text{target}}(x) = \int u_t^{\text{target}}(x \mid z) \, \frac{p_t(x \mid z)\, p_{\text{data}}(z)}{p_t(x)} \, \mathrm{d}z. \tag{3.7}$$

Then u_t^{target} generates the marginal probability path p_t. Equivalently, if

$$X_0 \sim p_{\text{init}}, \qquad \frac{\mathrm{d}}{\mathrm{d}t} X_t = u_t^{\text{target}}(X_t), \tag{3.8}$$

then

$$X_t \sim p_t, \qquad 0 \le t \le 1.$$

In particular, $X_1 \sim p_{\text{data}}$.

The value of the theorem is practical. Designing a transport between two full distributions would be difficult. Designing a transport from a random source point towards one observed training example is easy. If each conditional transport reaches its chosen training example, the

marginalised field transports the complete source distribution to the complete data distribution.

3.3 Learning the field by conditional flow matching

The marginal field in (3.7) is exactly the field required for generation, but it contains the unknown data distribution and thus cannot be evaluated directly. Were it available, one would minimise

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],\ X_t\sim p_t}\left[\left\|v_\theta(t,X_t) - u_t^{\mathrm{target}}(X_t)\right\|^2\right]. \tag{3.9}$$

Flow matching avoids that intractable target. Training instead samples a data example z, samples an intermediate state from its conditional path, and regresses onto the corresponding *conditional* velocity,

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],\ z\sim p_{\mathrm{data}},\ X_t\sim p_t(\cdot|z)}\left[\left\|v_\theta(t,X_t) - u_t^{\mathrm{target}}(X_t \mid z)\right\|^2\right]. \tag{3.10}$$

The central result of conditional flow matching is that this tractable loss has the same parameter gradients as the marginal objective:

Theorem 3.2 (Conditional Flow Matching: 9, Theorem 12)**.** *The marginal and conditional flow matching losses differ only by a constant independent of θ,*

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathcal{L}_{\mathrm{CFM}}(\theta) + C, \tag{3.11}$$

and therefore

$$\nabla_\theta\mathcal{L}_{\mathrm{FM}}(\theta) = \nabla_\theta\mathcal{L}_{\mathrm{CFM}}(\theta). \tag{3.12}$$

For a sufficiently expressive model, the minimiser of $\mathcal{L}_{\mathrm{CFM}}$ is the marginal vector field u_t^{target}.

This is also what makes flow matching useful at scale. Training requires no ODE integration. One samples a time, a data point and a conditional intermediate state, then evaluates an ordinary regression loss. The differential equation appears only when generating a sample after training.

For Euclidean data, a particularly elegant construction is obtained by drawing $x_0 \sim p_{\mathrm{init}}$ and connecting it directly to a data point x_1 by the straight-line trajectory

$$x_t = (1-t)x_0 + tx_1. \tag{3.13}$$

This trajectory implicitly defines a conditional probability path. Indeed, if $X_0 \sim p_{\mathrm{init}}$ and $X_t = (1-t)X_0 + tx_1$, then the distribution of X_t is, by definition, $p_t(\cdot \mid x_1)$. At $t = 0$ this distribution is p_{init}, while at $t = 1$ every trajectory reaches x_1, so that $p_1(\cdot \mid x_1) = \delta_{x_1}$. The density of this path therefore need not be evaluated explicitly.

The conditional vector field is obtained directly from the velocity of these trajectories. Differen-

tiating (3.13) with respect to time gives

$$u_t^{\text{target}}(x_t \mid x_1) = \frac{\mathrm{d}}{\mathrm{d}t} x_t = \frac{x_1 - x_t}{1 - t} = x_1 - x_0. \tag{3.14}$$

SigmaFlow uses this construction for fragment translations. In contrast to the variance-preserving diffusion path of SigmaDock, its conditional trajectories are straight and have constant velocity. This motivates the hypothesis that the resulting transport can be integrated accurately with fewer numerical steps, an empirical question tested in Section 6.2.

3.4 From straight lines to rotational geodesics

Everything above was written in $\mathbb{R}^d$. For translations this is already the construction SigmaFlow needs. For rotations, however, the straight-line interpolant in (3.13) is not available. Rotation matrices cannot be interpolated by ordinary addition without leaving SO(3), and the displacement $x_1 - x_0$ has no direct analogue between two rotations. The geometric objects introduced in Section 2.3 provide the corresponding replacements. The following presentation follows Solà et al. [23] and Chen and Lipman [3]

The construction follows the same logic as in Euclidean space. Fix a target point x_1 and draw $X_0 \sim p_{\text{init}}$. In $\mathbb{R}^d$, we defined a trajectory from X_0 to x_1 by straight-line interpolation. On a Riemannian manifold $(\mathcal{M}, g)$, the natural replacement is the shortest geodesic,

$$X_t = \operatorname{Exp}_{X_0}(t \operatorname{Log}_{X_0}(x_1)). \tag{3.15}$$

Here $\operatorname{Log}_{X_0}(x_1) \in T_{X_0}\mathcal{M}$ is the tangent vector at the source point that points towards x_1 along the shortest geodesic. Multiplying this vector by t moves a fraction t of the geodesic distance, and the exponential map returns the resulting tangent vector to the manifold. Consequently, $X_t = X_0$ at $t = 0$ and $X_t = x_1$ at $t = 1$, wherever the chosen shortest geodesic is unique.

As in the Euclidean construction, this interpolation rule implicitly defines a conditional probability path. For every t, let $p_t(\cdot \mid x_1)$ denote the distribution of X_t induced by the random source point X_0. At $t = 0$ this is the source distribution, while at $t = 1$ every trajectory reaches x_1 and hence $p_1(\cdot \mid x_1) = \delta_{x_1}$. The conditional density itself need not be evaluated explicitly. Instead, flow matching only requires the velocity of the trajectories that generate it.

For the geodesic in (3.15), the velocity at the current point x_t can be written as

$$u_t(x_t \mid x_1) = \frac{\operatorname{Log}_{x_t}(x_1)}{1 - t}. \tag{3.16}$$

The logarithmic map $\operatorname{Log}_{x_t}(x_1)$ is the tangent vector at x_t pointing along the remaining geodesic towards x_1. Dividing by the remaining time $1 - t$ therefore gives the velocity required to reach x_1 at $t = 1$ [3]. This is the geometric analogue of $(x_1 - x_t)/(1 - t)$ in (3.14) and will be explicitly derived for SO(3).

The conditional flow matching principle itself is unchanged. The difference is that velocities now belong to the tangent space $T_{x_t}\mathcal{M}$, so the prediction and target must be compared using the metric at the same point. The Riemannian conditional flow matching objective is therefore [3]

$$\mathcal{L}_{\mathrm{RCFM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],\, x_1\sim p_{\mathrm{data}},\, x_t\sim p_t(\cdot|x_1)} \left[\left\| v_\theta(t,x_t) - u_t(x_t \mid x_1) \right\|^2_{g(x_t)} \right]. \tag{3.17}$$

For SO(3), this general construction becomes explicit. Let R_0 be a source rotation and R_1 the crystallographic orientation. On SO(3), tangent spaces are related to the tangent space at the identity by

$$T_R\mathrm{SO}(3) = \{R\Omega : \Omega \in \mathfrak{so}(3)\}, \qquad \mathfrak{so}(3) = T_I\mathrm{SO}(3). \tag{3.18}$$

Indeed, multiplying a tangent vector $R\Omega \in T_R\mathrm{SO}(3)$ from the left by $R^{-1} = R^\top$ removes the current rotation and recovers $\Omega \in \mathfrak{so}(3)$. This identifies each tangent space $T_R\mathrm{SO}(3)$ with the common tangent space at the identity and is called left trivialisation.

Applying this identification to the displacement from R_0 to R_1 gives

$$\Omega = R_0^\top \operatorname{Log}_{R_0}(R_1) = \operatorname{Log}(R_0^\top R_1) \in \mathfrak{so}(3). \tag{3.19}$$

The final equality follows from

$$R_1 = R_0 \operatorname{Exp}(\Omega) \qquad \Longleftrightarrow \qquad R_0^\top R_1 = \operatorname{Exp}(\Omega). \tag{3.20}$$

Thus $\Omega = \operatorname{Log}(R_0^\top R_1)$ represents the Riemannian displacement from R_0 to R_1 in the fixed tangent space at the identity. It plays the same role as $x_1 - x_0$ in Euclidean space, encoding the direction and distance required to move from the source orientation to the target along their shortest geodesic [3].

To move only a fraction t of this displacement, we scale Ω by t, map it back onto SO(3) with the exponential map, and compose the resulting relative rotation with the source rotation. This gives

$$R_t = R_0 \operatorname{Exp}(t\Omega) = R_0 \operatorname{Exp}\Big(t \operatorname{Log}(R_0^\top R_1)\Big). \tag{3.21}$$

At $t = 0$ this returns R_0. At $t = 1$, $\operatorname{Exp}(\Omega) = R_0^\top R_1$, so

$$R_0 \operatorname{Exp}(\Omega) = R_0 R_0^\top R_1 = R_1.$$

Equation (3.21) is therefore the rotational counterpart of the Euclidean interpolation

$$x_t = x_0 + t(x_1 - x_0).$$

Just as before, drawing R_0 from the source distribution and applying this interpolation rule induces the conditional probability path used for training.

It remains to obtain the conditional velocity associated with this trajectory. Since Ω is fixed once the source and target rotations have been chosen, differentiating $R_t = R_0 \operatorname{Exp}(t\Omega)$ only requires differentiating the matrix exponential. For constant Ω,

$$\frac{\mathrm{d}}{\mathrm{d}t} \operatorname{Exp}(t\Omega) = \operatorname{Exp}(t\Omega)\Omega. \tag{3.22}$$

Therefore,

$$\begin{aligned} \dot{R}_t &= R_0 \frac{\mathrm{d}}{\mathrm{d}t} \operatorname{Exp}(t\Omega) \\ &= R_0 \operatorname{Exp}(t\Omega)\Omega \\ &= R_t \Omega. \end{aligned} \tag{3.23}$$

Unlike the Euclidean velocity $\dot{x}_t$, the velocity $\dot{R}_t$ lies in the tangent space $T_{R_t}\mathrm{SO}(3)$, which changes along the geodesic. Using the left trivialisation introduced above expresses this velocity in the fixed tangent space at the identity:

$$R_t^\top \dot{R}_t = R_t^\top R_t \Omega = \Omega. \tag{3.24}$$

The identity-tangent-space representation of the velocity is therefore constant along the entire conditional geodesic, just as $x_1 - x_0$ is constant along the Euclidean straight-line trajectory.

The same velocity can equivalently be written in terms of the remaining displacement from the current rotation R_t to the target R_1. Starting from

$$R_t = R_0 \operatorname{Exp}(t\Omega),$$

taking the transpose gives

$$\begin{aligned} R_t^\top &= \left(R_0 \operatorname{Exp}(t\Omega)\right)^\top \\ &= \operatorname{Exp}(t\Omega)^\top R_0^\top \\ &= \operatorname{Exp}(-t\Omega) R_0^\top. \end{aligned} \tag{3.25}$$

The final equality follows because $\Omega \in \mathfrak{so}(3)$ is skew-symmetric, so $\Omega^\top = -\Omega$. Multiplying by the target rotation then gives

$$\begin{aligned} R_t^\top R_1 &= \operatorname{Exp}(-t\Omega) R_0^\top R_1 \\ &= \operatorname{Exp}(-t\Omega) \operatorname{Exp}(\Omega) \\ &= \operatorname{Exp}((1-t)\Omega). \end{aligned} \tag{3.26}$$

The second equality uses $R_0^\top R_1 = \operatorname{Exp}(\Omega)$ by the definition of Ω. The final equality holds because both exponentials are generated by the same matrix Ω and therefore commute. Along the chosen shortest geodesic,

$$\operatorname{Log}(R_t^\top R_1) = (1-t)\Omega. \tag{3.27}$$

Consequently, the conditional rotational velocity can be written as

$$u_t^{\text{rot}} = \Omega = \text{Log}(R_0^\top R_1) = \frac{\text{Log}(R_t^\top R_1)}{1-t}. \tag{3.28}$$

The Euclidean and rotational constructions are therefore the same recipe expressed in the geometry of their respective spaces:

$$\begin{array}{lll} & \text{trajectory} & \text{conditional velocity} \\ \mathbb{R}^3: & x_t = x_0 + t(x_1 - x_0) & u_t = x_1 - x_0, \\ \text{SO}(3): & R_t = R_0 \,\text{Exp}\big(t\,\text{Log}(R_0^\top R_1)\big) & u_t = \text{Log}(R_0^\top R_1). \end{array} \tag{3.29}$$

The Euclidean and rotational constructions now have the same form. A source point follows a prescribed path to the data point, and its derivative provides the conditional velocity. On SO(3), subtraction becomes the logarithmic map and straight lines become geodesics. No heat kernel, Brownian motion or rotational score is needed to construct the flow path.

There is, however, one geometric detail that becomes important when this target is connected to the inherited SigmaDock architecture. In (3.24) we represented the tangent velocity by $R_t^\top \dot{R}_t$. This is one of two common ways to represent rotational velocity. The distinction between them determines whether a predicted angular velocity acts on a rotation from the left or from the right.

3.5 Body frame and spatial frame

A tangent velocity $\dot{R} \in T_R\text{SO}(3)$ can be represented in the tangent space at the identity in two natural ways. The choice depends on the coordinate frame in which the angular velocity is expressed. Consider a fragment rotating inside a fixed protein pocket. Its motion may be described relative to axes attached to the fragment itself, called the *body frame*, or relative to axes fixed in the surrounding pocket, called the *spatial frame*. Both describe exactly the same physical motion, but their coordinate representations are generally different.

Because R maps body coordinates to spatial coordinates, multiplication order identifies the frame. In $R\Omega$, Ω acts before the frame rotation and is body-fixed. In WR, W acts after that rotation and is spatial.

The representation already used in (3.24) is the body-frame convention. For a differentiable rotation path $R(t)$, define

$$\Omega = R^\top \dot{R} \in \mathfrak{so}(3). \tag{3.30}$$

Multiplying by R from the left recovers the tangent velocity,

$$\dot{R} = R\Omega. \tag{3.31}$$

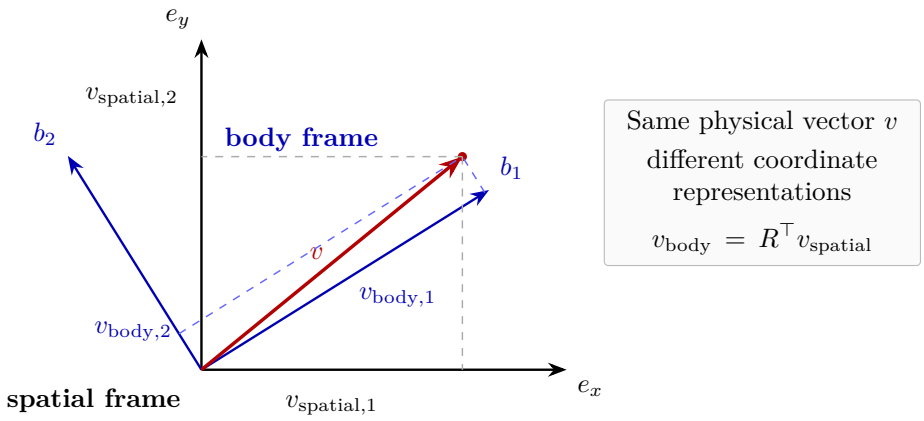


Figure 3.1: Schematic two-dimensional analogy of spatial and body-frame coordinates. The red arrow represents one physical vector v. The spatial frame is fixed to the environment, while the body frame is rotated with the fragment. The vector itself is unchanged, but its numerical components depend on the chosen basis. If R maps body-frame coordinates into the spatial frame, then $v_{\text{body}} = R^\top v_{\text{spatial}}$.

Reading the product from right to left, Ω acts on body-frame coordinates before R carries the resulting motion into spatial coordinates. The matrix Ω therefore describes the instantaneous angular velocity relative to axes that rotate together with the fragment.

Alternatively, the same tangent velocity can be represented in the spatial frame by

$$W = \dot{R}R^\top \in \mathfrak{so}(3). \tag{3.32}$$

To see why W again belongs to the Lie algebra, substitute $\dot{R} = R\Omega$ and use $\Omega^\top = -\Omega$:

$$\begin{aligned} W^\top &= \left(R\Omega R^\top\right)^\top \\ &= R\Omega^\top R^\top \\ &= -R\Omega R^\top \\ &= -W. \end{aligned} \tag{3.33}$$

Thus W is again skew-symmetric and therefore $W \in \mathfrak{so}(3)$. This gives

$$\dot{R} = WR. \tag{3.34}$$

Here the order is reversed. The rotation R first maps body-frame coordinates into the fixed spatial frame, after which W acts on the already rotated coordinates. Thus W describes the same instantaneous angular velocity relative to the fixed world coordinates of the protein pocket. The distinction can therefore be read directly from the multiplication order:

$$\underbrace{\dot{R} = R\Omega}_{\substack{\Omega \text{ acts in body coordinates} \\ \text{before the frame rotation}}}, \qquad \underbrace{\dot{R} = WR}_{\substack{W \text{ acts in spatial coordinates} \\ \text{after the frame rotation}}}. \tag{3.35}$$

The two matrices therefore encode the same physical tangent velocity in different coordinates.

Their relationship is precisely the corresponding change of frame. Substituting $\dot{R} = WR$ into the body-frame definition gives

$$\begin{aligned} \Omega &= R^\top \dot{R} \\ &= R^\top W R \\ &= \mathrm{Ad}_{R^{-1}}(W). \end{aligned} \tag{3.36}$$

Conversely,

$$W = R\Omega R^\top = \mathrm{Ad}_R(\Omega). \tag{3.37}$$

See Marsden and Ratiu [17] for a more detailed Lie-group formulation.

The matrices Ω and W are elements of $\mathfrak{so}(3)$. Using the identification between skew-symmetric matrices and three-vectors, their corresponding angular-velocity vectors may be written

$$\omega_{\text{body}} = \Omega^\vee, \qquad \omega_{\text{spatial}} = W^\vee. \tag{3.38}$$

Thus the distinction between body and spatial frame is not a distinction between two different rotations or physical motions. It is a distinction between two coordinate representations of the same tangent velocity.

SigmaFlow uses the body-frame convention. The rotational flow matching target derived in (3.28) is

$$u_t^{\text{rot}} = \mathrm{Log}(R_0^\top R_1) \in \mathfrak{so}(3), \tag{3.39}$$

which is precisely the body-frame representation obtained from $R_t^\top \dot{R}_t$. A numerical integration step therefore acts on the right,

$$R \longmapsto R\,\mathrm{Exp}\Big(\Delta t\, u_t^{\text{rot}}\Big). \tag{3.40}$$

This is consistent with the geodesic construction itself. If the exact conditional velocity is used for a small time increment Δt, then

$$R_t \longmapsto R_t\,\mathrm{Exp}(\Delta t\,\Omega),$$

which advances the rotation along the same geodesic defined in (3.21).

This convention becomes important because the inherited SigmaDock rigid-body readout naturally produces the opposite representation. Its torque and inertia are computed in world coordinates, so

$$\omega_F = I_F^{-1}\tau_F \tag{3.41}$$

is expressed in the spatial frame. This is consistent with SigmaDock's left-acting rotational update in (2.55). SigmaFlow, however, requires the body-frame velocity associated with (3.28). The inherited spatial-frame prediction must therefore be converted before it can be interpreted as the Riemannian flow matching velocity. In matrix form this is exactly the adjoint transformation

in (3.36). Chapter 4 gives the corresponding conversion explicitly for the inherited network readout.

The distinction is easy to miss in implementation because both coordinate representations ultimately contain three numbers. Using the wrong convention therefore leaves tensor shapes, network outputs and losses apparently well defined. Geometrically, however, the predicted angular velocity is then interpreted relative to the wrong coordinate axes. Keeping the frame convention explicit is therefore necessary to make the rotational target, network output and numerical integration rule describe the same motion.

3.6 The SigmaFlow probability path and objective

SigmaFlow now follows by applying the Euclidean construction to translations and the geodesic construction to rotations. Its source distribution is

$$p_0(z) = \mathcal{N}(0, I_{3N}) \otimes \mathcal{U}_{\mathrm{SO}(3)}^{\otimes N}, \tag{3.42}$$

which is the stationary limit (2.49) of SigmaDock's forward diffusion, although the released SigmaDock sampler itself uses the finite-noise rotational reference described in Section 2.4. For each training example, draw

$$x_0 \sim \mathcal{N}(0, I_3), \qquad R_0 \sim \mathcal{U}_{\mathrm{SO}(3)} \tag{3.43}$$

independently for each fragment. If (x_1, R_1) denotes its crystallographic translation and orientation, the conditional paths and targets are

$$x_t = (1-t)x_0 + tx_1, \qquad u_t^{\mathrm{trans}} = x_1 - x_0 = \frac{x_1 - x_t}{1-t}, \tag{3.44}$$

and

$$R_t = R_0 \operatorname{Exp}\Big(t \operatorname{Log}(R_0^\top R_1)\Big), \qquad u_t^{\mathrm{rot}} = \operatorname{Log}(R_0^\top R_1) = \frac{\operatorname{Log}(R_t^\top R_1)}{1-t}. \tag{3.45}$$

The two forms of each target are mathematically equal along the conditional path. The implementation uses the endpoint forms $x_1 - x_0$ and $\operatorname{Log}(R_0^\top R_1)$. They remain finite at $t = 1$ and, after the rotational velocity is expressed in the body frame, are constant along the conditional path. The apparently singular factors $1/(1-t)$ are therefore never required during target construction.

For N fragments the construction is componentwise because the model uses the direct-product geometry from Section 2.1. Writing the fragment index explicitly, the training loss is

$$\mathbb{E}_{t\sim\mathcal{U}[0,1],\, z_1\sim p_{\mathrm{data}}(\cdot|c),\, z_0\sim p_0}\left[\frac{1}{N}\sum_{i=1}^{N}\left(\lambda_{\mathrm{trans}}\big\|v_{\theta i}^{\mathrm{trans}} - u_{t,i}^{\mathrm{trans}}\big\|_2^2 + \lambda_{\mathrm{rot}}\big\|v_{\theta i}^{\mathrm{rot}} - u_{t,i}^{\mathrm{rot}}\big\|_F^2\right)\right]. \tag{3.46}$$

As in SigmaDock, the loss is averaged over fragments and retains the inherited relative weighting of translation and rotation, with $\lambda_{\mathrm{trans}} = 2$ and $\lambda_{\mathrm{rot}} = 0.5$. Equation (3.46) is simply the

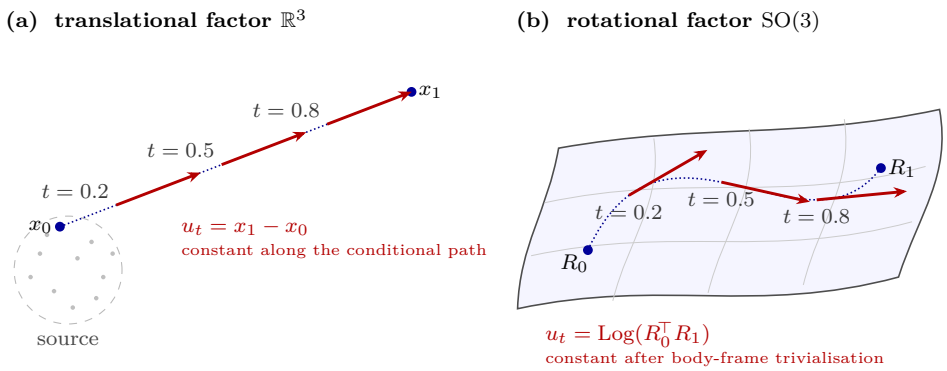


Figure 3.2: Conditional paths for one translational and one rotational fragment coordinate. **(a)** In $\mathbb{R}^3$, shown schematically in two dimensions, interpolation is linear and the target $x_1 - x_0$ is constant. **(b)** On SO(3), interpolation follows the shortest geodesic. The tangent arrow changes its location and physical orientation along the curved path, but its coordinates after body-frame trivialisation remain $\mathrm{Log}(R_0^\top R_1)$. Thus SigmaFlow uses the same conceptual target on both factors: the displacement from source to data expressed in the geometry of that factor. The surface is schematic; SO(3) itself is three-dimensional.

Riemannian CFM objective (3.17) written for the fragment product space.

The rotational loss uses the Frobenius norm on $\mathfrak{so}(3)$. For $A = \widehat{\omega}$,

$$\|A\|_F^2 = \mathrm{tr}(A^\top A) = 2\|\omega\|_2^2, \tag{3.47}$$

At inference the learned field is integrated with explicit Euler. Translation uses the ordinary Euclidean update, while rotation returns to the group with the exponential map,

$$x^{(k+1)} = x^{(k)} + v_\theta^{\mathrm{trans}} \Delta t, \qquad R^{(k+1)} = R^{(k)} \,\mathrm{Exp}\Big(v_\theta^{\mathrm{rot}} \Delta t\Big). \tag{3.48}$$

The right multiplication is the body-frame convention of Section 3.5. Because the conditional paths are straight on translations and shortest geodesics on rotations, they deliberately avoid unnecessary curvature in the target transport. Whether this translates into fewer network evaluations in the learned *marginal* flow is an important empirical question tested directly in Section 6.2.

Chapter 4

SigmaFlow-NE: The Minimal Conversion

4.1 Design principle and what actually changed

To isolate the generative mechanism, the implementation keeps all inherited components fixed unless flow matching requires a change. Table 4.1 summarises this partition.

Table 4.1: Partition of the system under the minimal-change constraint. Components marked identical were verified by file hash over the relevant source trees.

Component	Status	Note
EquiformerV2 backbone	identical	all backbone source files are hash-identical
Featurisation and Graph	identical	inherited
Data pipeline, fragmentation	identical	same conformers, same splits
Output block ($\ell = 1$ head)	identical	one field f_i, Eq. (2.50)
Newton-Euler reduction	identical	Eqs. (2.51)-(2.54)
Optimiser and schedule	identical	AdamW [16], same cosine schedule
Loss weights	identical	SigmaDock production values
Forward noising process	**removed**	replaced by an explicit probability path
Score rescaling by $\lambda(t)$	**removed**	score rescaling is no longer necessary and deleted
Probability path	**new**	Eqs. (3.44)-(3.45)
Training target	**new**	conditional velocity in place of score
Frame conversion	**new**	Eq. (4.2)
Sampler	**new**	deterministic Euler, Eq. (3.48)

4.2 The vector field readout

The network output and its Newton-Euler reduction are inherited exactly, so the per-fragment increments ΔT_F and $\widehat{\omega}_F$ of (2.54) are available unchanged. What differs is their interpretation. SigmaFlow reads them as a velocity, which requires no time-dependent rescaling, so the translational prediction is:

$$v_\theta^{\text{trans}} \;=\; \Delta T_F \;=\; \frac{1}{|F|}\sum_{i\in F} f_i. \tag{4.1}$$

The rotational prediction requires one further step. As noted in Section 3.5, $\widehat{\omega}_F$ is a spatial-frame quantity, whereas the target (3.45) and the integrator (3.48) use the body-frame convention. The prediction must therefore be expressed in the same frame before it can be compared with the target. Using a spatial-frame quantity directly would compare components expressed with respect to different coordinate axes. Applying the adjoint transformation of (3.36) gives the predicted rotational velocity:

$$v_\theta^{\mathrm{rot}} = R_t^\top \widehat{\omega}_F R_t. \tag{4.2}$$

Conjugating a skew-symmetric matrix by a rotation preserves skew-symmetry, so v_θ^{rot} remains a valid element of $\mathfrak{so}(3)$ and every downstream shape is unchanged.

Lastly note the implementation, the SLURM configuration of every run reported here, the sampling and evaluation chain, and the frozen result file from which all tables and figures in this thesis are regenerated are available at `https://github.com/jmuelleo/SigmaFlow`.

Chapter 5

SigmaFlow-TR: Translation and Rotation Specific Heads

5.1 Motivation

SigmaFlow-NE uses one atom-wise field for both channels. Translation reads its zeroth spatial moment, while rotation reads its first moment about the fragment centre. SigmaDock subsequently rescales these outputs separately as scores, but SigmaFlow interprets them directly as velocities. SigmaFlow-TR tests whether the two velocity targets benefit from separate output representations while retaining the shared EquiformerV2 trunk.

5.2 Construction

SigmaFlow-TR replaces the single output block by two blocks of identical configuration on a shared trunk, so the backbone up to the final attention block is untouched and only the readout is duplicated. Each block emits its own $\ell = 1$ field per atom, f_i^T and f_i^R. The Newton-Euler reduction is removed from the active path and replaced by mean pooling,

$$v_\theta^{\text{trans}} \;=\; \frac{1}{|F|}\sum_{i\in F} f_i^T, \qquad \bar{w}_F \;=\; \frac{1}{|F|}\sum_{i\in F} f_i^R, \qquad v_\theta^{\text{rot}} \;=\; R_t^\top \widehat{\bar{w}_F}\, R_t. \tag{5.1}$$

The translational expression is unchanged in form. The rotational expression differs substantively: torque and inverse inertia are gone, and the network emits an angular velocity directly. The frame conversion is retained unmodified, since it is required by the invariance of the target rather than by the particular readout. Mean pooling rather than summation is deliberate, since a sum would make the predicted velocity scale with fragment size and impose an uncontrolled size-dependent gain on the loss.

The cost is one additional attention block: 16,899,826 parameters versus 14,979,377, of which 1,920,449 belong to each of the two identically configured heads, an increase of 12.82%. Everything else, including the data pipeline, fragmentation, optimiser, schedule, loss weights, source

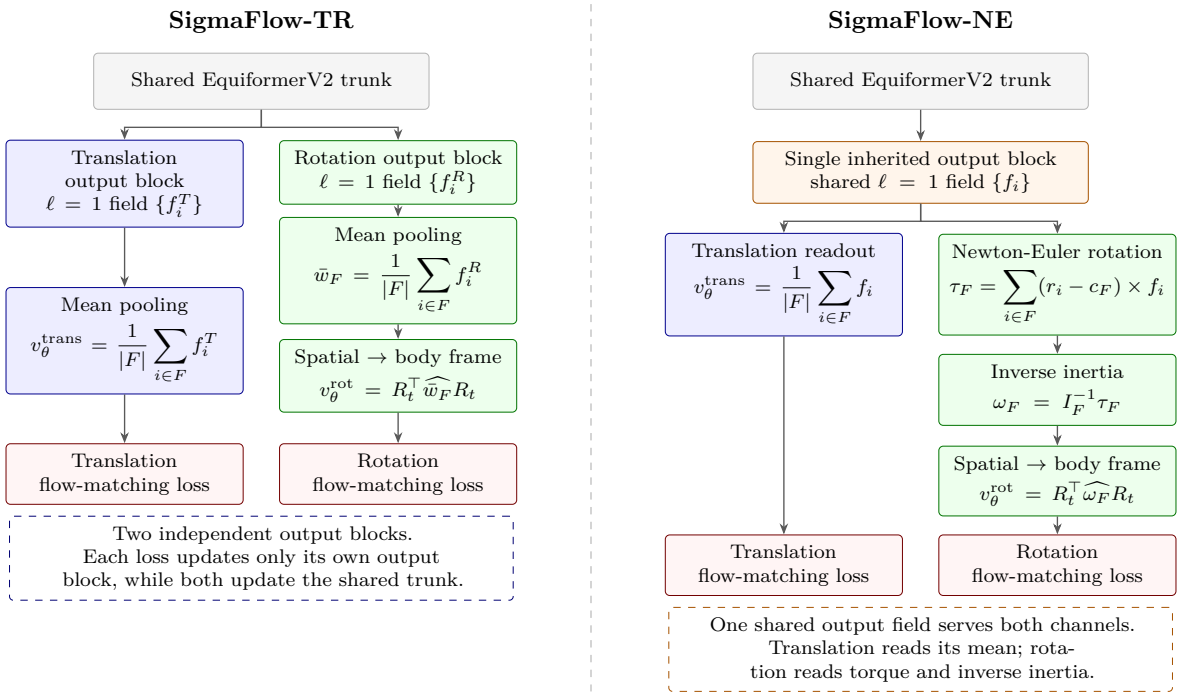


Figure 5.1: Conceptual comparison of the two SigmaFlow readouts.

distribution and sampler, is byte-identical to SigmaFlow-NE.

Chapter 6

Results

6.1 Experimental basis

The three models of Table 6.1 received the same wall-clock allocation on one L40S GPU each and differ only as recorded in Table 4.1. All models are trained on the general set of PDBbind v2020 [15], in the reprocessed R1 release (19,037 complexes) [20].

Table 6.1: The three training runs. Wall-clock is cumulative, since two models resumed after node failures and the successor allocations were shortened so that all three end within four minutes of each other. The horizon is the epoch count the annealing schedule was calibrated to, set per model from its measured throughput; all three take 594 optimiser steps per epoch at batch size 32.

model	Mechanism	Wall-clock	Horizon	Epoch reached	Schedule used
SigmaDock	diffusion	72:00:06	239	232	97.1%
SigmaFlow-NE	flow matching	72:00:03	255	232	91.0%
SigmaFlow-TR	flow matching	72:03:20	255	222	87.1%

6.1.1 The compute budget

Three descriptions must be distinguished when comparing these results with Prat et al. [21]. The training procedure reported in the paper, the configuration shipped with the released code, and the leaner setup used here. SigmaDock was reported to train for up to 256 epochs on four 80GB A100 GPUs under distributed data parallelism [21, App. E.3]. The experiments here instead use the reprocessed PDBbind v2020.R1 general set with a batch size of 32 on a single GPU. R1 was preferred because it reprocesses the remaining structures with the newer preparation workflow rather than merely removing 406 complexes. The fraction of crystallographic ligands passing all PoseBusters checks rises from 63.0% to 78.2%, while the intermolecular checks change by half a point [1]. The difference therefore reflects cleaner molecular preparation rather than easier protein-ligand geometry.

The released recipe also contains training enrichments that were not used here, most notably random rotational augmentation and PoseBusters-based filtering of starting conformers, together with longer parameter averaging and several smaller hyperparameter differences. The

runs in this thesis follow the code defaults where possible. This reduced the computational cost to roughly 72 GPU-hours against approximately 384 for the published budget, but also makes the reproduction intentionally leaner.

The reproduced SigmaDock baseline consequently remains 10 to 15 percentage points below the published results. The gap is plausibly explained by the smaller budget, the leaner recipe and the reprocessed dataset rather than by the generative mechanism. These differences do not favour SigmaFlow. All three models are trained under the same defaults and constraints, and SigmaDock progresses furthest through its own annealing schedule. The experiments should be read as a controlled comparison between diffusion and flow matching under a common lean regime, not as a reproduction of SigmaDock's published performance.

6.1.2 Performance estimation

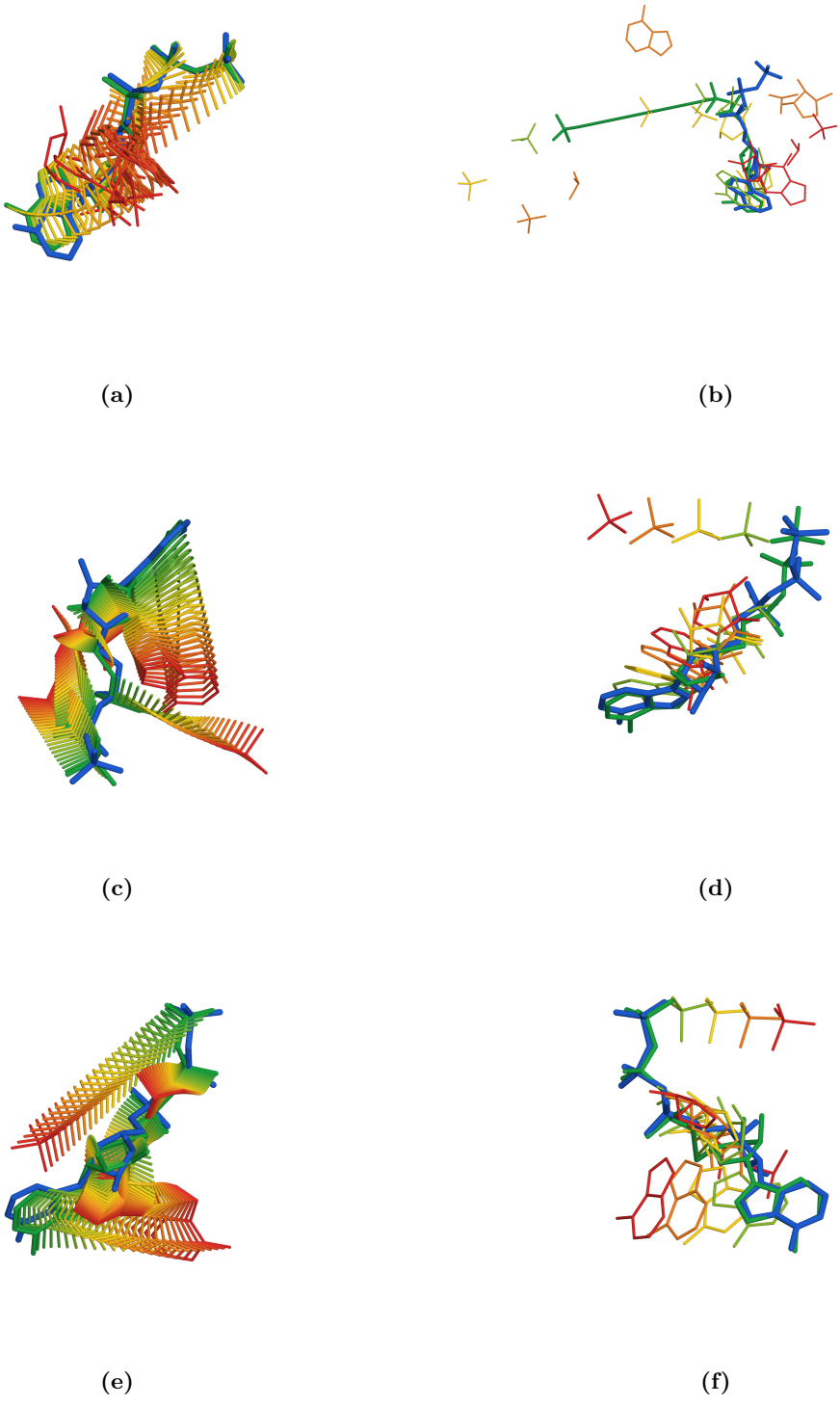
A Top-1 rate at K draws is an expectation over which K candidates happen to be drawn. All reported rates estimate it by drawing 10000 random subsets of size K for every complex, taking the candidate the ranker selects in each, and averaging first over subsets and then over complexes. Where K equals the number of generated candidates there is only one subset and the value is exact. Models are compared by a paired bootstrap test over complexes with 8000 resamples of the mean difference, which is the estimand the tables report.

6.2 Integration steps and what a candidate costs

SigmaDock's final accuracy depends strongly on ranking: one candidate satisfies the combined criterion in 24.1% of cases, while selection from 40 reaches 64.5%. This motivates exchanging some per-candidate quality for a larger pool at fixed compute. Reducing integration steps is the obvious way to do so.

The models respond very differently. From 25 to 5 steps, SigmaFlow loses under one percentage point of placement accuracy per draw, while SigmaDock falls from 50.5% to 7.0%. After mixed-score ranking its accuracy falls from 70.0% to 25.4%, and the combined criterion from 64.5% to 6.5%. All models lose some chemical validity at 5 steps, but the collapse is specific to diffusion. Figure 6.1 illustrates the likely reason for this. The flow trajectories are smoother and remain visibly smooth under coarse integration, whereas SigmaDock deteriorates sharply. This is the robustness required to exchange integration steps for additional candidates.

On PoseBusters, generating one candidate for each of the 308 complexes takes on average 39.0 minutes at 25 integration steps and 11.2 minutes at 5 steps across the two SigmaFlow variants. Fixed loading and output costs reduce the resulting speed-up from the asymptotic factor of 5 to 3.48. Consequently, the wall-clock budget required for 40 25-step candidates permits 140 5-step candidates. On Astex, the corresponding runtimes are 7.3 and 2.9 minutes, allowing 101 5-step candidates under the same budget.



(a) **(b)**

(c) **(d)**

(e) **(f)**

Figure 6.1: Generated trajectories for ligand ATP from PDB entry 7KRU. Rows are the models, in the order SigmaDock (a and b), SigmaFlow-NE (c and d) and SigmaFlow-TR (e and f). The left column integrates 25 steps and the right column 5. Every step is drawn at once, coloured from red at the first to green at the last, with the crystallographic pose in blue. In the intermediate steps the severed bonds are not drawn, so each rigid fragment is visible as a separate body travelling its own path. The final pose keeps its bonds and is drawn thicker. An animated version is available at https://jmuelleo.github.io/SigmaFlow/trajectories/.

For large screening campaigns the fixed cost is amortised and the ratio tends to the 5-fold step ratio. 40 candidates at 25 steps then correspond to 200 at 5, reported as the amortised regime.

6.3 PoseBusters

As shown by the "per draw"-column in Table 6.2, SigmaDock is ahead when it comes to single-sample performance. A single SigmaDock candidate satisfies the combined criterion 24.1% of the time against 16.8% and 19.2% for the flow models at the same step count. Read across to the mixed score and the order changes. At matched sampling wall-clock the flow models reach 69.7% and 68.7% against SigmaDock's 64.5%, and in the amortised regime of 200 draws 72.3% and 70.4%. On accuracy alone the corresponding figures at two hundred draws are 77.5% and 76.9% against 70.0%. At 200 draws the advantage is significant for both flow models, while the difference between SigmaFlow-NE and SigmaFlow-TR is not significant on PB308 at any draw count. Every model here falls short of the published figures, 79.9 and 80.5 against a best

Table 6.2: Performance of SigmaDock and the SigmaFlow variants on PB308 at the 72-hour endpoints. The left half gives Top-1 under three selection rules, the right half compares the mixed score against SigmaDock at 25 steps and 40 draws, so a positive number favours the row. The compute-matched comparison is the bold row, 5 integration steps at 140 draws. The 200-draw rows are the amortised regime of Section 6.2. The published row is SigmaDock as reported by Prat et al. [21] at 40 draws, reproduced for orientation and not measured here.

			Top-1, three selection rules			Mixed score, minus SigmaDock 25/40		
model	steps	candidates	per draw	mixed score	oracle	Pts.	95% interval	p
RMSD < 2 Å and PB-valid with protein								
SigmaDock	25	40	24.09	64.50	79.15		*reference*	
SigmaFlow-NE	25	40	16.75	68.08	78.50	+3.58	[−1.63, +8.47]	0.183
SigmaFlow-TR	25	40	19.21	65.15	77.20	+0.65	[−4.89, +5.86]	0.866
SigmaFlow-NE	5	140	12.95	69.70	82.38	+5.21	[+0.63, +9.85]	0.022
SigmaFlow-TR	5	140	15.10	68.70	80.43	+4.21	[−0.55, +8.87]	0.088
SigmaFlow-NE	5	200	12.95	72.31	85.02	+7.82	[+2.93, +12.70]	0.003
SigmaFlow-TR	5	200	15.10	70.36	82.41	+5.86	[+0.65, +11.07]	0.029
SigmaDock, published	25	40	–	79.9	–		–	
SigmaFlow-TR minus SigmaFlow-NE, mixed score								
at 25 / 40	25	40	–	–	–	−2.93	[−7.82, +1.95]	0.281
at 5 / 140	5	140	–	–	–	−1.00	[−4.79, +2.77]	0.597
at 5 / 200	5	200	–	–	–	−1.95	[−6.84, +2.93]	0.465
RMSD < 2 Å								
SigmaDock	25	40	50.53	70.03	93.16		*reference*	
SigmaFlow-NE	25	40	44.81	74.59	93.49	+4.56	[−0.65, +9.77]	0.085
SigmaFlow-TR	25	40	47.93	71.66	94.79	+1.63	[−3.58, +6.84]	0.572
SigmaFlow-NE	5	140	44.13	76.16	97.28	+6.12	[+1.67, +10.54]	0.004
SigmaFlow-TR	5	140	47.13	75.66	97.02	+5.63	[+1.25, +9.96]	0.011
SigmaFlow-NE	5	200	44.13	77.52	97.72	+7.49	[+2.61, +12.38]	0.002
SigmaFlow-TR	5	200	47.13	76.87	97.72	+6.84	[+2.28, +11.73]	0.004
SigmaDock, published	25	40	–	80.5	–		–	
SigmaFlow-TR minus SigmaFlow-NE, mixed score								
at 25 / 40	25	40	–	–	–	−2.93	[−7.82, +1.95]	0.261
at 5 / 140	5	140	–	–	–	−0.50	[−4.07, +2.97]	0.744
at 5 / 200	5	200	–	–	–	−0.65	[−5.21, +3.91]	0.852

of 72.3 and 77.5. The likely reason was discussed in Section 6.1.1 and should be kept in mind throughout.

6.4 The Astex Diverse Set

As you can see in Table 6.3 Astex shows the same pattern more strongly. At matched wall-clock, SigmaFlow-NE and SigmaFlow-TR reach 84.9% and 88.1% on the combined criterion against SigmaDock's 74.1%, gains of 10.7 and 14 points. At 200 candidates they reach 87.1% and 92.9%, with SigmaFlow-TR slightly above the published SigmaDock result of 90.6%. SigmaFlow-NE and SigmaFlow-TR cannot be ranked reliably against each other. SigmaFlow-TR leads on Astex but SigmaFlow-NE on PoseBusters, and only one training run was performed per model.

Table 6.3: The Astex Diverse Set, otherwise as in Table 6.2. The compute-matched count here is 101 draws at 5 steps.

			Top-1, three selection rules			Mixed score, minus SigmaDock 25/40		
model	steps	candidates	per draw	mixed score	oracle	Pts.	95% interval	p
RMSD < 2 Å and PB-valid with protein								
SigmaDock	25	40	38.65	74.12	89.41		*reference*	
SigmaFlow-NE	25	40	28.38	81.18	91.76	+7.06	[−3.53, +17.65]	0.213
SigmaFlow-TR	25	40	31.50	82.35	94.12	+8.24	[−2.35, +18.82]	0.148
SigmaFlow-NE	5	101	23.19	84.87	96.34	+10.76	[+3.35, +18.71]	0.004
SigmaFlow-TR	5	101	26.21	88.11	96.65	+13.99	[+6.27, +22.18]	< 0.001
SigmaFlow-NE	5	200	23.19	87.06	98.82	+12.94	[+3.53, +22.35]	0.014
SigmaFlow-TR	5	200	26.21	92.94	98.82	+18.82	[+10.59, +28.24]	< 0.001
SigmaDock, published	25	40	–	90.6	–		–	
SigmaFlow-TR minus SigmaFlow-NE, mixed score								
at 25 / 40	25	40	–	–	–	+1.18	[−7.06, +9.41]	0.904
at 5 / 101	5	101	–	–	–	+3.24	[−1.10, +7.86]	0.153
at 5 / 200	5	200	–	–	–	+5.88	[−1.18, +12.94]	0.126
RMSD < 2 Å								
SigmaDock	25	40	65.47	78.82	95.29		*reference*	
SigmaFlow-NE	25	40	60.62	85.88	98.82	+7.06	[+0.00, +15.29]	0.090
SigmaFlow-TR	25	40	63.32	85.88	98.82	+7.06	[−2.35, +16.47]	0.156
SigmaFlow-NE	5	101	61.08	87.65	98.75	+8.83	[+2.79, +15.21]	0.002
SigmaFlow-TR	5	101	64.06	89.74	99.42	+10.92	[+4.35, +18.09]	< 0.001
SigmaFlow-NE	5	200	61.08	89.41	98.82	+10.59	[+2.35, +18.82]	0.010
SigmaFlow-TR	5	200	64.06	92.94	100.00	+14.12	[+5.88, +22.35]	< 0.001
SigmaDock, published	25	40	–	90.6	–		–	
SigmaFlow-TR minus SigmaFlow-NE, mixed score								
at 25 / 40	25	40	–	–	–	+0.00	[−8.24, +8.24]	1.000
at 5 / 101	5	101	–	–	–	+2.09	[−2.01, +6.87]	0.329
at 5 / 200	5	200	–	–	–	+3.53	[−2.35, +9.41]	0.330

Figure 6.2 shows why candidate count matters. SigmaDock leads for small candidate sets, but the flow models overtake it as the number of candidates grows. At equal integration steps the crossing is between 20 and 35 candidates on PoseBusters and below ten on Astex. Beyond roughly 25 candidates SigmaDock gains little, whereas the flow curves continue to rise. This suggests that the flow models generate structurally broader candidate sets, which is tested directly in Chapter 7. Reducing SigmaDock itself to 5 integration steps does not yield a benefit. Its performance remains near the bottom across candidate counts, showing that additional samples cannot compensate once per-sample quality has collapsed. In these experiments, flow matching makes this trade-off viable by preserving useful sample quality at low step counts, allowing compute to be reallocated from longer trajectories to larger candidate sets. The prescribed simple transport paths are the plausible explanation for this low-step robustness.

Another thing to note is that the oracle-minus-mixed gap measures selection loss. On PB308 it

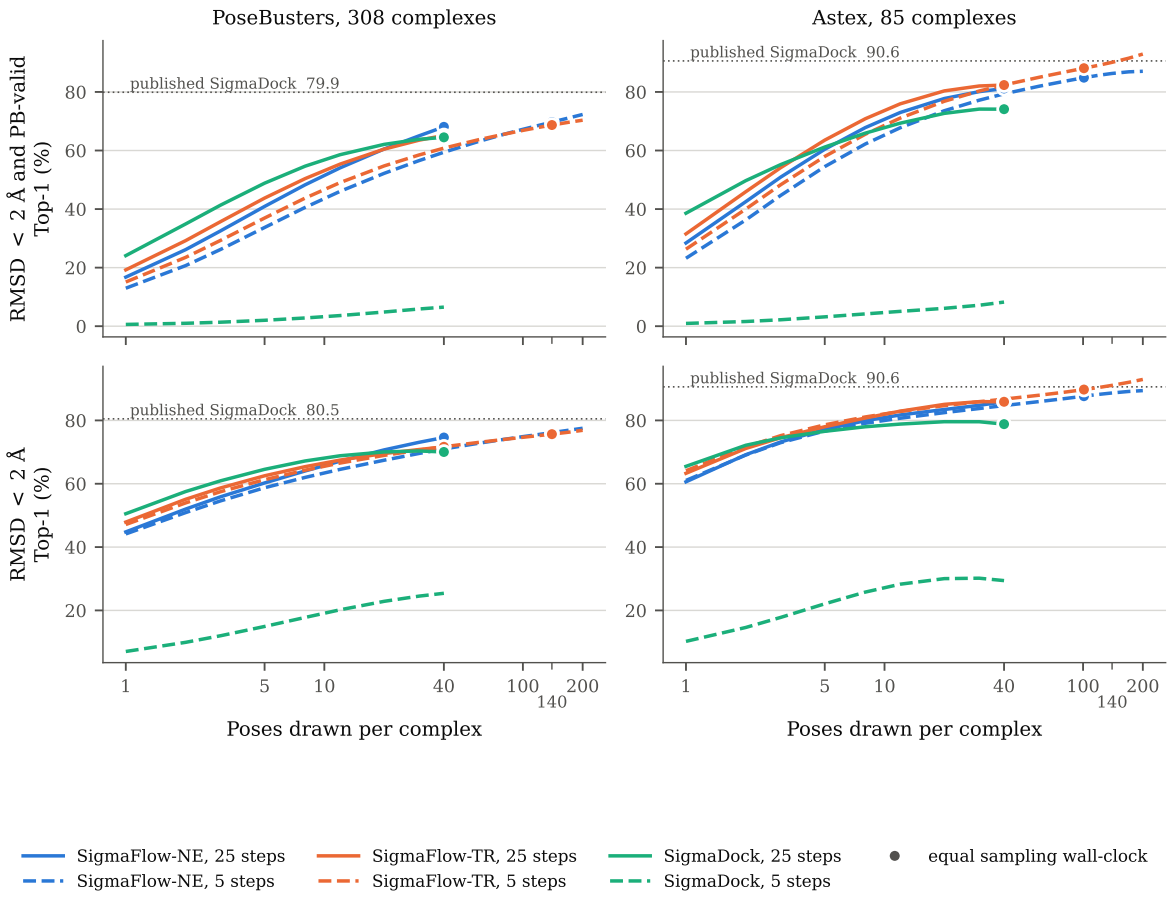


Figure 6.2: Top-1 under SigmaDock's mixed score against the number of poses drawn per complex, at the training endpoints, for both benchmarks and both success criteria.

is 12.71 and 12.05 points for the two flow models at 200 candidates, against 14.65 for SigmaDock at 40. A substantial part of available candidate-set accuracy is therefore lost at selection for every model, an observation we will come back to in Chapter 7.

6.5 Performance over fragment count

Performance also varies with the number of rigid fragments a ligand is cut into, shown in Figure 6.3. The classes are unevenly filled, and the right-hand end of the PoseBusters panel should not be read as a trend, since eleven fragments or more account for six complexes in total and a single pose can move such a point by 50 percentage points. Over the populated range the decline is real. The rank correlation between fragment count and success is negative for all 5 cells and significant in every one, with ρ between -0.15 and -0.19 at $p < 0.01$. After six fragments success falls from between 70% and 79% to between 34% and 46%, and it does so for the diffusion baseline and both flow models alike. Larger ligands are therefore harder for all three, which is what one expects when each additional fragment adds six dimensions to the state the sampler has to get right. The Astex panel shows no comparable decline, but the benchmark contains only ten ligands with seven or more fragments, compared with 49 of the 308 PoseBusters complexes. The evidence therefore rests primarily on a single benchmark,

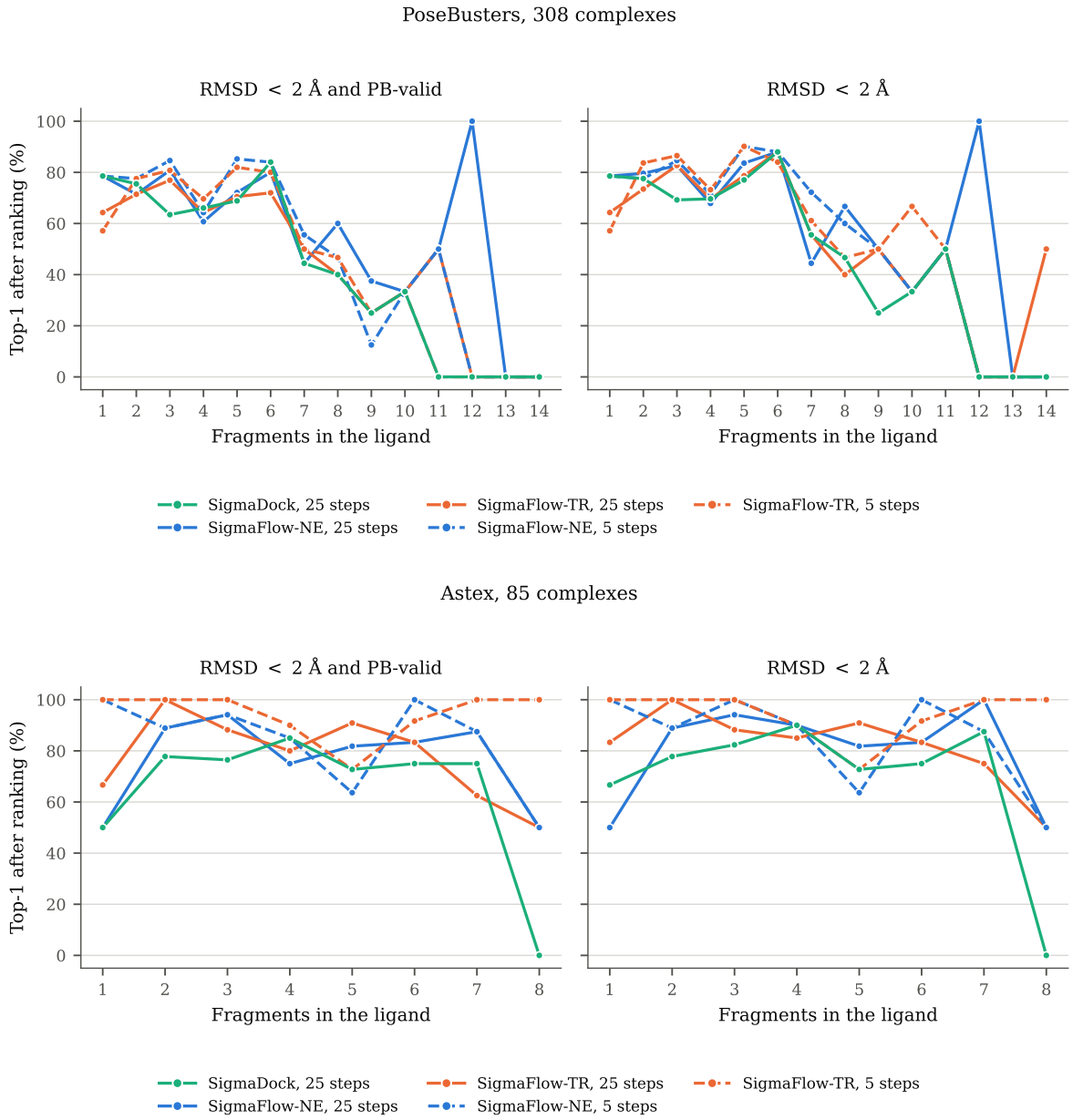


Figure 6.3: Top-1 under the mixed score against the number of rigid fragments the ligand is cut into, on PoseBusters (top, $n = 308$) and Astex (bottom, $n = 85$). Colour is the model and line style the number of integration steps, as in Figure 6.2. SigmaDock at 5 steps is omitted because its performance is not competitive.

while the high-fragment regime in which the effect would matter most is sparsely represented in both. This makes it difficult to determine conclusively how strongly docking difficulty increases with fragment count.

6.6 Position against published methods

Figures 6.4 and 6.5 place the three models beside the methods Prat et al. [21] reports. Two points stand out. On PoseBusters every model here stays below the published model, by 7.6

points at best on the combined criterion, which is the reproduction gap of Section 6.1.1 rather than a statement about mechanisms. On Astex SigmaFlow-TR at 5 steps and 200 draws reaches 92.9% on the combined criterion, more than the published 90.6%, and it is the only cell in either table where a 72-hour run of this thesis exceeds the published figure. Both flow models also clear every classical and deep-learning baseline on both benchmarks, while the diffusion model at 5 steps does not.

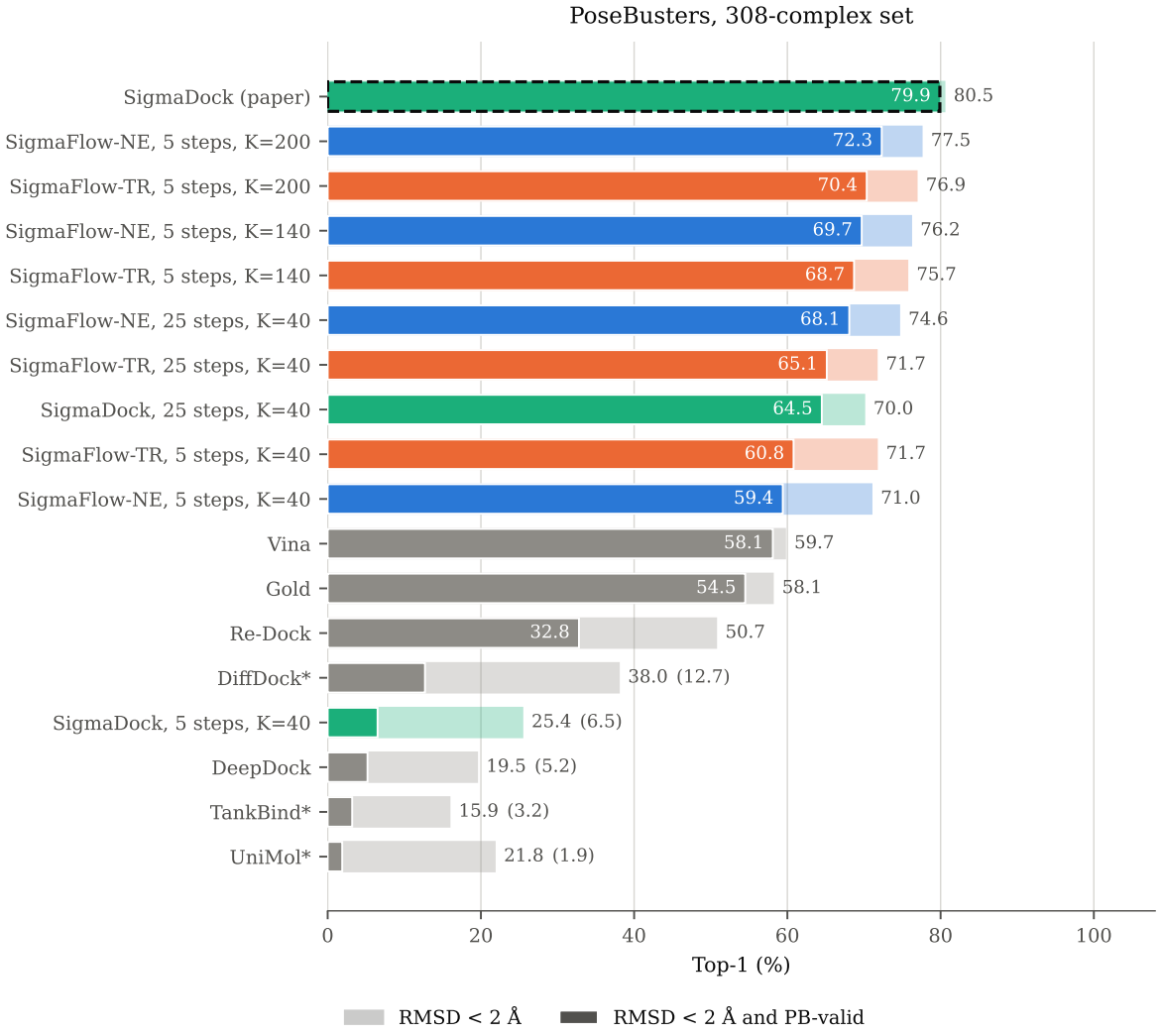


Figure 6.4: Top-1 on PoseBusters, published methods and the three models of this thesis, ordered by the combined criterion.

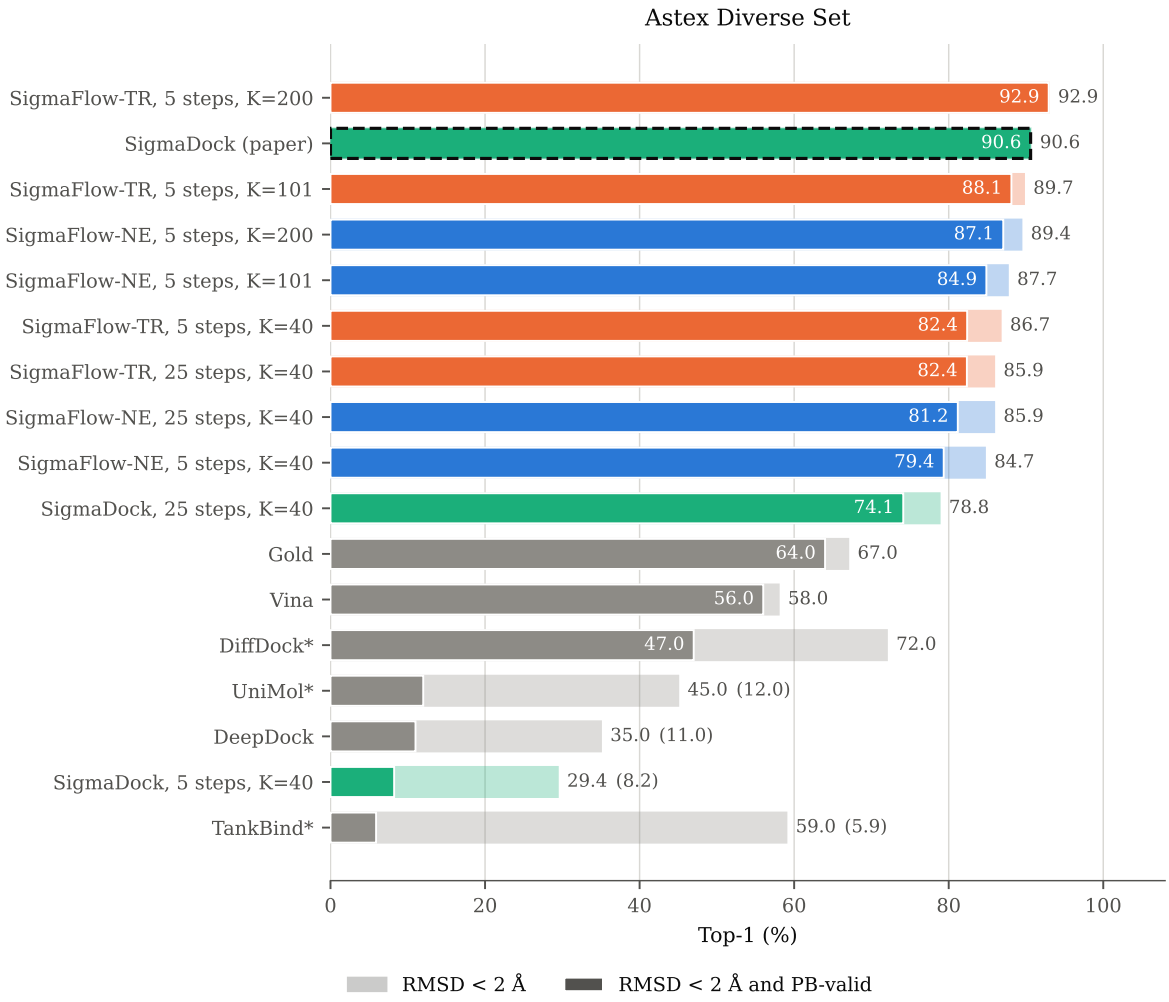


Figure 6.5: Top-1 on the Astex Diverse Set, encoded as in Figure 6.4.

Chapter 7

Generated distribution

Chapter 6 established the central trade-off. SigmaDock is stronger per draw, but SigmaFlow remains useful at 5 integration steps and can spend the saved compute on more candidates. Ranking then makes the flow models competitive or better. Yet candidate count alone cannot explain this. The last 5 candidates are worth 0.27 points to SigmaDock at 25 steps, against 1.41 for SigmaFlow-NE and 0.68 for SigmaFlow-TR. On placement accuracy SigmaDock loses 0.23 points over the same 5 draws while the flow models gain 0.67 and 0.41. At 5 steps and 200 candidates both flow models still gain about 0.10 points on placement accuracy and 0.19 and 0.11 on the combined criterion. Both mechanisms therefore saturate, but SigmaFlow does so significantly later.

This suggests that what matters is not only how many candidates are drawn but how many distinct structural hypotheses they represent. This chapter therefore asks three questions. Do the models generate distributions with different structural diversity? Does the shape of those distributions reveal prediction uncertainty? And when a successful pose exists but Top-1 fails, is the remaining error within a structural mode or in choosing the mode itself?

7.1 Training trajectories

The endpoint comparison alone cannot determine whether the differences are temporary consequences of the training budget. Snapshots written every six hours during the same three runs provide the trajectory.

Figure 7.1 confirms two features of the endpoint results. The ordering in per-draw performance becomes stable from approximately epoch 50 onwards, with SigmaDock stronger per draw than either flow model, so the endpoint difference is not created by where training happened to stop. The 5-step trajectories of the flow models also stay close to their 25-step counterparts throughout, ending at 44.0% and 47.2% accuracy against 44.8% and 47.9% at 25 steps. SigmaDock behaves differently. Its 5-step trajectory remains near the bottom of the panel and ends at 7.1% against 50.5% at 25 steps. The low-step robustness the trade requires is therefore present throughout training rather than only at the endpoint. Lastly note that Figure 7.1 also shows that the three

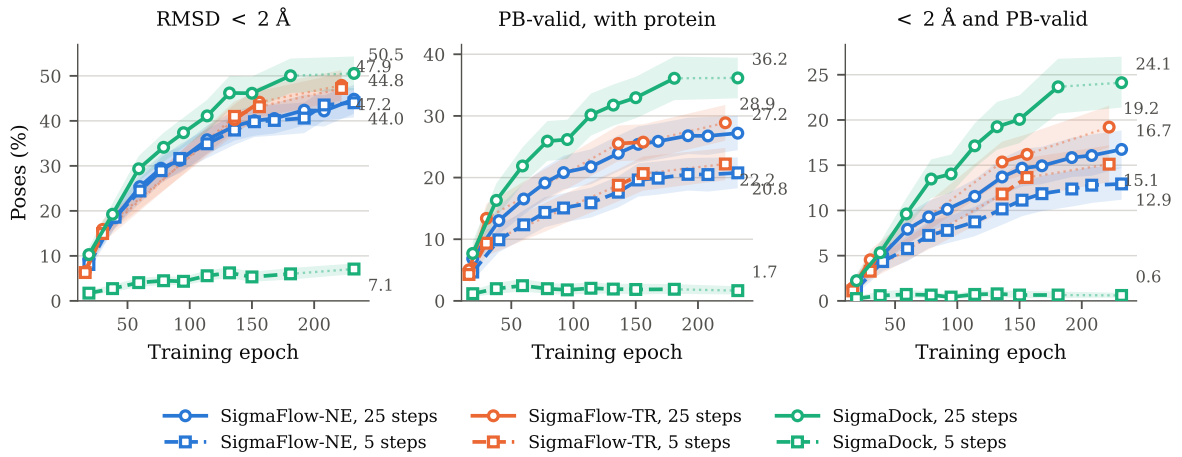


Figure 7.1: Per-draw rates over training for all three models at 25 and 5 integration steps, evaluated on PB308 with generated input conformers and no ranking, so the curves measure generation alone. Markers are snapshots, bands are 95% bootstrap intervals over complexes, and the final value of each curve is printed at its right end. The axis is the training epoch rather than wall-clock, because the models differ in throughput and two of them resumed after node failures, which restarts a clock but not an epoch counter. Faint dotted segments bridge gaps of more than 40 epochs, where a snapshot mechanism failed during a continuation run and no measurement exists.

models are not equally converged at the end. Over their final snapshot interval SigmaDock still gains 0.010 points of placement accuracy per epoch, against 0.108 for SigmaFlow-NE and 0.057 for SigmaFlow-TR, in the same order as the fraction of the annealing schedule each completed. The comparison is therefore conservative with respect to the flow models, since the diffusion baseline is the closest to convergence.

7.2 Effective poses

A generated distribution lives on a product of fragment translations and rotations and is difficult to inspect directly. For ranking the question is simpler. Do two candidates represent meaningfully different binding hypotheses, or are they effectively repeated samples of the same pose?

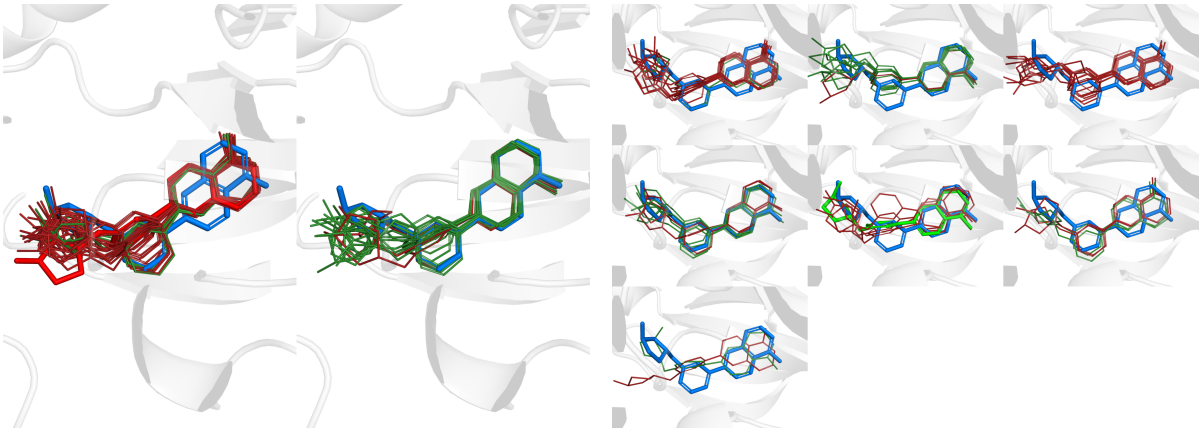
We operationalise the distinction by clustering candidates on their pairwise RMSD, as detailed in Algorithm 1. The threshold is 2 Å, matching the definition of a successful pose in the evaluation. So each cluster can be read as one effective structural hypothesis at that resolution.

Three choices matter for the interpretation. The poses are already expressed in the global coordinates, so no superposition is applied and the spatial information at issue is preserved. Complete linkage merges two groups only once their largest cross distance falls below δ, which guarantees that every pair inside a cluster lies within the threshold rather than being connected indirectly through intermediate samples. The distance is computed over the common atom ordering rather than over molecular automorphisms, so a pose related to another by a symmetry of the ligand graph counts as distinct. This was done to keep computational demands low and

Algorithm 1 Clustering generated poses into effective structural hypotheses

Input: generated poses $\{x_1, \ldots, x_K\}$ in the receptor frame and threshold $\delta = 2\,\text{Å}$
for every pair (i, j) **do**
    $d_{ij} \leftarrow \text{RMSD}(x_i, x_j)$ over the common atom ordering, without superposition
end for
Build the hierarchy by agglomerative clustering on d with complete linkage
Cut the hierarchy at distance δ
Return: clusters $\mathcal{C}_1, \ldots, \mathcal{C}_M$

makes the reported cluster counts an upper bound in absolute terms. It affects all three models identically and therefore leaves the comparison between them unchanged.



(a) SigmaDock, two modes.

(b) SigmaFlow-NE, seven modes.

Figure 7.2: The 40 candidates generated for complex 7PIH_7QW, clustered into binding modes. Each panel is one mode, ordered by size. Thin sticks are the candidates, blue is the crystallographic pose, and the thick candidate is the one the ranker returned, coloured by its own outcome. Green means the candidate satisfies the success criterion, RMSD $< 2\,$Å together with all PoseBusters checks, and red means it fails at least one. SigmaDock places 62.5% of its draws in its first mode and returns a candidate from it although its second mode is correct. SigmaFlow-NE spreads over seven modes, the largest holding 22.5%, and returns a successful candidate. An interactive visualisation with further examples is available at `https://jmuelleo.github.io/SigmaFlow/modes/`.

The distributions differ substantially. At 40 candidates and 25 steps on PoseBusters, SigmaDock generates 8.83 clusters per complex against 13.92 for SigmaFlow-NE and 13.24 for SigmaFlow-TR, paired increases of 5.09 and 4.42 ($p < 0.0001$). This is 0.22 effective poses per draw against 0.35 and 0.33.

SigmaDock is also more concentrated. Its largest cluster contains 45.9% of candidates against 27.5% and 31.5% for the flow models. On Astex the corresponding values are 57.7%, 36.1% and 39.1%. Even within modes, SigmaDock is tighter, size-weighted pairwise RMSD is 1.153 Å against 1.351 Å and 1.326 Å. It therefore generates fewer modes, places more mass on its dominant mode and samples more tightly within each mode. These results are summarised in Table 7.1 and can also be observed in Figure 7.2 This helps explain the sampling curves. 40 SigmaDock draws represent fewer than nine effective poses at 2 Å, while 40 flow draws represent about thirteen to fourteen. A broader distribution requires more samples to explore,

Table 7.1: Structure of the generated pose distribution on PoseBusters after clustering at 2 Å. Cluster count is the mean number of structural clusters per complex, largest cluster is the fraction of generated poses in the largest cluster, and within-cluster RMSD is the size-weighted mean pairwise distance inside clusters.

model	steps	candidates	clusters	largest cluster	within-cluster RMSD
SigmaDock	25	40	8.83	45.9%	1.153 Å
SigmaFlow-NE	25	40	13.92	27.5%	1.351 Å
SigmaFlow-TR	25	40	13.24	31.5%	1.326 Å
SigmaFlow-NE	5	200	46.43	18.9%	1.292 Å
SigmaFlow-TR	5	200	43.88	22.2%	1.268 Å

consistent with SigmaFlow overtaking only once enough candidates are drawn. Furthermore, similar results are found at a finer 1 Å or 0.5 Å resolution.

These results indicate that prescribing simple, time-independent transport paths changes the sampling distribution substantially. Compared with diffusion, the linear and geodesic paths used by SigmaFlow produce a broader, less concentrated distribution that longer continues to generate structurally distinct alternatives as additional samples are drawn.

7.3 Distributional concentration as uncertainty

The same observation has a second implication. A concentrated candidate distribution means that independent source draws repeatedly lead to essentially the same binding hypothesis. A dispersed distribution means the model keeps producing competing hypotheses. The latter is naturally associated with ambiguity about the complex, which suggests that the generated distribution itself carries information about prediction reliability.

For some complexes nearly all candidates collapse into one dominant mode. For others substantial mass is spread across incompatible modes, which often differ in how part of the ligand is rotated rather than in gross placement. One such mode may contain near-native poses while another represents an incorrect binding mode. The distribution therefore records disagreement between plausible solutions even though the generator was never trained to output an uncertainty estimate.

A simple statistic already captures much of this. Let

$$c_{\max} = \frac{\max_m |\mathcal{C}_m|}{K} \tag{7.1}$$

denote the fraction of candidates in the largest cluster. Used alone to predict whether the selected pose succeeds, $c_{\max}$ gives at 25 steps AUC-ROCs of 0.78, 0.78 and 0.79 for SigmaDock, SigmaFlow-NE and SigmaFlow-TR, also showing that the signal is not specific to flow matching.

The effect is also large in absolute terms. For SigmaDock the success rate is 84.3% in the highest

concentration tercile against 34.3% in the lowest. The corresponding values are 88.7% versus 41.7% for SigmaFlow-NE and 83.3% versus 34.9% for SigmaFlow-TR. The distribution of the samples is therefore an uncertainty signal. Concentrated predictions are more often reliable, while competing modes indicate harder complexes.

7.4 Where docking failures occur

Distributional uncertainty indicates when a prediction may be unreliable but not why it failed. A failed Top-1 prediction can arise at different stages. The generator may never produce a successful pose, or a successful pose may exist in the candidate pool and be rejected by ranking. The second case separates further into choosing the wrong structural mode and choosing the wrong candidate within the correct mode.

On PoseBusters, complete generation failure remains an important limitation. For SigmaDock at 25 steps no successful pose is generated for 20.9% of complexes, against 21.5% and 22.8% for the flow models at the same step count. Raising the 5-step pool to 200 candidates lowers these rates to 15% and 17.6%. Additional coverage therefore reduces generation failure without eliminating it.

Among SigmaDock failures, 41.3% still contain a successful pose somewhere in the pool, against 32.7% for SigmaFlow-NE and 34.6% for SigmaFlow-TR at 25 steps, and 45.9% for SigmaFlow-NE and 40.7% for SigmaFlow-TR at 5 steps with 200 candidates. On Astex the distinction is sharper, where 90.9% and 83.3% of failed predictions for the two 5-step flow models nevertheless contain a successful pose. In those cases the required structure has already been generated and the remaining error is entirely one of selection.

7.5 The cluster oracle

The clustering localises these selection errors. A cluster oracle is allowed to identify a cluster containing a successful pose, but the ordinary mixed score must still choose within that cluster. Comparison with the full pose oracle therefore separates between-mode from within-mode ranking error.

Table 7.2: Top-1 success on PoseBusters under the ordinary mixed-score selector, a cluster oracle that names a cluster containing a successful pose and then applies the mixed score within it, and the full pose oracle. Clusters are defined at 2 Å.

model	steps	candidates	mixed score	cluster oracle	full oracle	remaining gap
SigmaDock	25	40	64.50	77.85	79.15	1.30
SigmaFlow-NE	25	40	68.08	78.18	78.50	0.33
SigmaFlow-TR	25	40	65.15	76.55	77.20	0.65
SigmaFlow-NE	5	200	72.31	83.71	85.02	1.30
SigmaFlow-TR	5	200	70.36	81.76	82.41	0.65

Naming the correct structural cluster raises Top-1 success by 10 to 13 percentage points across the PoseBusters configurations. After that intervention only 0.33 to 1.30 percentage points remain between the mixed score and the full pose oracle. Repeating the analysis at a 1 Å clustering threshold gives the same qualitative result with a remaining gap no larger than about one point, so the conclusion is not an artefact of the resolution.

Once the correct mode is known, the existing mixed score is already very effective at choosing within it. The recoverable ranking error therefore lies primarily *between* structural modes rather than *within* the correct one. Improving the fine ranking of geometrically similar candidates offers little access to the remaining oracle gap. A useful selector instead has to distinguish competing binding hypotheses that may each contain internally plausible poses.

The comparison also clarifies where SigmaFlow's advantage comes from. At the same 25-step schedule, differences largely disappear under the cluster oracle. SigmaDock, SigmaFlow-NE and SigmaFlow-TR reach 77.9%, 78.2% and 76.6%, despite ordinary Top-1 rates of 64.5%, 68.1% and 65.2%. The same-schedule gain therefore appears to arise mainly from the broader candidate distribution giving the ranker more favourable alternatives. With 5 steps and 200 candidates, a second effect emerges. Cluster-oracle success rises to 83.7% for SigmaFlow-NE and 81.8% for SigmaFlow-TR, with full-oracle rates of 85% and 82.4%. The improvement therefore appears to operate through two related channels. At a fixed candidate count, SigmaFlow's broader distribution gives the ranker more structurally distinct realisations of the available binding modes, increasing the chance that a successful pose is also recognised as favourable by the mixed score. Because low-step sampling remains cheap and additional draws continue to explore new regions of the distribution, larger candidate sets then provide a second benefit by increasing coverage of successful binding modes themselves.

7.6 Implications for selection and uncertainty

The distributional analysis reveals two complementary limitations. Generation remains imperfect, and on PoseBusters a substantial fraction of failed predictions contain no successful candidate and cannot be repaired by any ranker. Drawing more SigmaFlow candidates reduces this through broader coverage. Once a successful pose is available, most of the remaining recoverable error lies in identifying its structural mode. The mixed score already ranks poses well within a correct mode, suggesting that future ranking may benefit more from learning which cluster is promising than from refining individual pose-level scores. This reduces the decision from many poses to a smaller number of structural modes and allows information to be aggregated across multiple members of each cluster, potentially providing a stronger signal of whether a mode is correct.

Furthermore the generated distribution signals when a problem is difficult. Largest-cluster concentration predicts final success across both mechanisms. But *detecting* ambiguity is different from *resolving* it. Repeated sampling often reveals that mass is spread across incompatible

binding hypotheses, but that alone does not identify which hypothesis is correct.

The central result of this chapter is therefore not that SigmaFlow generates more poses. It generates a more diverse candidate distribution in the experiments studied here, so additional sampling continues to expose the ranker to structurally distinct alternatives after SigmaDock has begun to saturate. The same distribution provides an intrinsic signal of prediction reliability. As coverage improves, the remaining recoverable error increasingly becomes a problem of choosing between competing binding modes. Improving that decision, rather than generating or finely reranking more poses, is the natural next target.

Chapter 8

Discussion

8.1 What the comparison establishes

The comparison supports four conclusions.

Flow matching tolerates coarse integration where diffusion does not. The 5-step gap appears throughout training, not only at the endpoint.

At equal sampling cost, the flow models perform better after ranking. The direction is consistent across both benchmarks and both success criteria. SigmaFlow matches or exceeds the reproduced SigmaDock baseline both at the 25-step, 40-candidate setting and when inference compute is reallocated towards many low-step samples, although not every individual same-schedule comparison is statistically significant. Most importantly, the central hypothesis of this thesis is strongly supported: reducing SigmaFlow to 5 integration steps and spending the saved compute on a much larger candidate set significantly improves Top-1 success. At 200 candidates, this gain is significant for both SigmaFlow variants on both benchmarks and under both the 2 Å RMSD and combined PoseBusters-valid criteria.

This coexists with better single-draw performance from SigmaDock. The advantage of SigmaFlow appears when inference compute is allocated across a candidate set rather than to one draw.

Candidate diversity and ranking are part of the mechanism. SigmaDock's ranked curve saturates earlier, while SigmaFlow supplies more structural modes per draw. Once a successful mode exists, the mixed score performs well within it but can choose the wrong mode.

Note that only one training run was performed per model, and SigmaFlow-TR leads SigmaFlow-NE on Astex but trails it on PoseBusters. The readout variants therefore cannot be ranked reliably.

8.2 Limitations

The training budget and the departures from the published recipe are the binding limitation, and Section 6.1.1 discusses both. Beyond that, only one training run was performed per model, so every interval and p value is computed over sampling seeds and complexes rather than over repeated trainings. This matters most for the readout comparison, where the effect sizes are smallest. The comparison matches wall-clock rather than optimisation steps, and SigmaFlow-TR reached 222 epochs against the 232 of SigmaFlow-NE, so it is the model with the least training.

8.3 Future work

8.3.1 Matching the published training protocol

A first priority for future work is to resolve the remaining SigmaDock reproduction gap. Additional runs designed to more closely match the published training protocol, including the original training set, unexpectedly completed before submission, but only a few hours before the deadline and too late for a full analysis. A preliminary evaluation still places these reproductions roughly 6-10 percentage points below the reported benchmark, leaving the source of the discrepancy unresolved. At the same time, the qualitative compute trade-off appears to persist, with 200 5-step SigmaFlow samples continuing to outperform 40 25-step SigmaDock samples. SigmaFlow-NE even reaches a 94.1% success rate on the combined criterium on Astex. Future work should therefore isolate the remaining differences in training and implementation and repeat each configuration across random seeds to determine whether the observed variation is systematic or reflects run-to-run variability.

8.3.2 Learning to select the structural mode

Section 7.5 showed that naming the correct cluster is worth 10 to 13 percentage points and nearly exhausts what better ranking can achieve. Two models suggest themselves. The first scores clusters rather than individual candidates and estimates which structural mode contains a successful pose, trained on generated samples in the manner of DiffDock [6] and evaluated on a held-out set. The second maps the generated distribution together with the mixed score to a calibrated probability that the delivered prediction is correct, making the signal of Section 7.3 usable at inference time.

8.3.3 Informative source distributions

The source is a free modelling choice under flow matching and a fixed consequence of the forward process under diffusion, which makes it the most distinctively flow-specific degree of freedom available, and SigmaFlow leaves it unexploited by inheriting (2.49) unchanged. The translational source is already informative, since pocket-centred coordinates place its mean near the answer. On the rotational factor there is no counterpart, introducing prior information seems like a promising future direction of work.

8.3.4 Likelihood-based pose confidence

The flow formulation offers a second route to confidence that diffusion does not offer as directly, namely reading a likelihood off the generative process itself. Under the usual regularity conditions, the deterministic ODE used for sampling defines a continuous normalising flow [4], and the instantaneous change of variables gives

$$\log p_1(x_1) \;=\; \log p_0(x_0) \;-\; \int_0^1 \operatorname{div} v_\theta(t, x_t)\, \mathrm{d}t. \tag{8.1}$$

The same holds on a Riemannian manifold with the divergence taken with respect to the Riemannian volume [3], which on SO(3) with the bi-invariant metric is the Haar measure of Section 2.3.1, so the construction is available on $\mathcal{M}$ without further choices.

The immediate use is ranking the K generated candidates by conditional log-density, which needs no additional training and no labelled samples and is therefore directly comparable against a learned classifier on identical candidate sets.

8.3.5 Rectified Flows

Another natural extension is to apply rectification to SigmaFlow by retraining on endpoint pairs induced by the learned flow, with the aim of making the marginal transport trajectories straighter, as described in Liu et al. [14]. This could further improve very low-step sampling efficiency, but it would be important to test whether these gains are obtained without reducing the structural diversity that appears to make additional SigmaFlow samples useful.

Chapter 9

Conclusion

This thesis replaced SigmaDock's score-based diffusion with Riemannian flow matching while retaining its fragment representation, architecture, data pipeline and evaluation protocol. The motivation came from SigmaDock itself: single draws are often poor, yet ranking a candidate set raises performance substantially. This suggested reallocating inference compute from integration depth to candidate count.

Simply reducing SigmaDock's integration steps fails because its sampler collapses at 5 steps. SigmaFlow remains much more robust. Its individual candidates are generally weaker, but 5-step sampling makes enough additional candidates affordable that final ranked performance improves at matched wall-clock and network-evaluation budgets on both benchmarks.

The candidate distribution explains why the trade remains useful. SigmaFlow generates more structural modes per draw, places less mass in its dominant mode and is less concentrated within modes. Extra draws therefore provide genuinely new alternatives for longer, whereas SigmaDock saturates earlier. The same distribution carries uncertainty information: largest-cluster concentration predicts final success with ROC-AUC around 0.78 across both mechanisms.

Generation is still a major failure source on PoseBusters. When a successful candidate does exist, however, a cluster oracle nearly closes the full pose oracle, showing that most recoverable ranking error lies between binding modes rather than within the correct one. Learning to select those modes is therefore the most direct ranking extension.

The conclusions remain limited by the lean training budget and a single training run per model. Future work should therefore test these findings under the full SigmaDock training recipe and exploit design freedoms specific to flow matching, particularly informative source distributions and rectified flows. The generated sample distribution also offers a natural basis for further uncertainty quantification and for developing cluster-level binding-mode selection, which may be more effective than ranking individual poses directly.

Bibliography

[1] Martin Buttenschön. *Deep learning-based protein–ligand docking*. Dphil thesis, University of Oxford, 2025. Department of Statistics, The Queen's College.

[2] Martin Buttenschön, Garrett M. Morris, and Charlotte M. Deane. Posebusters: Ai-based docking methods fail to generate physically valid poses or generalise to novel sequences. *Chemical Science*, 15:3130–3139, 2024.

[3] Ricky T. Q. Chen and Yaron Lipman. Flow matching on general geometries. In *International Conference on Learning Representations*, 2024.

[4] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, 2018.

[5] Gregory S. Chirikjian and Alexander B. Kyatkin. *Engineering Applications of Noncommutative Harmonic Analysis: With Emphasis on Rotation and Motion Groups*. CRC Press, Boca Raton, 2001.

[6] Gabriele Corso, Hannes Stärk, Bowen Jing, Regina Barzilay, and Tommi Jaakkola. DiffDock: Diffusion steps, twists, and turns for molecular docking. In *International Conference on Learning Representations*, 2023.

[7] Valentin De Bortoli, Emile Mathieu, Michael Hutchinson, James Thornton, Yee Whye Teh, and Arnaud Doucet. Riemannian score-based generative modelling. In *Advances in Neural Information Processing Systems*, 2022.

[8] Michael J. Hartshorn, Marcel L. Verdonk, Gianni Chessari, Suzanne C. Brewerton, Wijnand T. M. Mooij, Paul N. Mortenson, and Christopher W. Murray. Diverse, high-quality test set for the validation of protein-ligand docking performance. *Journal of Medicinal Chemistry*, 50(4):726–741, 2007.

[9] Peter Holderrieth and Ezra Erives. Introduction to flow matching and diffusion models, 2026. URL https://diffusion.csail.mit.edu/. arXiv:2506.02070.

[10] Wengong Jin, Siranush Sarkizova, Xun Chen, Nir Hacohen, and Caroline Uhler. Unsupervised protein-ligand binding energy prediction via neural Euler's rotation equation. In *Advances in Neural Information Processing Systems*, 2023.

[11] Adam Leach, Sebastian M. Schmon, Matteo T. Degiacomi, and Chris G. Willcocks. Denoising diffusion probabilistic models on SO(3) for rotational alignment. In *ICLR Workshop on Geometrical and Topological Representation Learning*, 2022.

[12] Yi-Lun Liao, Brandon Wood, Abhishek Das, and Tess Smidt. Equiformerv2: Improved equivariant transformer for scaling to higher-degree representations. In *International Conference on Learning Representations*, 2024.

[13] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

[14] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023.

[15] Zhihai Liu, Minyi Su, Li Han, Jie Liu, Qifan Yang, Yan Li, and Renxiao Wang. Forging the basis for developing protein–ligand interaction scoring functions. *Accounts of Chemical Research*, 50(2):302–309, 2017.

[16] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[17] Jerrold E. Marsden and Tudor S. Ratiu. *Introduction to Mechanics and Symmetry*. Springer, 2nd edition, 1999.

[18] Andrew T. McNutt, Paul Francoeur, Rishal Aggarwal, Tomohide Masuda, Rocco Meli, Matthew Ragoza, Jocelyn Sunseri, and David Ryan Koes. Gnina 1.0: molecular docking with deep learning. *Journal of Cheminformatics*, 13(43), 2021.

[19] Rocco Meli and Philip C. Biggin. spyrmsd: symmetry-corrected RMSD calculations in Python. *Journal of Cheminformatics*, 12:49, 2020.

[20] PDBbind+. PDBbind+ data release v2020.R1: The PDBbind v2020 complexes retained in v2024, reprocessed, 2024. URL https://www.pdbbind-plus.org.cn/download. General set, 19,037 protein-ligand complexes. Accessed 3 August 2026.

[21] Alvaro Prat, Leo Zhang, Charlotte M. Deane, Yee Whye Teh, and Garrett M. Morris. SigmaDock: Untwisting molecular docking with fragment-based SE(3) diffusion. In *International Conference on Learning Representations*, 2026.

[22] Rodrigo Quiroga and Marcos A. Villarreal. Vinardo: A scoring function based on autodock vina improves scoring, docking, and virtual screening. *PLoS ONE*, 11(5):e0155183, 2016.

[23] Joan Solà, Jérémie Deray, and Dinesh Atchuthan. A micro Lie theory for state estimation in robotics. *arXiv preprint arXiv:1812.01537*, 2018.

[24] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[25] Jason Yim, Andrew Campbell, Andrew Y. K. Foong, Michael Gastegger, José Jiménez-Luna, Sarah Lewis, Victor Garcia Satorras, Bastiaan S. Veeling, Regina Barzilay, Tommi Jaakkola, and Frank Noé. Fast protein backbone generation with SE(3) flow matching. *arXiv preprint arXiv:2310.05297*, 2023.

[26] Jason Yim, Brian L. Trippe, Valentin De Bortoli, Emile Mathieu, Arnaud Doucet, Regina Barzilay, and Tommi Jaakkola. SE(3) diffusion model with application to protein backbone generation. In *International Conference on Machine Learning*, 2023.