

Determining the Milky Way gravitational potential without selection functions

TAAVET KALDA ¹ AND GREGORY M. GREEN ¹

¹*Max-Planck-Institut für Astronomie, Königstuhl 17, D-69117 Heidelberg, Germany*

ABSTRACT

Selection effects, such as interstellar extinction and varying survey depth, complicate efforts to determine the gravitational potential – and thus the distribution of baryonic and dark matter – throughout the Milky Way galaxy using stellar kinematics. We present a new variant of the “Deep Potential” method of determining the gravitational potential from a snapshot of stellar positions and velocities that does not require any modeling of spatial selection functions. Instead of modeling the full six-dimensional phase-space distribution function $f(\vec{x}, \vec{v})$ of observed kinematic tracers, we model the conditional velocity distribution $p(\vec{v} | \vec{x})$, which is unaffected by a purely spatial selection function. We simultaneously learn the gravitational potential $\Phi(\vec{x})$ and the underlying spatial density of the entire tracer population $n(\vec{x})$ – including unobserved stars – using the collisionless Boltzmann equation under the stationarity assumption. The advantage of this method is that unlike the spatial selection function, all of the quantities we model, $p(\vec{v} | \vec{x})$, $\Phi(\vec{x})$, and $n(\vec{x})$, typically vary smoothly in both position and velocity. We demonstrate that this “conditional” Deep Potential method is able to accurately recover the gravitational potential in a mock dataset with a complex three-dimensional dust distribution that imprints fine angular structure on the selection function. Because we do not need to model the spatial selection function, our new method can effectively scale to large, complex datasets while using relatively few parameters, and is thus well-suited to *Gaia* data.

Keywords: Milky Way dynamics (1051) — Stellar dynamics (1596) — Astrostatistics (1882) — Neural networks (1933)

1. INTRODUCTION

While the standard cosmological model involving cold dark matter (CDM) has been highly successful in predicting the large-scale structure of the Universe, the distribution of dark matter on sub-galactic scales, which is shaped both by baryonic feedback and the properties of the dark matter, remains less well understood (J. S. Bullock & M. Boylan-Kolchin 2017). As the only galaxy that we can study star by star, the Milky Way provides a unique opportunity to measure the distribution of dark matter on small scales. The gravitational potential is sourced by all matter; when combined with observationally informed models of the distribution of stars and gas, the potential thus contains information about the distribution of dark matter on small scales.

The most direct way of measuring the gravitational potential – by observing the accelerations of stars orbiting in the Galaxy – is at present beyond our technical

reach in all but a limited number of cases (H. Silverwood & R. Easther 2019; S. Chakrabarti et al. 2020, 2022; A. Moran et al. 2024). A number of methods have therefore been developed to infer the gravitational potential from a snapshot of stellar kinematics, typically involving assumptions about the stationarity and symmetry (*e.g.*, axisymmetry) of the Milky Way. Among these methods are Jeans modeling (J. H. Jeans 1915; J. Binney 1980; M. Cappellari 2008), Schwarzschild modeling (M. Schwarzschild 1979; R. C. E. van den Bosch et al. 2008), and action-based modeling (J. Binney 2012a,b; J. Bovy & H.-W. Rix 2013; W. H. Trick et al. 2016; E. Vasiliev 2019).

In recent years, the European Space Agency’s *Gaia* mission (Gaia Collaboration et al. 2016) has dramatically increased the quantity and quality of measured stellar parallaxes, proper motions, and radial velocities available. As of *Gaia* Data Release 3, high-quality, six-dimensional phase-space measurements are available for ≈ 34 million stars (Gaia Collaboration et al. 2023). The size of this dataset not only makes it possible – but also makes it necessary – to consider more flexible, nonpara-

metric models of Galactic structure that do justice to the richness of the data.

G. M. Green & Y.-S. Ting (2020) and G. M. Green et al. (2023) introduced a new method, “Deep Potential,” which makes minimal assumptions about the form of the gravitational potential and distribution function, using neural networks and normalizing flows (E. G. Tabak & C. V. Turner 2012; G. Papamakarios et al. 2019) to flexibly model these functions. Deep Potential uses the collisionless Boltzmann equation (CBE) to find the gravitational potential that renders the observed distribution function as stationary as possible. Subsequent work has extended this method and applied it to both mock and real *Gaia* datasets (J. An et al. 2021; A. P. Naik et al. 2022; M. R. Buckley et al. 2023; T. Kalda et al. 2024; E. Putney et al. 2024; T. Kalda & G. M. Green 2025; S. H. Lim et al. 2025).

In this work, we present a new variant of Deep Potential that obviates the need to model spatial selection functions, by learning the velocity distribution conditional on position, $p(\vec{v} | \vec{x})$, from the observed kinematic tracers, and then learning the true spatial density of the full tracer population (including those that are unobserved due to selection effects), $n(\vec{x})$, from the collisionless Boltzmann equation. We demonstrate that this method is able to accurately recover the gravitational potential on a mock dataset “observed” through a complex three-dimensional dust distribution. This method should alleviate a key challenge in applying Deep Potential to *Gaia* data, and may even open up the possibility of applying the method to spectroscopic surveys with more patchy sky coverage and more complex selection functions, such as SDSS-V (J. A. Kollmeier et al. 2017).

2. DERIVATION OF METHOD

In the original (“vanilla”) formulation of Deep Potential (G. M. Green & Y.-S. Ting 2020; G. M. Green et al. 2023), one fits a normalizing flow to represent the phase-space density $f(\vec{x}, \vec{v})$ of a population of kinematic tracers (*i.e.*, stars) with observed positions $\vec{x}$ and velocities $\vec{v}$ (*e.g.*, from *Gaia*). One then calculates the gradients $\partial_{\vec{x}} f(\vec{x}, \vec{v})$ and $\partial_{\vec{v}} f(\vec{x}, \vec{v})$ at a set of phase-space coordinates drawn from the distribution function. Using the collisionless Boltzmann equation (CBE), and given a model of the gravitational potential $\Phi(\vec{x})$, one can calculate the rate of change of the distribution function at any point in phase space:

$$\partial_t f(\vec{x}, \vec{v}) = \sum_{i=1}^3 \left[\frac{\partial \Phi(\vec{x})}{\partial x_i} \frac{\partial}{\partial v_i} - v_i \frac{\partial}{\partial x_i} \right] f(\vec{x}, \vec{v}). \quad (1)$$

Under the stationarity assumption, the left-hand side ($\partial_t f$) must vanish. Modeling the potential as a neu-

ral network (which takes a 3-vector representing $\vec{x}$ and outputs a scalar representing Φ), one minimizes (with respect to the network parameters) a loss function that penalizes both non-stationarity ($\partial_t f \neq 0$) and negative matter densities ($\nabla^2 \Phi < 0$):

$$L_{\text{vanilla}} = \left\langle \sinh^{-1}(\alpha |\partial_t f|) + \lambda \sinh^{-1}(\beta \max\{-\nabla^2 \Phi, 0\}) \right\rangle_{\vec{x}, \vec{v} \sim f}, \quad (2)$$

where α , β and λ (all set to unity throughout this work) are hyperparameters that control the shape and relative importance of the two penalties, and the average is taken over points in $\vec{x}$ and $\vec{v}$ drawn from the modeled distribution function. Minimizing L_{vanilla} yields a model of the gravitational potential $\Phi(\vec{x})$ of the system (up to a physically meaningless additive constant).

This approach requires a fair sample of phase-mixed kinematic tracers. In the presence of dust extinction, varying survey depth and other selection effects, one generally obtains a biased sample of the distribution function. To good approximation, the selection function is usually a function of position, but not of velocity: $p(\text{observed} | \vec{x}, \vec{v}) = S(\vec{x})$. We denote the distribution of observed kinematic tracers as

$$f_{\text{obs}}(\vec{x}, \vec{v}) = f(\vec{x}, \vec{v}) S(\vec{x}). \quad (3)$$

One approach to dealing with the selection function is to weight the observed tracers by $S^{-1}(\vec{x})$ when learning the distribution function, in order to obtain an unbiased estimate of the true distribution function (T. Kalda & G. M. Green 2025). However, this approach requires an accurate estimate of the selection function $S(\vec{x})$ at the location of each observed tracer, and amplifies noise in regions of low $S(\vec{x})$.

E. Putney et al. (2024) instead treats $\ln S(\vec{x})$ as an additional function to be learned from the CBE, analogously to $\Phi(\vec{x})$. Writing the CBE in terms of $\ln f(\vec{x}, \vec{v}) = \ln f_{\text{obs}}(\vec{x}, \vec{v}) - \ln S(\vec{x})$,

$$\partial_t \ln f(\vec{x}, \vec{v}) = \sum_{i=1}^3 \left\{ \frac{\partial \Phi(\vec{x})}{\partial x_i} \frac{\partial}{\partial v_i} \ln f_{\text{obs}}(\vec{x}, \vec{v}) - v_i \frac{\partial}{\partial x_i} [\ln f_{\text{obs}}(\vec{x}, \vec{v}) - \ln S(\vec{x})] \right\}. \quad (4)$$

Because the terms involving the gravitational potential and selection function depend differently on velocity, it is possible to simultaneously learn both using nearly the same loss function as before (Eq. 2). Replacing $\partial_t f$ with $\partial_t \ln f$ and adding a term that penalizes large deviations of $S(\vec{x})$ from unity (with corresponding hyperparameter γ_S , which we set to zero in this work), one obtains the

Method	Functions learned from		Notes
	Observed tracers	CBE	
“Vanilla”	$f(\vec{x}, \vec{v})$	$\Phi(\vec{x})$	$S(\vec{x})$ must be modeled based on prior knowledge.
“Selection function”	$f_{\text{obs}}(\vec{x}, \vec{v})$	$\Phi(\vec{x}), S(\vec{x})$	
“Conditional”	$p(\vec{v} \vec{x})$	$\Phi(\vec{x}), n(\vec{x})$	$n_{\text{obs}}(\vec{x})$ can be optionally learned from the tracers.

Table 1. Overview of the three Deep Potential variants discussed in this work, indicating which functions are learned from the observed kinematic tracers (using normalizing flows) and from the collisionless Boltzmann equation (CBE; using neural networks).

loss function:

$$L_{\text{selfn}} = \left\langle \sinh^{-1}(\alpha |\partial_t \ln f|) + \lambda \sinh^{-1}(\beta \max\{-\nabla^2 \Phi, 0\}) + \gamma_S (\ln S)^2 \right\rangle_{\vec{x}, \vec{v} \sim f_{\text{obs}}}. \quad (5)$$

One attractive aspect of this “selection function” approach is that it obviates the need to determine the selection function beforehand. Additionally, if one has an initial estimate of the selection function (*e.g.*, based on knowledge of the survey and dust properties), one can first learn an initial estimate of the distribution function using weighted kinematic tracers, and then treat $S(\vec{x})$ as a learned correction to the selection function. However, one difficulty in this approach is that the selection function $S(\vec{x})$ typically has highly complicated spatial structure (in particular, fine angular structure, as viewed from Earth, imprinted by dust extinction and varying survey depth), which is difficult for a neural network model to capture. The observed distribution function $f_{\text{obs}}(\vec{x}, \vec{v})$ also typically contains detailed angular structure that even large normalizing flow models struggle to accurately capture.

We therefore propose a method that only requires measuring the distribution of velocities, conditional on position. As in [M. R. Buckley et al. \(2023\)](#), one can decompose the distribution function as $f(\vec{x}, \vec{v}) = n(\vec{x})p(\vec{v} | \vec{x})$, where $n(\vec{x})$ is the true spatial density of kinematic tracers (including those that are not observed due to selection effects). As the conditional velocity distribution, $p(\vec{v} | \vec{x})$, is unaffected by a purely spatial selection function, it can be learned from the observed kinematic tracers using a conditional normalizing flow. The CBE can be rewritten in terms of $n(\vec{x})$ and $p(\vec{v} | \vec{x})$:

$$\partial_t \ln f(\vec{x}, \vec{v}) = \sum_{i=1}^3 \left\{ \frac{\partial \Phi(\vec{x})}{\partial x_i} \frac{\partial}{\partial v_i} \ln p(\vec{v} | \vec{x}) - v_i \frac{\partial}{\partial x_i} [\ln p(\vec{v} | \vec{x}) + \ln n(\vec{x})] \right\}. \quad (6)$$

One can model the true spatial density of tracers, $\ln n(\vec{x})$, as a neural network, and use the loss function

Eq. (7) to simultaneously learn both $\ln n(\vec{x})$ (up to an additive constant) and $\Phi(\vec{x})$ from the data. The key to this “conditional” approach is that the model of the selection function is replaced by a model of the true spatial distribution of the tracers. While the former typically has fine spatial (particularly angular) structure, the latter does not.

While this method appears to discard the information contained in the spatial distribution of the observed tracers, it has a number of major advantages. First, the observed spatial distribution of tracers may be more trouble than it is worth, as it contains selection-function-induced features that can “poison” the measured gradients of the distribution function. Second, as the true spatial density of the kinematic tracers is typically much smoother than the observed spatial density (as well as the selection function), it can be effectively modeled by a neural network. The conditional velocity distribution $p(\vec{v} | \vec{x})$ is also typically smooth, making it easier to accurately capture with a conditional normalizing flow. Finally, spatial information from the observed distribution of kinematic tracers, $n_{\text{obs}}(\vec{x})$, can be reintroduced in a manner that is less likely to contaminate the spatial gradients in the CBE. Specifically, one can add in an additional term $\langle [\ln n(\vec{x}) - \ln n_{\text{obs}}(\vec{x})]^2 \rangle$ to the loss function that encourages the modeled $n(\vec{x})$ to be similar to the observed spatial density:

$$L_{\text{conditional}} = \left\langle \sinh^{-1}(\alpha |\partial_t \ln f|) + \lambda \sinh^{-1}(\beta \max\{-\nabla^2 \Phi, 0\}) + \gamma_n (\ln n - \ln n_{\text{obs}})^2 \right\rangle_{\vec{x}, \vec{v} \sim f_{\text{obs}}}. \quad (7)$$

If an initial estimate of $S(\vec{x})$ is available, then in the above loss, $n_{\text{obs}}(\vec{x})$ can be replaced by a normalizing flow model of the true spatial density $n(\vec{x})$, learned from weighted samples of the observed positions of the tracers. In this work, we set $\gamma_n = 0$.

Table 1 summarizes the three variants of Deep Potential, indicating which functions are learned from the observed kinematic tracers and which are learned from the CBE.

3. TEST CASE: MOCK GALAXY WITH 3D DUST DISTRIBUTION

Here, we compare the effectiveness of the three variants of Deep Potential sketched out above (“vanilla,” “selection function,” and “conditional”) on a mock dataset.

3.1. Data generation

We generate mock data based on a Plummer sphere model (H. C. Plummer 1911) with a complex three-dimensional distribution of dust. The Plummer sphere potential depends on radius r as

$$\Phi(r) = -\frac{GM_0}{(r^2 + a^2)^{\frac{1}{2}}}, \quad (8)$$

where $G = 1$ is the gravitational constant, M_0 is the total mass of the system, and a is a scale radius. We choose $a = 5$ kpc and set M_0 such that the density at $r = 0$ is unity (in arbitrary mass units per kpc³). The Plummer potential admits a stationary distribution function that depends only on specific energy $E = \frac{1}{2}v^2 + \Phi$:

$$f(\vec{x}, \vec{v}) \propto \begin{cases} (-E)^{\frac{7}{2}}, & E < 0 \\ 0, & E \geq 0 \end{cases}. \quad (9)$$

We generate a three-dimensional distribution of dust by modeling the logarithm of the extinction density, $\ln[\rho_{\text{dust}}/(\text{mag kpc}^{-1})]$, as a Gaussian process with power spectrum $P(k) \propto k^{-3}$ (with a high-frequency cutoff at $k \approx 4$ kpc⁻¹), a mean of -0.5 , and unit variance. The left panels of Fig. 1 show a two-dimensional slice through the dust extinction density and the integrated extinction. We draw stars from the distribution function, subject to the constraint $r < 8$ kpc, and assign each star an absolute magnitude M drawn from a normal distribution with mean $\bar{M} = 0.5$ mag and standard deviation $\sigma_M = 2$ mag. We place the observer at $r = 0$, and calculate the apparent magnitude of each star, based on its absolute magnitude, distance modulus μ and extinction A . We impose an apparent magnitude cut of $m_{\text{lim}} = 15$. The implied selection function is then a function of distance modulus μ and extinction A :

$$\begin{aligned} S(\mu, A) &= \int_{-\infty}^{m_{\text{lim}} - \mu - A} p(M | \bar{M}, \sigma_M) dM \\ &= \frac{1}{2} \left[1 + \text{erf} \left(\frac{m_{\text{lim}} - \bar{M} - \mu - A}{\sqrt{2} \sigma_M} \right) \right]. \end{aligned} \quad (10)$$

We repeatedly draw stars until we obtain 2^{22} (~ 4.19 million) observed stars. The right panels of Fig. 1 show a two-dimensional slice through the selection function and

a projection of the distribution of observed stars. Both are finely structured due to dust extinction, illustrating the difficulty of capturing these structures with neural networks.

3.2. Results with three approaches

We run all three variants of Deep Potential on our mock dataset. We train on 75% of the data and split off the remaining 25% as a validation set. For each variant, we use an automatic Bayesian hyperparameter optimizer, **optuna** (T. Akiba et al. 2019), to determine the model architecture and training procedure that minimize the unregularized validation losses (both when learning the distribution function and when learning the potential). We train continuous normalizing flows using Conditional Flow Matching (Y. Lipman et al. 2022; A. Tong et al. 2023). We model all fields learned using the CBE and the neural networks used by Flow Matching as ResNets (K. He et al. 2016). The details of our model architectures and training procedures are described in Appendix A.

For each method, we obtain a model of the gravitational potential field $\Phi(\vec{x})$, from which we can calculate the gravitational density field $\rho(\vec{x}) = \nabla^2 \Phi / 4\pi G$. For the “selection function” method, we also obtain a model of the selection function $S(\vec{x})$, and for the “conditional” method, we obtain a model of the true spatial density of tracers $n(\vec{x})$, which includes the unobserved tracers (*e.g.*, those obscured by dust).

Fig. 2 shows the recovered gravitational densities and their residuals (vs. the true density), both as a function of radius r and the true selection function $S(\vec{x})$. The “vanilla” method, which attempts to recover the true selection function by upweighting each observed tracer by $1/S(\vec{x})$, struggles to recover the density as the number of observed tracers falls off beyond $r \gtrsim 2$ kpc or for $S(\vec{x}) \lesssim 0.5$. Both the “selection function” and “conditional” methods are able to accurately recover the gravitational density out to $r \sim 4$ kpc. The “selection function” method recovers $S(\vec{x})$ to within $\sim 10\%$ accuracy for $S(\vec{x}) \gtrsim 0.25$, while the “conditional” method recovers $n(\vec{x})$ to within $\sim 10\%$ accuracy for all $S(\vec{x})$. Fig. 3 shows the recovery of $S(\vec{x})$ and $n(\vec{x})$ by the “selection function” and “conditional” methods, respectively, in the $z = 0$ plane. Because $S(\vec{x})$ contains fine angular structure, a far larger network (~ 1.4 million parameters) is required to model it accurately than is required for $n(\vec{x})$ (~ 47 thousand parameters). The residuals in $S(\vec{x})$ appear as long radial spurs, illustrating the difficulty of recovering the angular structures imprinted by dust extinction. In contrast, the fractional residuals in $n(\vec{x})$ are smaller and smoother in $\vec{x}$. The difficulty of

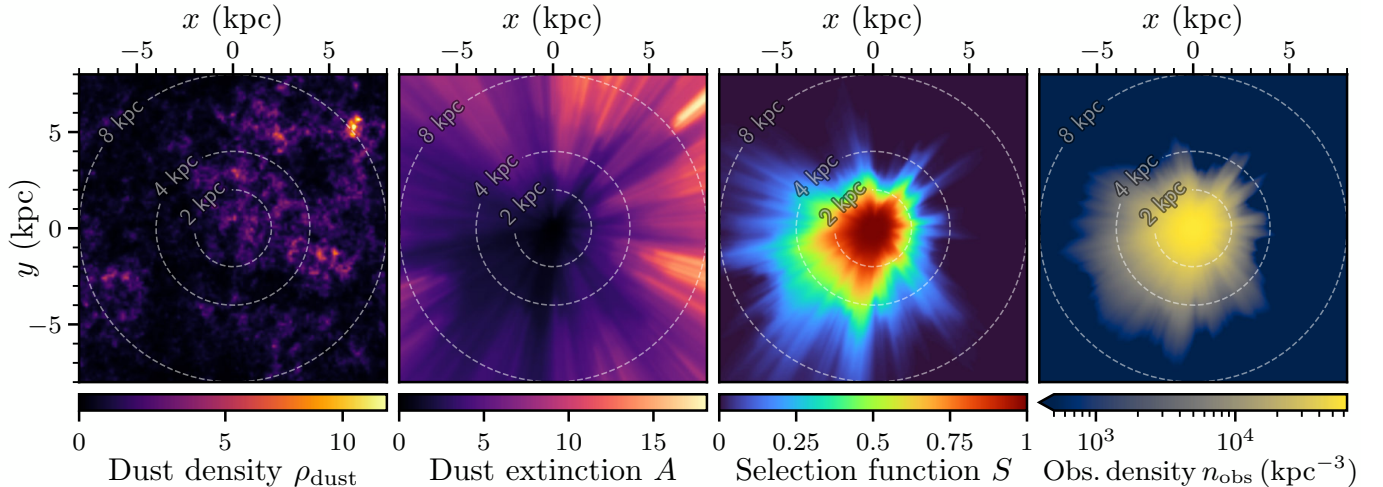


Figure 1. Two-dimensional slices (the $z = 0$ plane) through the three-dimensional dust extinction density (left) and integrated extinction (as viewed from the origin; middle left), the resulting selection function (middle right), and the observed spatial distribution of stars (right). The observer is assumed to be at the origin. Dust extinction imprints fine angular structure on the selection function and the observed spatial distribution of stars, which can be difficult to capture with neural networks.

accurately capturing the angular structure in $S(\vec{x})$ and $n_{\text{obs}}(\vec{x})$ is further illustrated in Appendix B.

Unsurprisingly, all three methods deteriorate when the observed tracers become too sparse, as can be seen in the third row of Fig. 2, with the “vanilla” method deteriorating the most rapidly. Given that the “conditional” method requires orders of magnitude fewer parameters to achieve the same accuracy as the “selection function” method, we expect it to outperform the latter on real data with even more spatially complex selection functions.

4. CONCLUSIONS

We have demonstrated a new variant of the “Deep Potential” approach to determining the gravitational potential of a system (such as the Milky Way) from a frozen snapshot of kinematic tracers (*e.g.*, stars), which does not require modeling of finely structured spatial selection functions. Instead of modeling the full six-dimensional distribution function of the observed tracers, $f_{\text{obs}}(\vec{x}, \vec{v})$, we model the conditional velocity distribution $p(\vec{v} | \vec{x})$, which is unaffected by a purely spatial selection function. We simultaneously learn the gravitational potential $\Phi(\vec{x})$ and the underlying spatial density of the entire tracer population $n(\vec{x})$ using the collisionless Boltzmann equation under the stationarity assumption. The advantage of this method is that all of the quantities we model – $p(\vec{v} | \vec{x})$, $\Phi(\vec{x})$, and $n(\vec{x})$ – typically vary smoothly in position and velocity.

We have shown that both the “conditional” and “selection function” variants of the Deep Potential method are able to accurately recover the gravitational potential in a mock dataset with a complex three-dimensional

dust distribution that imprints fine angular structure on the selection function. In our mock data, the dust distribution has structure down to a scale of ~ 0.25 kpc, which corresponds to an angular scale of 15 deg at a distance of 1 kpc, or 1.8 deg at 8 kpc. In the real world, dust clouds have structure all the way down to sub-parsec scales. As the selection function becomes more complex, we expect the “conditional” method to outperform the “selection function” method, as the latter will struggle to accurately capture fine angular structure in both the selection function and in $\ln n_{\text{obs}}(\vec{x})$. The “conditional” method does not require learning either of these fields. Already at the resolution of our mock three-dimensional dust map, the model representing $\ln S(\vec{x})$ requires ~ 1.4 million parameters. In contrast, the model representing the full underlying tracer population, $\ln n(\vec{x})$, which is unaffected by selection effects, requires just ~ 47 thousand parameters (see Table 2). This discrepancy will become larger as the three-dimensional dust distribution becomes more finely structured. For this reason, we suggest that the “conditional” method is a promising approach for recovering the gravitational potential from real *Gaia* data.

This “conditional” Deep Potential method could be applied not only to *Gaia* data, but also to spectroscopic surveys with even more complex selection functions or uneven sky coverage, such as SDSS-V. The key requirement is that $p(\vec{v} | \vec{x})$ vary smoothly enough with position $\vec{x}$ that the conditional normalizing flow can effectively interpolate between patchily observed regions of the Galaxy.

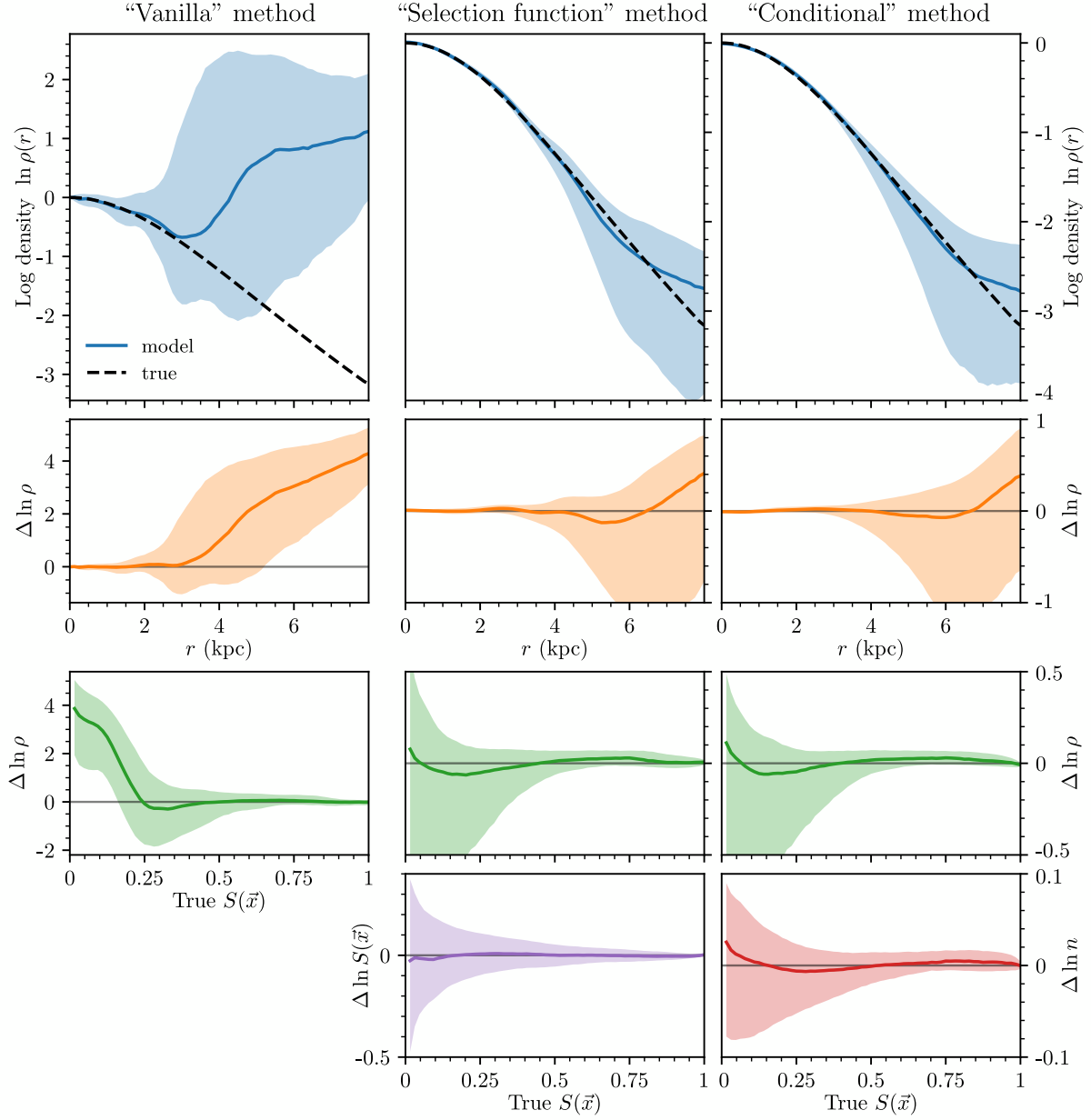


Figure 2. The gravitational densities ($\rho = \nabla^2 \Phi / 4\pi G$) and their residuals (vs. the true densities), as recovered by the three variants of Deep Potential. **Top row:** $\ln \rho$ as a function of radius r , with the solid curves showing the median (in spherical shells) and the shaded envelopes enclosing the 16th to 84th percentiles. The dotted curves show the true density. **Second row:** Residuals $\Delta \ln \rho = \ln \rho_{\text{model}} - \ln \rho_{\text{true}}$ as a function of r . **Third row:** Residuals as a function of the true selection function $S(\vec{x})$, illustrating the deterioration in the recovered density as the fraction of stars observed decreases. **Bottom row:** Residuals in the selection function $S(\vec{x})$ (as recovered by the “selection function” method) and true spatial density of tracers $n(\vec{x})$ (as recovered by the “conditional” method), each as a function of the true $S(\vec{x})$. In the top three rows, the vertical axes are the same for the “selection function” and “conditional” methods.

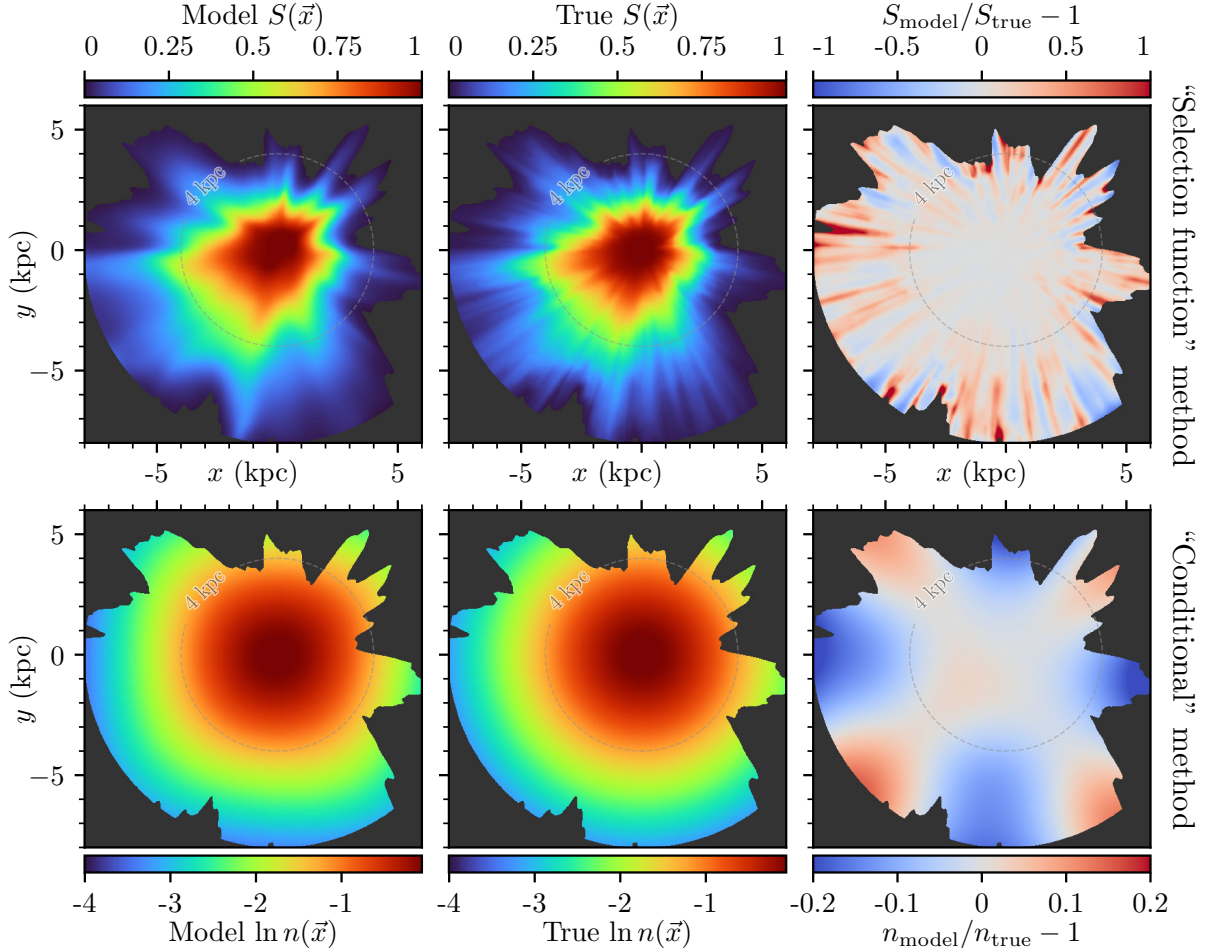


Figure 3. Recovery of the selection function and true spatial density of tracers by the “selection function” and “conditional” methods, respectively, in a two-dimensional slice through the $x-y$ plane (at $z=0$). **Top row:** The recovered selection function (left), true selection function (middle), and residuals (right) for the “selection function” method. **Bottom row:** The recovered true spatial density of tracers (left), true spatial density of tracers (middle), and residuals (right) for the “conditional” method. Dust imprints fine angular structure on the selection function, necessitating much larger models (~ 1.4 million parameters in this case) than are needed to capture the much smoother true spatial density of tracers (~ 47 thousand parameters). As seen in the top-right panel, the selection-function residuals appear as long radial spurs, illustrating the difficulty of recovering the angular structures in $S(\vec{x})$.

ACKNOWLEDGMENTS

GG and TK are supported by a Sofja Kovalevskaja Award to GG from the Alexander von Humboldt Foundation.

Computations were performed on the HPC system Vera at the Max Planck Computing and Data Facility.

AUTHOR CONTRIBUTIONS

TK originated the idea of using the conditional distribution function $p(\vec{v} | \vec{x})$ to determine the gravitational potential, derived the mathematical formalism, wrote the code, ran Deep Potential on the test cases, generated most of the figures, and wrote the appendices. GG cre-

ated the mock data for the test cases, generated Fig. 1, wrote most of the initial draft of the manuscript, and provided advice and feedback on the method, code, and figures.

This work makes use of the following software packages: `astropy` (Astropy Collaboration et al. 2013, 2018, 2022), `JAX` (J. Bradbury et al. 2018), `matplotlib` (J. D. Hunter 2007), `numpy` (C. R. Harris et al. 2020), `optuna` (T. Akiba et al. 2019), `python` (G. Van Rossum & F. L. Drake 2009), and `scipy` (P. Virtanen et al. 2020).

Software citation information aggregated using [The Software Citation Station](#) (T. Wagg & F. S. Broekgaarden 2024; T. Wagg et al. 2025).

APPENDIX

A. MODEL ARCHITECTURES AND TRAINING PROCEDURES

We implement all models using the JAX framework (J. Bradbury et al. 2018). The hyperparameters for the neural network architectures and the training procedures, determined via the `optuna` optimization framework (T. Akiba et al. 2019), are summarized in Table 2.

All of the scalar fields and normalizing flow vector fields are implemented as Residual Networks (ResNets). Each block returns $x + f(x)$, where $f(x)$ consists of a linear layer, followed by a $\text{SiLU}(x) = x/(1 + \exp(-x))$ activation function and another linear layer. All of the models are optimized using the `AdamW` optimizer. We employ a cosine annealing learning rate schedule, which decays the learning rate from the initial value specified in Table 2 to close to zero over the course of training.

A.1. Normalizing flows

We employ Continuous Normalizing Flows (CNFs; see Section 4 of G. Papamakarios et al. 2019) to model the probability densities $n_{\text{obs}}(\vec{x})$ and $p(\vec{v} \mid \vec{x})$. Unlike discrete flows that construct a transformation via a sequence of discrete layers, CNFs define the mapping from a base distribution (usually a unit Gaussian, $p_0(\vec{z})$) to the target distribution ($p_1(\vec{z})$) via an Ordinary Differential Equation (ODE), $d\vec{z}/dt = \vec{u}_\theta(\vec{z}, t)$, where $\vec{u}_\theta$ is a learnable vector field parameterized by a neural network with weights θ , and t is an integration variable (not to be confused with the real time of any physical system). The log-probability of the target distribution is defined by the transport equation

$$\ln p_1(\vec{z}) = \ln p_0(\vec{z}) - \int_{t=0}^{t=1} dt \nabla \cdot \vec{u}_t(\vec{z}, t). \quad (\text{A1})$$

A significant downside of CNFs is the computational cost of integrating the ODE during inference, as it requires the use of auto-differentiable integrators to compute the gradients in the log-density.

To train the CNF, we employ Conditional Flow Matching (CFM; Y. Lipman et al. 2022; A. Tong et al. 2023), a recent method that is gaining traction, largely due to its ability to train the vector field in an inference-free manner, which greatly speeds up training and facilitates the use of more complex models. The method regresses the trainable vector field $\vec{u}_\theta(\vec{z}, t)$ onto a target vector field that draws straight paths between samples from the base distribution (noise), and the samples drawn from the data distribution. This avoids the costly ODE integration required during training in maximum-likelihood approaches such as FFJORD (W. Grathwohl et al. 2018).

As the choice of vector field to transform from the base to the target distribution is highly degenerate with θ , different forms of regularization were explored to straighten the flow trajectories and reduce numerical stiffness during inference. Perhaps the most promising was the use of minibatch optimal transport (minibatch-OT), which modifies the Conditional Flow Matching sampling scheme to respect the optimal transport map between the base and target distributions. Since the full optimal transport scheme is computationally expensive (the Hungarian algorithm scales with $\mathcal{O}(N^3)$), we perform it in minibatches of size 256. In practice, 512 epochs’ worth of minibatch OT pairings were precomputed and later used via a dataloader during the training of the vector field. We found 512 to be a sufficient threshold, as the sampling time provides an extra layer of stochasticity, provided that after 512 training epochs, the pairings were reused and reshuffled. Unfortunately, minibatch-OT does not trivially extend to conditional normalizing flows (H. K. Cheng & A. Schwing 2025) and hence was only employed for the training of the observed spatial density.

We additionally employed kinetic regularization which punishes the “average kinetic energy” of the trajectories, penalizing $\int \|\vec{u}_\theta\| dt$, encouraging straighter flows, and Jacobian regularization, which punishes the rate of change of volume in t , by penalizing the Frobenius norm of $\partial u_\theta(\vec{z}_i, t)/\partial z_j$ (C. Finlay et al. 2020). However, `optuna` decided not to make use of Jacobian regularization, likely due to its high computational cost and marginal gains.

Before feeding t into the vector field network, we embed it using a sinusoidal embedding of dimension `time_embedding_dim`, then concatenate it with the other inputs. The frequencies are spaced geometrically between 1 and 1024. Conditioning parameters, such as the velocity, were handled by linearly appending them to the input; however, more sophisticated methods exist (E. Perez et al. 2017).

To summarize, the CFM loss function is computed by sampling $t \in [0, 1]$ (we let t be the square root of a uniform random variable on $[0, 1]$), sampling z_0, z_1 from the base and target distributions respectively (optionally with a

minibatch-OT map), such that the loss for one draw is given by

$$\mathcal{L}_{\text{CFM}} = \|u_\theta(t, z_t) - u_t(z_t)\|^2 + \lambda_{\text{kinetic}} \|u_\theta(t, z_t)\|^2 + \lambda_{\text{Jacobian}} \|\partial u(\vec{z}_i, t)/\partial z_j\|_F^2, \quad (\text{A2})$$

$$z_t = tz_1 + (1-t)z_0, \quad (\text{A3})$$

$$u_t = z_1 - z_0. \quad (\text{A4})$$

For all three methods, we model the full six-dimensional distribution function using the decomposition $f_{\text{obs}}(\vec{x}, \vec{v}) = n_{\text{obs}}(\vec{x})p(\vec{v} | \vec{x})$, where both distributions are represented as continuous normalizing flows. For the “vanilla” method, as the spatial density $n_{\text{obs}}(\vec{x})$ should represent an estimate of the true density, we must weight the samples by the inverse of the selection function. As $S(\vec{x})$ approaches zero, the weights will diverge. To avoid this, we clip the weights to be ≤ 10 . For the “selection function” and “conditional” methods, we do not need to weight the samples, as in the former case, the selection function is learned from the CBE, while in the latter case, we only learn the conditional velocity distribution $p(\vec{v} | \vec{x})$ from the samples and later learn the true spatial distribution of the tracers $n(\vec{x})$ from the CBE.

A.2. Scalar function networks

The gravitational potential $\Phi(\vec{x})$, the selection function correction $\ln S(\vec{x})$, and the true number density $\ln n(\vec{x})$ are all scalar functions that take 3-vectors (representing $\vec{x}$) as inputs. To assist the networks in capturing the complex angular structure of the selection function (as visualized in Figs. 1 and 4), we augment the input coordinates for the $\ln S(\vec{x})$ and $n_{\text{obs}}(\vec{x})$ networks. In addition to Cartesian coordinates, we compute real Spherical Harmonics $Y_l^m(\theta, \phi)$ up to a maximum degree of `SH_embedding_lmax`. These harmonic features are concatenated with the input vector, allowing the network to explicitly leverage angular basis functions suitable for modeling extinction features on the sky. We found that while higher degrees of spherical harmonics help with training, values ≥ 10 increase numerical stiffness and make the inference unstable. Additionally, perhaps unsurprisingly, spherical harmonics were not utilized for the learning of $\ln n(\vec{x})$ because it is less sensitive to angular features arising from extinction.

We use `optuna` to perform a Bayesian search over the hyperparameter space. For the normalizing flows, the objective is to maximize the log-likelihood of the validation set. For the CBE-based networks (Φ , S , n), the objective is to minimize the total physics loss (stationarity penalties plus regularization terms; see Eqs. 2, 5 and 7) on the validation set.

A.3. Computational costs

Compared to T. Kalda & G. M. Green (2025), the performance of the normalizing flows has undergone dramatic improvements. From most to least important, the improvements arose from switching from FFJORD to Conditional Flow Matching, using ResNets and positional embeddings (instead of Multilayer Perceptrons), transitioning from TensorFlow 2 (M. Abadi et al. 2015) to JAX, and using the SiLU activation (instead of tanh). The speed-up was observed to be larger than a factor of 100 for more complex flows. This speed-up allowed us to use `optuna` (requiring many trials) to optimize hyperparameters. The computational times of different components of the model on an Nvidia A100 are listed in Table 2.

B. RECOVERY OF SELECTION AND DISTRIBUTION FUNCTIONS

Here, we present additional figures illustrating the recovery of various fields by the “selection function” and “conditional” methods.

Fig. 4 shows a sky plot (at $r = 4\text{ kpc}$) of the “selection function” method’s recovery of the selection function in comparison with the ground truth. Dust extinction imprints fine angular patterns on the selection function. In particular, highly extincted areas have low numbers of “detected” mock stars, and hence recovery of the selection function is limited by shot noise.

Fig. 5 compares our normalizing flow representations of $n_{\text{obs}}(\vec{x})$ and $p(\vec{v} | \vec{x})$ with the ground truth. The conditional velocity distribution is recovered to within $\sim 2\%$ accuracy all the way down to $S(\vec{x}) \sim 0.01$. In contrast, due to the complex patterns imprinted by the spatial selection function, the residuals in the recovered $n_{\text{obs}}(\vec{x})$ are larger and display a large positive trend at low $S(\vec{x})$. The “conditional” method only makes use of the conditional velocity distribution, while the “selection function” method additionally relies on the modeled $n_{\text{obs}}(\vec{x})$.

REFERENCES

- | | |
|--|--|
| <p>Abadi, M., Agarwal, A., Barham, P., et al. 2015, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, https://www.tensorflow.org/</p> | <p>Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. 2019, in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining</p> |
|--|--|

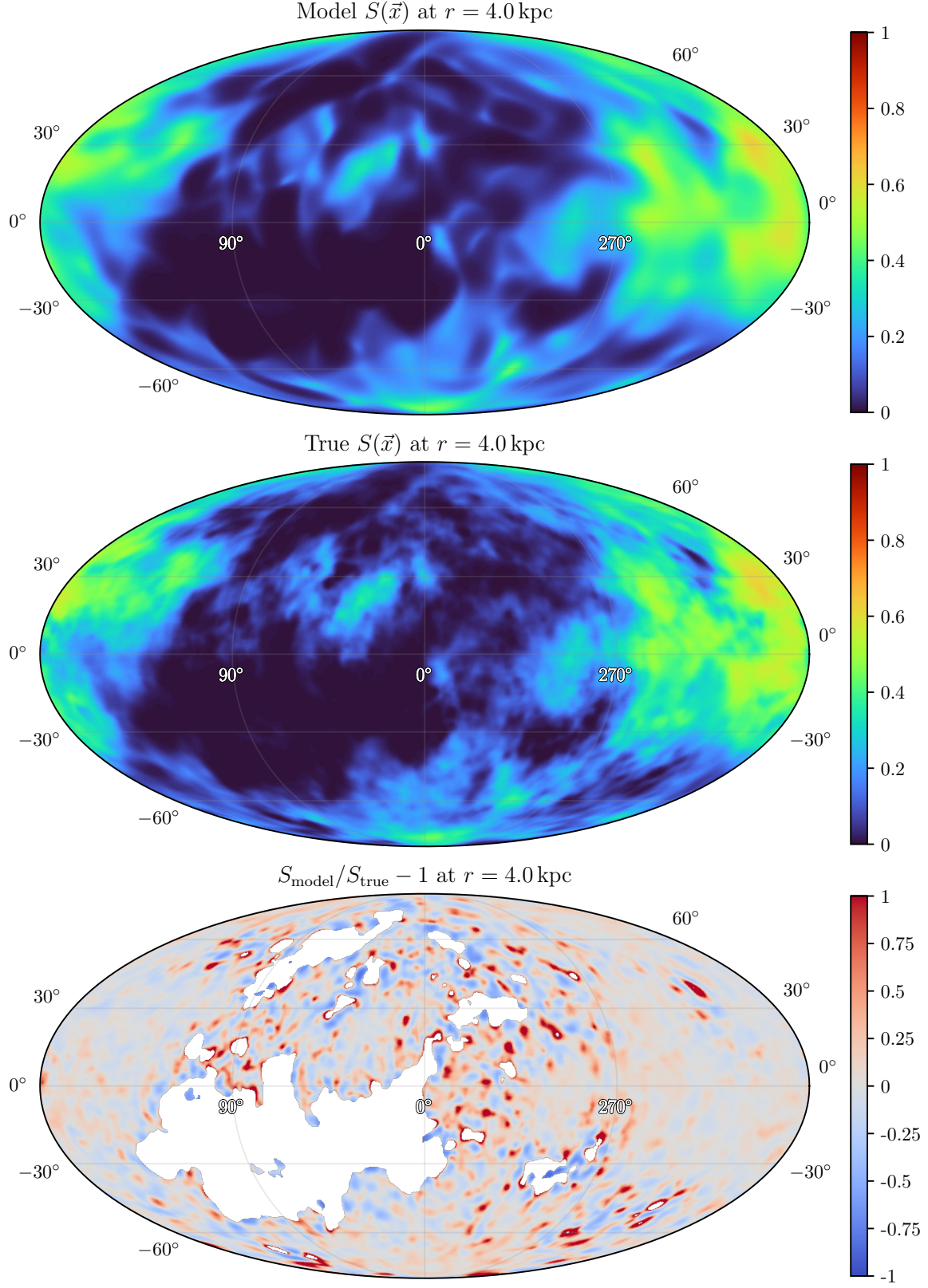


Figure 4. Sky plots of the “selection function” method’s prediction of the selection function $S(\vec{x})$, the true selection function, and their fractional residuals on the surface of a sphere of radius 4 kpc, centered on the mock observer. Areas with $S_{\text{true}} < 0.01$ are masked. The differences can be attributed to shot noise in areas of low completeness, and to the difficulty of capturing fine angular structure with a neural network. Nevertheless, with a mock selection function with structure down to ~ 0.25 kpc scales, our selection-function implementation performs well, albeit at the cost of large model sizes (here, ~ 1.4 million parameters).

	Parameter	Method		
		Vanilla	Selection function	Conditional
General	Base Network	ResNet		
	Activation	SiLU		
	Optimizer	AdamW		
	LR Scheduler	Cosine annealing		
Minibatch-OT	epochs	512		
	batch_size	256		
$n_{\text{obs}}(\vec{x})$	width $\times$ depth	115×6		
	time_embedding_dim	50		
	SH_embedding_lmax	3		
	learning_rate	2.0×10^{-3}		
	epochs	1558		
	# of parameters	166 656		
	Training time	47 min training + 90 min sampling		
$p(\vec{v} \mid \vec{x})$	width $\times$ depth	57×10		
	Kinetic regularization	2.25×10^{-5}		
	learning_rate	1.16×10^{-3}		
	epochs	852		
	# of parameters	69 580		
	Training time	23 min training + 45 min sampling		
$\Phi(\vec{x})$	width $\times$ depth	58×6	29×8	29×8
	ℓ_2 regularization	0.23	0.86	0.31
	learning_rate	5.7×10^{-3}	7.9×10^{-3}	8.4×10^{-3}
	epochs	155	116	116
	# of parameters	42 054	14 533	14 533
	Training time	12 min	–	–
$S(\vec{x}), n(\vec{x})$	width $\times$ depth	–	240×12	75×4
	SH_embedding_lmax	–	3	0
	ℓ_2 regularization	–	0.73	0.16
	# of parameters	–	1 398 970	46 585
	Training time	–	20 min	12 min

Table 2. Hyperparameters of the different “Deep Potential” methods used in this study. The hyperparameters were tuned using the `optuna` Python package. The training time benchmarks were computed on an Nvidia A100 GPU. The $n_{\text{obs}}(\vec{x})$ and $p(\vec{v} \mid \vec{x})$ flows were trained in common for all three methods, while the $\Phi(\vec{x})$ networks were trained separately for each method. For the “selection function” and “conditional” methods, we used the same network structure for Φ , determined by `optuna` (based on the “selection function” method), but allowed `optuna` to determine a separate learning rate and regularization strength for the “conditional” case. The $S(\vec{x})$ and $n(\vec{x})$ networks were only trained for the “selection function” and “conditional” methods, respectively. These last two networks were trained simultaneously with the potential, and thus share learning rates and training times with Φ .

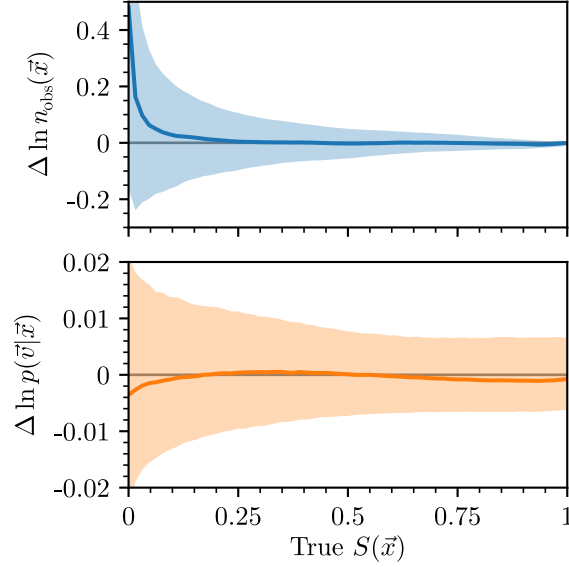


Figure 5. Top panel: Residuals between our normalizing flow model of the distribution of observed kinematic tracers and the ground truth: $\Delta \ln n_{\text{obs}}(\vec{x}) \equiv \ln n_{\text{obs}}(\vec{x}) - \ln [n(\vec{x}) S(\vec{x})]$, where $n(\vec{x})$ and $S(\vec{x})$ are the true spatial density of tracers (including those that are not observed) and selection function, respectively. **Bottom panel:** Residuals between our conditional normalizing flow model of the velocity distribution and the ground truth. The conditional distribution $p(\vec{v} | \vec{x})$ is recovered to within $\sim 2\%$, even for $S(\vec{x})$ as low as ~ 0.01 . Due to the complex nature of the spatial selection function, the fractional errors in the recovered $n_{\text{obs}}(\vec{x})$ are larger, and the median residual diverges substantially from zero at low $S(\vec{x})$.

- An, J., Naik, A. P., Evans, N. W., & Burrage, C. 2021, MNRAS, 506, 5721, doi: [10.1093/mnras/stab2049](https://doi.org/10.1093/mnras/stab2049)
- Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., et al. 2013, A&A, 558, A33, doi: [10.1051/0004-6361/201322068](https://doi.org/10.1051/0004-6361/201322068)
- Astropy Collaboration, Price-Whelan, A. M., Sipőcz, B. M., et al. 2018, AJ, 156, 123, doi: [10.3847/1538-3881/aabc4f](https://doi.org/10.3847/1538-3881/aabc4f)
- Astropy Collaboration, Price-Whelan, A. M., Lim, P. L., et al. 2022, ApJ, 935, 167, doi: [10.3847/1538-4357/ac7c74](https://doi.org/10.3847/1538-4357/ac7c74)
- Binney, J. 1980, MNRAS, 190, 873, doi: [10.1093/mnras/190.4.873](https://doi.org/10.1093/mnras/190.4.873)
- Binney, J. 2012a, MNRAS, 426, 1324, doi: [10.1111/j.1365-2966.2012.21757.x](https://doi.org/10.1111/j.1365-2966.2012.21757.x)
- Binney, J. 2012b, MNRAS, 426, 1328, doi: [10.1111/j.1365-2966.2012.21692.x](https://doi.org/10.1111/j.1365-2966.2012.21692.x)
- Bovy, J., & Rix, H.-W. 2013, ApJ, 779, 115, doi: [10.1088/0004-637X/779/2/115](https://doi.org/10.1088/0004-637X/779/2/115)
- Bradbury, J., Frostig, R., Hawkins, P., et al. 2018, JAX: composable transformations of Python+NumPy programs, 0.3.13 <http://github.com/google/jax>
- Buckley, M. R., Lim, S. H., Putney, E., & Shih, D. 2023, MNRAS, 521, 5100, doi: [10.1093/mnras/stad843](https://doi.org/10.1093/mnras/stad843)
- Bullock, J. S., & Boylan-Kolchin, M. 2017, ARA&A, 55, 343, doi: [10.1146/annurev-astro-091916-055313](https://doi.org/10.1146/annurev-astro-091916-055313)
- Cappellari, M. 2008, MNRAS, 390, 71, doi: [10.1111/j.1365-2966.2008.13754.x](https://doi.org/10.1111/j.1365-2966.2008.13754.x)
- Chakrabarti, S., Stevens, D. J., Wright, J., et al. 2022, ApJL, 928, L17, doi: [10.3847/2041-8213/ac5c43](https://doi.org/10.3847/2041-8213/ac5c43)
- Chakrabarti, S., Wright, J., Chang, P., et al. 2020, ApJL, 902, L28, doi: [10.3847/2041-8213/abb9b5](https://doi.org/10.3847/2041-8213/abb9b5)
- Cheng, H. K., & Schwing, A. 2025, arXiv e-prints, arXiv:2503.10636, doi: [10.48550/arXiv.2503.10636](https://doi.org/10.48550/arXiv.2503.10636)
- Finlay, C., Jacobsen, J.-H., Nurbekyan, L., & Oberman, A. M. 2020, arXiv e-prints, arXiv:2002.02798, doi: [10.48550/arXiv.2002.02798](https://doi.org/10.48550/arXiv.2002.02798)
- Gaia Collaboration, Prusti, T., de Bruijne, J. H. J., et al. 2016, A&A, 595, A1, doi: [10.1051/0004-6361/201629272](https://doi.org/10.1051/0004-6361/201629272)
- Gaia Collaboration, Vallenari, A., Brown, A. G. A., et al. 2023, A&A, 674, A1, doi: [10.1051/0004-6361/202243940](https://doi.org/10.1051/0004-6361/202243940)
- Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., & Duvenaud, D. 2018, arXiv e-prints, arXiv:1810.01367, doi: [10.48550/arXiv.1810.01367](https://doi.org/10.48550/arXiv.1810.01367)
- Green, G. M., & Ting, Y.-S. 2020, arXiv e-prints, arXiv:2011.04673, doi: [10.48550/arXiv.2011.04673](https://doi.org/10.48550/arXiv.2011.04673)
- Green, G. M., Ting, Y.-S., & Kamdar, H. 2023, ApJ, 942, 26, doi: [10.3847/1538-4357/aca3a7](https://doi.org/10.3847/1538-4357/aca3a7)
- Harris, C. R., Millman, K. J., van der Walt, S. J., et al. 2020, Nature, 585, 357, doi: [10.1038/s41586-020-2649-2](https://doi.org/10.1038/s41586-020-2649-2)
- He, K., Zhang, X., Ren, S., & Sun, J. 2016, in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1, doi: [10.1109/CVPR.2016.90](https://doi.org/10.1109/CVPR.2016.90)
- Hunter, J. D. 2007, Computing in Science & Engineering, 9, 90, doi: [10.1109/MCSE.2007.55](https://doi.org/10.1109/MCSE.2007.55)

- Jeans, J. H. 1915, MNRAS, 76, 70,
doi: [10.1093/mnras/76.2.70](https://doi.org/10.1093/mnras/76.2.70)
- Kalda, T., & Green, G. M. 2025, ApJ, 992, 84,
doi: [10.3847/1538-4357/adf8ea](https://doi.org/10.3847/1538-4357/adf8ea)
- Kalda, T., Green, G. M., & Ghosh, S. 2024, MNRAS, 527,
12284, doi: [10.1093/mnras/stae011](https://doi.org/10.1093/mnras/stae011)
- Kollmeier, J. A., Zasowski, G., Rix, H.-W., et al. 2017,
arXiv e-prints, arXiv:1711.03234,
doi: [10.48550/arXiv.1711.03234](https://doi.org/10.48550/arXiv.1711.03234)
- Lim, S. H., Putney, E., Buckley, M. R., & Shih, D. 2025,
JCAP, 2025, 021, doi: [10.1088/1475-7516/2025/01/021](https://doi.org/10.1088/1475-7516/2025/01/021)
- Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., &
Le, M. 2022, arXiv e-prints, arXiv:2210.02747,
doi: [10.48550/arXiv.2210.02747](https://doi.org/10.48550/arXiv.2210.02747)
- Moran, A., Mingarelli, C. M. F., Van Tilburg, K., & Good,
D. 2024, PhRvD, 109, 123015,
doi: [10.1103/PhysRevD.109.123015](https://doi.org/10.1103/PhysRevD.109.123015)
- Naik, A. P., An, J., Burrage, C., & Evans, N. W. 2022,
MNRAS, 511, 1609, doi: [10.1093/mnras/stac153](https://doi.org/10.1093/mnras/stac153)
- Papamakarios, G., Nalisnick, E., Jimenez Rezende, D.,
Mohamed, S., & Lakshminarayanan, B. 2019, arXiv
e-prints, arXiv:1912.02762,
doi: [10.48550/arXiv.1912.02762](https://doi.org/10.48550/arXiv.1912.02762)
- Perez, E., Strub, F., de Vries, H., Dumoulin, V., &
Courville, A. 2017, arXiv e-prints, arXiv:1709.07871,
doi: [10.48550/arXiv.1709.07871](https://doi.org/10.48550/arXiv.1709.07871)
- Plummer, H. C. 1911, MNRAS, 71, 460,
doi: [10.1093/mnras/71.5.460](https://doi.org/10.1093/mnras/71.5.460)
- Putney, E., Shih, D., Lim, S. H., & Buckley, M. R. 2024,
arXiv e-prints, arXiv:2412.14236,
doi: [10.48550/arXiv.2412.14236](https://doi.org/10.48550/arXiv.2412.14236)
- Schwarzschild, M. 1979, ApJ, 232, 236, doi: [10.1086/157282](https://doi.org/10.1086/157282)
- Silverwood, H., & Easter, R. 2019, Publications of the
Astronomical Society of Australia, 36, e038,
doi: [10.1017/pasa.2019.25](https://doi.org/10.1017/pasa.2019.25)
- Tabak, E. G., & Turner, C. V. 2012, Communications on
Pure and Applied Mathematics, 66, 145,
doi: [10.1002/cpa.21423](https://doi.org/10.1002/cpa.21423)
- Tong, A., Fatras, K., Malkin, N., et al. 2023, arXiv e-prints,
arXiv:2302.00482, doi: [10.48550/arXiv.2302.00482](https://doi.org/10.48550/arXiv.2302.00482)
- Trick, W. H., Bovy, J., & Rix, H.-W. 2016, ApJ, 830, 97,
doi: [10.3847/0004-637X/830/2/97](https://doi.org/10.3847/0004-637X/830/2/97)
- van den Bosch, R. C. E., van de Ven, G., Verolme, E. K.,
Cappellari, M., & de Zeeuw, P. T. 2008, MNRAS, 385,
647, doi: [10.1111/j.1365-2966.2008.12874.x](https://doi.org/10.1111/j.1365-2966.2008.12874.x)
- Van Rossum, G., & Drake, F. L. 2009, Python 3 Reference
Manual (Scotts Valley, CA: CreateSpace)
- Vasiliev, E. 2019, MNRAS, 482, 1525,
doi: [10.1093/mnras/sty2672](https://doi.org/10.1093/mnras/sty2672)
- Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020,
Nature Methods, 17, 261, doi: [10.1038/s41592-019-0686-2](https://doi.org/10.1038/s41592-019-0686-2)
- Wagg, T., Broekgaarden, F., Van-Lane, P., Wu, K., &
Gültekin, K. 2025, TomWagg/software-citation-station:
v1.4, v1.4 Zenodo, doi: [10.5281/zenodo.17654855](https://doi.org/10.5281/zenodo.17654855)
- Wagg, T., & Broekgaarden, F. S. 2024, arXiv e-prints,
arXiv:2406.04405. <https://arxiv.org/abs/2406.04405>