Measuring Galactic Dark Matter through Unsupervised Machine Learning

Sung Hak Lim
Rutgers University

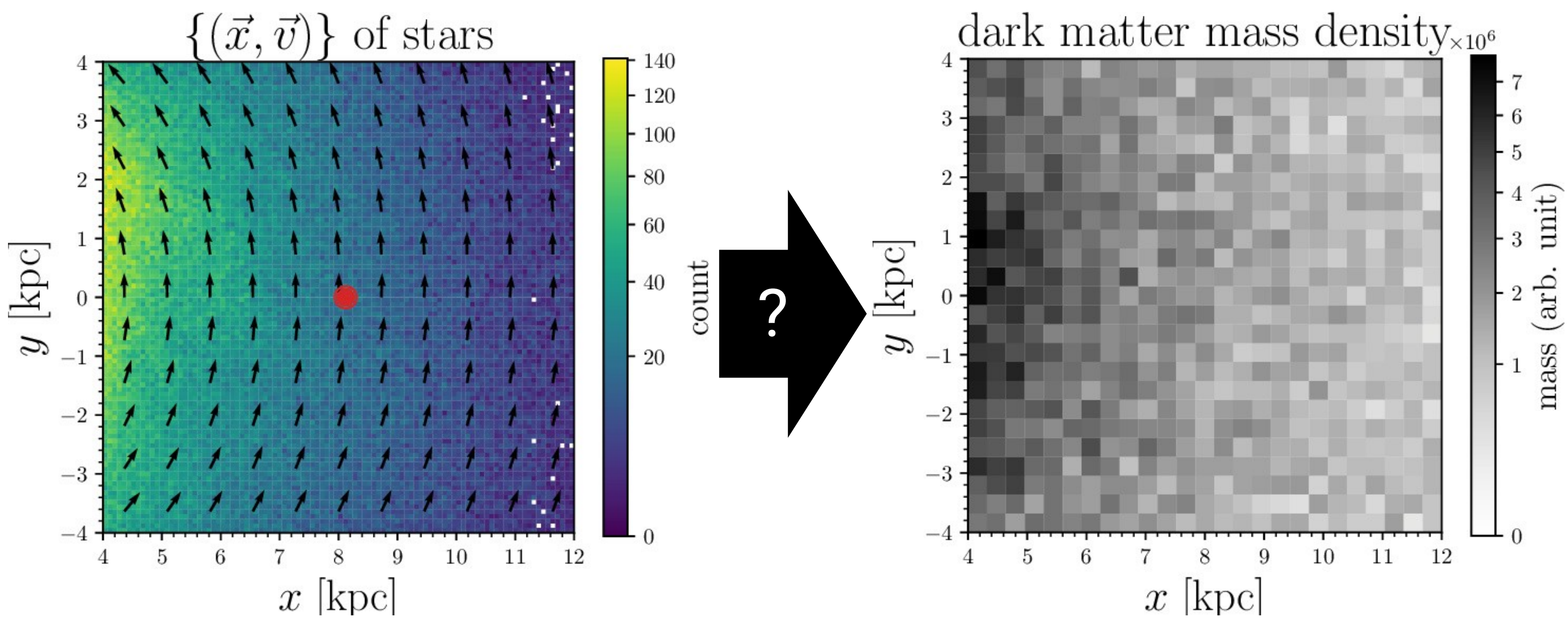
Yonsei University

June 2022

Based on
M. R. Buckley, **SHL**, E. Putney, and D. Shih, arXiv:2205.01129

Introduction

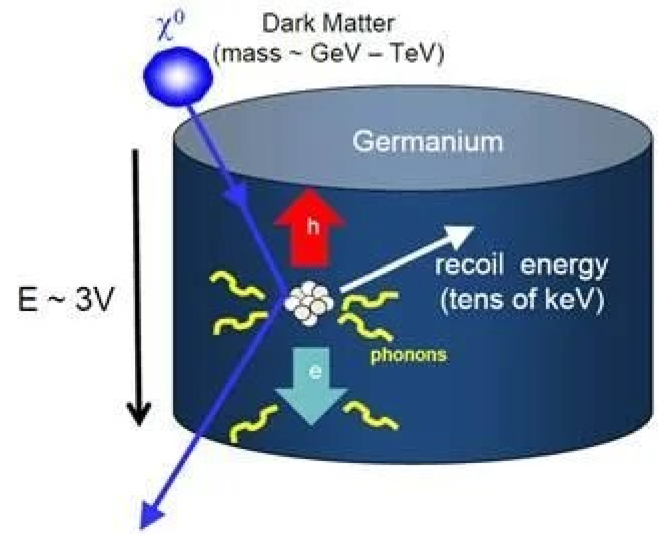
Main Topic:



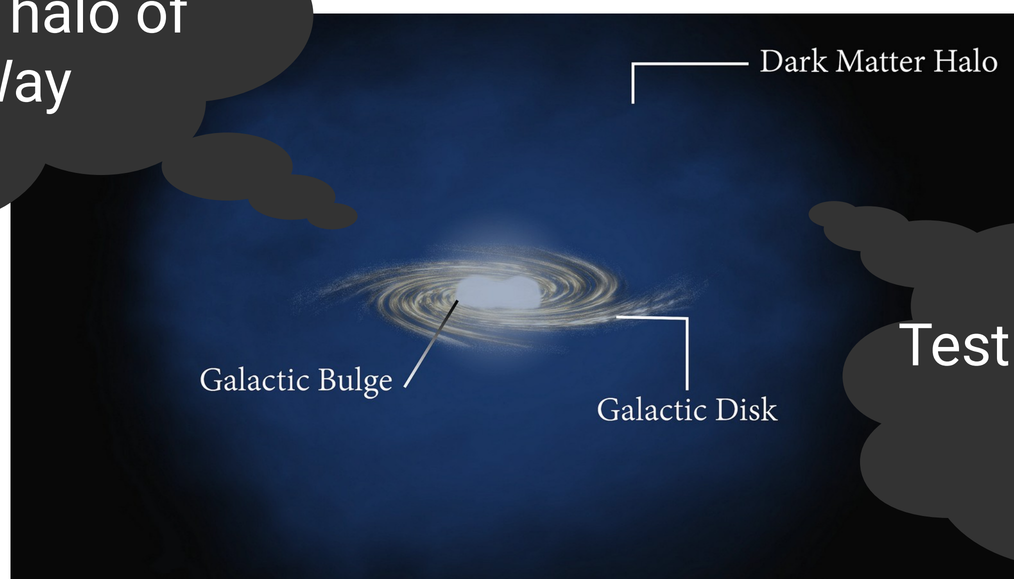
How can we infer the **local galactic dark matter density** from observed position and velocities of stars without assuming **symmetries** and **models**?

Why measuring galactic dark matter is important?

Inputs to Direct Detection experiments

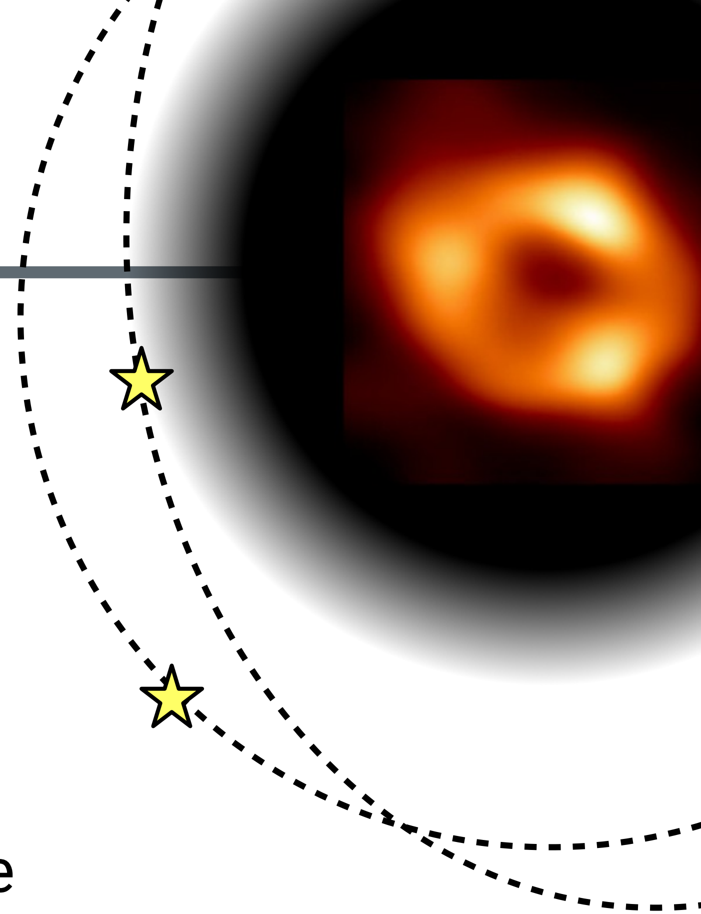


Understanding the dark matter halo of the Milky Way



Testing modified gravity solutions?

Orbits in Galaxy



One way to measure the galactic dark matter is by analyzing the kinematics of stars tracing the galactic gravitational field.

But typical galaxies like the Milky Way have more than billions of stars!

Assuming that the interaction between stars are less relevant (collisionless), more practical way of understanding this gravitation system with a large number of particles is using the phase space density.

$$f(\vec{x}, \vec{v})d\vec{x}d\vec{v}$$

Each star is then regarded as a sample from this phase-space density.

Image from Event Horizon Telescope Collaboration/National Science Foundation

Equation of Motion: (Collisionless) Boltzmann Equation

The equation of motion for the phase space density is called the (collisionless) Boltzmann equation.

$$\left[\frac{\partial}{\partial t} + \vec{v} \cdot \frac{\partial}{\partial \vec{x}} + \vec{a} \cdot \frac{\partial}{\partial \vec{v}}\right] f(\vec{x}, \vec{v}) = 0, \quad \vec{a} = -\frac{d\Phi(\vec{x})}{d\vec{x}}$$

Assuming that the galaxy is in dynamic equilibrium (df/dt = 0), we could estimate the acceleration field a(x) from the Milky Way snapshot at the current time.

But recovering the phase space density from the observed motion of stars is trivial?

$$\{(\vec{x}, \vec{v})\} \rightarrow f(\vec{x}, \vec{v})$$

This regression problem is a 6D density modeling problem. Constructing a smooth density estimation without any assumption is not a trivial task... But we have a huge dataset!

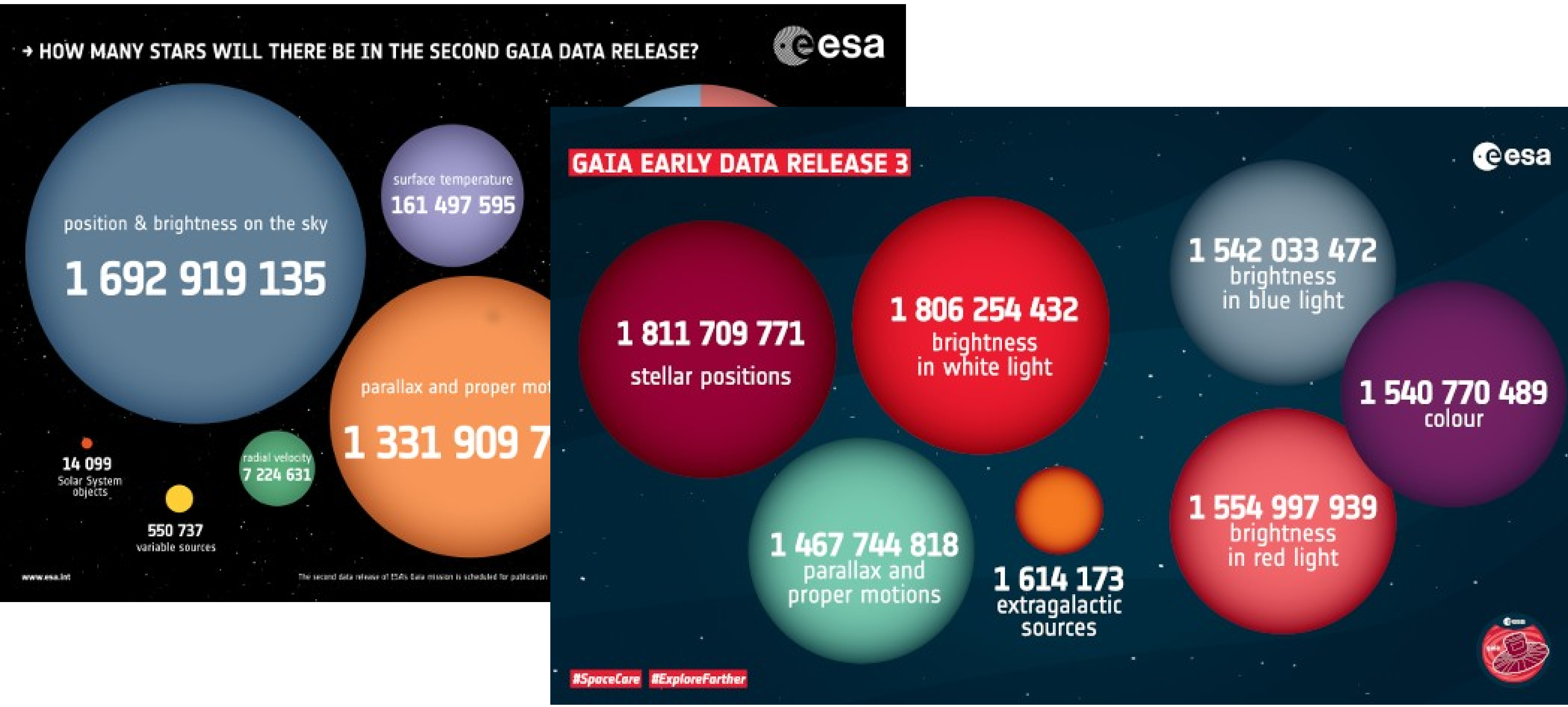
Gaia Dataset

Gaia is a European space mission providing astrometry, photometry, and spectroscopy of more than billion stars in the Milky Way.

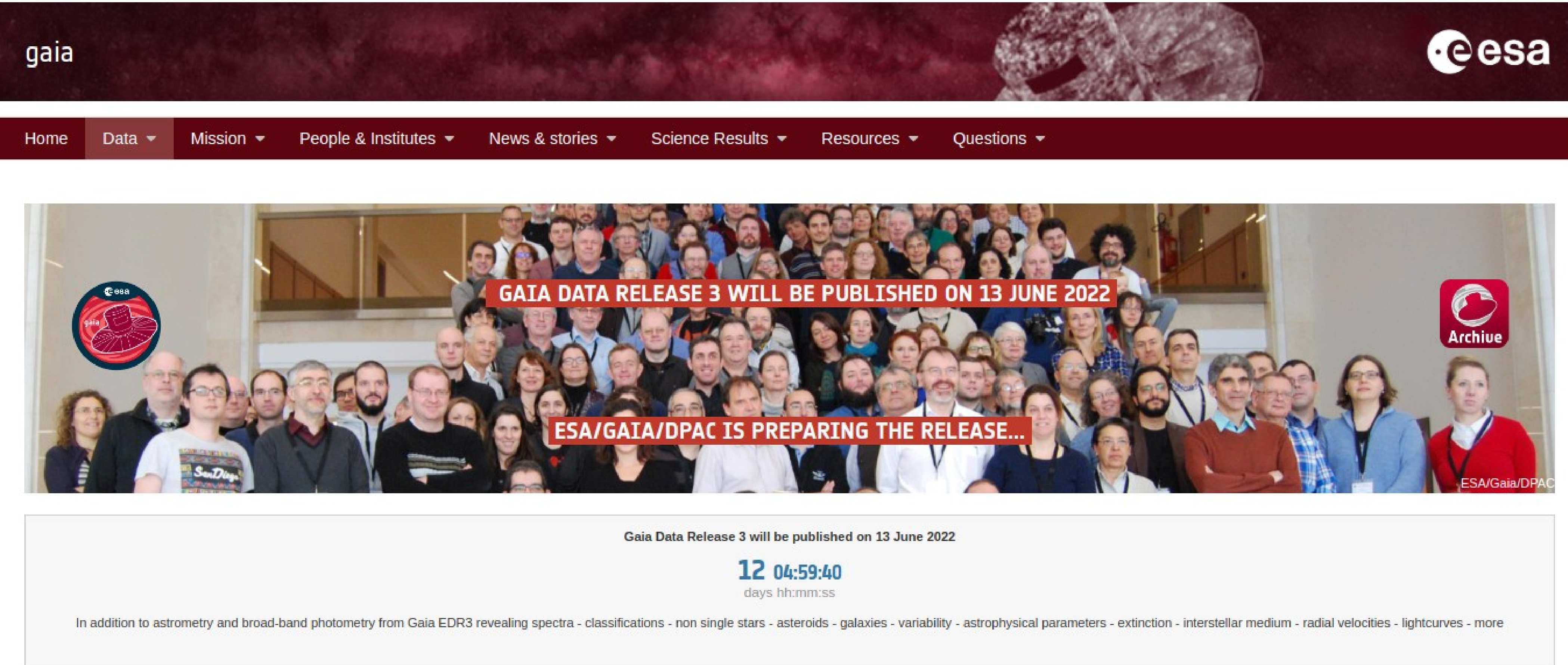
Image from https://gea.esac.esa.int/archive/visualization/

Gaia Dataset

Gaia is a European space mission providing astrometry, photometry, and spectroscopy of more than billion stars in the Milky Way.



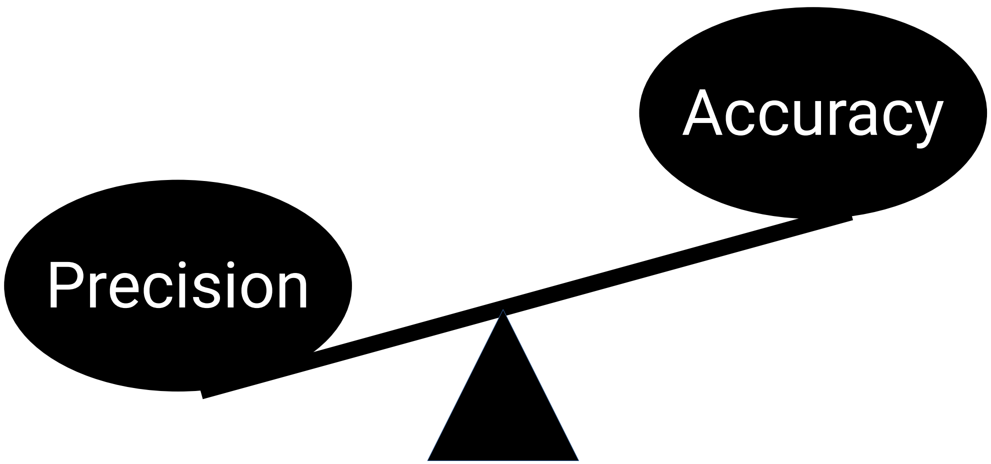
Gaia Dataset



New dataset will come out soon!

Accuracy Matters

Thanks to recent progress in observing stars in the Milky Way, we can measure **the dark matter density in the Solar neighborhood** in very high precision using model-based analysis.



When sufficient number of data are available, using overconstrained models may results in inaccurate results.

Need of analysis without assumed **symmetries** and **models**?

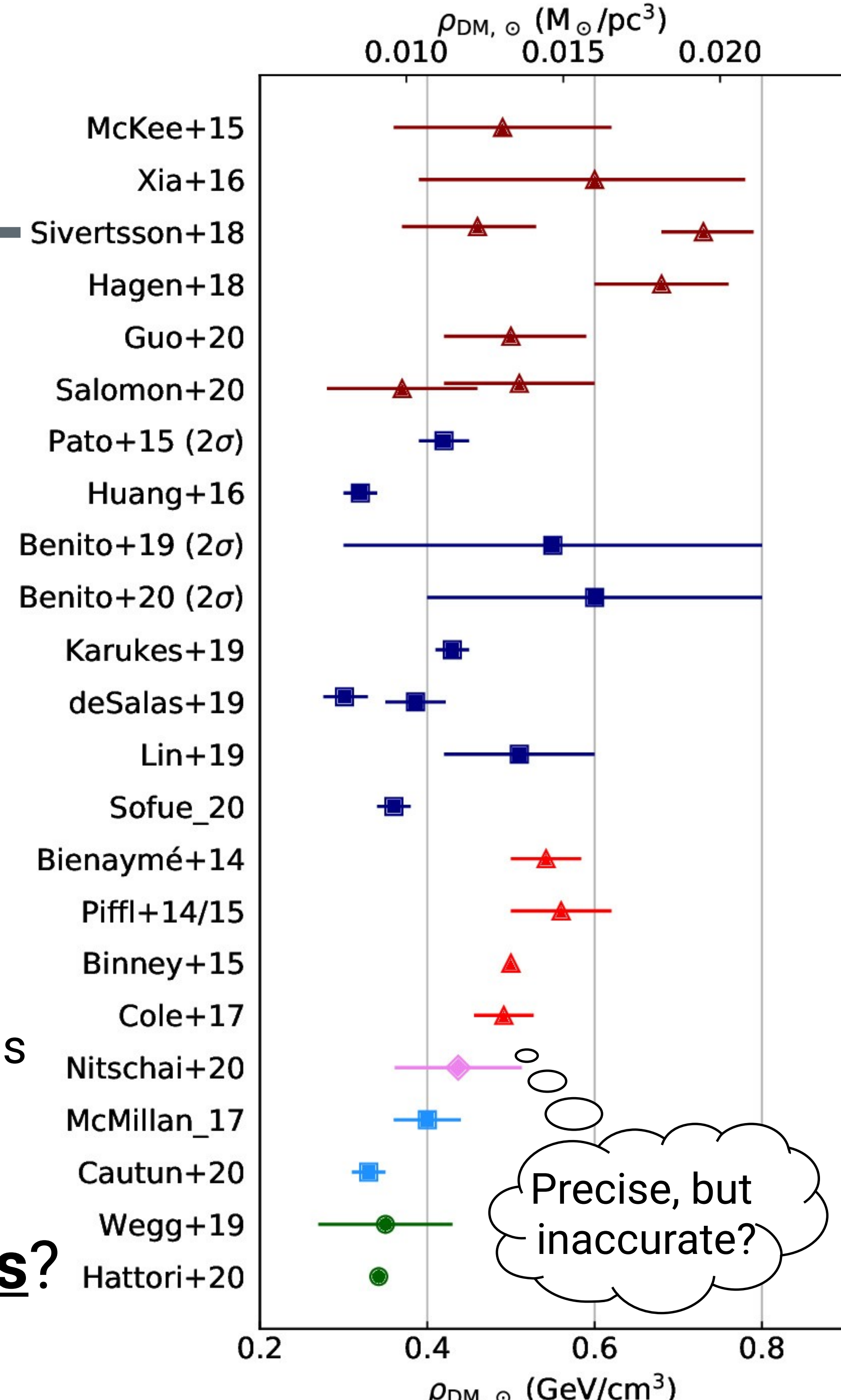


Figure from P. F. de Salas, and A. Widmark, arXiv:2012.11477

Machine Learning Approach

Recent progress in machine learning allows us to estimate probability density functions in high dimensions with high fidelity.

$$\rightarrow f(\vec{x}, \vec{v})$$

Related works:
G. M. Green, et. al., arXiv:2011.04673
A. P. Naik, et. al., arXiv:2112.07657
J. An, et. al., arXiv:2106.06981
G. M. Green, et. al., arXiv:2205.02244

As a proof-of-concept, we will test this idea on an N-body simulated galaxy.

Outline of Strategy

Mock data
$\{(\vec{x}, \vec{v})\}$


Grab stars near the Sun in an N-body simulated galaxy

Phase space density
$f(\vec{x}, \vec{v})$

Neural Networks for Density Estimation:
Normalizing Flows

$$\vec{u}_0 \to \vec{u}_1 \to \cdots \to \vec{u}_n = (\vec{x}, \vec{v})$$

Gravitational accel.
$\vec{a}(\vec{x})$

Solving Boltzmann Equation

$$\left[\vec{v} \cdot \frac{\partial}{\partial \vec{x}} + \vec{a} \cdot \frac{\partial}{\partial \vec{v}}\right] f(\vec{x}, \vec{v}) = 0$$

Mass density
$\rho(\vec{x})$

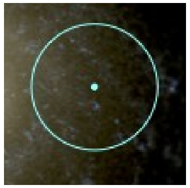
Solving Gauss's Equation

$$-4\pi G\rho = \nabla \cdot \vec{a}$$

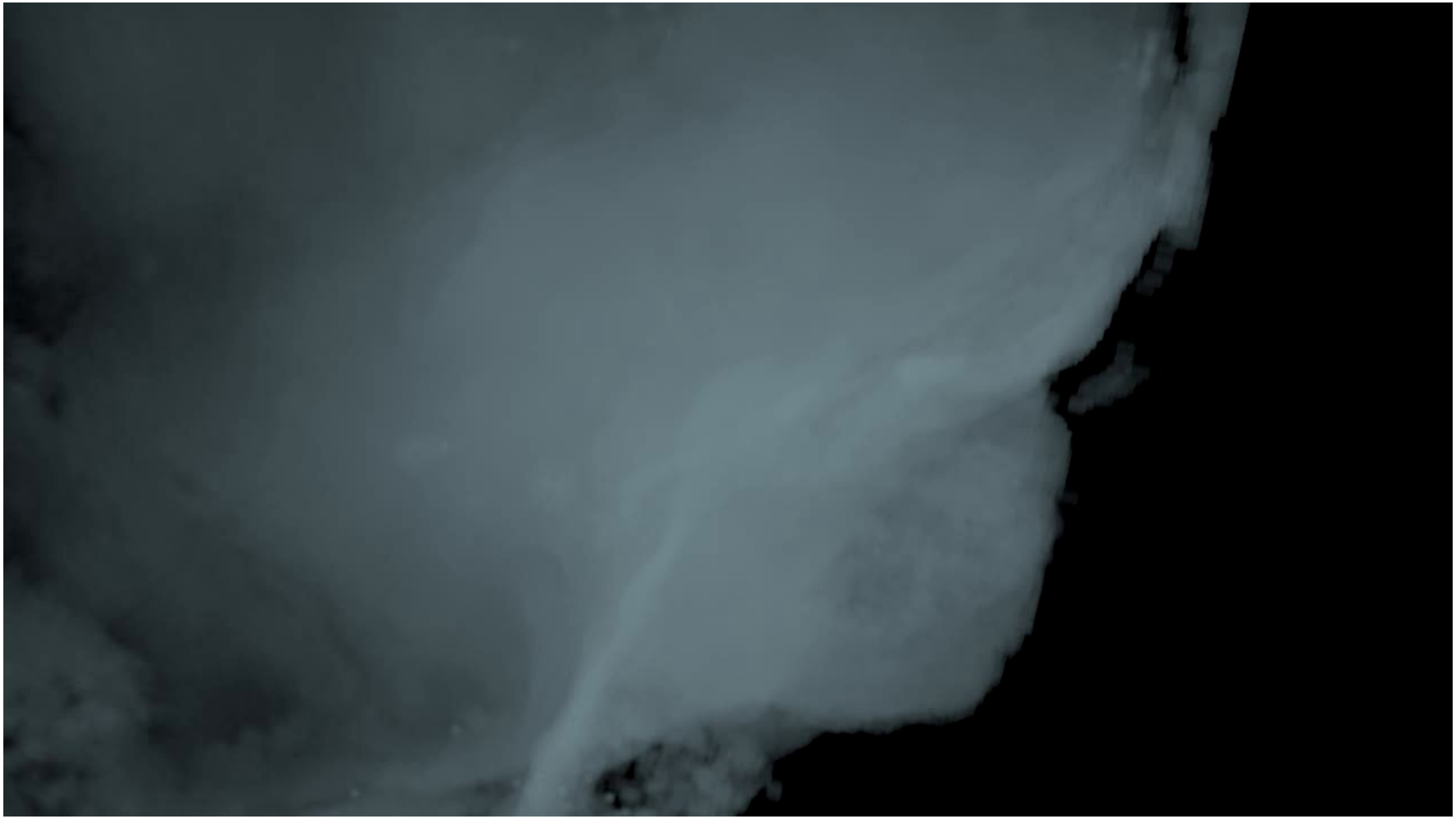
Training Dataset

Mock data

$\{(\vec{x}, \vec{v})\}$

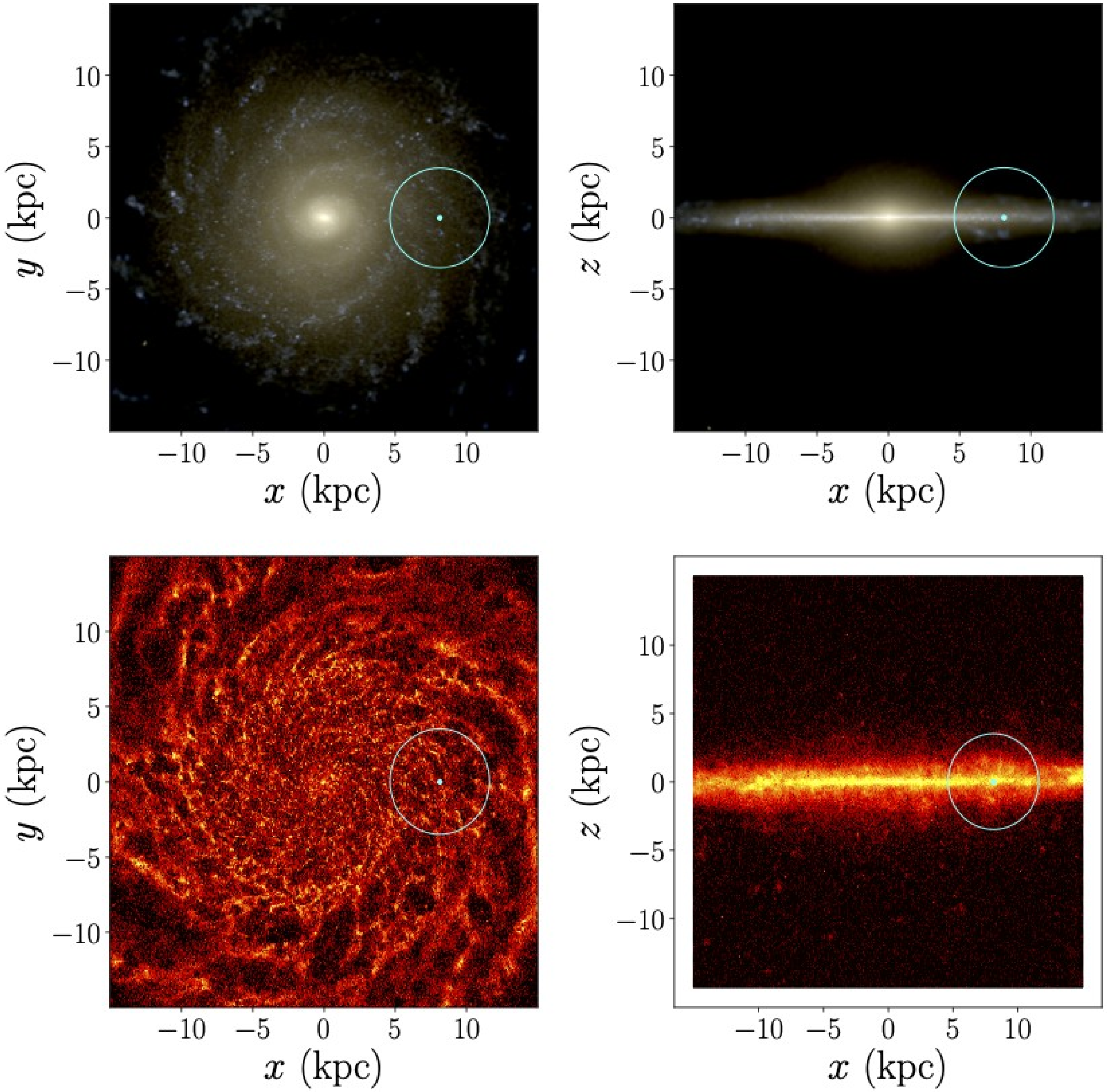


Introducing h277



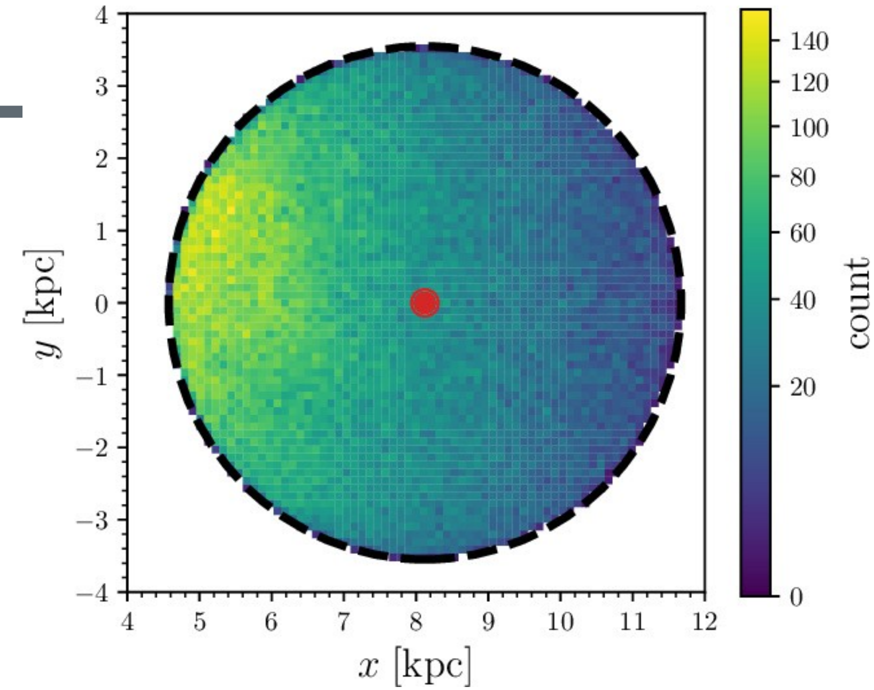
https://b2share.eudat.eu/records/c9f232d8ac804785aad35004177a704e

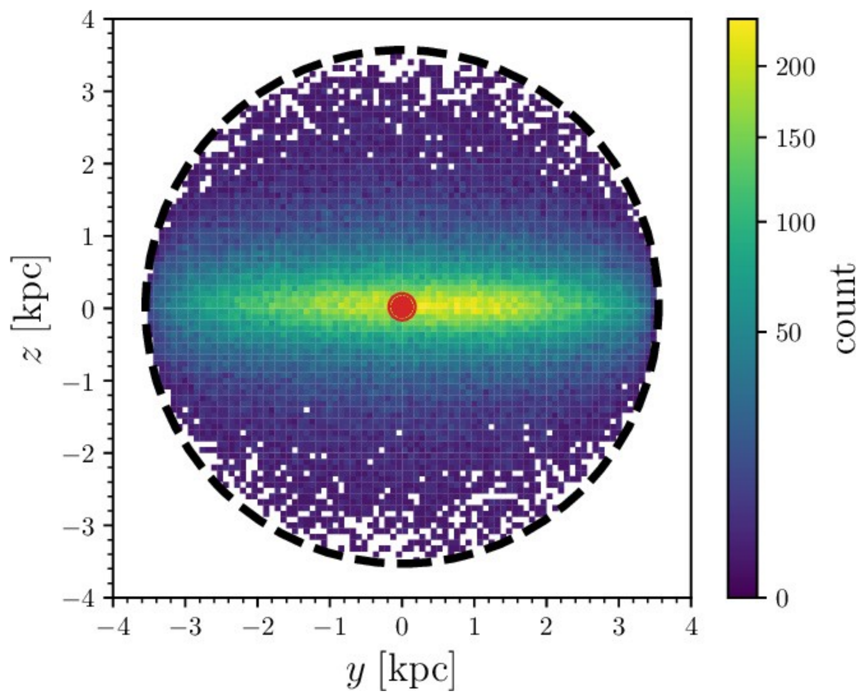
h277 at present

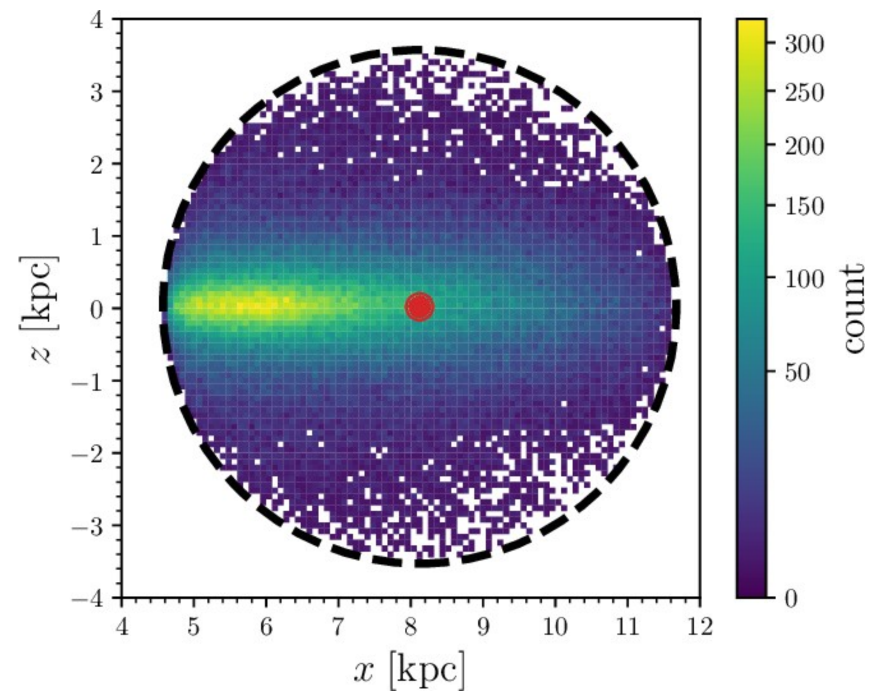


Training Dataset

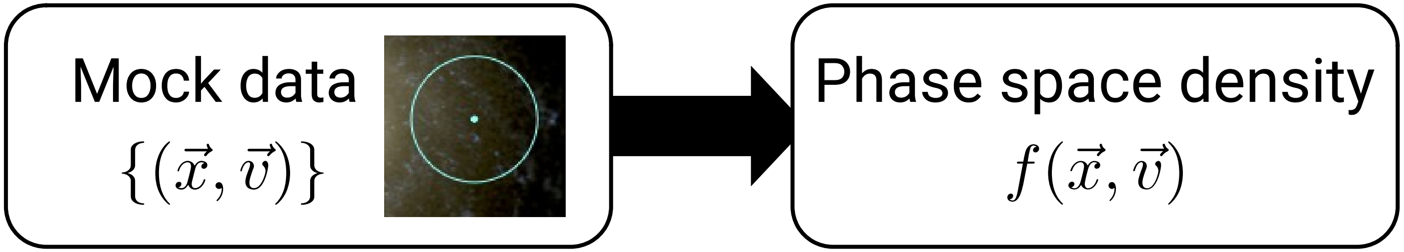
- number of stars
  153,174 (<< size of Gaia 6D dataset)
- observer's location
  [8.122, 0., 0.0208] kpc
- observing radius = 3.5 kpc
- simulation resolution: 0.173 kpc
- Using only kinematic information:
  position and velocity

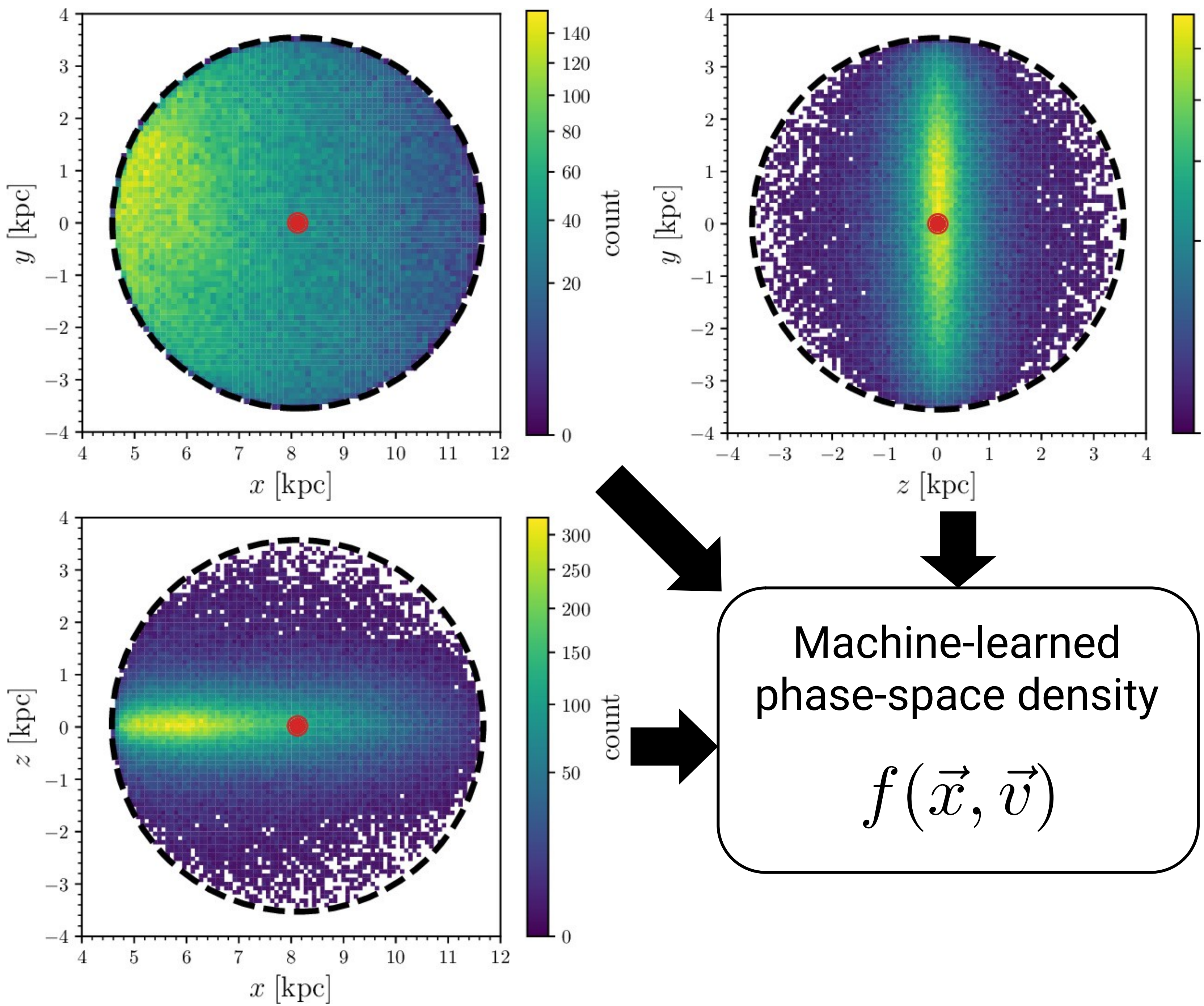






Phase Space Density Estimation through Flows

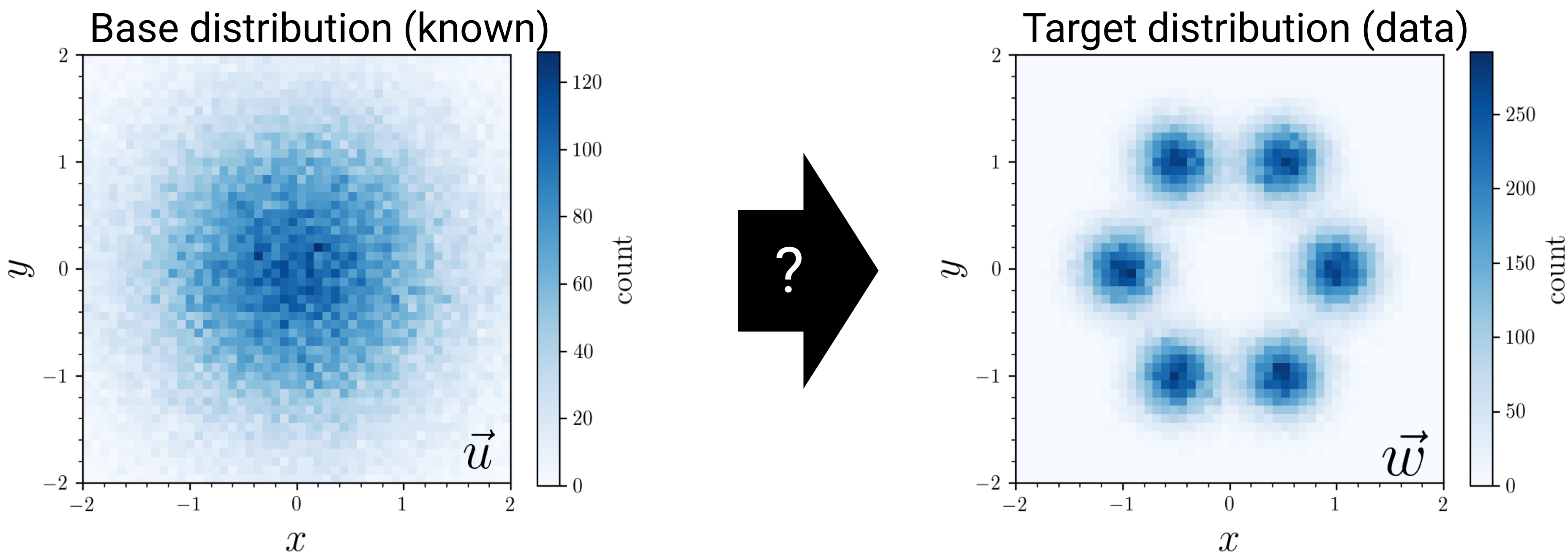




y [kpc]
x [kpc]
count
z [kpc]
Machine-learned
phase-space density
$f(\vec{x}, \vec{v})$

Normalizing Flows: Neural Network learning a Transformation

Normalizing Flows (NFs) is an artificial neural network that learns a transformation of random variables.



Main idea: if we could find out such transformation, we can use the transformation formula for the density estimation:

$$p_W(\vec{w}) = p_U(\vec{u}) \cdot \left| \frac{d\vec{u}}{d\vec{w}} \right|$$

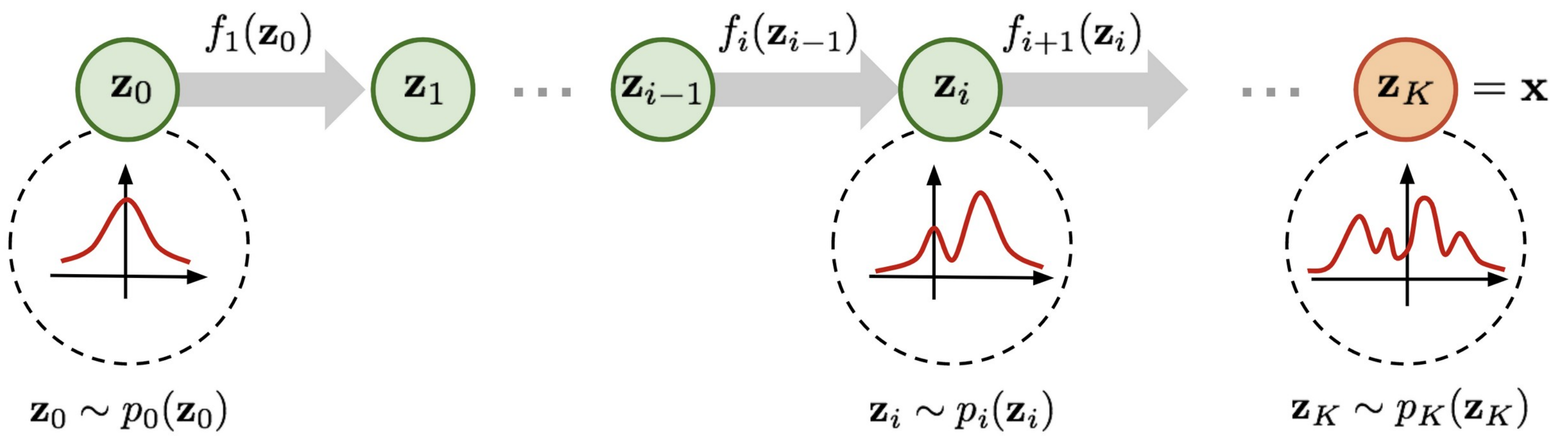
We will use this model for estimating the phase space density f(x,v) from the data.

Normalizing Flows: Neural Network learning a Transformation

Normalizing Flows (NFs) is a chain of simple trainable functions (neural networks) learning a transformation of random variables.

$$\vec{w} = f(\vec{u}) = f_K \circ \cdots \circ f_2 \circ f_1(\vec{u})$$

* Simple = inverse and Jacobian determinant are easily computable.



What can normalizing flows do?

- Density Estimation:

$$p_W(\vec{w}) = p_U(\vec{u}) \cdot \left| \frac{d\vec{u}}{d\vec{w}} \right|$$

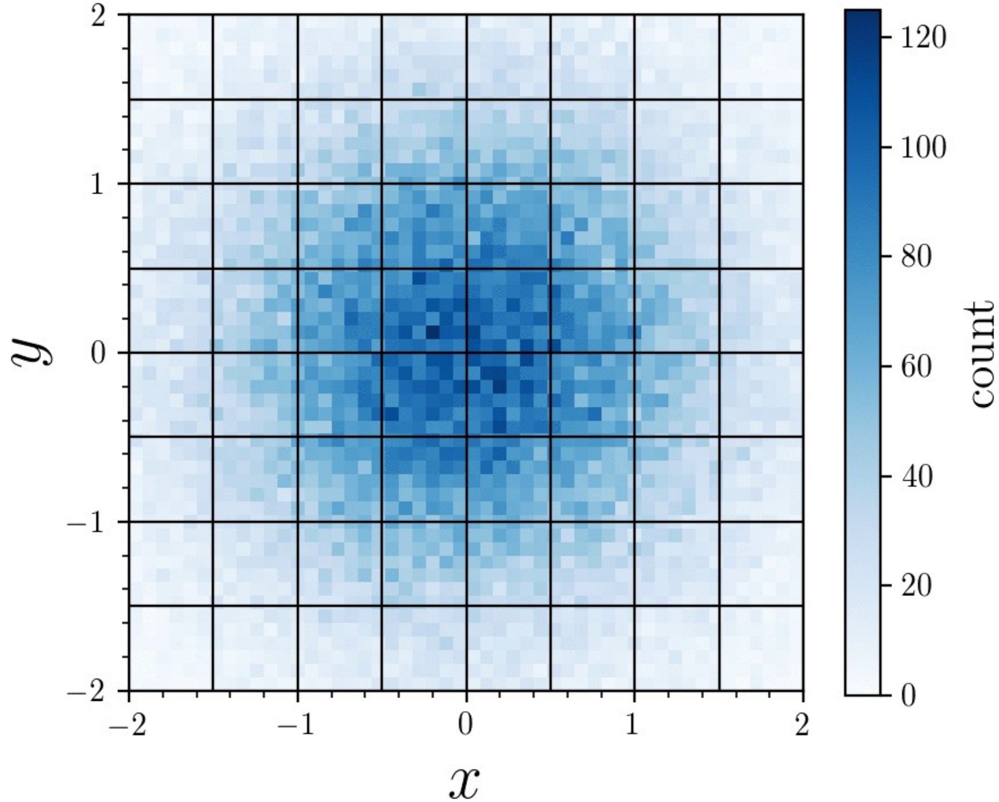
- Sample Generation:

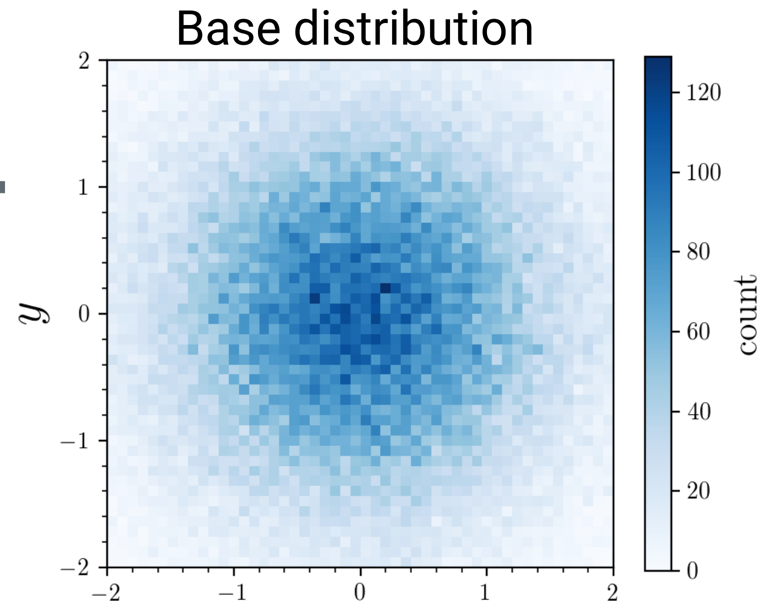
$$\vec{w} = f(\vec{u})$$

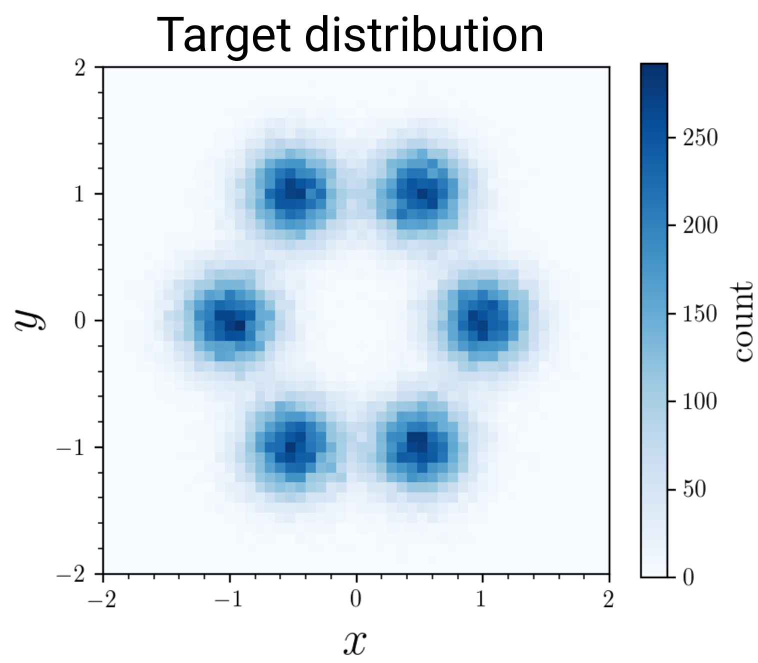
We will see that these two features of normalizing flows allows us to use them for estimating galactic acceleration and mass density.

Image from https://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models.html

Normalizing Flows: How it works?







* result of a continuous normalizing flow learning infinitesimal transformations

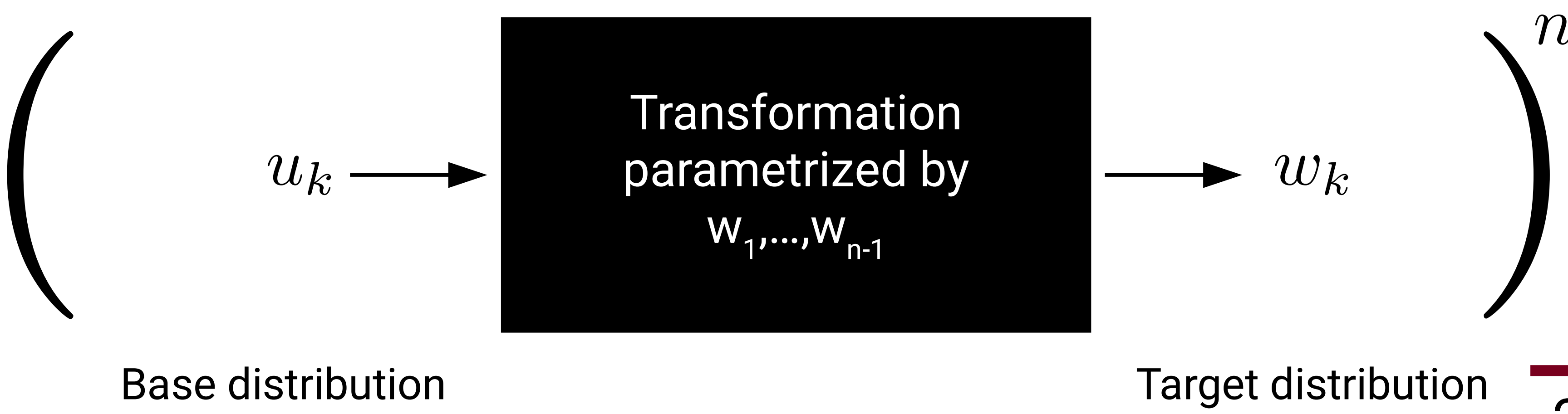
Autoregressive Models

But constructing a simple multivariate bijection is a non-trivial task.
→ one solution: **autoregressive models**

Autoregressive models are motivated from the chain rule of probability.

$$p(\vec{w}) = p(w_1) \times p(w_2|w_1) \times \cdots \times p(w_n|w_1, \cdots, w_{n-1})$$

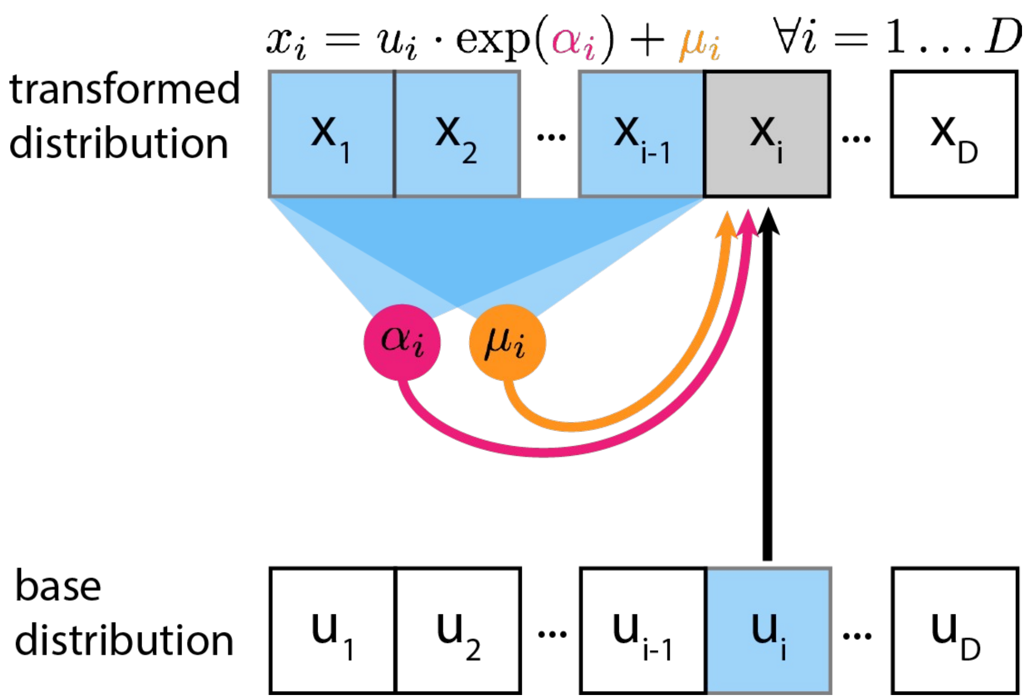
Instead of building a bijection transforming all the variables at once, we can simply model the density as a product of conditional probability of each variable.



Masked Autoregressive Flow

One of the simplest setup is using a linear transformation conditioned on previous components.
This flow is called Masked Autoregressive Flow (MAF).

$$w_k = u_k \sigma(w_1, \cdots, w_{k-1}) + \mu(w_1, \cdots, w_{k-1})$$



Conditioning variables makes the transformation non-linear.

We use a chain of these flows
to fit the phase-space density from the training dataset.

Image from https://blog.evjang.com/2018/01/nf2.html

MAFs for modeling phase space density

We may directly attempts to construct the full 6D phase-space density, but using two separate MAFs for position and velocity is better.

$$f(\vec{x}, \vec{v})$$

MAFs for
full phase space

$$f(\vec{x}, \vec{v}) = \nu(\vec{x})p(\vec{v}|\vec{x})$$

MAFs for
position space

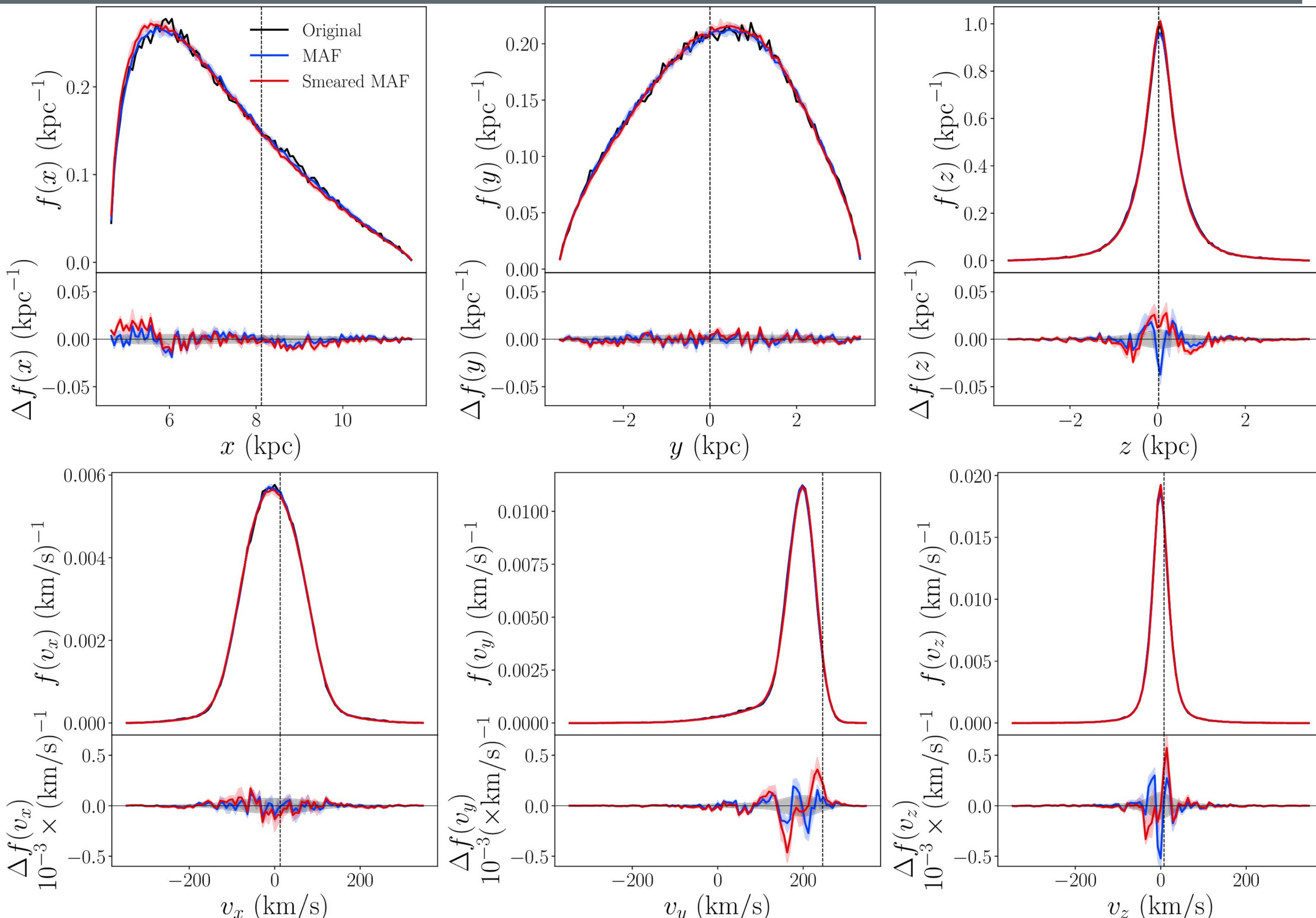
$\times$

MAFs for
velocity space

Advantage: we could directly sample velocities at given position.

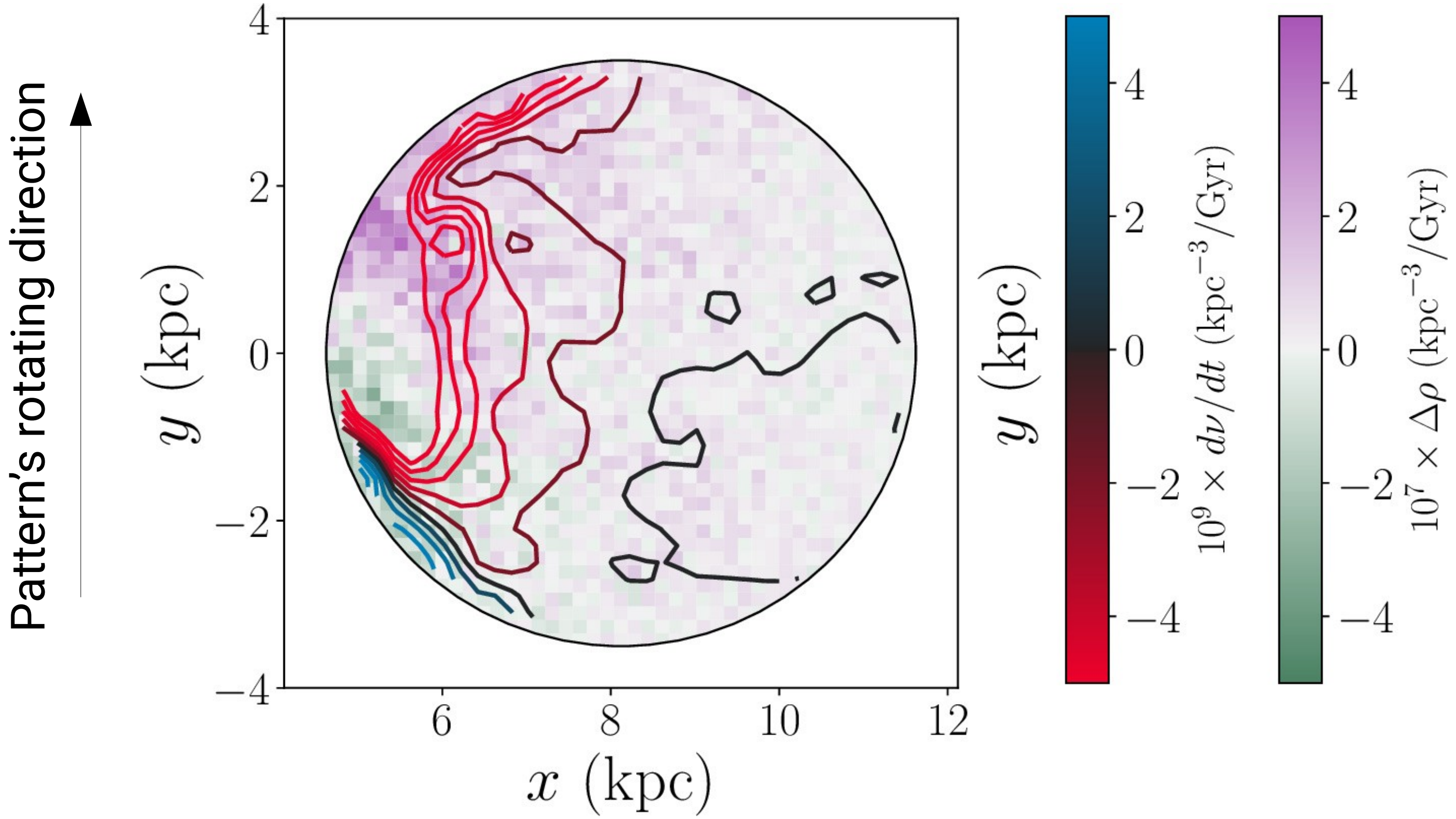
This decomposition is very useful for calculating derived quantities using Monte-Carlo integration on velocity integrals.

Results:
Phase-space density Estimation



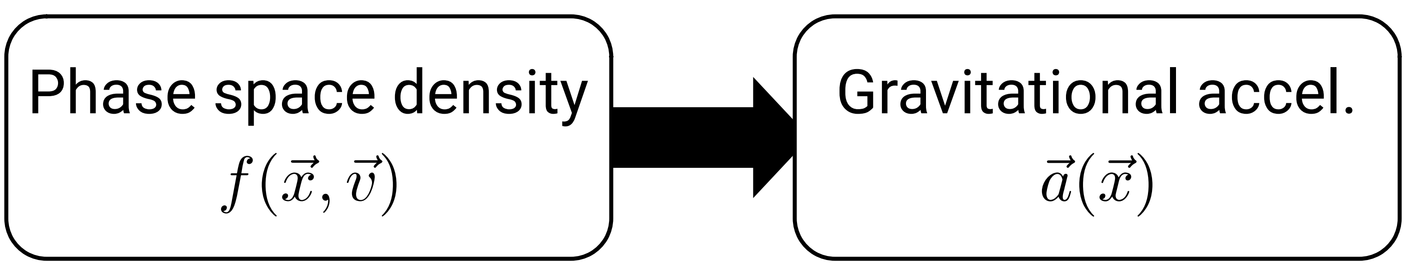
Application: Measuring local departure from equilibrium

If galactic acceleration can be measured using other methods, such as pulsar and binary timing, we can measure the time derivatives from the given snapshot of the Milky Way,



$$\frac{d\nu}{dt} = \int d^3\vec{v} \left[\vec{v} \cdot \frac{df}{d\vec{x}} + \vec{a} \cdot \frac{df}{d\vec{v}}\right]$$

Gravitational Acceleration Estimation: Solving Boltzmann Equation



Acceleration Estimation

Now we have the estimated phase-space density estimation on our hand. Let's try to solve the Boltzmann equation.

$$\left[\vec{v}\cdot\frac{\partial}{\partial\vec{x}}+\vec{a}\cdot\frac{\partial}{\partial\vec{v}}\right]f(\vec{x},\vec{v})=0, \qquad \vec{a}=-\frac{d\Phi(\vec{x})}{d\vec{x}}$$

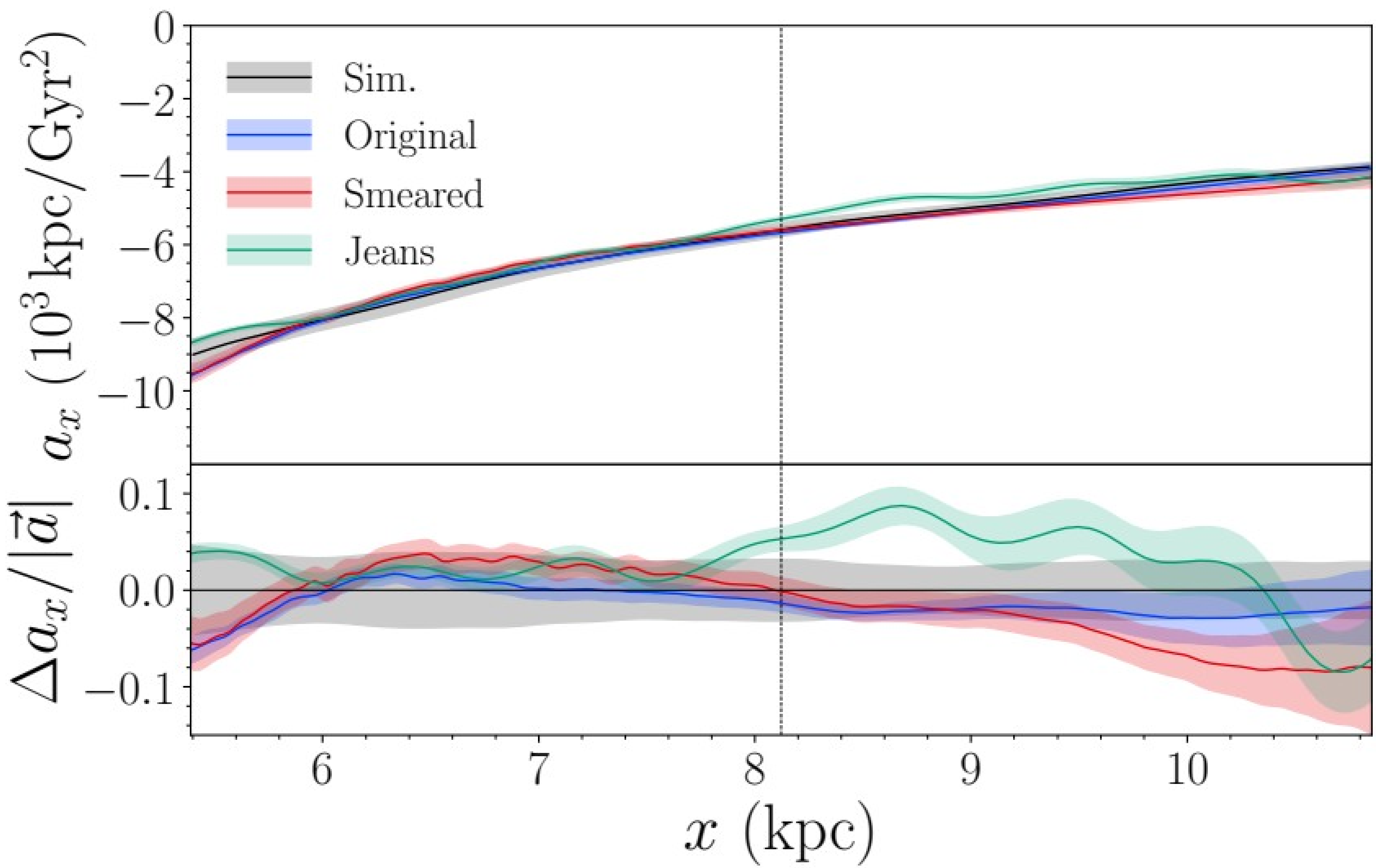
- Underdetermined in a point of view at each star position.
- Overdetermined in a point of view of phase-space density.

Given the fact that we could resample velocities at given position multiplie times, we can solve the overdetermined system using least square minimization.

$$\mathcal{L}(\vec{x})=\frac{1}{N}\sum_{\alpha=1}^{N}\left|\left[\vec{v}^{\alpha}\cdot\frac{\partial}{\partial\vec{x}}+\vec{a}\cdot\frac{\partial}{\partial\vec{v}}\right]f(\vec{x},\vec{v}^{\alpha})\right|^2$$

We draw 10,000 samples per position to reduce the MC integration error below the statistical and measurement errors.

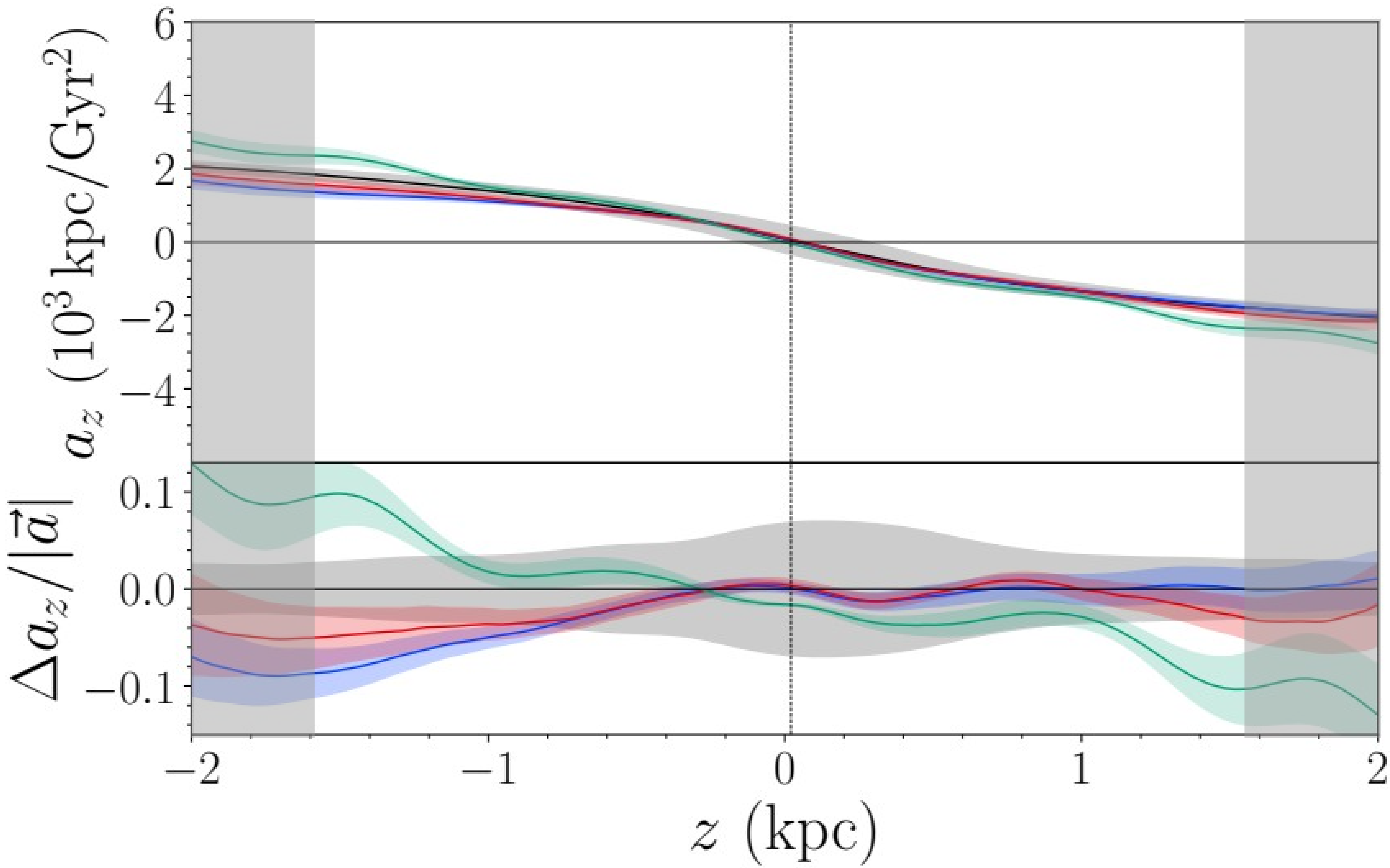
Acceleration along x-axis



Our method can find out the acceleration within 5% accuracy!

Acceleration along z-axis

Stat is low
1000~1500 / kpc^3

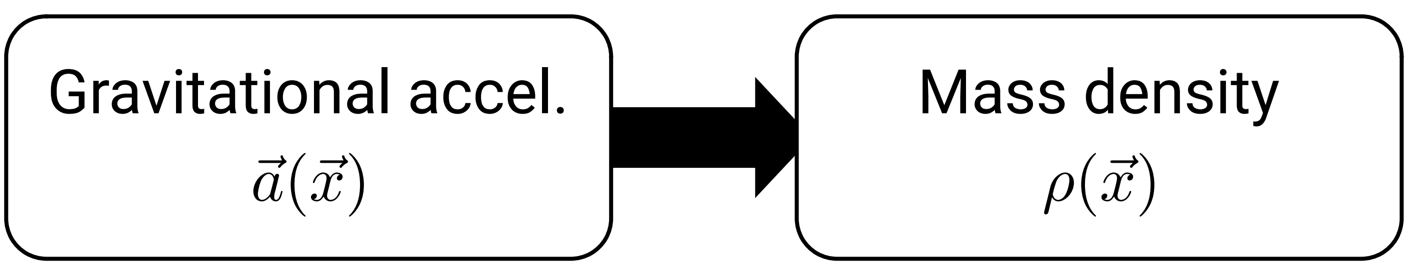


Our method can find out the acceleration within 5% accuracy!

Acceleration at the Sun

component	dataset	acceleration (kpc/Gyr^2)	(stat.)	(syst.)
a_x	sim.-truth	-5608	–	± 193
	original	-5672	$\pm$ 43	–
	smeared	-5597	$\pm$ 67	$\pm$ 40
	Jeans	-5305	$\pm$ 67	$\pm$ 19
a_y	sim.-truth	41	–	± 116
	original	344	$\pm$ 51	–
	smeared	237	$\pm$ 45	$\pm$ 29
	Jeans	0	–	–
a_z	sim.-truth	86	–	± 381
	original	60	$\pm$ 30	–
	smeared	72	$\pm$ 30	$\pm$ 13
	Jeans	-38	$\pm$ 3	$\pm$ 1

Mass Density Estimation: Solving Gauss's Equation



Solving Gauss's Equation

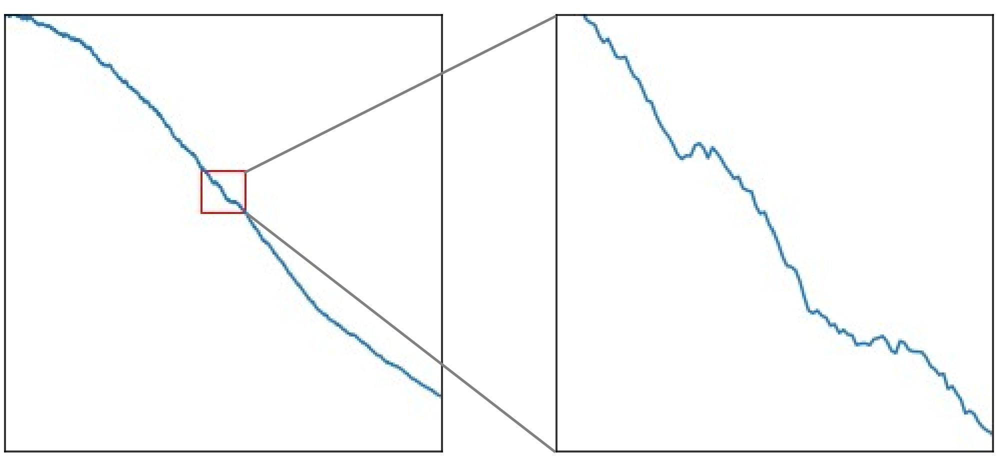
Since we have a smooth acceleration estimator, we may directly take another derivative to estimate the local mass density.

Gauss's Law: $-4\pi G\rho = \nabla \cdot \vec{a}$

Mass density function:
genuine point-wise feature

Compatible?

Estimated accelerations:
finite resolution
finite training samples



Note that differentiation is essentially an error-amplifying process.

We may end up with losing precision because of noise or inductive bias of interpolation!

Smoothed Mass Density Estimation

Instead, we will estimate kernel smoothed mass density:

$$-4\pi G\rho * K_h = (\nabla \cdot \vec{a}) * K_h$$

(Smoothed)
mass density function:
at kernel bandwidth scale

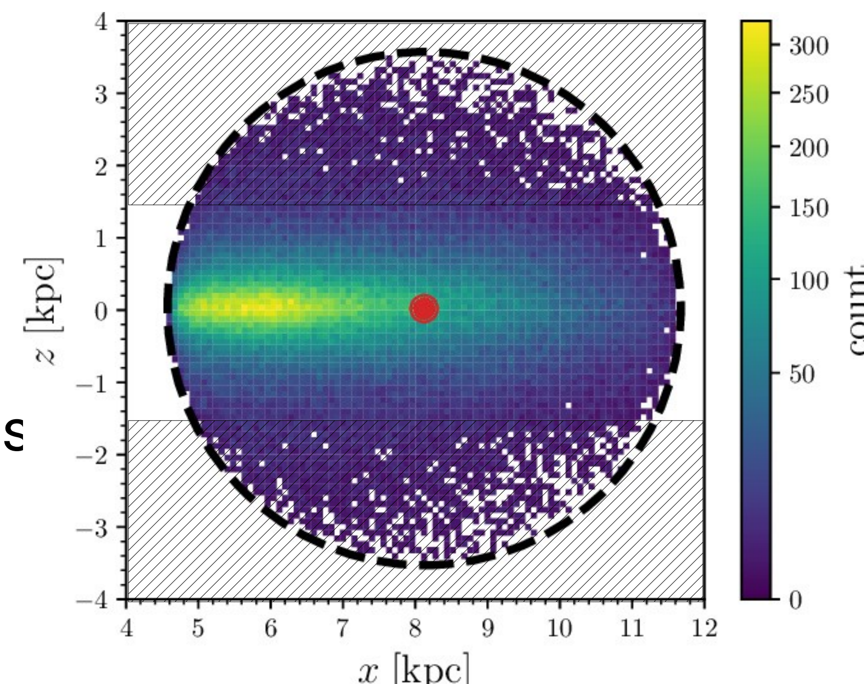
Compatible!

(Smoothed)
estimated accelerations:
at kernel bandwidth scale

For kernels, bandwidths larger than the simulation resolution (0.173 kpc) is ideal for our purpose.

But not much stars are available at high-|z|, so we use the following bandwidths

$$(h_x, h_y, h_z) = (1.0, 1.0, 0.2) \text{ kpc}$$



No need of evaluating 2nd order derivative directly

Another advantage of using kernel smoothed mass density is that we do not need to evaluate the 2nd order derivative of the network.

$$-4\pi G\rho * K_h = (\nabla \cdot \vec{a}) * K_h = \int d^3\vec{x}' (\nabla \cdot \vec{a})(\vec{x}') K_h(\vec{x} - \vec{x}')$$

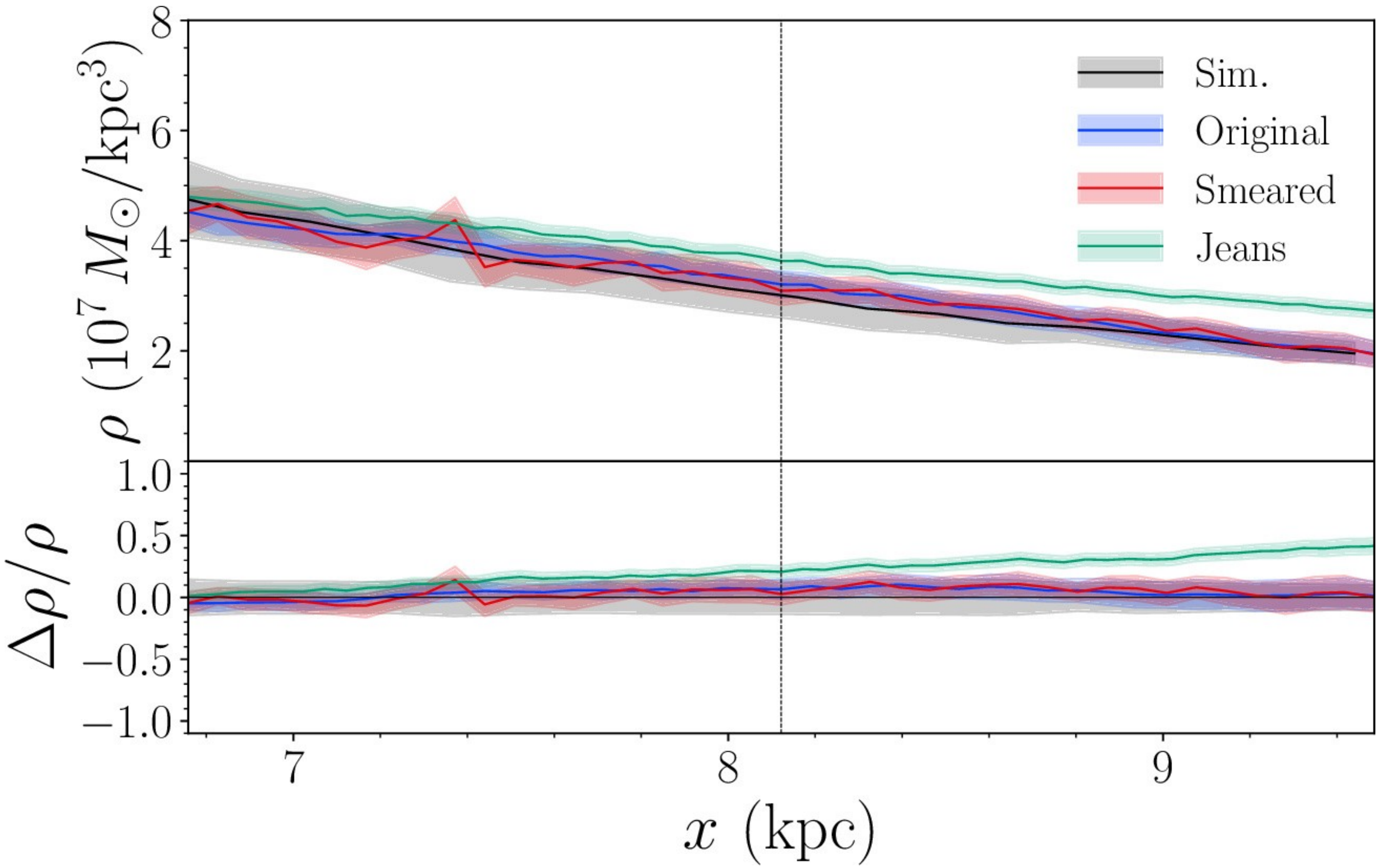
$$= \oint d^2\vec{x}' \hat{n} \cdot \vec{a}(\vec{x}') K_h(\vec{x} - \vec{x}') + \int d^3\vec{x}' \vec{a}(\vec{x}') \cdot \nabla K_h(\vec{x} - \vec{x}')$$

To estimate the smoothed mass density at given position, we do the following:

1. Draw samples from kernel boundary and kernel itself.
2. Evaluate accelerations at perturbed positions.
3. Solve the above Gauss's equation

Warning: this step is very time consuming!
10000 x 3200
~ 30M network evaluations per point!

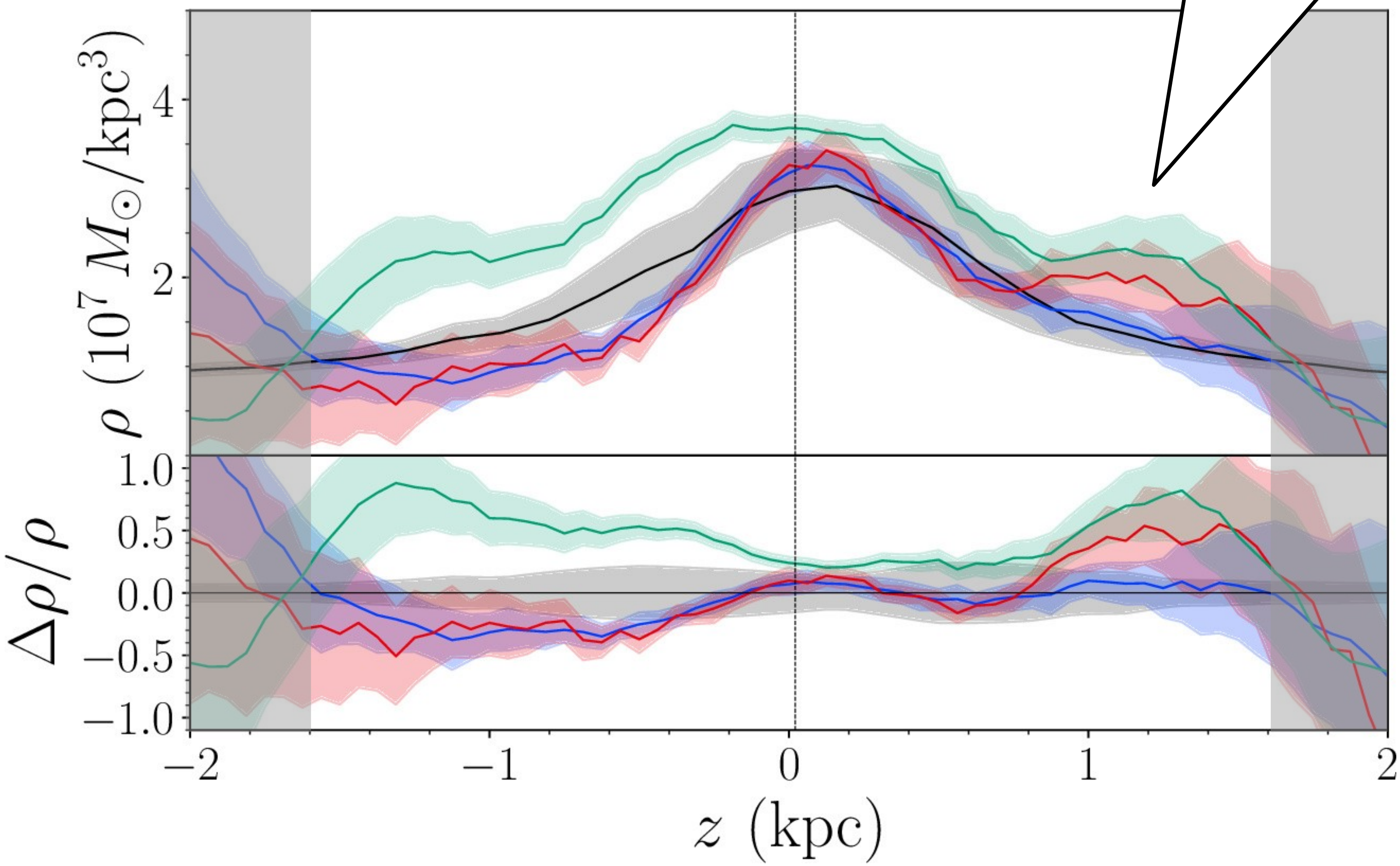
Mass Density along x-axis



Our method can find out the mass density within 10~20% accuracy!

Mass Density along z-axis

We can measure
the mass density
away from the disk plane!



Our method can find out the mass density within 10~20% accuracy!

Mass Density at the Sun

Dataset	Mass Density ρ ($10^7\ M_\odot/\mathrm{kpc}^3$)	(stat.)	(syst.)
sim.-truth	3.06	–	± 0.37
original	3.33	± 0.17	–
smeared	3.37	± 0.17	± 0.15
Jeans	3.67	± 0.13	± 0.05

Conclusion

- We introduced flow-based neural network that accurately fits the phase space density without assuming symmetries and models. The learned phase space density can be used for the mass density estimation by solving the collisionless Boltzmann equation.
- We successfully demonstrated our mass density estimation method to a fully cosmological simulation of a Milky Way-like galaxy, which doesn't impose equilibrium and axisymmetry explicitly
- We successfully achieved ~20% accuracy in mass density estimation using only O[10^5] tracer stars, which is significantly smaller than the Gaia dataset.
- Our method do not need to assume symmetries or models, so that the estimation is truly a local density estimation, which can be useful for revealing hidden information in the Milky way.
- By using accurately measured accelerations, our method also provides a way to analyze local departure from equilibrium.
- With upcoming new dataset from Gaia, it would be an exciting time to test this kind of ideas in a real-world dataset!

Backups

Traditional Approach: Jeans Equation and Parametric Models

Instead of considering full phase-space velocity,
do the method of moments on velocity distributions.
(6D problem → 3D problem)

$$\frac{\partial}{\partial t}\nu\langle v_i\rangle + \frac{\partial}{\partial x_i}\nu\langle v_i v_j\rangle - \nu(\vec{x})a_j = 0$$

$$\nu(\vec{x}) = \int d^3\vec{v}\, f(\vec{x},\vec{v})$$

$$\langle v_i\rangle_{\vec{x}} = \int d^3\vec{v}\, v_i f(\vec{x},\vec{v})$$

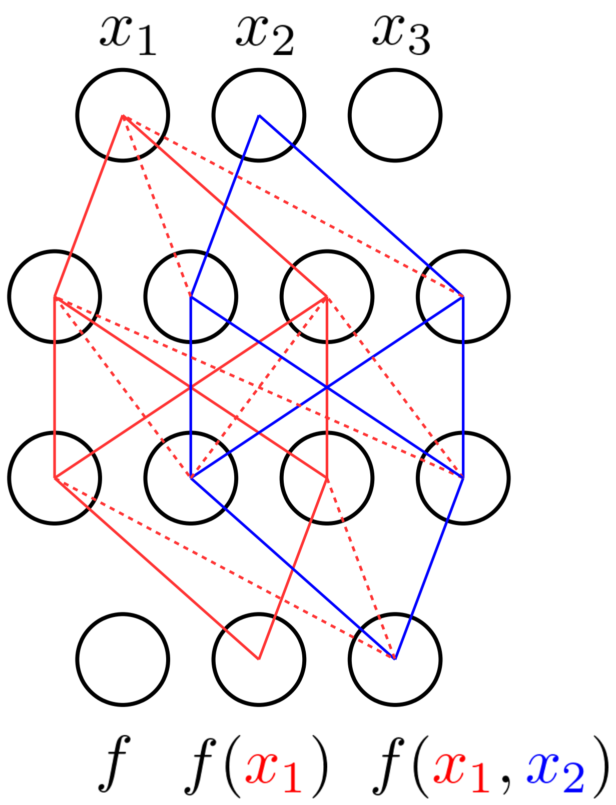
$$\langle v_i v_j\rangle_{\vec{x}} = \int d^3\vec{v}\, v_i v_j f(\vec{x},\vec{v})$$

Assume symmetries to reduce the dimension further.
Introduce parametric models to covert the problem into
a parametric regression with a few parameters.

Example)

$$\nu(r) = \frac{3}{4\pi a^3}\left(1+\frac{r^2}{a^2}\right)^{-5/2} \qquad \rho(r) = \rho_s\left(\frac{r}{r_s}\right)^{-\gamma}\left(1+\frac{r}{r_s}\right)^{-(\beta-\gamma)/\alpha}$$

MADE block



$$p(\vec{u}) = p(u_1) \times p(u_2|u_1) \times \cdots \times p(u_n|u_{1:n-1})$$

$$h_k \to u_k = h_k \sigma(u_1, \cdots, u_{k-1}) + \mu(u_1, \cdots, u_{k-1})$$

Normalizing Flows

What can normalizing flows do?

- Density Estimation:

The probability density function (PDF) of *Y* is given in terms of the PDF of *X* and the transformation *f*.

$$p_Y(y) = p_X(x) \cdot \left| \frac{dx}{dy} \right| = p_X(f^{-1}(y)) \cdot \left| \frac{df^{-1}(y)}{dy} \right|$$

- Sample Generation:

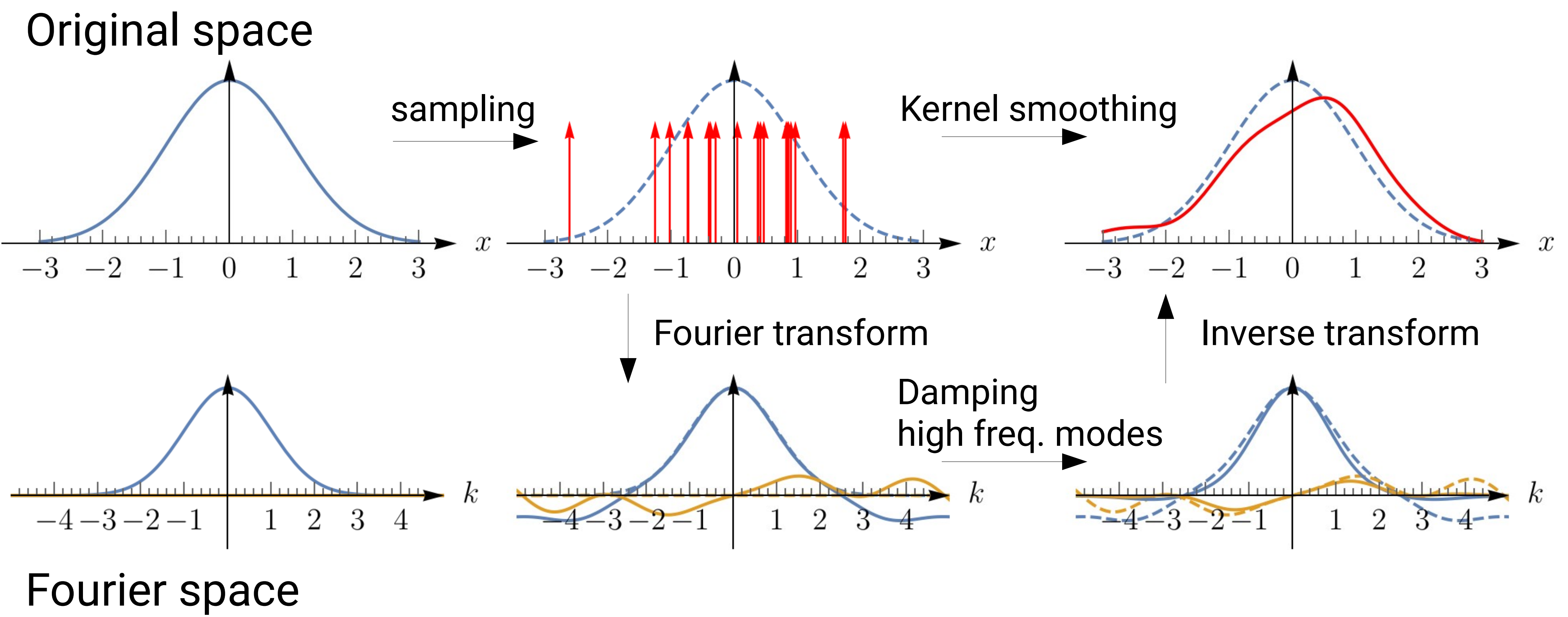
Draw samples from the base distribution, and
transform them into the samples from the learned distribution.

$$X \sim \mathcal{N}(0,1)$$

$$Y = f(X;\theta) = f_K \circ \cdots \circ f_2 \circ f_1(X;\theta)$$

We will see that these two features of normalizing flows allows us to use them for estimating galactic acceleration and mass density.

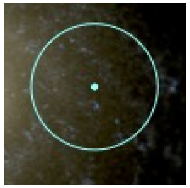
Kernel Smoothing Example: Data from Gaussian Distribution



Resampling-based Uncertainty Estimation

Mock data

$\{(\vec{x}, \vec{v})\}$



Phase space density

$f(\vec{x}, \vec{v})$

Gravitational accel.

$\vec{a}(\vec{x})$

Mass density

$\rho(\vec{x})$

Uncertainty Estimation

Neural networks h
→ Error propagation through the formula is complicated

Sampling based methods:
- Bootstrapping
- Monte Carlo Error Propagation

Nevertheless, our network is dealing with small dimensional dataset and training time is quite fast (~3 hours / network)

Bootstrapping

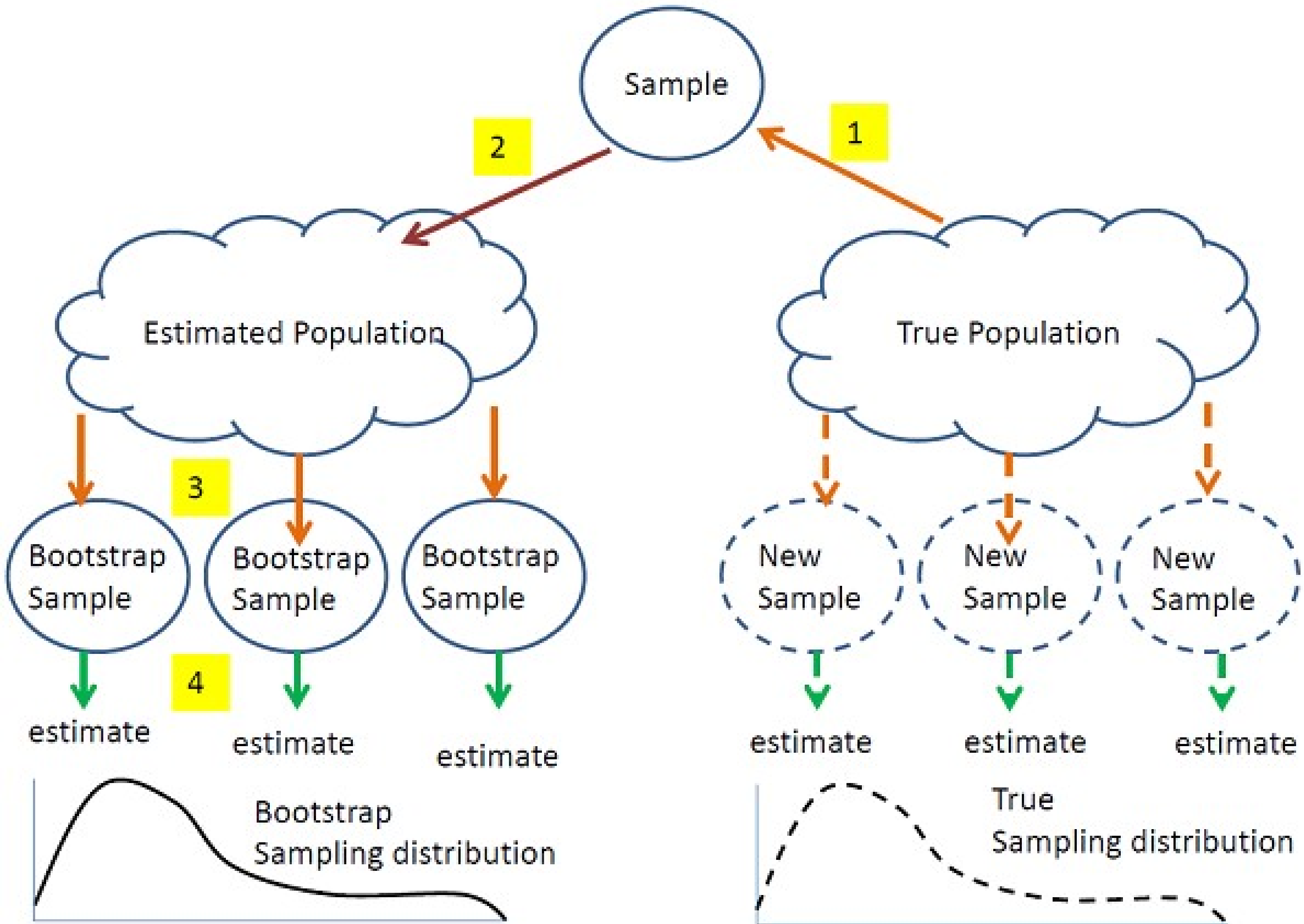
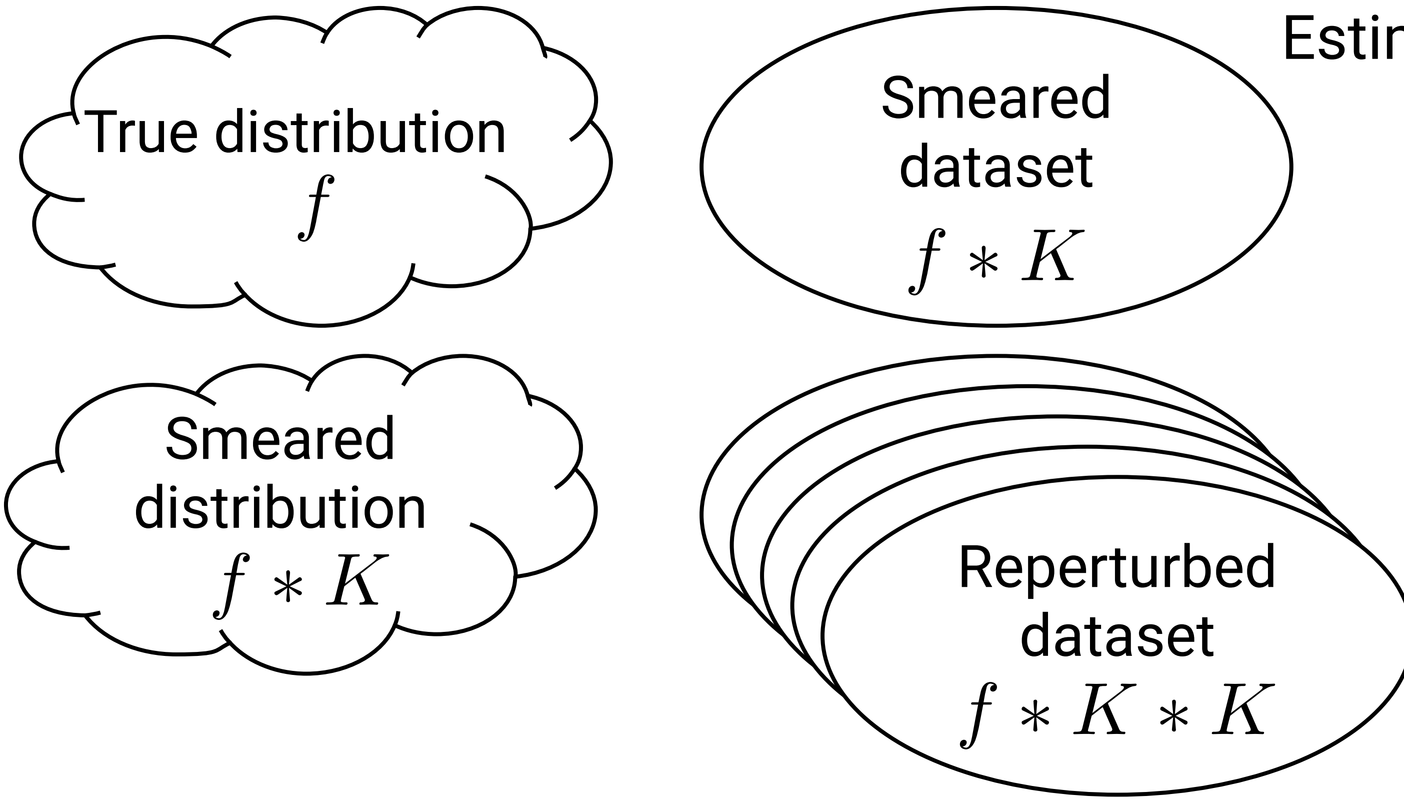


Image from https://online.stat.psu.edu/stat555/node/119/

Monte Carlo Error Propagation

Measurement error heats up the density function!
Estimation can be biased!



As long as the relative uncertainty is sufficiently small,
the leading order measurement uncertainty and bias
can be estimated from smeared dataset and reperturbed dataset.