Improving Flow Matching for Posterior Inference with Physics-based Controls

Benjamin Holzschuh [1] **Nils Thuerey** [1]

Abstract

Flow-based generative modeling is a powerful tool for solving inverse problems in physical sciences that can be used for sampling and likelihood evaluation with much lower inference times than traditional methods. We propose to refine flows with additional control signals based on an underlying physics model. In our experiments, this control signal is represented by gradients with respect to a differentiable cost function. We train a neural network to aggregate a pretrained flow and physics-based control signal to yield a hybrid update. We evaluate the refinements against classical MCMC methods for modeling strong gravitational lens systems, a challenging inverse problem in astronomy. We demonstrate that including physics-based controls improves the accuracy by 57%, making them competitive with MCMC methods while being 12x to 83x faster for inference.

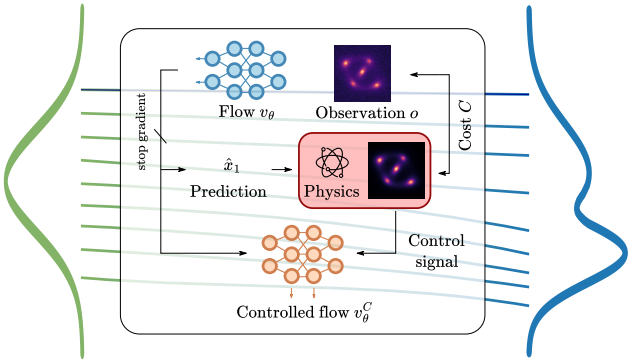

Figure 1: Overview of our proposed framework. We consider a (pretrained) flow network v_θ. We use the predicted flow at the current trajectory point x_t at time t to obtain an estimate $\hat{x}_1$ for the final prediction x_1. We assess the quality of the obtained estimate by using a physics-based model represented by a cost function C. This cost function yields a control signal in which direction to improve the estimate $\hat{x}_1$. An additional network learns to combine the predicted flow with the control signal to give a new controlled flow. By combining learning-based updates with suitable control signals, we avoid local optima and manage to obtain high-accuracy samples with low inference times.

1. Introduction

Acquiring posterior distributions given measurement data is of paramount scientific interest, with real-world applications ranging from particle physics (Baydin et al., 2019), over the inference of gravitational waves (Dax et al., 2021) to predictions of dynamical systems such as weather forecasting (Gneiting & Raftery, 2005). For an observation o and model parameters x the likelihood $p(o|x)$ corresponds to how strongly we believe a model with parameters x causes o to occur. In Bayesian modeling, we are interested in the posterior $p(x|o)$, which is proportional to the likelihood times the prior $p(x)$ and tells us which parameters are most likely to explain the observation.

Infering the posterior based on samples from traditional likelihood-based methods can be expensive for high-dimensional data and when likelihood evaluations are costly. Simulation-based inference (Cranmer et al., 2020) represents the posterior as a parametric function $q(x|o)$, which is a learnable conditional density estimator that can be trained purely by simulations $o \sim p(o|x)$ alone. By investing an upfront cost of training the density estimator, we can sample and compute likelihoods from $q(x|o)$ much faster than other methods, thereby amortizing the training cost over many observations.

Traditionally, normalizing flows (Rezende & Mohamed, 2015; Dinh et al., 2017; Papamakarios et al., 2019) have been a popular class of density estimators used in many areas of science. To compute likelihoods and for sampling, normalizing flows transform a noise distribution to the posterior distribution. With the advent of diffusion models (Ho et al., 2020) and flow matching (Liu et al., 2023; Lipman et al., 2023) it became clear that the mapping between noise and the posterior can be defined a priori, for example by specifying a corruption process that transforms any data distribution to a standard Gaussian. The resulting continuous-time models outperform discrete architecture in many areas

[1]TUM School of Computation, Information and Technology, Technical University of Munich, Garching, Germany. Correspondence to: Benjamin Holzschuh <benjamin.holzschuh@tum.de>.

Accepted by the Structured Probabilistic Inference & Generative Modeling workshop of ICML 2024, Vienna, Austria.

and training larger models is much more scalable.

We propose a simple strategy to reintroduce control signals into the flow network. We refine an existing pretrained flow-based models with arbitrary control signals by aggregating the learned flow and control signals into a controlled flow. The aggregation network is very small compared to the pretrained flow network, and we find that freezing the weights of the pretrained network works very well; thus, refining needs only a minimal amount of additional parameters and compute.

To demonstrate how these refinements affect the accuracy of samples and the posterior, we consider modeling strong gravitational lens systems (Vegetti & Koopmans, 2009; Vegetti et al., 2023), an inverse problem in astrophysics that is challenging and requires precise posteriors for accurate modeling of observations. In galaxy-scale strong lenses, light from a source galaxy is deflected by the gravitational potential of a galaxy between the source and observer, causing multiple images of the source to be seen. Since these images and their distortions are sensitive to the distribution of matter on small scales, this can act as a probe for different dark matter models. With upcoming and current sky surveys (Laureijs et al., 2011) expected to release large data catalogs in the near future, the number of known lenses will increase dramatically by several orders of magnitude. Traditional computational approaches require between several minutes to many hours or days to model a single lens system. Therefore, there is an urgent need to reduce the compute and inference with learning-based methods.

In this experiment we show that using flow matching and our proposed physics-based control signals, we obtain posterior distributions for lens modeling that are competitive with the posteriors obtained by MCMC-based methods, but with several orders of magnitude faster inference times.

2. Flow Matching

Continuous-time flow models transform samples from a sampling distribution p_0 to samples of a target or posterior distribution p_1. This mapping can be expressed via a neural ordinary equation (ODE)

$$dx_t = v_\theta(t, x_t)dt, \tag{1}$$

where $v_\theta(t, x_t)$ represents a neural network with parameters θ. Early works (Chen et al., 2018; Grathwohl et al., 2019) optimize $v_\theta(t, x)$ using maximum likelihood training, which is computationally demanding and difficult to scale to larger networks. Instead, in flow matching the network $v_\theta(t, x)$ is trained by regressing a vector field $u(t, x)$ that generates probability paths that map from p_0 to p_1.

Generating probability paths We say that a smooth vector field $u : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$, called *velocity*, generates the probability paths p_t, if it satisfies the continuity equation $\frac{\partial p}{\partial t} = -\nabla \cdot (p_t u_t)$. Informally, this means that we can sample from the distribution p_t by sampling $x_0 \sim p_0$ and then solving the ODE $dx = u(t, x)dx$ with initial condition x_0. In the following, we will denote $u(t, x)$ by $u_t(x)$. To regress the velocity field, we can define the **flow matching** objective

$$\mathcal{L}_{\text{FM}}(\theta) := \mathbb{E}_{t\sim\mathcal{U}(0,1),x\sim p_t(x)} \left|\left|v_\theta(t, x) - u_t(x)\right|\right|^2 . \tag{2}$$

In order to compute this loss, we need to sample from the probability distribution $p_t(x)$ and know the velocity $u_t(x)$. However, in general $u_t(x)$ is not accessible.

Conditioning variable To solve this problem, we can apply a trick by introducing a latent variable z distributed according to $q(z)$ and define the conditional likelihoods $p_t(x|z)$ that depend on the latent variable so that $p_t(x) = \int p_t(x|z)q(z)dz$. Interestingly, if the conditional likelihoods are generated by the velocities $u_t(x|z)$, then the velocity $u_t(x)$ can be written in terms of $u_t(x|z)$ and $p_t(x|z)$ with $u_t(x) := \mathbb{E}_{q(z)}[u_t(x|z)p_t(x|z)/p_t(x)]$. We can choose paths $p_t(x|z)$ that are easy to sample from and for which we know the generating velocities $u_t(x|z)$. Next, we define the **conditional flow matching** loss

$$\mathcal{L}_{\text{CFM}}(\theta) := \mathbb{E}_{t,q(z),p_t(x|z)} \left|\left|v_\theta(t, x) - u_t(x|z)\right|\right|^2 , \tag{3}$$

In contrast to the flow matching loss (2), this loss is tractable and can be used for optimization. Now, one can show (Tong et al., 2023) that if $p_t(x) > 0$ for all $x \in \mathbb{R}^d$, then

$$\nabla_\theta \mathcal{L}_{\text{FM}}(\theta) = \nabla_\theta \mathcal{L}_{\text{CFM}}(\theta). \tag{4}$$

This means that we can train $v_\theta(x, t)$ to regress $u_t(x)$ generating the mapping between p_0 and p_1 by optimizing the conditional flow matching loss (3).

2.1. Couplings

The above framework allows for many degrees of freedom when specifying the mapping from p_0 to p_1 via the conditioning variable z and the conditional likelihoods p_t.

Conditional optimal transport One particularly intuitive and simple choice is to consider the coupling $q(z) = p_1(x)$ (Lipman et al., 2023) together with conditional probability and generating velocity

$$p_t(x|x_1) = \mathcal{N}(x|\, tx_1, (1 - (1 - \sigma_{\min})t)I) \tag{5}$$

$$u_t(x|x_1) = \frac{x_1 - (1 - \sigma_{\min})x}{1 - (1 - \sigma_{\min})t}, \tag{6}$$

where $\sigma_{\min} > 0$. Conditioned on x_1, this coupling transports a point $x_0 \sim \mathcal{N}(0, I)$ from the sampling distribution to the posterior distribution on the linear trajectory tx_1 ending in x_1 but decreasing the standard deviation from 1 to

a smoothing constant $\sigma_{\min}$. In this case, the transport path coincides with the optimal transport between two Gaussian distributions.

3. Controls for Improved Accuracy

While flow-based models $v_\theta(t, x)$ gradually transform samples from p_0 to p_1 in many steps during inference via solving the ODE (1), there is no direct feedback loop between the underlying physics-based model, the current point on the trajectory x_t, and the observation o. We would like to reintroduce this feedback loop into the inference and training.

Conditioning of flows Flows $v_\theta(t, x)$ can be conditioned on an observation o through an additional input $v_\theta(t, x, o)$, therefore modeling the conditional densities $p_t(x|o)$ (Song et al., 2021). For example, in classifier free-guidance (Ho & Salimans, 2022), this conditioning is randomly dropped and set to 0 during training. The resulting models can then be used for both conditional and unconditional generation. A fundamental problem here is that the conditioning o is static, whereas we propose to have a dynamic control mechanism that depends on the trajectory x_t, the observation, and an underlying control signal. The latter should relate x_t and observation using a physics-based model represented through a cost function C. As the accuracy of neural networks is inherently limited by the finite size of their weights, and smaller networks are attractive from a computational perspective, physics-based control has the potential to yield high accuracy with lean and efficient neural network models.

1-step prediction An additional issue is that the current trajectory x_t might not be close to a good estimate of a posterior sample x_1, especially at the beginning of inference, where x_0 is drawn from the sampling distribution. Instead of applying the cost function C to the current estimate x_t, we extrapolate x_t forward in time to obtain an estimated $\hat{x}_1$

$$\hat{x}_1 = x_t + (1 - t)v_\theta(t, x_t, o) \tag{7}$$

to alleviate this issue.

Flows based on linear conditional transportation paths have empirically been shown to have trajectories with less curvature (Lipman et al., 2023) compared to, for example, diffusion models, thus enabling inference in fewer steps and providing better estimates for $\hat{x}_1$.

Self-conditioning Conditioning a model on something that depends on its own output can be seen as a form of self-conditioning. We showcase an adapted version of self-conditioning (Chen et al., 2023) in algorithm 1. The main idea of the algorithm is that instead of providing x_t to the network, the input is comprised of a tuple $[x_t; \hat{x}_1]$, where $\hat{x}_1$ is either 0 or the 1-step prediction (7). By conditioning a model on its own output, we simulate feedback, which lets the network learn to identify and correct its own mistakes without the need to backpropagate gradients through multiple network passes.

3.1. Controls

Aiming for high inference accuracy, we extend self-conditioning via physics-based control signals to include an additional feedback loop between the model output and an underlying physics-based prior.

Control signal c Given an observation o and the estimated prediction $\hat{x}_1$, we can calculate how well $\hat{x}_1$ explains o via some cost function C. The cost function can also depend directly on the likelihood $p(o|\hat{x}_1)$. For a differentiable cost function C, we define the control signal via

$$c(\hat{x}_1, o) := [C(\hat{x}_1, o); -\nabla_x C(\hat{x}_1, o)]. \tag{8}$$

Note that while we recommend using informative control signals, we can use any control that depends on $\hat{x}_1$ and o. In our astrophysics experiment in section 4, we use an auto-differentiable ray-tracer (Galan et al., 2022) to evaluate how well the estimate $\hat{x}_1$ matches the observation o.

Controlled flow v_θ^C We pretrain the flow network $v_\theta(t, x, o)$ without any control signals to make sure that we can realize the best achievable performance possible based on learning alone. Then, in a second training phase, we introduce the control network $v_\theta^C(t, v, c)$. The task of the control network is to aggregate the pretrained flow with the control signal c. The control network is much smaller than the flow network, making up ca. 10% of the weights θ in our experiments. We freeze the network weights of v_θ and train with the conditional flow matching loss (3) for a small number of additional steps. This reduces training time and compute since we do not need to backpropagate through $v_\theta(x, t)$. We did not find that freezing the weights of v_θ affects the performance negatively.

Time-dependence We notice that if the estimate $\hat{x}_1$ is bad and the corresponding cost $C(\hat{x}_1, o)$ is high, gradients and control signals can be highly unreliable. In our main experiment in section 4, we empirically find that the estimates $\hat{x}_1$ improve for $t \geq 0.8$. Therefore, we only train the control network for $t \geq 0.8$, which allows for focusing on control signals containing useful information. For $t < 0.8$, we directly output the pretrained flow $v_\theta(t, x, o)$.

We summarize our proposed control-based training in algorithm 2, assuming conditional optimal transport paths (5). For a more compact representation, we omit the time dependence. Note that for each training step, we require one evaluation of v_θ, whereas self-conditioning requires two evaluations.

Algorithm 1 FM with Self-conditioning

Input: Training distribution q_1, flow network v_θ, $\sigma_{\min}$
while Training **do**
  $(x_1, o) \sim q_1$; $z \leftarrow \mathcal{N}(0, I)$; $s \leftarrow \mathcal{U}(0, 1)$
  $x \leftarrow tx_1 + (1 - (1 - \sigma_{\min})t)z$; $\hat{x}_1 \leftarrow 0$
  $v \leftarrow \text{stopgrad}(v_\theta(t, [x, \hat{x}_1], o))$
  if $s > 0.5$ **then**
    $\hat{x}_1 \leftarrow x + (1 - t)v$
  $\tilde{v} \leftarrow v_\theta(t, [x, \hat{x}_1], o)$
  $u_t(x|x_1, o) \leftarrow \frac{x_1 - (1-\sigma_{\min})x}{1-(1-\sigma_{\min})t}$
  $\mathcal{L}_{\text{CFM}} \leftarrow ||\tilde{v} - u_t(x|x_1, o)||_2^2$
  $\theta \leftarrow \text{Update}(\theta, \nabla_\theta \mathcal{L}_{\text{CFM}}(\theta))$
return: v_θ

Algorithm 2 FM with Control Signals

Input: Training distribution q_1, pretrained network v_θ, control network v_θ^C, $\sigma_{\min}$
while Training **do**
  $(x_1, o) \sim q_1$; $z \leftarrow \mathcal{N}(0, I)$
  $x \leftarrow tx_1 + (1 - (1 - \sigma_{\min})t)z$
  $v \leftarrow \text{stopgrad}(v_\theta(t, x, o))$
  $\hat{x}_1 \leftarrow x + (1 - t)v$
  $c \leftarrow \text{control}(\hat{x}_1, o)$
  $\tilde{v} \leftarrow v_\theta^C(t, v, c) + v$
  $u_t(x|x_1, o) \leftarrow \frac{x_1 - (1-\sigma_{\min})x}{1-(1-\sigma_{\min})t}$
  $\mathcal{L}_{\text{CFM}} \leftarrow ||\tilde{v} - u_t(x|x_1, o)||_2^2$
  $\theta \leftarrow \text{Update}(\theta, \nabla_\theta \mathcal{L}_{\text{CFM}}(\theta))$
return: v_θ, v_θ^C

4. Strong Gravitational Lensing

Strong gravitational lensing is a physical phenomenon whereby the light rays by a distant object, such as a galaxy, are deflected by an intervening massive object, such as another galaxy or a galaxy cluster. As a result, one observes multiple distorted images of the background source. We aim to recover both the lens and source light distribution as well as the lens mass density distribution. This experiment focuses on realistic simulated observations for which we know the ground truths.

Lens modeling The *lens equation* relates coordinates on the source plane β and the observed image plane Θ via the deflection angle α induced by the mass profile or gravitational potential of the lens galaxy. For each component we use a parametric model (Singular Isothermal Ellipsoid for the lens, Sérsic profiles for the light). The likelihood is measured by the χ^2-statistic, which is the modeled image plane Θ minus the observation o divided by the noise. To solve the lensing equation, we make use of the publicly available raytracing code by (Galan et al., 2022). We stress that even small perturbations of the model parameters can cause the χ_2 to increase significantly; see figure 3 in the appendix.

Pretraining The network v_θ is trained via flow matching (Lipman et al., 2023). The flow network v_θ consists of a convolutional feature extraction neural network represented by a shallow convolutional network with 8 layers and fed with x_t and t to a feed-forward network with residual blocks and skip-connections. Overall, it comprises ca. 480K parameters, making it fast for inference.

Finetuning with control signals The control network v_θ^C is represented by a feed-forward network with residual blocks and gated linear units. This architecture accounts for ca. 11% of all parameters in the combined model. The control signals are obtained from simulating an observation based on the predicted estimate $\hat{x}_1$ via ray-tracing based on the parametric models, calculating the χ_2-statistic and computing gradients with respect to the estimate $\hat{x}_1$. See figure 2 for a comparison of the simulated reconstructions.

Table 1: Evaluation with respect to average χ_2 and inference time for the posterior distribution.

Method	Avg. $\chi_2 \downarrow$	Modeling Time $\downarrow$
HMC	1.67	$\sim$ 275x (2200s)
AIES	1.42	$\sim$ 83x (671s)
Flow Matching		
10 steps	1.71	1x (**8s**)
+ Controls		
10 steps	1.48	$\sim$ 3x (26s)
50 steps	**1.40**	$\sim$ 6x (50s)

4.1. Evaluation and Discussion

As reference posteriors, we include Hamiltonian Monte Carlo (HMC) with No-U-Turn sampler (Hoffman et al., 2014) and Affine-Invariant Ensemble Sampling (Goodman & Weare, 2010, AIES), which are both two popular MCMC-methods in astronomy.

χ_2-statistic We show an evaluation of all methods in table 1. For both flow matching methods, we use Euler integration. The average χ_2 is computed over 24 randomly chosen validation systems, where for each, we draw 100 samples from the posterior. If we compute the χ_2 for the ground truth parameters, we obtain a value of 1.17 due to the noise in the observation. Since we cannot overfit to noise with the parametric models, this represents a lower bound for χ_2 in this experiment. Including the physics-based control

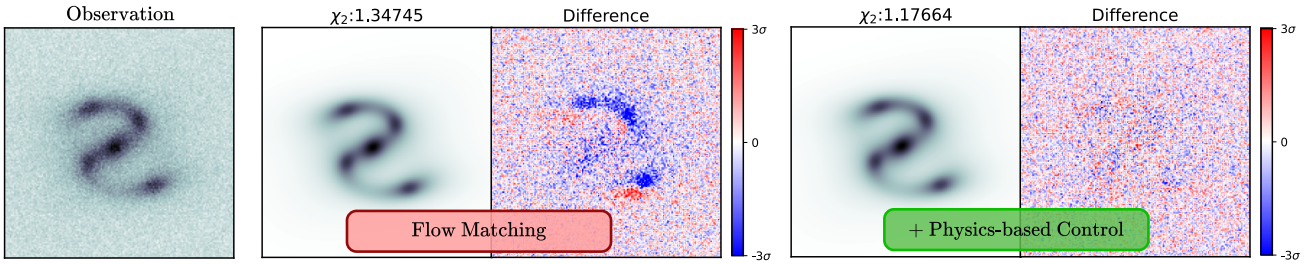

Figure 2: Reconstruction of observation (left) with parameters sampled from the posterior. Flow matching is purely learning-based and shows noticeable residuals in the reconstruction, whereas adding the physics-based control signal removes remaining residuals. Our proposed method has wider posteriors than the two MCMC reference methods and covers their support.

improves the χ_2 from 1.71 to 1.40, representing an improvement of 57% relative to the best modeling. We define the modeling time as the average compute time required to produce 1000 credible posterior samples. Both HMC and AIES require significant warmup times before producing the first samples from the posterior, which we include in the table. However, after warmup, it is relatively cheap to obtain new samples. On the other hand, flow matching does not require any warmup time and the modeling time increases linearly with the number of posterior samples. All methods were implemented in JAX (Bradbury et al., 2018) and used the same hardware. The measurements in table 1 highlight that our method yields an accuracy that surpasses AIES, while being more than 13x faster.

This evaluation demonstrates that learning-based methods are highly competitive even in small to moderate-sized problems with established MCMC-based methods in terms of accuracy, clearly beating them in terms of inference time. Flow matching with our proposed control signals is especially interesting because it is not affected as much by the curse of dimensionality as traditional inference methods and allows for having non-trivial learnable high-dimensional priors. However, before these methods are widely trusted, they need to demonstrate their competitiveness with traditional methods. Our results show that this is indeed the case, which opens up exciting avenues for applying and developing approaches targeting similar and adjacent inverse problems in science.

5. Conclusion

We presented a method for improving flow-based models with additional control signals. This allows us to refine an existing flow with only a few additional weights and little training time. We thereby efficiently bridge the gap between purely learning-based methods for simulation-based inference and optimization with hand-crafted cost functions within the framework of flow matching. This improvement is critical for scientific applications where high accuracy and trustworthiness in the methods are required. Purely learning-based methods face significant difficulties in producing very accurate samples, as there is usually no feedback during inference of how good samples are. In this paper, we demonstrated that we do not need large network sizes or tremendous amounts of data to train accurate models that are competitive with established MCMC methods if we include suitable control signals from physics-based models. We believe this work makes an important step towards making posterior inference in science more accurate, understandable, and reliable.

References

Baydin, A. G., Shao, L., Bhimji, W., Heinrich, L., Naderiparizi, S., Munk, A., Liu, J., Gram-Hansen, B., Louppe, G., Meadows, L., Torr, P. H. S., Lee, V. W., Cranmer, K., Prabhat, and Wood, F. Efficient probabilistic inference in the quest for physics beyond the standard model. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 5460–5473, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/6d19c113404cee55b4036fce1a37c058-Abstract.html.

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P. A., Horsfall, P., and Goodman, N. D. Pyro: Deep universal probabilistic programming. *J. Mach. Learn. Res.*, 20:28:1–28:6, 2019. URL http://jmlr.org/papers/v20/18-403.html.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Chen, T., Zhang, R., and Hinton, G. E. Analog

bits: Generating discrete data using diffusion models with self-conditioning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=3itjR9QxFw.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 6572–6583, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/69386f6bb1dfed68692a24c8686939b9-Abstract.html.

Cranmer, K., Brehmer, J., and Louppe, G. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020.

Dax, M., Green, S. R., Gair, J., Macke, J. H., Buonanno, A., and Schölkopf, B. Real-time gravitational wave science with neural posterior estimation. *Physical review letters*, 127(24):241103, 2021.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=HkpbnH9lx.

Galan, A., Vernardos, G., Peel, A., Courbin, F., and Starck, J. L. Using wavelets to capture deviations from smoothness in galaxy-scale strong lenses. *Astronomy and Astrophysics*, 668:A155, December 2022. doi: 10.1051/0004-6361/202244464.

Gneiting, T. and Raftery, A. E. Weather forecasting with ensemble methods. *Science*, 310(5746):248–249, 2005.

Goodman, J. and Weare, J. Ensemble samplers with affine invariance. *Communications in applied mathematics and computational science*, 5(1):65–80, 2010.

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., and Duvenaud, D. FFJORD: free-form continuous dynamics for scalable reversible generative models. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rJxgknCcK7.

Ho, J. and Salimans, T. Classifier-free diffusion guidance. *CoRR*, abs/2207.12598, 2022. doi: 10.48550/ARXIV.2207.12598. URL https://doi.org/10.48550/arXiv.2207.12598.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

Hoffman, M. D., Gelman, A., et al. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

Kelley, C. T. *Iterative methods for optimization*. SIAM, 1999.

Laureijs, R., Amiaux, J., Arduini, S., Augueres, J.-L., Brinchmann, J., Cole, R., Cropper, M., Dabin, C., Duvet, L., Ealet, A., et al. Euclid definition study report. *arXiv preprint arXiv:1110.3193*, 2011.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=PqvMRDCJT9t.

Liu, X., Gong, C., and Liu, Q. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=XVjTT1nw5z.

Papamakarios, G., Sterratt, D. C., and Murray, I. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In Chaudhuri, K. and Sugiyama, M. (eds.), *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pp. 837–848. PMLR, 2019. URL http://proceedings.mlr.press/v89/papamakarios19a.html.

Phan, D., Pradhan, N., and Jankowiak, M. Composable effects for flexible and accelerated probabilistic programming in numpyro. *arXiv preprint arXiv:1912.11554*, 2019.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. In Bach, F. R. and Blei, D. M. (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1530–1538. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/rezende15.html.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Tong, A., Malkin, N., Huguet, G., Zhang, Y., Rector-Brooks, J., Fatras, K., Wolf, G., and Bengio, Y. Conditional flow matching: Simulation-free dynamic optimal transport. *CoRR*, abs/2302.00482, 2023. doi: 10.48550/ARXIV.2302.00482. URL https://doi.org/10.48550/arXiv.2302.00482.

Vegetti, S. and Koopmans, L. V. Bayesian strong gravitational-lens modelling on adaptive grids: objective detection of mass substructure in galaxies. *Monthly Notices of the Royal Astronomical Society*, 392(3):945–963, 2009.

Vegetti, S., Birrer, S., Despali, G., Fassnacht, C., Gilman, D., Hezaveh, Y., Levasseur, L. P., McKean, J., Powell, D., O'Riordan, C., et al. Strong gravitational lensing as a probe of dark matter. *arXiv preprint arXiv:2306.11781*, 2023.

A. Implementation Details

Lens models We consider the following models:

- For modeling the lens we use an Singular Isothermal Ellipsoid (SIE) model with 6 parameters: the Einstein radius θ_E, the ellipticities e_1 and e_2 and x_{center} and y_{center}. There is shear, for which we only consider γ_1 and γ_2 as free parameters.
- The source is modeled by a Sersic profile with free parameters being the amplitude, the half-light radius, the Sersic index n, the ellipticities e_1 and e_2 as well as the positions x_{center} and y_{center}.
- The lens light is modeled in the same way as the source, although when generating the mock data, we fix the position as well as ellipticities to be the same as the lens mass model. For training and inference, we infer positions for both lens mass and lens light model, so the model could produce different values for them. The MCMC methods use the same parameter for both lens light and lens mass position.

We list all priors in table 2, table 3 and table 4. We do not have priors on the ellipticities e_1 and e_2 directly, but we obtain them from priors on the position angle and axis ratio. Also, we obtain the shear parameters from γ_1 and γ_2 from ϕ_{ext} and γ_{ext} by converting them polar to cartesian coordinates. For simulation-based inference (SBI), we also include the two parameters ra_0 and dec_0 for the shear, which are always set to 0 when generating the training data sets, but in general our network could infer other values. Overall, there are 23 parameters for v_θ, which fully describe the simulation setup. However, in our dataset there are only 17 free parameters. The MCMC methods only infer the reduced set of parameters, making use of the dependencies between them.

Measurement instruments Observations have 160 times 160 pixels. The pixel size is 0.04 arc seconds. We use a Gaussian points spread function (PSF) with full width at half maximum (FWHM) of 0.3. The there is Gaussian background noise with a root mean-squared values of 0.01 and an exposure time of 1000s.

Data For training the flow network, we require a dataset consisting of pairs of ground truth parameters x and corresponding observations o. Several instrument-specific measurement effects are included when simulating the observations. We include background and Poisson noise and smoothing by a point-spread function (PSF). These directly affect the posterior, as more noise and a stronger PSF will widen the posterior distribution. We generate 125.000 data samples for training and 25.000 for validation.

Setup of MCMC-based methods We setup both baselines methods as follows:

1. Hamiltonian Monte Carlo: we use the No-U-Turn samples together with a window adaptation strategy for a better mass matrix and step sizes. The window adaptation strategy from *blackjax* requires an initial guess. We obtain this initial guess by perturbing the ground truth parameters by adding a Gaussian with mean 0 and covariance matrix σI, where $\sigma = 0.01$. Then, we apply BFGS (Kelley, 1999) for a maximum number of 500 iterations using an implementation from scipy. We begin sampling after the adaptation strategy has finished warming up.
2. Affine-Invariant Ensemble Sampling: we use DEMove and StretchMove both with probability 0.5. There are 400 chains and we warm up for 20K steps before starting sampling.

Both methods are implemented in numpyro (Phan et al., 2019; Bingham et al., 2019) and optimized with JAX, so their runtimes are comparable with each other.

Network architectures and training

- Our flow network v_θ comprises a lightweight feature extraction network, representated by a convolutional neural network, which is consists of 6 downsampling blocks with 1 layer each a 32 channels and kernel size 3. As post-processing of the output, we apply GroupNorm, silu and an additional 2DConv layer with kernel size 3 and a single channel. We reshape the output and feed it through a final dense layer. The output of the feature extraction has the same dimensionality as the parameters x.

Table 2: Priors for lens mass model parameters

Parameter	Prior
x_{center}	$\mathcal{U}(-0.2, 0.2)$
y_{center}	$\mathcal{U}(-0.2, 0.2)$
position angle ϕ	$\mathcal{U}(0, 180)$
axis ratio q	$\mathcal{U}(0.25, 1)$
external shear orientation ϕ_{ext}	$\mathcal{U}(0, 180)$
external shear strength γ_{ext}	$\mathcal{U}(0, 0.1)$
Einstein radius θ_E	$\mathcal{U}(0.5, 2.0)$

Table 3: Priors for the source light

Parameter	Prior
amplitude	$\mathcal{U}(5.0, 10.0)$
half-light radius	$\mathcal{U}(0.5, 2.0)$
Sersic index n	$\mathcal{U}(1.5, 4.0)$
position angle ϕ	$\mathcal{U}(0, 180)$
axis ratio q	$\mathcal{U}(0.25, 1)$
x_{center}	$\mathcal{U}(-0.2, 0.2)$
y_{center}	$\mathcal{U}(-0.2, 0.2)$

Table 4: Priors for the lens light

Parameter	Prior
amplitude	$\mathcal{U}(5.0, 10.0)$
half-light radius	$\mathcal{U}(0.5, 2.0)$
Sersic index n	$\mathcal{U}(1.5, 4.0)$

- An additional dense feed-forward neural network receives the concatenated the time t, x_t and extracted features as input. The feed-forward neural neural networks consists of 8 residual blocks with hidden dimension 128 and elu activation.
- The control network v_θ^C is represented by a small feed-forward neural network, consisting of 3 residual blocks with 64 hidden layers and 3 residual blocks with 32 hidden layers. We condition each block on the time via gated linear units and use a 16 dimensional time embedding.

For training, we use a batch size of 256 for the flow network v_θ. When training v_θ^C, we decrease the batch size to 16. We use the Adam optimizer with learning rate 10^{-4} and weight decay 10^{-5}. Training v_θ was done on a single NVIDIA Ampere A100 GPU for ca. 45 hours and 1.250K steps. We trained v_θ^C for an additional 24 hours and 350K steps. A lot of the training time was spent on running evaluation metrics, so it can be substantially improved. For inference, all methods were run on a single NVIDIA RTX 2070 Ti.

B. Posteriors and Reconstructions for Lens Modeling

We show how small perturbations in the lens system's parameters affect the simulated observation in figure 3. We show extended plots of the posteriors as well as reconstructions based on non-cherry-picked random samples from the posterior for three lens systems: See figure 4, figure 5 and figure 6 for systems 1. See figure 7, figure 8 and figure 9 for systems 2. See figure 10, figure 11 and figure 12 for systems 3.

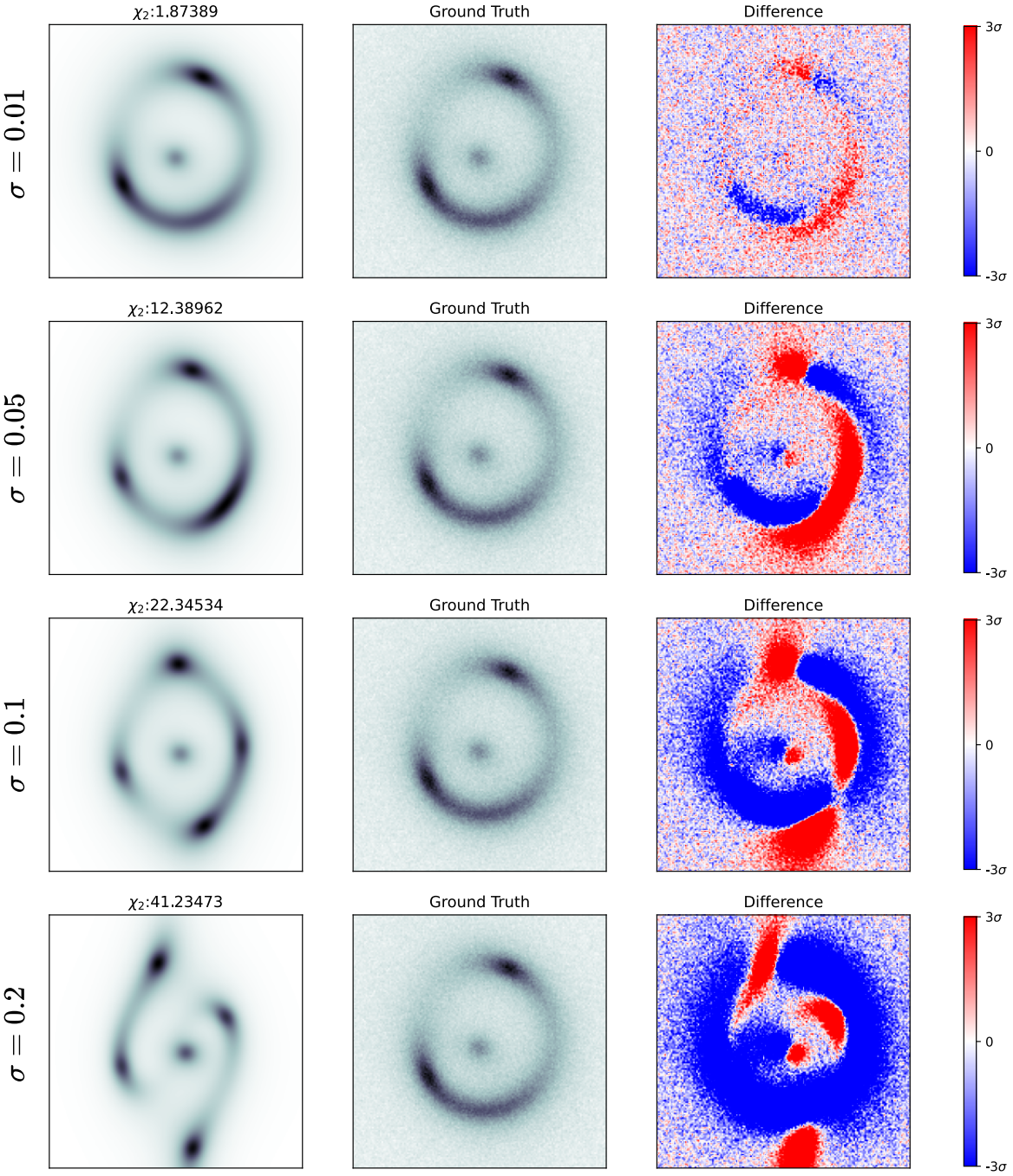

Figure 3: We show how a small noise σ affects the simulated observation. We add a normal Gaussian with mean 0 and standard deviation σ to a lens system's ground truth parameters x. Then, we plot the simulated observation based on the noised parameters and show the residuals.

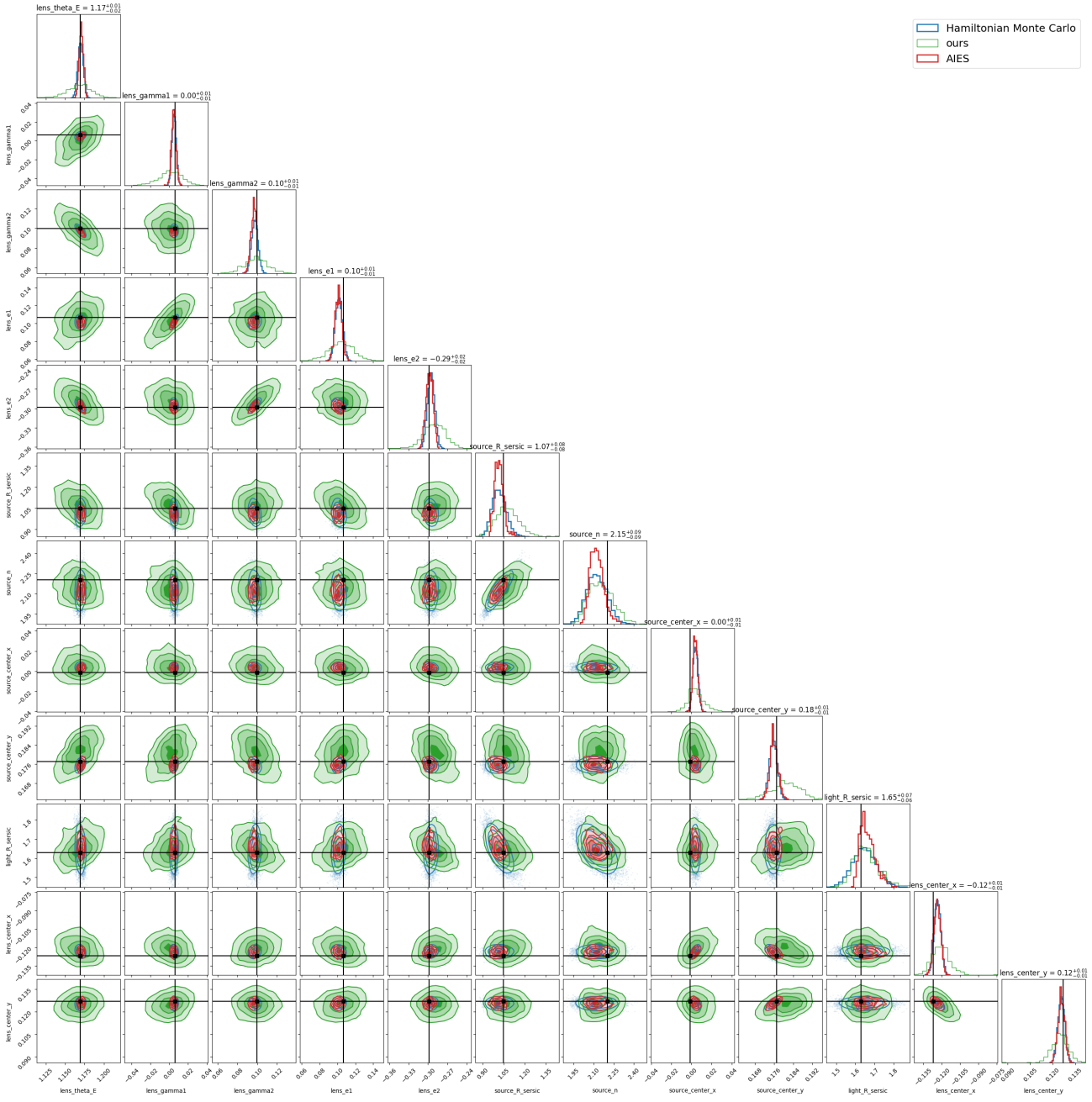

Figure 4: System 1: extended plot of the posterior for figure 2.

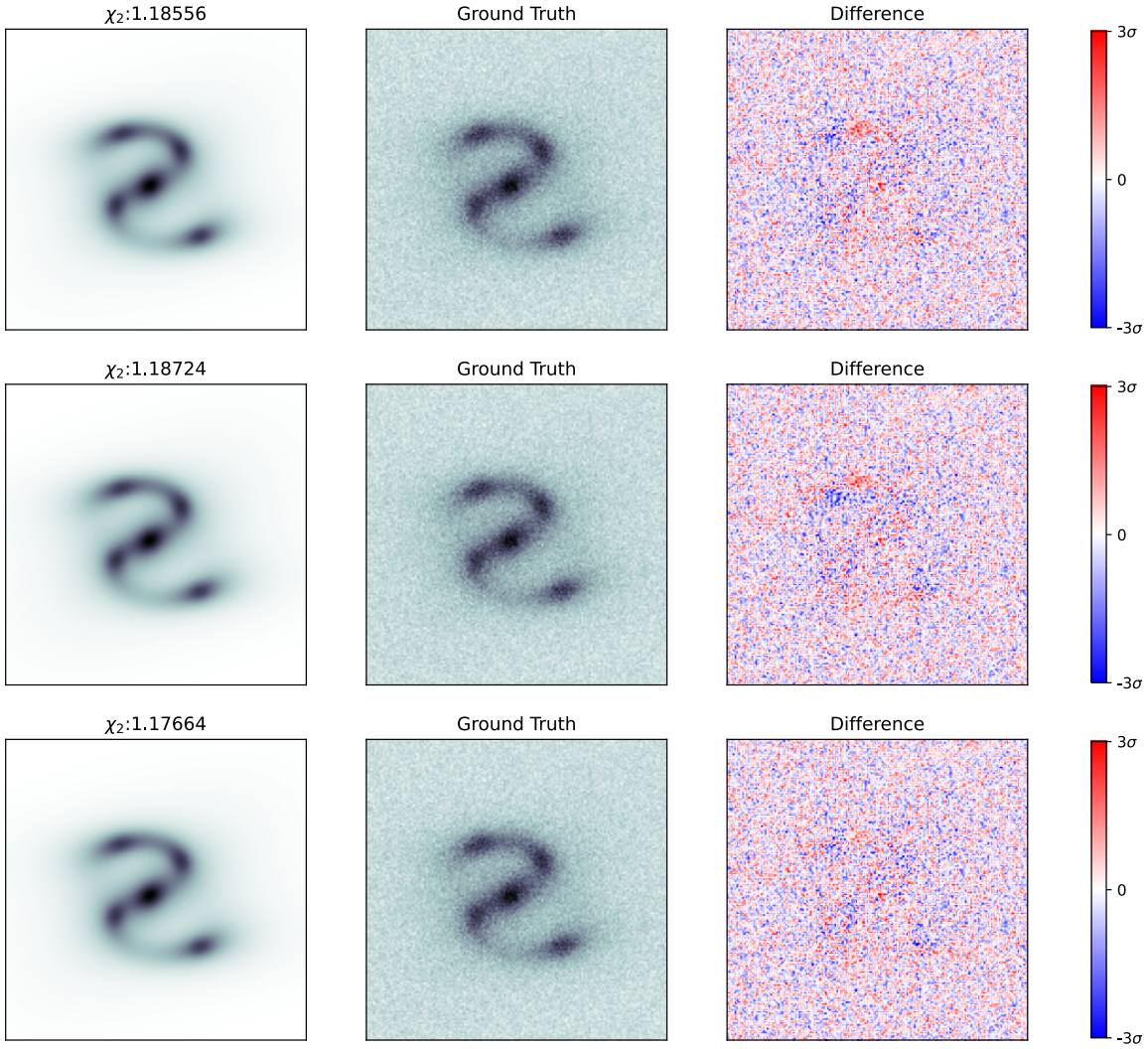

Figure 5: System 1: Three non cherry-picked simulations based on random samples from the posterior obtained with **flow matching + physics-based controls**.

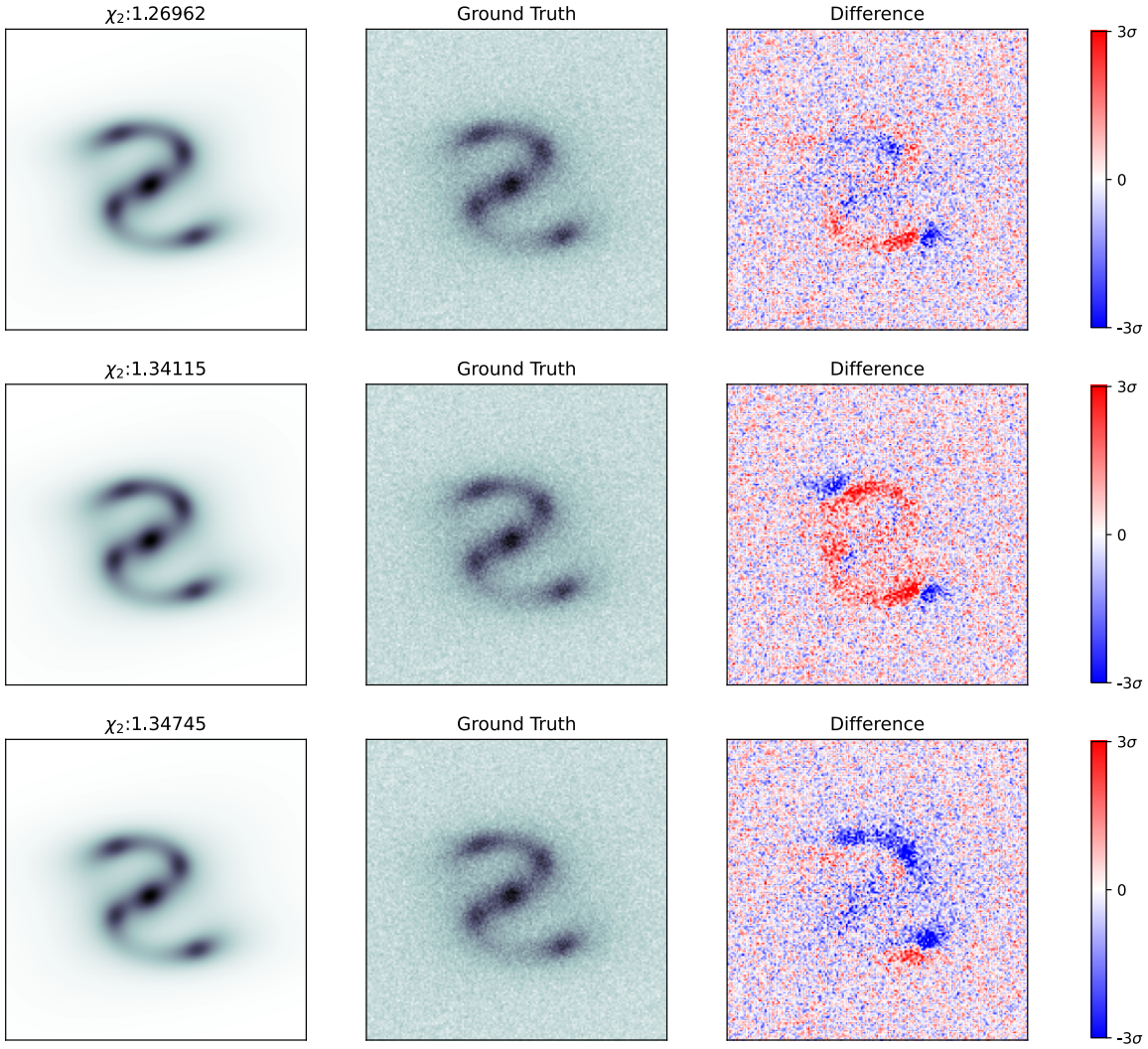

Figure 6: System 1: Three non cherry-picked simulations based on random samples from the posterior obtained with **flow matching** and no controls.

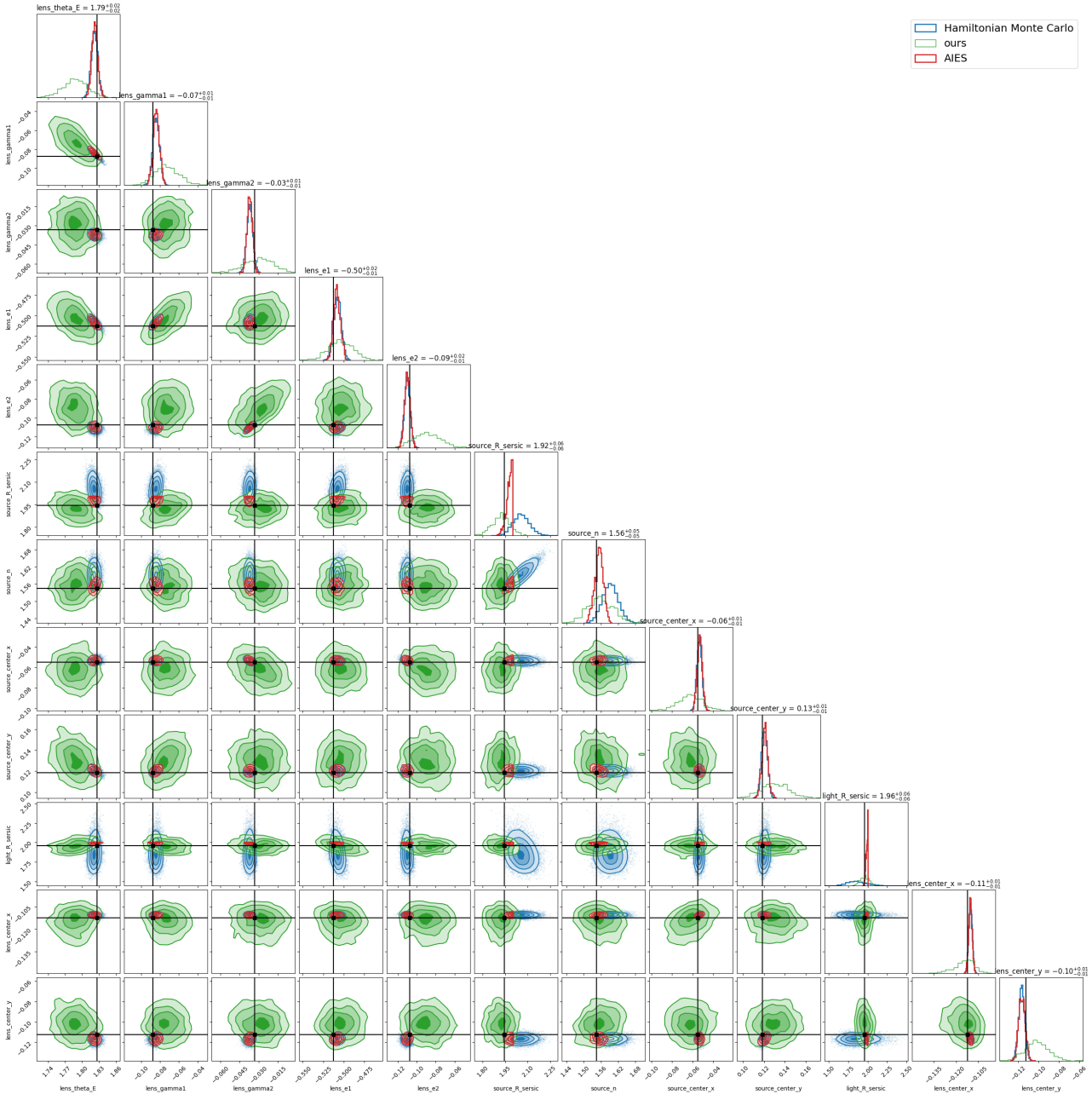

Figure 7: System 2: posterior plot.

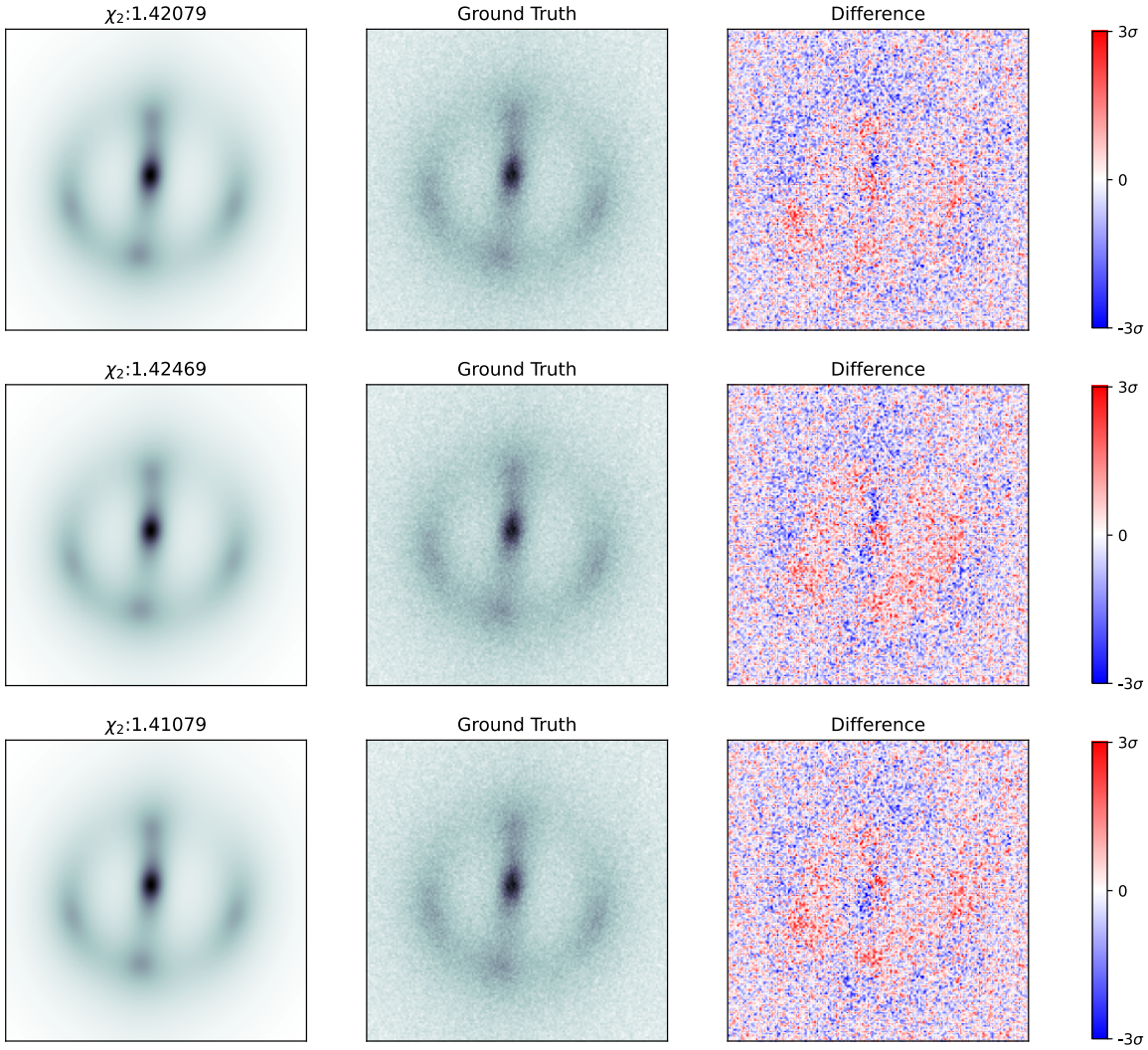

Figure 8: System 2: Three non-cherry-picked simulations based on random samples from the posterior obtained with **flow matching + physics-based controls**.

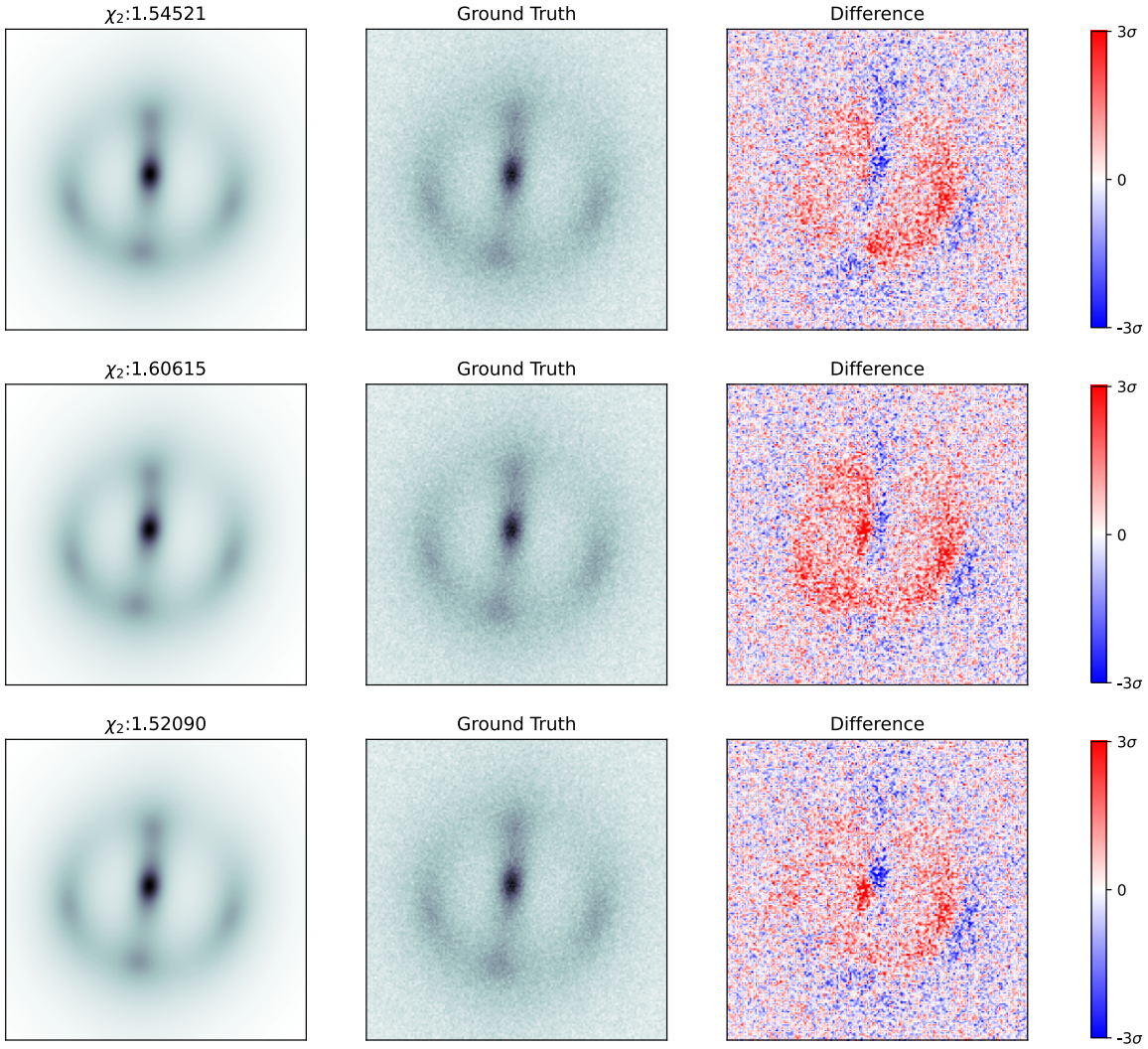

Figure 9: System 2: Three non cherry-picked simulations based on random samples from the posterior obtained with **flow matching** and no controls.

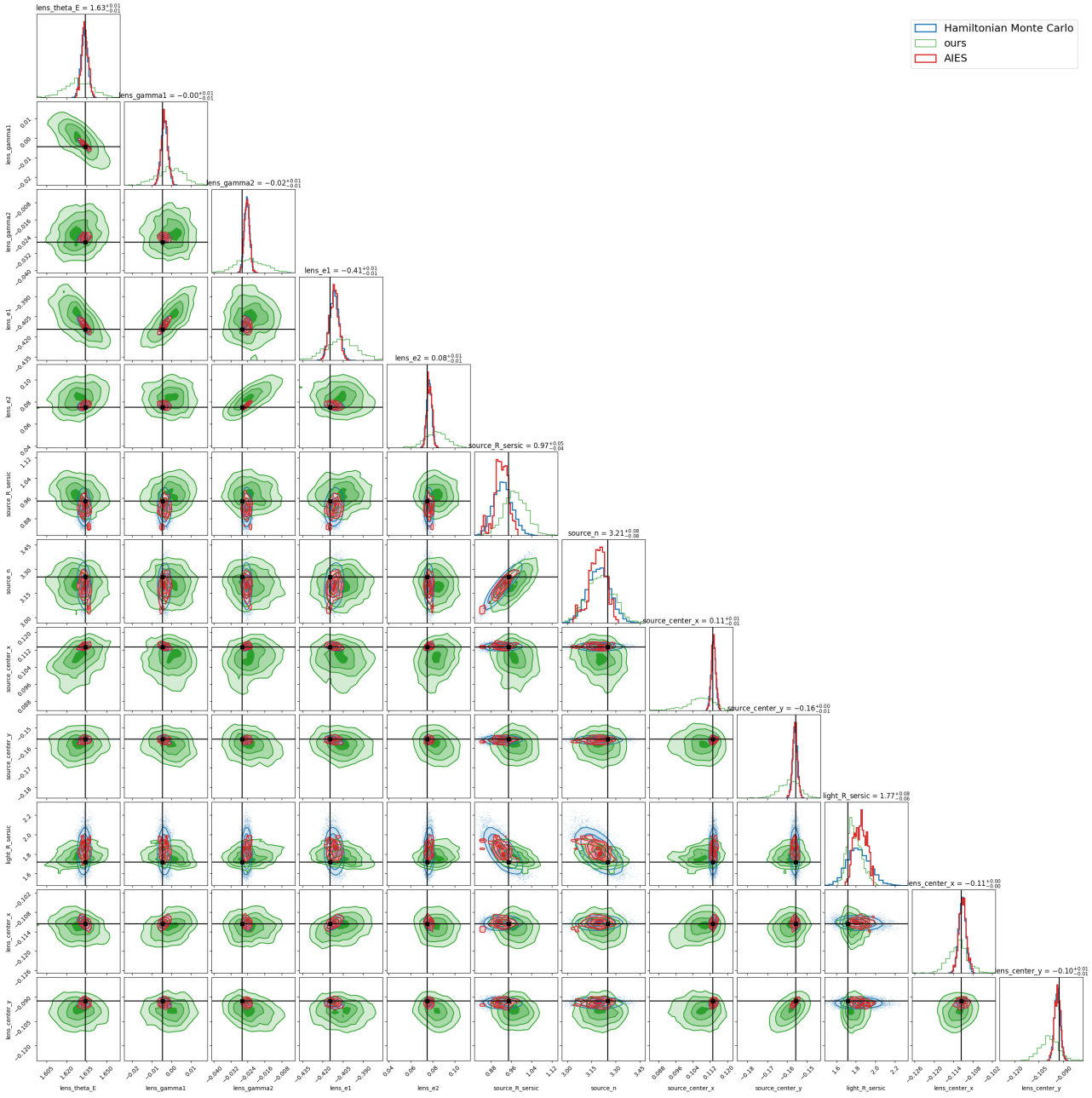

Figure 10: System 3: posterior plot.

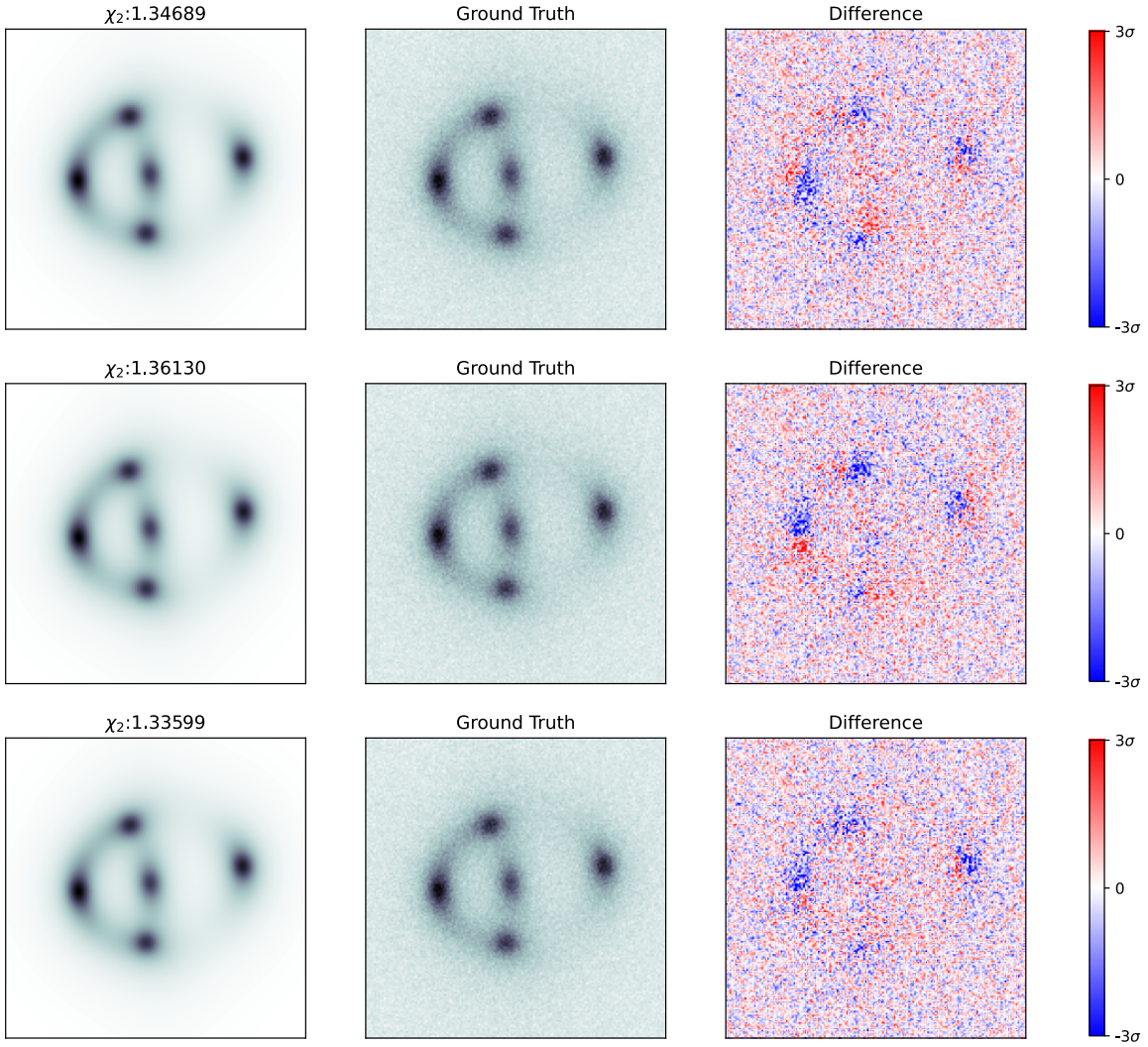

Figure 11: System 3: Three non-cherry-picked simulations based on random samples from the posterior obtained with **flow matching + physics-based controls**.

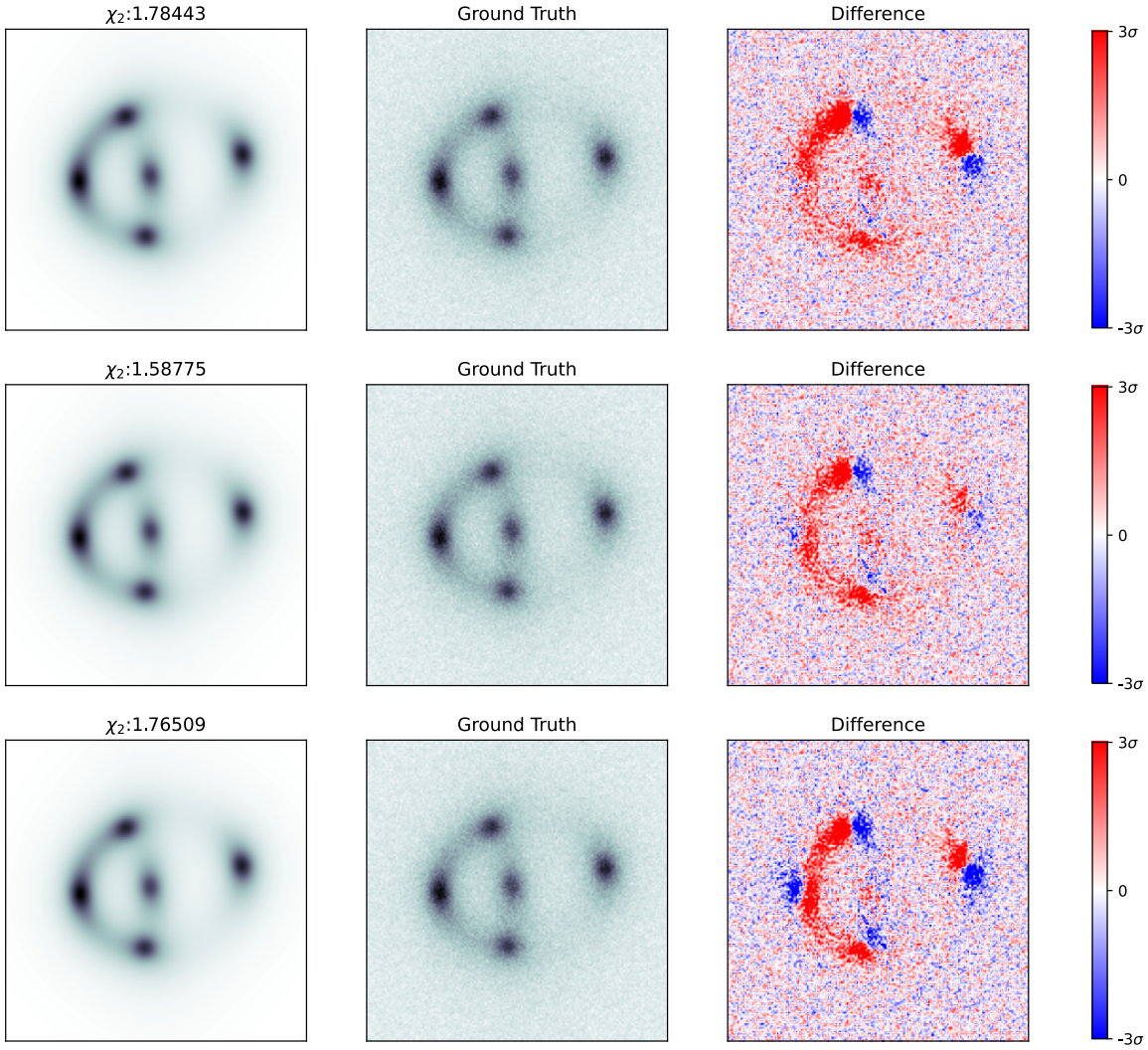

Figure 12: System 3: Three non-cherry-picked simulations based on random samples from the posterior obtained with **flow matching** and no controls.