The Journal of Open Source Software

Re-Envisioning Numerical Information Field Theory (NIFTy.re): A Library for Gaussian Processes and Variational Inference

Gordian Edenhofer [1,2,3¶], **Philipp Frank** [1], **Jakob Roth** [1,2,4], **Reimar H. Leike** [5], **Massin Guerdi** [2], **Lukas I. Scheel-Platz** [2,4,6], **Matteo Guardiani** [1,2], **Vincent Eberle** [1,2], **Margret Westerkamp** [1,2], and **Torsten A. Enßlin** [1,2]

1 Max Planck Institute for Astrophysics, Karl-Schwarzschild-Straße 1, 85748 Garching bei München, Germany **2** Ludwig Maximilian University of Munich, Geschwister-Scholl-Platz 1, 80539 München, Germany **3** University of Vienna, Türkenschanzstrasse 17, A-1180 Wien, Austria **4** Technical University of Munich, Boltzmannstr. 3, 85748 Garching bei München, Germany **5** No Affiliation **6** Helmholtz Zentrum München, Ingolstädter Landstraße 1, 85764 Neuherberg, Germany ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 19 February 2024
Published: unpublished

Summary

Imaging is the process of transforming noisy, incomplete data into a space that humans can interpret. NIFTy is a Bayesian framework for imaging and has already successfully been applied to many fields in astrophysics. Previous design decisions held the performance and the development of methods in NIFTy back. We present a rewrite of NIFTy, coined NIFTy.re, which reworks the modeling principle, extends the inference strategies, and outsources much of the heavy lifting to JAX. The rewrite dramatically accelerates models written in NIFTy, lays the foundation for new types of inference machineries, improves maintainability, and enables interoperability between NIFTy and the JAX machine learning ecosystem.

Statement of Need

Imaging commonly involves millions to billions of pixels. Each pixel usually corresponds to one or more correlated degrees of freedom in the model space. Modeling this many degrees of freedom is computationally demanding. However, imaging is not only computationally demanding but also statistically challenging. The noise in the data requires a statistical treatment and needs to be accurately propagated from the data to the uncertainties in the final image. To do this, we require an inference machinery that not only handles extremely high-dimensional spaces, but one that does so in a statistically rigorous way.

NIFTy is a Bayesian imaging library (Arras et al., 2019; Selig et al., 2013; Steininger et al., 2019). It is designed to infer the million- to billion-dimensional posterior distribution in the image space from noisy input data. At the core of NIFTy lies a set of powerful Gaussian Process (GP) models and accurate Variational Inference (VI) algorithms.

NIFTy.re is a rewrite of NIFTy in JAX (Bradbury et al., 2018) with all relevant previous GP models, new, more flexible GP models, and a more flexible machinery for approximating posterior distributions. Being written in JAX, NIFTy.re effortlessly runs on accelerator hardware such as the GPU and TPU, vectorizes models whenever possible, and just-in-time compiles code for additional performance. NIFTy.re switches from a home-grown automatic differentiation engine that was used in NIFTy to JAX's automatic differentiation engine. This lays the foundation for new types of inference machineries that make use of the higher order derivatives provided by JAX. Through these changes, we envision to harness significant gains in maintainability of NIFTy.re compared to NIFTy and a faster development cycle for new features.

We expect NIFTy.re to be highly useful for many imaging applications and envision many applications within and outside of astrophysics (Arras et al., 2019, 2022; Eberle et al., 2022, 2023; Frank et al., 2017; S. Hutschenreuter et al., 2022; Sebastian Hutschenreuter et al., 2023; Leike et al., 2020; Leike & Enßlin, 2019; Mertsch & Phan, 2023; J. Roth et al., 2023; Jakob Roth et al., 2023; Scheel-Platz et al., 2023; Tsouros et al., 2024; Welling et al., 2021; Westerkamp et al., 2023). NIFTy.re has already been successfully used in two galactic tomography publications (Edenhofer et al., 2023; Leike et al., 2022). A very early version of NIFTy.re enabled a 100-billion-dimensional reconstruction using a maximum posterior inference. In a newer publication, NIFTy.re was used to infer a 500-million-dimensional posterior distribution using VI (Knollmüller & Enßlin, 2019). The latter publication extensively used NIFTy.re's GPU support to reduce the runtime by two orders of magnitude compared to the CPU. With NIFTy.re bridging ideas from NIFTy to JAX, we envision many new possibilities for inferring classical machine learning models with NIFTy's inference methods and a plethora of opportunities to use NIFTy-components such as the GP models in classical neural network frameworks.

NIFTy.re competes with other GP libraries as well as with probabilistic programming languages and frameworks. Compared to GPyTorch (Hensman et al., 2015), GPflow (De G. Matthews et al., 2017), george (Ambikasaran et al., 2015), or TinyGP (Foreman-Mackey et al., 2024), NIFTy.re focuses on GP models for structured spaces and does not assume the posterior to be analytically accessible. Instead, NIFTy.re tries to approximate the true posterior using VI. Compared to classical probabilistic programming languages such as Stan (Carpenter et al., 2017) and frameworks such Pyro (Bingham et al., 2019), NumPyro (Phan et al., 2019), pyMC3 (Salvatier et al., 2016), emcee (Foreman-Mackey et al., 2013), dynesty (Koposov et al., 2023; Speagle, 2020), or BlackJAX (Cabezas & Louf, 2023), NIFTy.re focuses on inference in extremely high-dimensional spaces. NIFTy.re exploits the structure of probabilistic models in its VI techniques (Frank et al., 2021). With NIFTy.re, the GP models and the VI machinery are now fully accessible in the JAX ecosystem and NIFTy.re components interact seamlessly with other JAX packages such as BlackJAX and JAXopt/Optax (Blondel et al., 2021; DeepMind et al., 2020).

Core Components

NIFTy.re brings tried and tested structured GP models and VI algorithms to JAX. GP models are highly useful for imaging problems, and VI algorithms are essential to probing high-dimensional posteriors, which are often encountered in imaging problems. NIFTy.re infers the parameters of interest from noisy data via a stochastic mapping that goes in the opposite direction, from the parameters of interest to the data.

NIFTy and NIFTy.re build up hierarchical models for the posterior. The log-posterior function reads $\ln p(\theta|d) := l(d, f(\theta)) + \ln p(\theta) + \mathrm{const}$ with log-likelihood l, forward model f mapping the parameters of interest θ to the data space, and log-prior $\ln p(\theta)$. The goal of the inference is to draw samples from the posterior $p(\theta|d)$.

What is considered part of the likelihood versus part of the prior is ill-defined. Without loss of generality, NIFTy and NIFTy.re re-formulate models such that the prior is always standard Gaussian. They implicitly define a mapping from a new latent space with a priori standard Gaussian parameters ξ to the parameters of interest θ. The mapping $\theta(\xi)$ is incorporated into the forward model $f(\theta(\xi))$ in such a way that all relevant details of the prior model are encoded in the forward model. This choice of re-parameterization (Rezende & Mohamed, 2015) is called standardization. It is often carried out implicitly in the background without user input.

Gaussian Processes

One standard tool from the `NIFTy.re` toolbox is the so-called correlated field GP model from `NIFTy`. This model relies on the harmonic domain being easily accessible. For example, for pixels spaced on a regular Cartesian grid, the natural choice to represent a stationary kernel is the Fourier domain. In the generative picture, a realization s drawn from a GP then reads $s = \mathcal{FT} \circ \sqrt{P} \circ \xi$ with $\mathcal{FT}$ the (fast) Fourier transform, $\sqrt{P}$ the square-root of the power-spectrum in harmonic space, and ξ standard Gaussian random variables. In the implementation in `NIFTy.re` and `NIFTy`, the user can choose between two adaptive kernel models, a non-parametric kernel $\sqrt{P}$ and a Matérn kernel $\sqrt{P}$ (Arras et al., 2022; Guardiani, 2022 for details on their implementation). A code example that initializes a non-parametric GP prior for a 128×128 space with unit volume is shown in the following.

```python
from nifty8 import re as jft

dims = (128, 128)
cfm = jft.CorrelatedFieldMaker("cf")
cfm.set_amplitude_total_offset(offset_mean=2, offset_std=(1e-1, 3e-2))
# Parameters for the kernel and the regular 2D Cartesian grid for which it is
# defined
cfm.add_fluctuations(
  dims,
  distances=tuple(1.0 / d for d in dims),
  fluctuations=(1.0, 5e-1),
  loglogavgslope=(-3.0, 2e-1),
  flexibility=(1e0, 2e-1),
  asperity=(5e-1, 5e-2),
  prefix="ax1",
  non_parametric_kind="power",
)
# Get the forward model for the GP prior
correlated_field = cfm.finalize()
```

Not all problems are well described by regularly spaced pixels. For more complicated pixel spacings, `NIFTy.re` features Iterative Charted Refinement (Edenhofer et al., 2022), a GP model for arbitrarily deformed spaces. This model exploits nearest neighbor relations on various coarsenings of the discretized modeled space and runs very efficiently on GPUs. For one-dimensional problems with arbitrarily spaced pixels, `NIFTy.re` also implements multiple flavors of Gauss-Markov processes.

Building Up Complex Models

Models are rarely just a GP prior. Commonly, a model contains at least a few non-linearities that transform the GP prior or combine it with other random variables. For building more complex models, `NIFTy.re` provides a `Model` class that offers a somewhat familiar object-oriented design yet is fully JAX compatible and functional under the hood. The following code shows how to build a slightly more complex model using the objects from the previous example.

```python
from jax import numpy as jnp


class Forward(jft.Model):
  def __init__(self, correlated_field):
    self._cf = correlated_field
    # Tracks a callable with which the model can be initialized. This is not
    # strictly required, but comes in handy when building deep models. Note, the
    # init method (short for "initialization" method) is not to be confused with
```

```python
        # the prior, which is always standard Gaussian.
        super().__init__(init=correlated_field.init)

    def __call__(self, x):
        # NOTE, any kind of masking of the output, non-linear and linear
        # transformation could be carried out here. Models can also be combined and
        # nested in any way and form.
        return jnp.exp(self._cf(x))


forward = Forward(correlated_field)

data = jnp.load("data.npy")
lh = jft.Poissonian(data).amend(forward)
```

All GP models in `NIFTy.re` as well as all likelihoods behave like instances of `jft.Model`, meaning that JAX understands what it means if a computation involves `self`, other `jft.Model` instances, or their attributes. In other words, `correlated_field`, `forward`, and `lh` from the code snippets shown here are all so-called pytrees in JAX, and, for example, the following is valid code `jax.jit(lambda l, x: l(x))(lh, x0)` with `x0` some arbitrarily chosen valid input to `lh`. Inspired by equinox (Kidger & Garcia, 2021), individual attributes of the class can be marked as non-static or static via `dataclass.field(metadata=dict(static=...))` for the purpose of compiling. Depending on the value, JAX will either treat the attribute as an unknown placeholder or as a known concrete attribute and potentially inline it during compilation. This mechanism is extensively used in likelihoods to avoid inlining large constants such as the data and to avoid expensive re-compilations whenever possible.

Variational Inference

`NIFTy.re` is built for models with millions to billions of degrees of freedom. To probe the posterior efficiently and accurately, `NIFTy.re` relies on VI. Specifically, `NIFTy.re` implements Metric Gaussian Variational Inference (MGVI) and its successor geometric Variational Inference (geoVI) Frank (2022). At the core of both MGVI and geoVI lies an alternating procedure in which one switches between optimizing the Kullback–Leibler divergence for a specific shape of the variational posterior and updating the shape of the variational posterior. MGVI and geoVI define the variational posterior via samples, specifically, via samples drawn around an expansion point. The samples in MGVI and geoVI exploit model-intrinsic knowledge of the posterior's approximate shape, encoded in the Fisher information metric and the prior curvature (Frank et al., 2021).

`NIFTy.re` implements both MGVI and geoVI and allows for much finer control over the way samples are drawn and updated compared to `NIFTy`. Furthermore, `NIFTy.re` exposes stand-alone functions for drawing MGVI and geoVI samples from any arbitrary model with a likelihood from `NIFTy.re` and a forward model that is differentiable by JAX. In addition to stand-alone sampling functions, `NIFTy.re` also provides tools to configure and execute the alternating Kullback–Leibler divergence optimization and sample adaption at a lower abstraction level. These tools are provided in a JAXopt/Optax-style optimizer class (Blondel et al., 2021; DeepMind et al., 2020).

A typical minimization with `NIFTy.re` is shown in the following. It retrieves six independent, antithetically mirrored samples from the approximate posterior via 25 iterations of alternating between optimization and sample adaption. The final result is stored in the `samples` variable. A convenient one-shot wrapper for the code below is `jft.optimize_kl`. By virtue of all modeling tools in `NIFTy.re` being written in JAX, it is also possible to combine `NIFTy.re` tools with BlackJAX (Cabezas & Louf, 2023) or any other posterior sampler in the JAX ecosystem.

```python
from jax import random
```

```python
key = random.PRNGKey(42)
key, sk = random.split(key, 2)
# NIFTy is agnostic w.r.t. the type of inputs it gets as long as they support
# core arithmetic properties. Tell NIFTy to treat our parameter dictionary as a
# vector.
samples = jft.Samples(pos=jft.Vector(lh.init(sk)), samples=None, keys=None)

delta = 1e-4
absdelta = delta * jft.size(samples.pos)

opt_vi = jft.OptimizeVI(lh, n_total_iterations=25)
opt_vi_st = opt_vi.init_state(
  key,
  # Implicit definition for the accuracy of the KL-divergence approximation;
  # typically on the order of 2-12
  n_samples=lambda i: 1 if i < 2 else (2 if i < 4 else 6),
  # Parametrize the conjugate gradient method at the heart of the sample-drawing
  draw_linear_kwargs=dict(
    cg_name="SL", cg_kwargs=dict(absdelta=absdelta / 10.0, maxiter=100)
  ),
  # Parametrize the minimizer used in the nonlinear update of the samples
  nonlinearly_update_kwargs=dict(
    minimize_kwargs=dict(
      name="SN", xtol=delta, cg_kwargs=dict(name=None), maxiter=5
    )
  ),
  # Parametrize the minimization of the KL-divergence cost potential
  kl_kwargs=dict(minimize_kwargs=dict(name="M", xtol=delta, maxiter=35)),
  sample_mode=lambda i: "nonlinear_resample" if i < 3 else "nonlinear_update",
)
for i in range(opt_vi.n_total_iterations):
  print(f"Iteration {i+1:04d}")
  # Continuously updates the samples of the approximate posterior distribution
  samples, opt_vi_st = opt_vi.update(samples, opt_vi_st)
  print(opt_vi.get_status_message(samples, opt_vi_st))
```


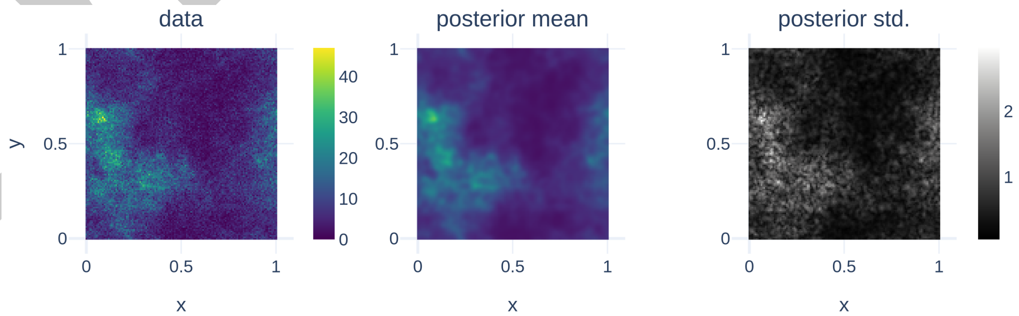

Figure 1: Data (left), posterior mean (middle), and posterior uncertainty (right) for a simple toy example.

Figure 1 shows an exemplary posterior reconstruction employing the above model. The posterior mean agrees with the data but removes noisy structures. The posterior standard deviation is approximately equal to typical differences between the posterior mean and the data.

Performance of `NIFTy.re` compared to `NIFTy`

We test the performance of `NIFTy.re` against `NIFTy` for the simple yet representative model from above. To assess the performance, we compare the time required to apply $M_p := F_p + \mathbb{1}$ to random input with F_p denoting the Fisher metric of the overall likelihood at position p and $\mathbb{1}$ the identity matrix. Within `NIFTy.re`, the Fisher metric of the overall likelihood is decomposed into $J_{f,p}^{\dagger} N^{-1} J_{f,p}$ with $J_{f,p}$ the implicit Jacobian of the forward model f at p and N^{-1} the Fisher-metric of the Poisson likelihood. We choose to benchmark M_p as a typical VI minimization in `NIFTy.re` and `NIFTy` is dominated by calls to this function.

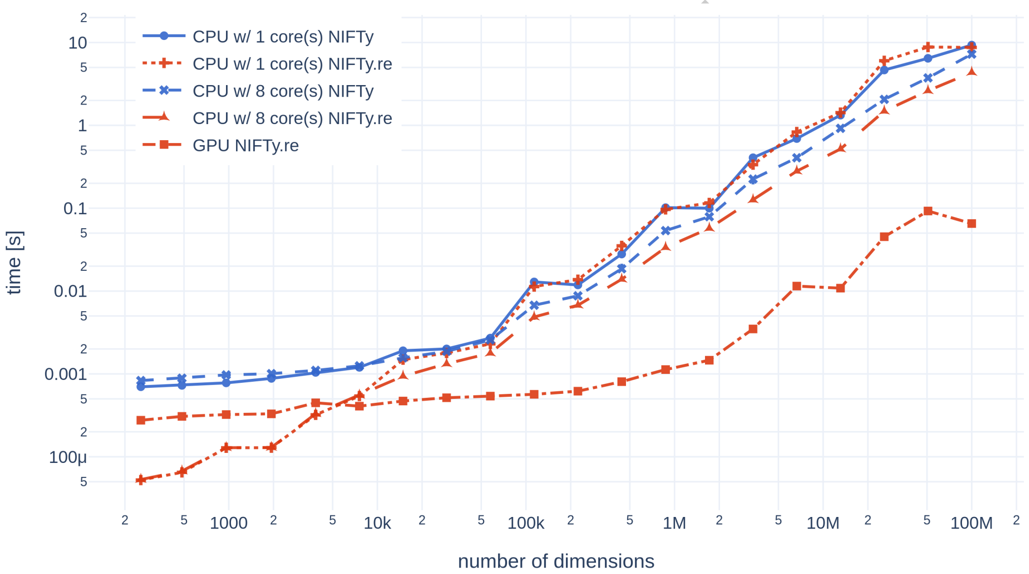

Figure 2: Median evaluation time of applying the Fisher metric plus the identity metric to random input for `NIFTy.re` and `NIFTy` on the CPU (one and eight core(s) of an Intel Xeon Platinum 8358 CPU clocked at 2.60G Hz) and the GPU (A100 SXM4 80 GB HBM2). The quantile range from the 16%- to the 84%-quantile is obscured by the marker symbols.

Figure 2 shows the median evaluation time in `NIFTy` of applying M_p to a new, random tangent position and the evaluation time in `NIFTy.re` of building M_p and applying it to a new, random tangent position for exponentially larger models. The 16%-quantiles and the 84%-quantiles of the timings are obscured by the marker symbols. We choose to exclude the build time of M_p in `NIFTy` from the comparison, putting `NIFTy` at an advantage, as its automatic differentiation is built around calls to M_p with p rarely varying. We ran the benchmark on one CPU core, eight CPU cores, and on a GPU on a compute-node with an Intel Xeon Platinum 8358 CPU clocked at 2.60G Hz and an NVIDIA A100 SXM4 80 GB HBM2 GPU. The benchmark used `jax==0.4.23` and `jaxlib==0.4.23+cuda12.cudnn89`. We vary the size of the model by increasing the size of the two-dimensional square image grid.

For small image sizes, `NIFTy.re` on the CPU is about one order of magnitude faster than `NIFTy`. Both reach about the same performance at an image size of roughly 15,000 pixels and continue to perform roughly the same for larger image sizes. The performance increases by a factor of three to four with eight cores for `NIFTy.re` and `NIFTy`, although `NIFTy.re` is slightly better at using the additional cores. On the GPU, `NIFTy.re` is consistently about one to two orders of magnitude faster than `NIFTy` for images larger than 100,000 pixels.

We believe the performance benefits of `NIFTy.re` on the CPU for small models stem from the reduced python overhead by just-in-time compiling computations. At image sizes larger than roughly 15,000 pixels, both evaluation times are dominated by the fast Fourier transform and are hence roughly the same as both use the same underlying implementation (Reinecke, 2024). Models in `NIFTy.re` and `NIFTy` are often well aligned with GPU programming models and thus consistently perform well on the GPU. Modeling components such as the new GP models implemented in `NIFTy.re` are even better aligned with GPU programming paradigms and yield even higher performance gains (Edenhofer et al., 2022).

Conclusion

We implemented the core GP and VI machinery of the Bayesian imaging package `NIFTy` in JAX. The rewrite moves much of the heavy-lifting from home-grown solutions to JAX, and we envision significant gains in maintainability of `NIFTy.re` and a faster development cycle moving forward. The rewrite accelerates typical models written in `NIFTy` by one to two orders of magnitude, lays the foundation for new types of inference machineries by enabling higher order derivatives via JAX, and enables the interoperability of `NIFTy`'s VI and GP methods with the JAX machine learning ecosystem.

Acknowledgements

Gordian Edenhofer acknowledges support from the German Academic Scholarship Foundation in the form of a PhD scholarship ("Promotionsstipendium der Studienstiftung des Deutschen Volkes"). Philipp Frank acknowledges funding through the German Federal Ministry of Education and Research for the project "ErUM-IFT: Informationsfeldtheorie für Experimente an Großforschungsanlagen" (Förderkennzeichen: 05D23EO1). Jakob Roth acknowledges financial support by the German Federal Ministry of Education and Research (BMBF) under grant 05A20W01 (Verbundprojekt D-MeerKAT). Matteo Guardiani, Vincent Eberle, and Margret Westerkamp acknowledge financial support from the "Deutsches Zentrum für Luft- und Raumfahrt e.V." (DLR) through the project Universal Bayesian Imaging Kit (UBIK, Förderkennzeichen 50OO2103). Lukas Scheel-Platz acknowledges funding from the European Research Council (ERC) under the European Union's Horizon Europe research and innovation programme under grant agreement No 101041936 (EchoLux).

References

Ambikasaran, S., Foreman-Mackey, D., Greengard, L., Hogg, D. W., & O'Neil, M. (2015). Fast Direct Methods for Gaussian Processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *38*, 252. https://doi.org/10.1109/TPAMI.2015.2448083

Arras, P., Frank, P., Haim, P., Knollmüller, J., Leike, R. H., Reinecke, M., & Enßlin, T. A. (2022). Variable structures in M87* from space, time and frequency resolved interferometry. *Nature Astronomy*, *6*, 259–269. https://doi.org/10.1038/s41550-021-01548-0

Arras, P., Frank, P., Leike, R., Westermann, R., & Enßlin, T. A. (2019). Unified radio interferometric calibration and imaging with joint uncertainty quantification. *Astronomy & Astrophysics*, *627*, A134. https://doi.org/10.1051/0004-6361/201935555

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P. A., Horsfall, P., & Goodman, N. D. (2019). Pyro: Deep universal probabilistic programming. *Journal of Machine Learning Research*, *20*, 28:1–28:6. http://jmlr.org/papers/v20/18-403.html

Blondel, M., Berthet, Q., Cuturi, M., Frostig, R., Hoyer, S., Llinares-López, F., Pedregosa, F., & Vert, J.-P. (2021). Efficient and modular implicit differentiation. *arXiv Preprint arXiv:2105.15183*.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). *JAX: Composable* *transformations of Python+NumPy programs* (Version 0.3.13). http://github.com/google/jax

Cabezas, L., Alberto, & Louf, R. (2023). *Blackjax: A sampling library for JAX* (Version v1.1.0). http://github.com/blackjax-devs/blackjax

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., & Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of Statistical Software*, *76*(1), 1–32. https://doi.org/10.18637/jss.v076.i01

De G. Matthews, A. G., Van Der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., León-Villagrá, P., Ghahramani, Z., & Hensman, J. (2017). GPflow: A gaussian process library using tensorflow. *Journal of Machine Learning Research*, *18*(1), 1299–1304.

DeepMind, Babuschkin, I., Baumli, K., Bell, A., Bhupatiraju, S., Bruce, J., Buchlovsky, P., Budden, D., Cai, T., Clark, A., Danihelka, I., Dedieu, A., Fantacci, C., Godwin, J., Jones, C., Hemsley, R., Hennigan, T., Hessel, M., Hou, S., … Viola, F. (2020). *The DeepMind JAX Ecosystem*. http://github.com/google-deepmind

Eberle, V., Frank, P., Stadler, J., Streit, S., & Enßlin, T. (2022). Efficient representations of spatially variant point spread functions with butterfly transforms in bayesian imaging algorithms. *Physical Sciences Forum*, *5*(1). https://doi.org/10.3390/psf2022005033

Eberle, V., Frank, P., Stadler, J., Streit, S., & Enßlin, T. (2023). Butterfly Transforms for Efficient Representation of Spatially Variant Point Spread Functions in Bayesian Imaging. *Entropy*, *25*(4), 652. https://doi.org/10.3390/e25040652

Edenhofer, G., Leike, R. H., Frank, P., & Enßlin, T. A. (2022). *Sparse kernel gaussian processes through iterative charted refinement (ICR)*. arXiv. https://doi.org/10.48550/ARXIV.2206.10634

Edenhofer, G., Zucker, C., Frank, P., Saydjari, A. K., Speagle, J. S., Finkbeiner, D., & Enßlin, T. (2023). A Parsec-Scale Galactic 3D Dust Map out to 1.25 kpc from the Sun. *arXiv e-Prints*, arXiv:2308.01295. https://doi.org/10.48550/arXiv.2308.01295

Foreman-Mackey, D., Hogg, D. W., Lang, D., & Goodman, J. (2013). emcee: The MCMC Hammer. *Publications of the Astronomical Society of the Pacific*, *125*(925), 306. https://doi.org/10.1086/670067

Foreman-Mackey, D., Yu, W., Yadav, S., Becker, M. R., Caplar, N., Huppenkothen, D., Killestein, T., Tronsgaard, R., Rashid, T., & Schmerler, S. (2024). *dfm/tinygp: The tiniest of Gaussian Process libraries* (Version v0.3.0). Zenodo. https://doi.org/10.5281/zenodo.10463641

Frank, P. (2022). Geometric variational inference and its application to bayesian imaging. *Physical Sciences Forum*, *5*(1). https://doi.org/10.3390/psf2022005006

Frank, P., Leike, R. H., & Enßlin, T. A. (2021). Geometric variational inference. *Entropy*, *23*(7), 853. https://doi.org/10.3390/e23070853

Frank, P., Steininger, T., & Enßlin, T. A. (2017). Field dynamics inference via spectral density estimation. *Physical Review E*, *96*(5), 052104. https://doi.org/10.1103/PhysRevE.96.052104

Guardiani, P. A. K., Matteo AND Frank. (2022). Causal, bayesian, & non-parametric modeling of the SARS-CoV-2 viral load distribution vs. Patient's age. *PLOS ONE*, *17*(10), 1–21. https://doi.org/10.1371/journal.pone.0275011

Hensman, J., G. Matthews, A. G. de, & Ghahramani, Z. (2015). Scalable variational gaussian process classification. In G. Lebanon & S. V. N. Vishwanathan (Eds.), *AISTATS* (Vol. 38). JMLR.org. http://dblp.uni-trier.de/db/conf/aistats/aistats2015.html#HensmanMG15

Hutschenreuter, S., Anderson, C. S., Betti, S., Bower, G. C., Brown, J.-A., Brüggen, M., Carretti, E., Clarke, T., Clegg, A., Costa, A., Croft, S., Van Eck, C., Gaensler, B. M., de Gasperin, F., Haverkorn, M., Heald, G., Hull, C. L. H., Inoue, M., Johnston-Hollitt, M., … Enßlin, T. A. (2022). The Galactic Faraday rotation sky 2020. *Astronomy & Astrophysics*, *657*, A43. https://doi.org/10.1051/0004-6361/202140486

Hutschenreuter, Sebastian, Haverkorn, M., Frank, P., Raycheva, N. C., & Enßlin, T. A. (2023). Disentangling the Faraday rotation sky. *arXiv e-Prints*, arXiv:2304.12350. https://doi.org/10.48550/arXiv.2304.12350

Kidger, P., & Garcia, C. (2021). Equinox: Neural networks in JAX via callable PyTrees and filtered transformations. *Differentiable Programming Workshop at Neural Information Processing Systems 2021*.

Knollmüller, J., & Enßlin, T. A. (2019). *Metric gaussian variational inference*. arXiv. https://doi.org/10.48550/ARXIV.1901.11033

Koposov, S., Speagle, J., Barbary, K., Ashton, G., Bennett, E., Buchner, J., Scheffler, C., Cook, B., Talbot, C., Guillochon, J., Cubillos, P., Ramos, A. A., Johnson, B., Lang, D., Ilya, Dartiailh, M., Nitz, A., McCluskey, A., & Archibald, A. (2023). *Joshspeagle/dynesty: v2.1.3* (Version v2.1.3). Zenodo. https://doi.org/10.5281/zenodo.8408702

Leike, R. H., Edenhofer, G., Knollmüller, J., Alig, C., Frank, P., & Enßlin, T. A. (2022). The Galactic 3D large-scale dust distribution via Gaussian process regression on spherical coordinates. *arXiv e-Prints*, arXiv:2204.11715. https://doi.org/10.48550/arXiv.2204.11715

Leike, R. H., & Enßlin, T. A. (2019). Charting nearby dust clouds using Gaia data only. *Astronomy & Astrophysics*, *631*, A32. https://doi.org/10.1051/0004-6361/201935093

Leike, R. H., Glatzle, M., & Enßlin, T. A. (2020). Resolving nearby dust clouds. *Astronomy & Astrophysics*, *639*, A138. https://doi.org/10.1051/0004-6361/202038169

Mertsch, P., & Phan, V. H. M. (2023). Bayesian inference of three-dimensional gas maps. II. Galactic HI. *Astronomy & Astrophysics*, *671*, A54. https://doi.org/10.1051/0004-6361/202243326

Phan, D., Pradhan, N., & Jankowiak, M. (2019). Composable Effects for Flexible and Accelerated Probabilistic Programming in NumPyro. *arXiv e-Prints*, arXiv:1912.11554. https://doi.org/10.48550/arXiv.1912.11554

Reinecke, M. (2024). *DUCC: Distinctly useful code collection* (Version 0.33.0). https://gitlab.mpcdf.mpg.de/mtr/ducc

Rezende, D. J., & Mohamed, S. (2015). Variational inference with normalizing flows. *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, 1530–1538. http://proceedings.mlr.press/v37/rezende15.html

Roth, Jakob, Arras, P., Reinecke, M., Perley, R. A., Westermann, R., & Enßlin, T. A. (2023). Bayesian radio interferometric imaging with direction-dependent calibration. *Astronomy & Astrophysics*, *678*, A177. https://doi.org/10.1051/0004-6361/202346851

Roth, J., Li Causi, G., Testa, V., Arras, P., & Ensslin, T. A. (2023). Fast-cadence High-contrast Imaging with Information Field Theory. *The Astronomical Journal*, *165*(3), 86. https://doi.org/10.3847/1538-3881/acabc1

Salvatier, J., Wiecki, T. V., & Fonnesbeck, C. (2016). Probabilistic programming in python using PyMC3. *PeerJ Computer Science*, *2*, e55. https://doi.org/10.7717/peerj-cs.55

Scheel-Platz, L. I., Knollmüller, J., Arras, P., Frank, P., Reinecke, M., Jüstel, D., & Enßlin, T. A. (2023). Multicomponent imaging of the Fermi gamma-ray sky in the spatio-spectral domain. *Astronomy & Astrophysics*, *680*, A2. https://doi.org/10.1051/0004-6361/202243819

Selig, M., Bell, M. R., Junklewitz, H., Oppermann, N., Reinecke, M., Greiner, M., Pachajoa, C., & Ensslin, T. A. (2013). *NIFTY: A versatile Python library for signal inference* (p. ascl:1302.013). Astrophysics Source Code Library, record ascl:1302.013.

Speagle, J. S. (2020). DYNESTY: a dynamic nested sampling package for estimating Bayesian posteriors and evidences. *493*(3), 3132–3158. https://doi.org/10.1093/mnras/staa278

Steininger, T., Dixit, J., Frank, P., Greiner, M., Hutschenreuter, S., Knollmüller, J., Leike, R. H., Porqueres, N., Pumpe, D., Reinecke, M., Šraml, M., Varady, C., & Enßlin, T. A. (2019). NIFTy 3 - Numerical Information Field Theory: A Python Framework for Multicomponent Signal Inference on HPC Clusters. *Annalen Der Physik*, *531*(3), 1800290. https://doi.org/10.1002/andp.201800290

Tsouros, A., Edenhofer, G., Enßlin, T., Mastorakis, M., & Pavlidou, V. (2024). Reconstructing Galactic magnetic fields from local measurements for backtracking ultra-high-energy cosmic rays. *Astronomy & Astrophysics*, *681*, A111. https://doi.org/10.1051/0004-6361/202346423

Welling, C., Frank, P., Enßlin, T., & Nelles, A. (2021). Reconstructing non-repeating radio pulses with Information Field Theory. *Journal of Cosmology and Astroparticle Physics*, *2021*(4), 071. https://doi.org/10.1088/1475-7516/2021/04/071

Westerkamp, M., Eberle, V., Guardiani, M., Frank, P., Platz, L., Arras, P., Knollmüller, J., Stadler, J., & Enßlin, T. (2023). First spatio-spectral Bayesian imaging of SN1006 in X-ray. *arXiv e-Prints*, arXiv:2308.09176. https://doi.org/10.48550/arXiv.2308.09176