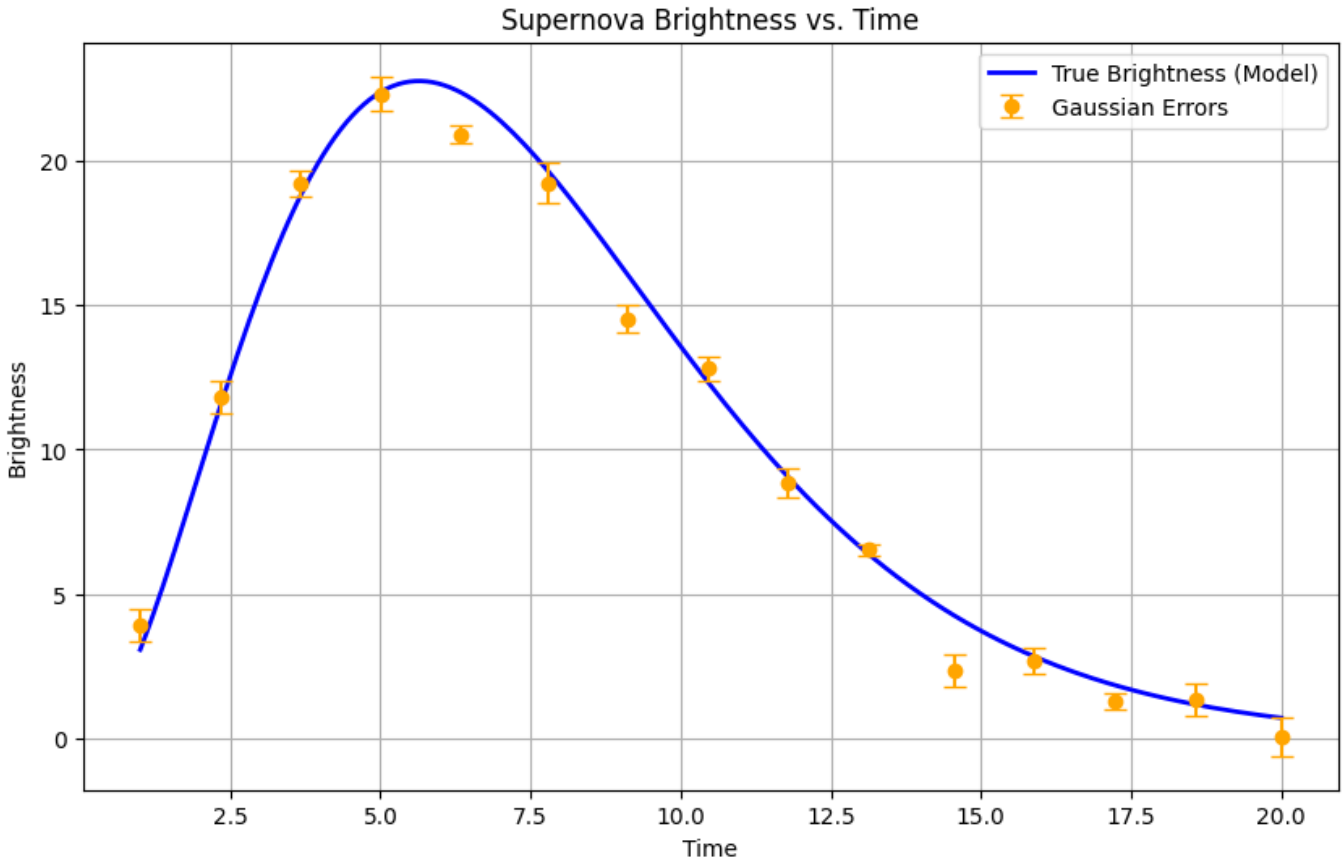
2025/02/01

Here we try using the vanilla HMC to fit a toy model for the supernova brightness curve.

$$ F(t;\vec{\theta}) = \frac{ \theta_0 t^3}{\text{exp}({\theta_1 t}) - 1} $$



From this I generated synthetic samples with heteroskedastic Gaussian errors.

The (unnormalised) likelihood is then given as

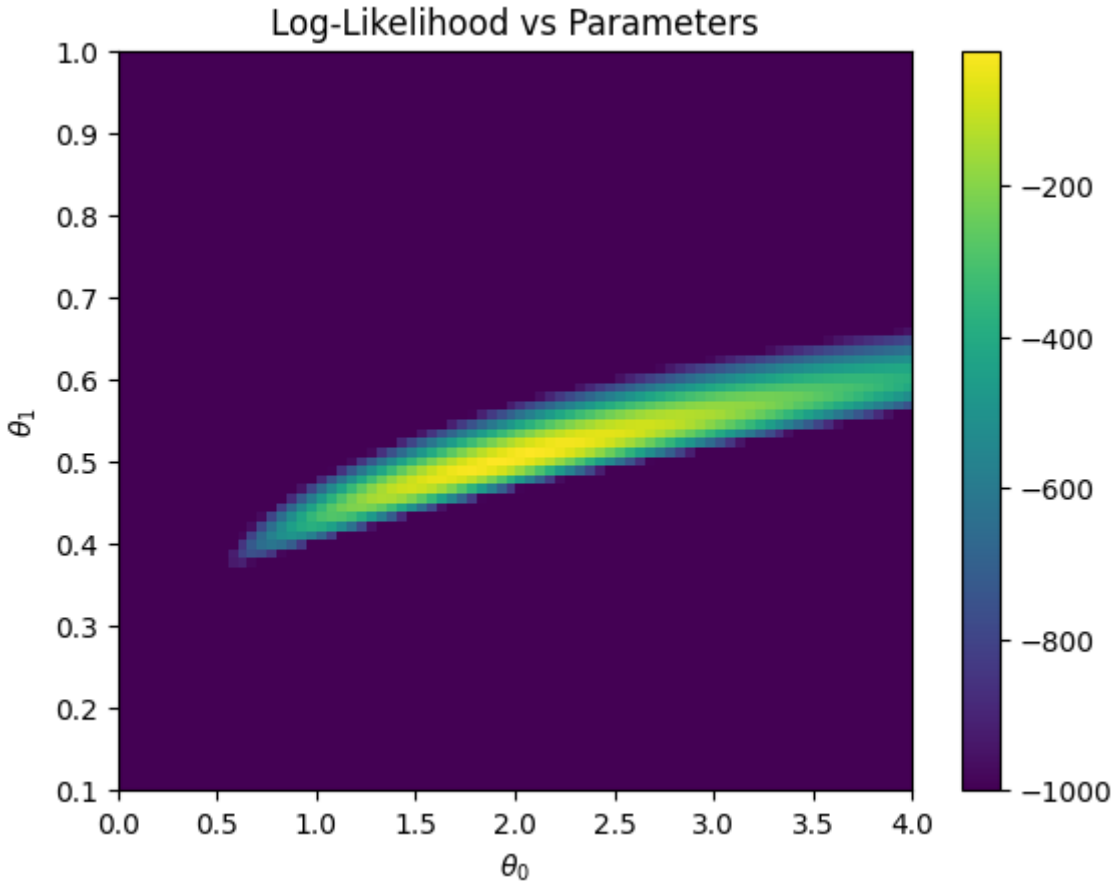
$$ P(\text{Data}|\vec{\theta}) = \Pi_t : \text{exp}(-\frac{(F_t(\vec{\theta}) - \text{Data}_t)^2}{2\sigma_t^2}) $$

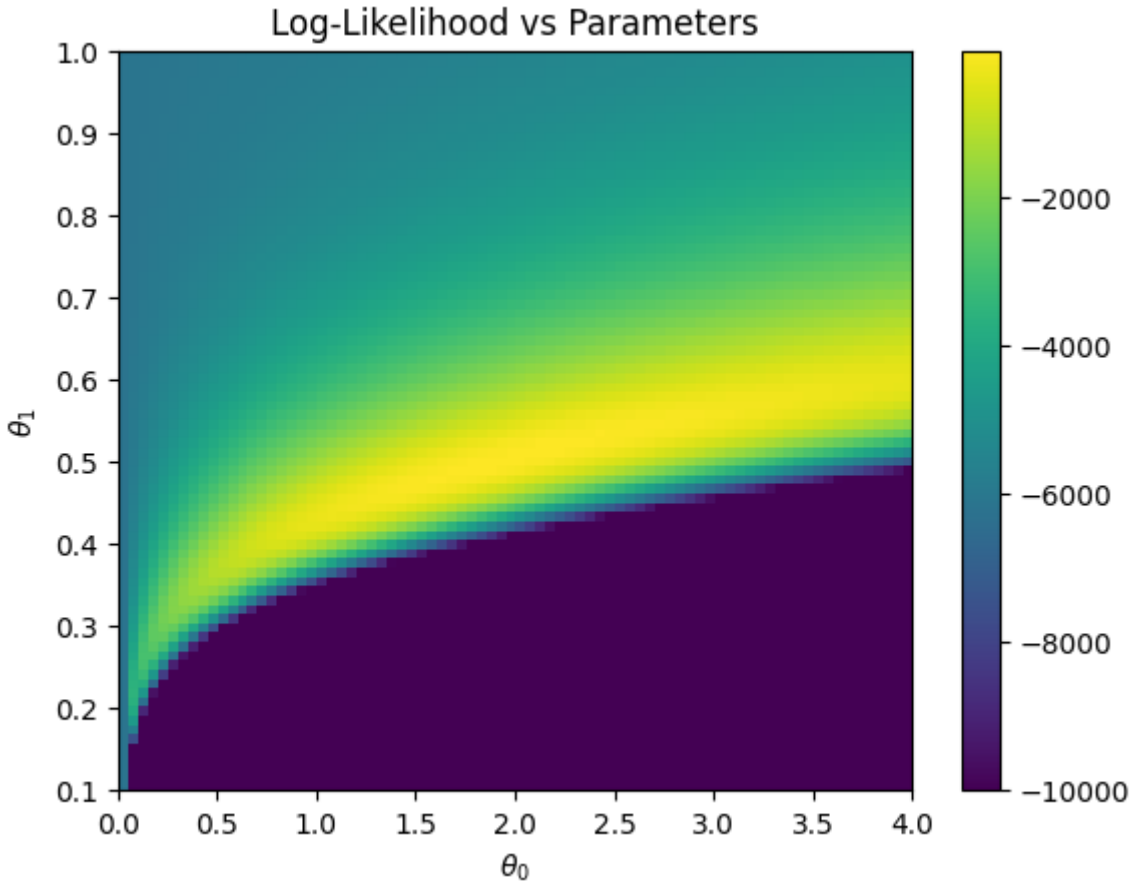
with log-likelihood

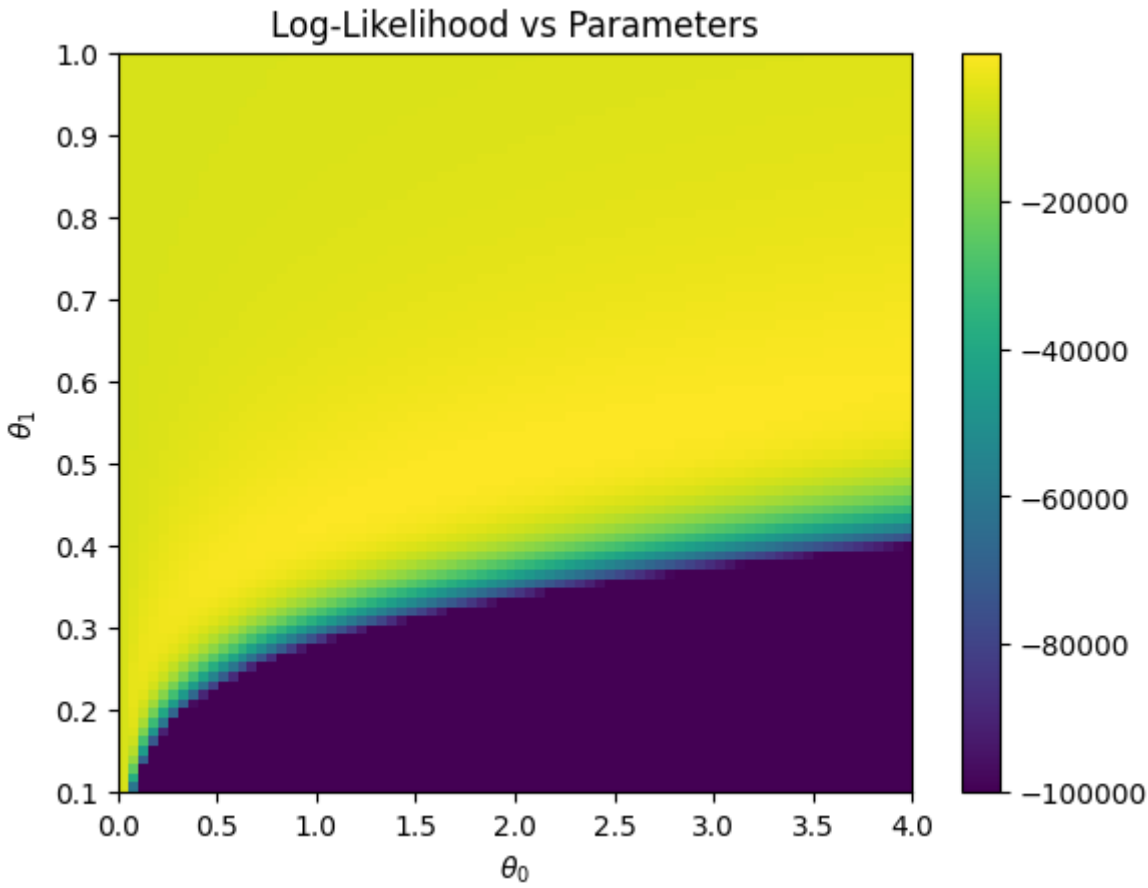
$$ \text{log}P(\text{Data}|\vec{\theta}) = \Sigma_t : -\frac{(F_t(\vec{\theta}) - \text{Data}_t)^2} {2\sigma_t^2} $$

Issue with limit at t=0 so we artificially exclude this

Nonetheless, it leads to an unpleasant probability distribution on theta. In these plots I limited the maximum modulus of the values to imrpove visability

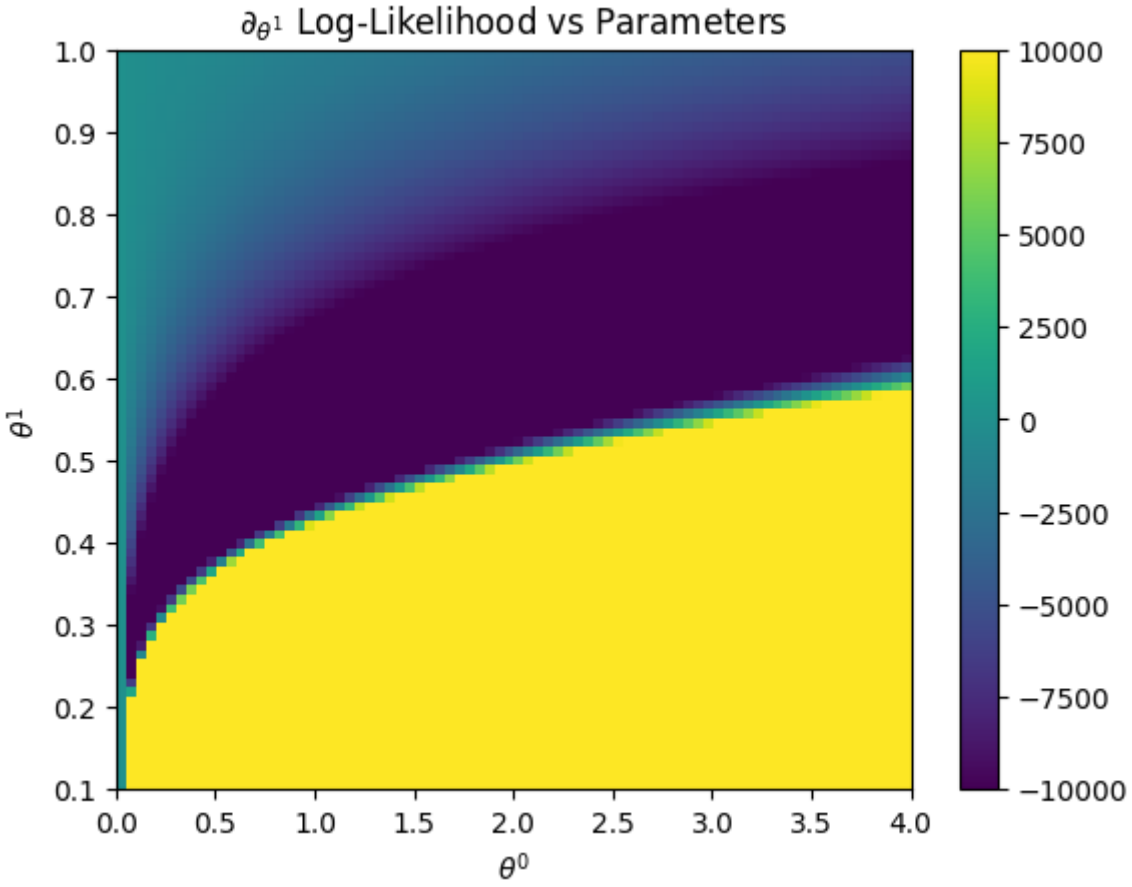

Log-Likelihood vs Parameters
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
θ_1
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
θ_0
−200
−400
−600
−800
−1000

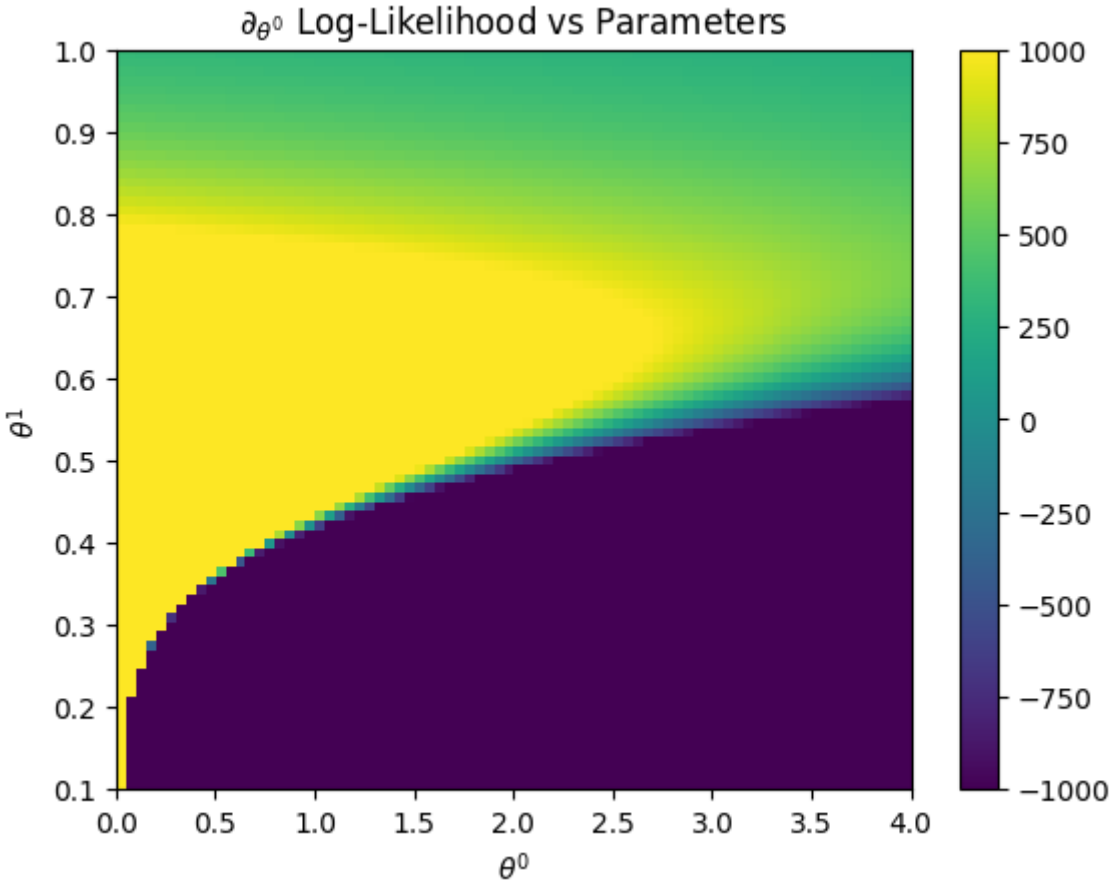

Log-Likelihood vs Parameters
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
θ_1
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
θ_0
−2000
−4000
−6000
−8000
−10000



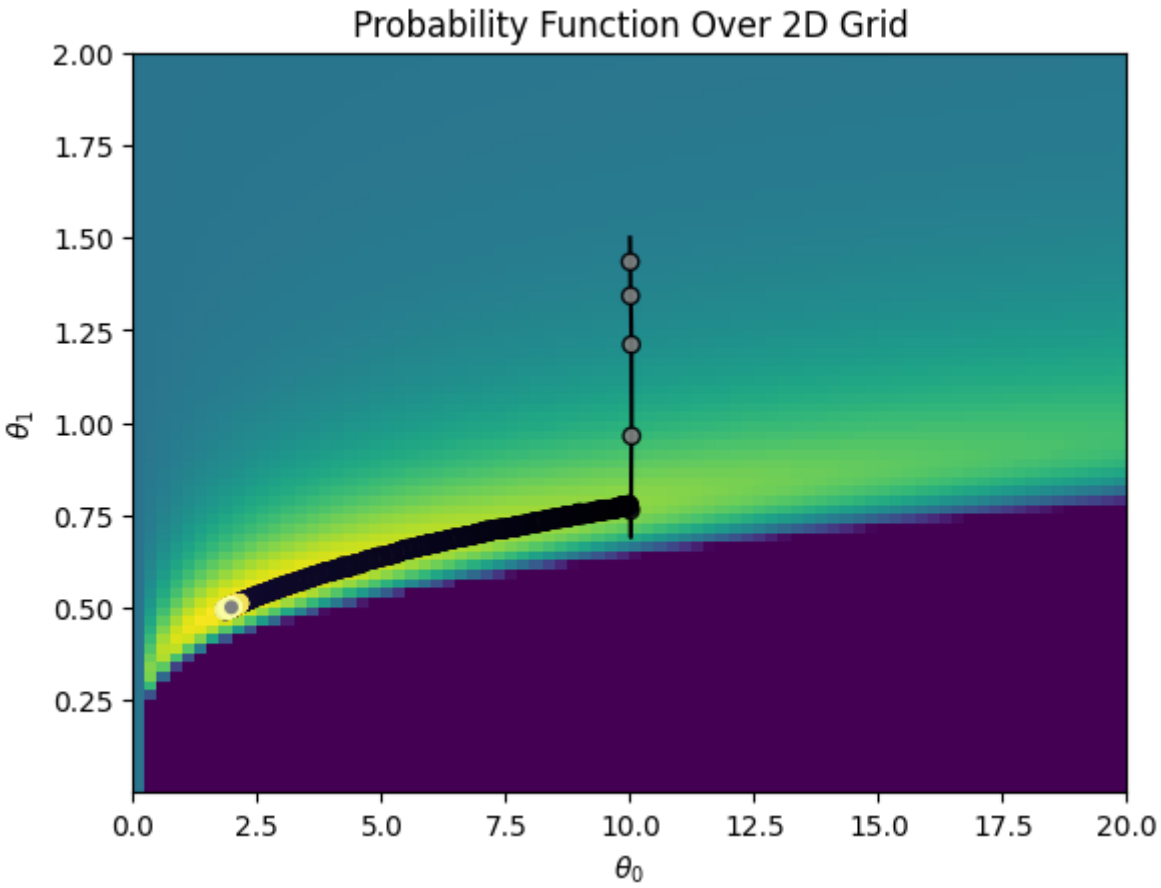
Note that the region where theta_1 is too high is relatively flat. This is because the model converges towards a flat line as theta_1 tends to infinity. The region where theta_2 is too low is much more sensitive as low theta causes the model to diverge.

Note that the gradient wrt theta_1 is an order of magnitude greater than for theta_2
Therefore we expect relatively vertical lines to be followed by HMC



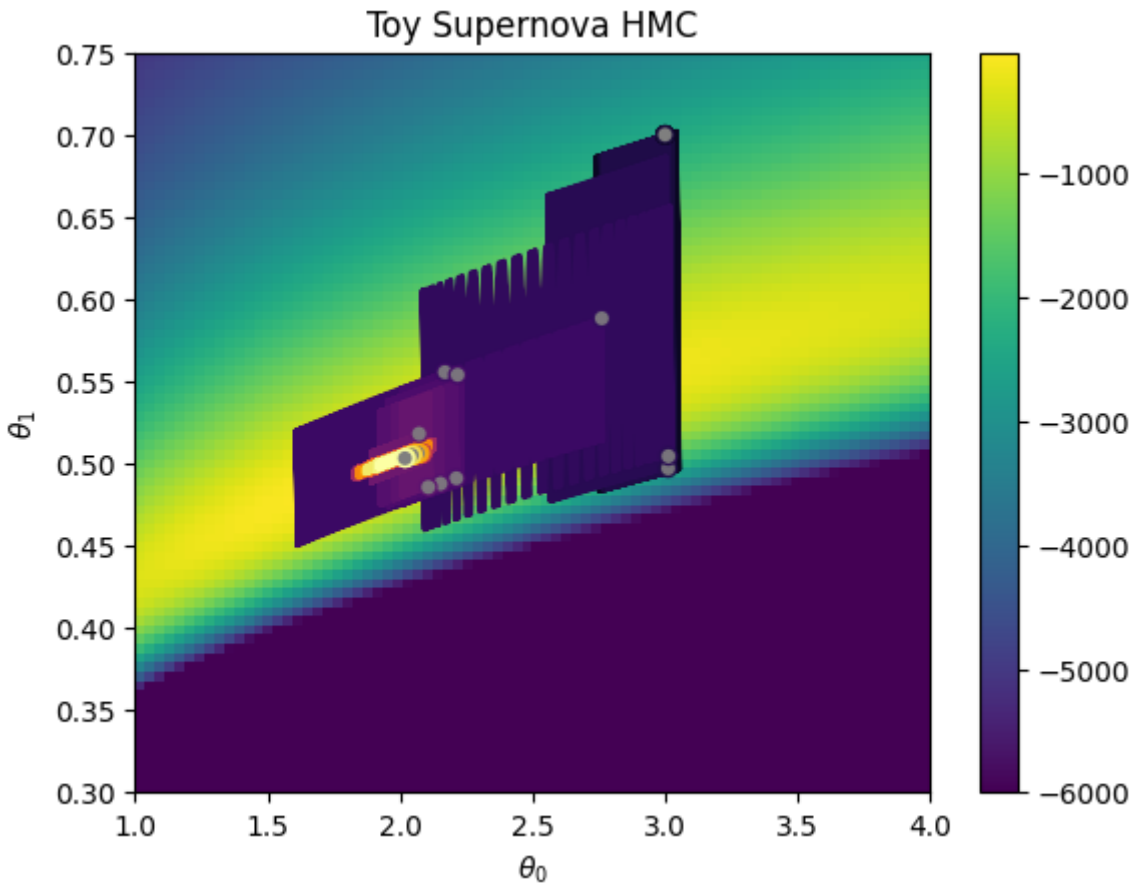


This gives us some very annoying burn-in:

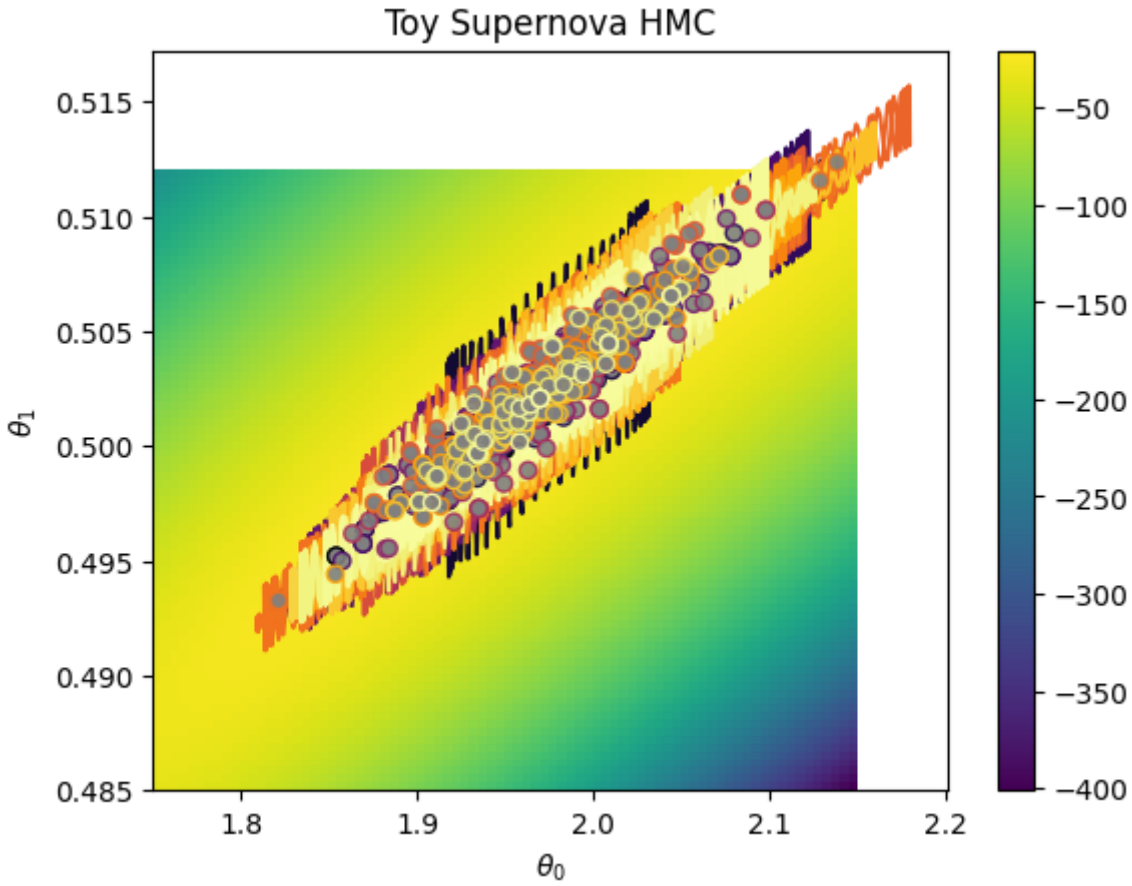


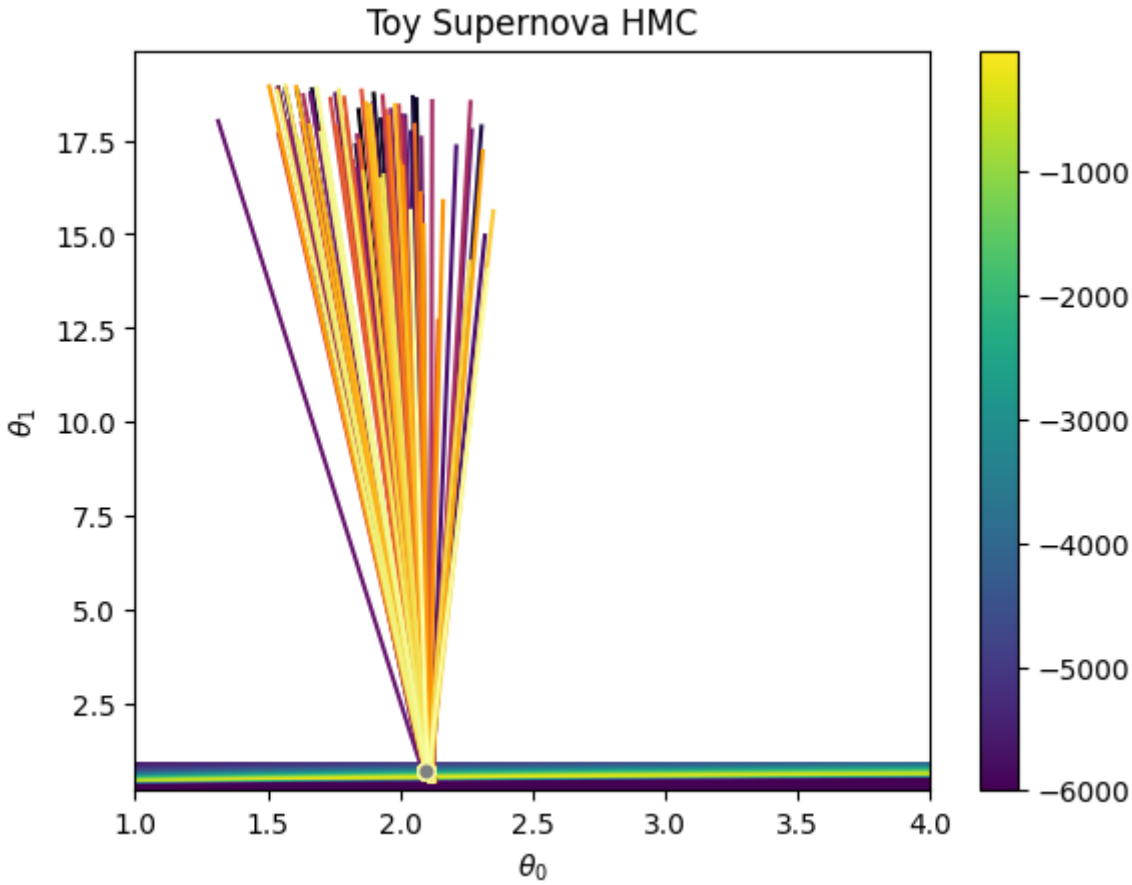
The much steeper region on the approach means we need to use small integration steps but once the chain has reached the ridge, this makes it very slow to approach the maximum. Perhaps we need to use adaptive step sizes.

We can also get lissajoux with dt = 0.0001 and L = 10000



One unfortunate feature of our toy supernova example is the very steep slope when theta1 is too low. We see that dt = 0.001 and L = 200 is a good choice of algorithm paramters when we start near the maximum. But if we start further away, this leads to a numerical overshoot and bad things happen. This could likely be fixed using the no-u-turn algorithm which incorperates a means to stop integration when the Hamiltonian is no longer conserved - and still maintain detailed balance. Another nice feature of the no-u-turn is that it samples uniformly along the integration path. In the case of the Lissajoux-like figures, this approximates sampling uniformly ocer that region.





To Do:

- [ ] Implement KS test

- [ ] Implement no-u-turn
- [ ] Possibly implement adaptive hyper-parameters but this won't solve the issue that
- [ ] Possibly
- [ ] Prove that non-linear reflections aren't volume-preserving
    - [ ] Demonstarting how the Gaussian momentum selection interacts with the Hamiltonian motion might be a good first step
    - [ ] Possibly find a way to compensate for this

2025/02/05

Empirical results suggest that non-linear reflections do in fact warp the probability distribution.

Comparing a 2D normal (var=1, cov=0) with samples outside the mask thrown away vs a 2D normal implementing bouncing off the mask yields very different distributions.

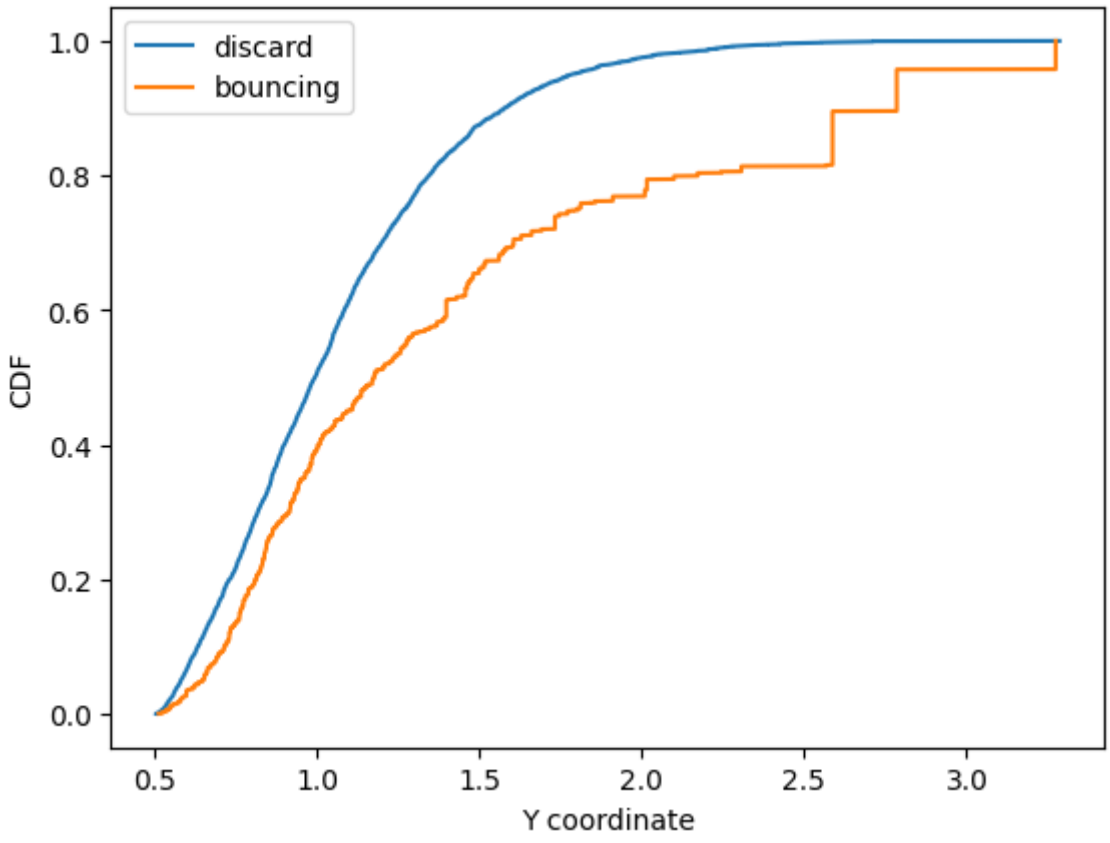
Mask used was y = 1/2 x**2

The simple case of throwing away samples outside the mask yields average 1.060498, variance 0.1445

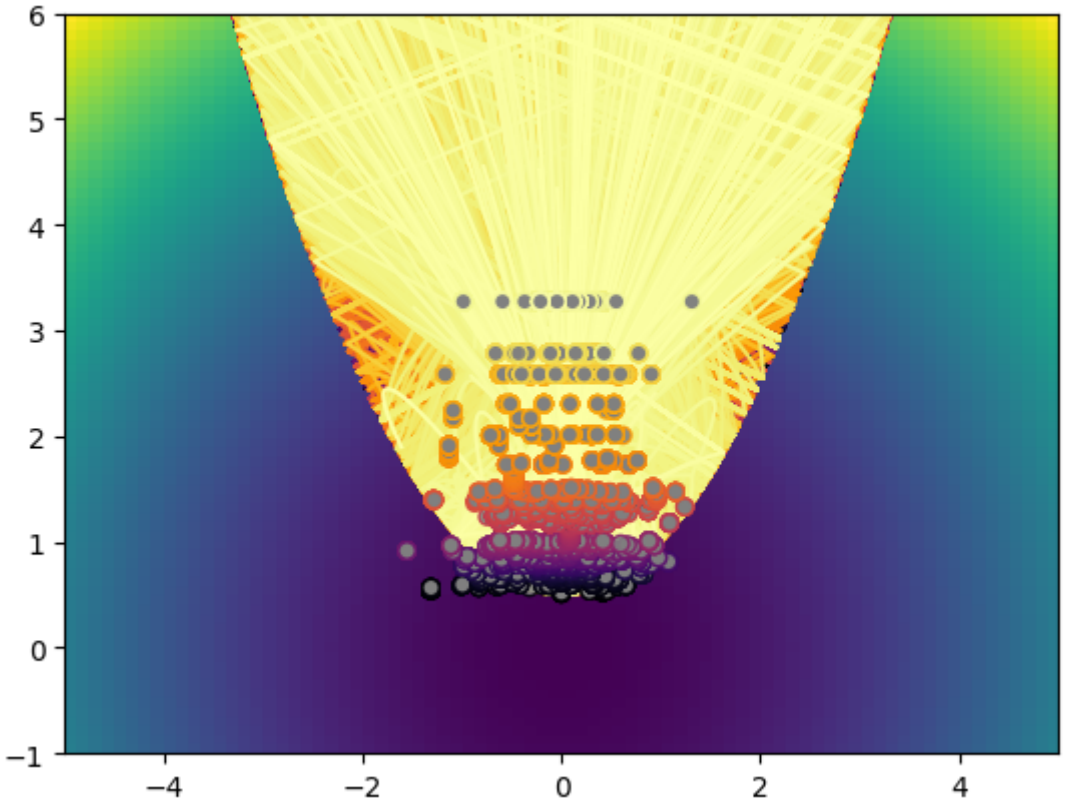
Bouncing HMC yields average 1.444963, variance 0.5756

The two-sample Kolmogorov-Smirnov test yields a p value of 1E-215 (KS statistic 0.2409)

However, when I plot the two CDFs, the boucning one has a very poor distribution:

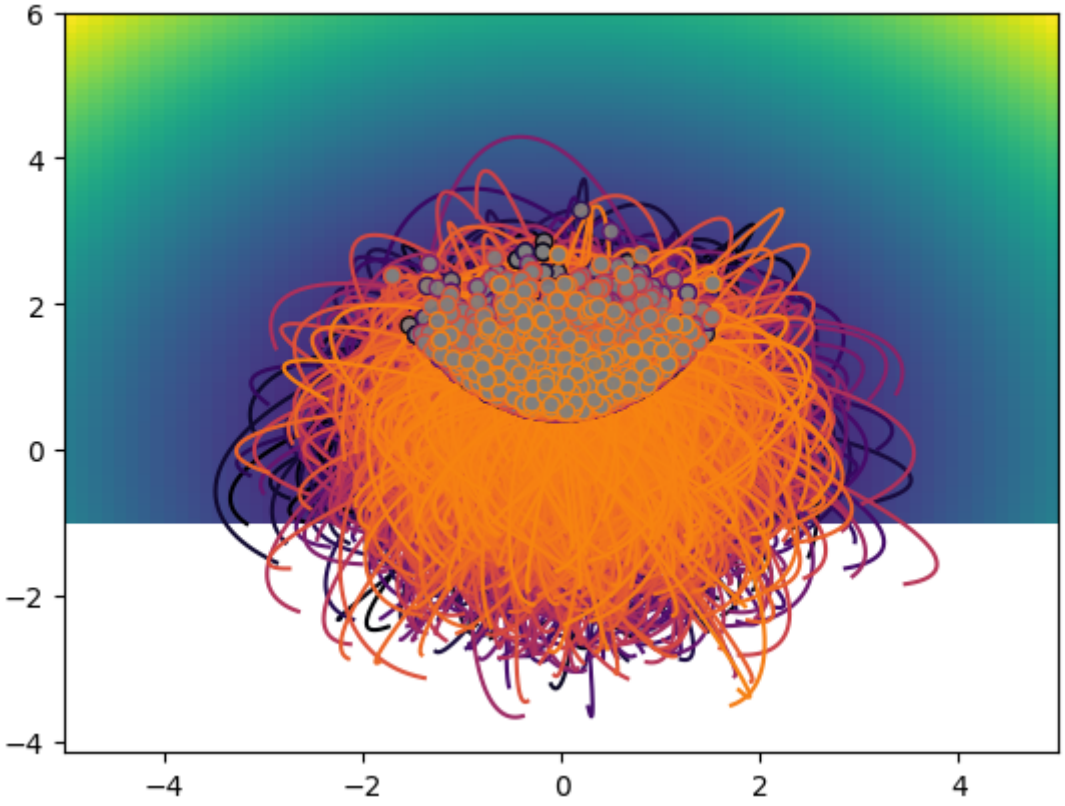


Plotting the samples from the bouncing HMC run tells us why...



This tells us that the chain progressively drifts away from the centre. Weirdly it seems to spend some time at discrete vertical positions - we should try and explain this behaviour theoretically. It might be some nasty interaction between the mask geometry and the underlying probability distribution. It is especially conerning that we appear to have collected some samples outside the masked region.

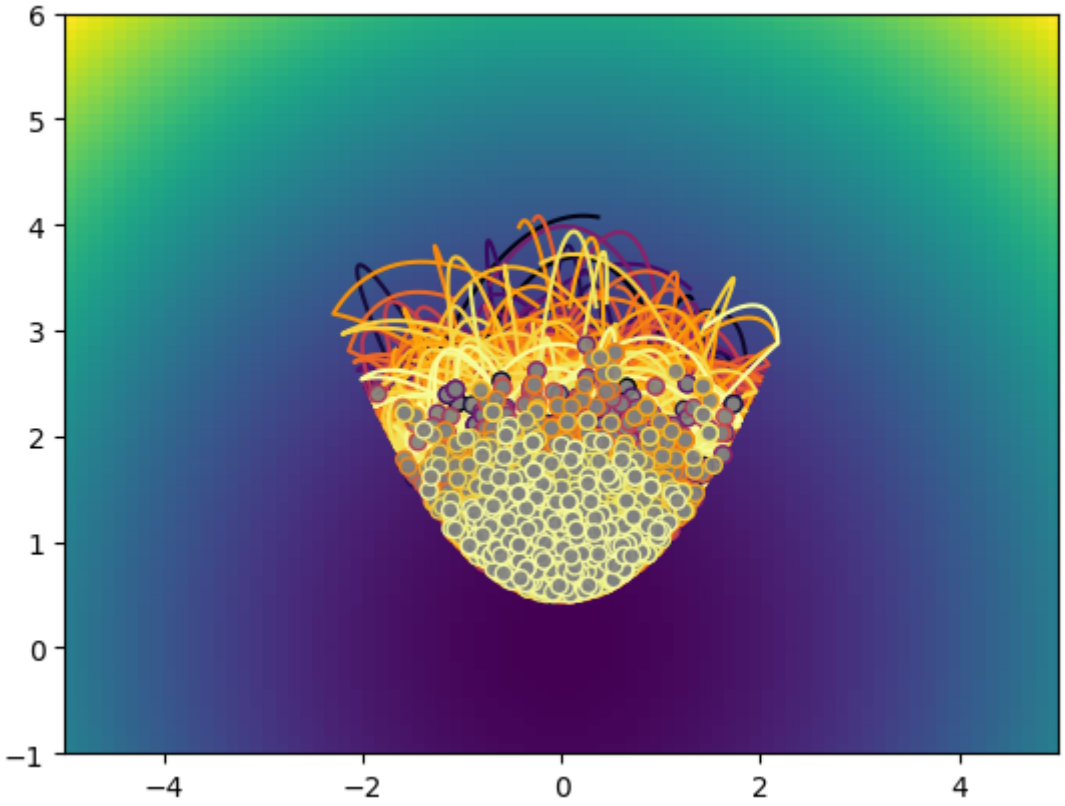
Here's what it looks like when we sample as if the mask weren't there and simply discard those points outside of the mask:

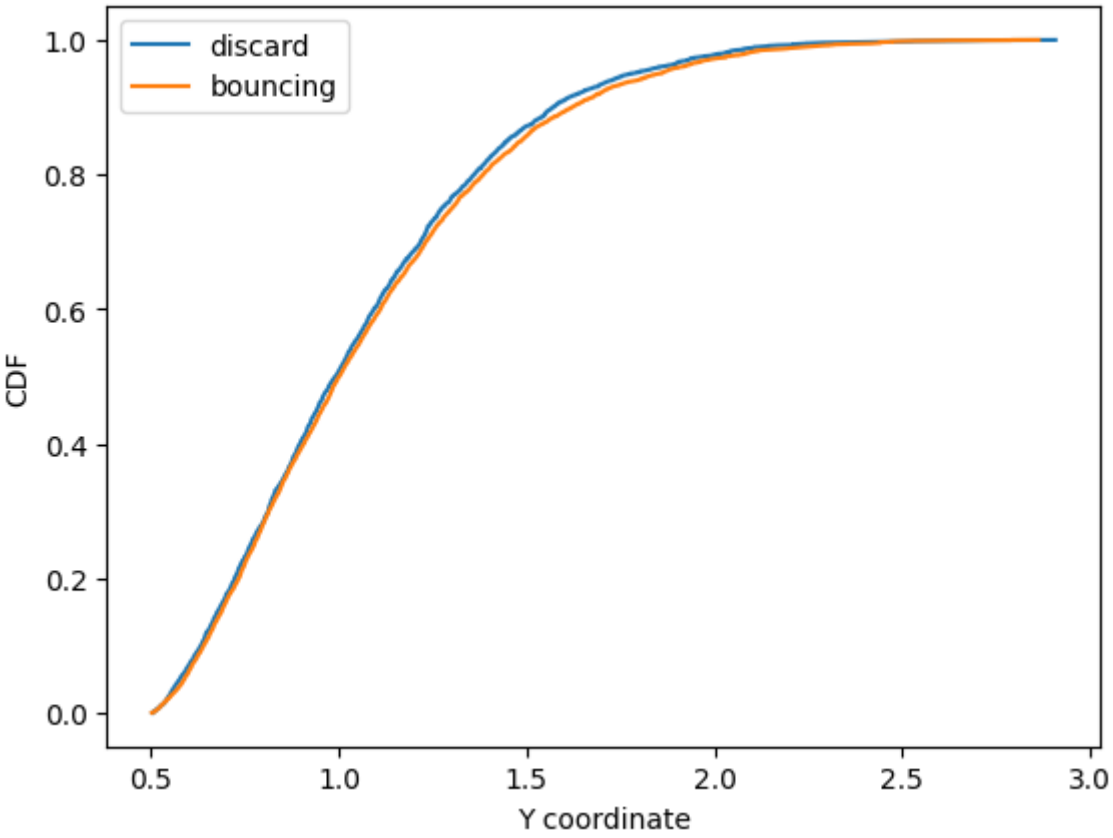


Note that we have arcs which have one end terminating outside of the mask. This is expected as these are points we originally sampled but later discarded as outside of the mask.

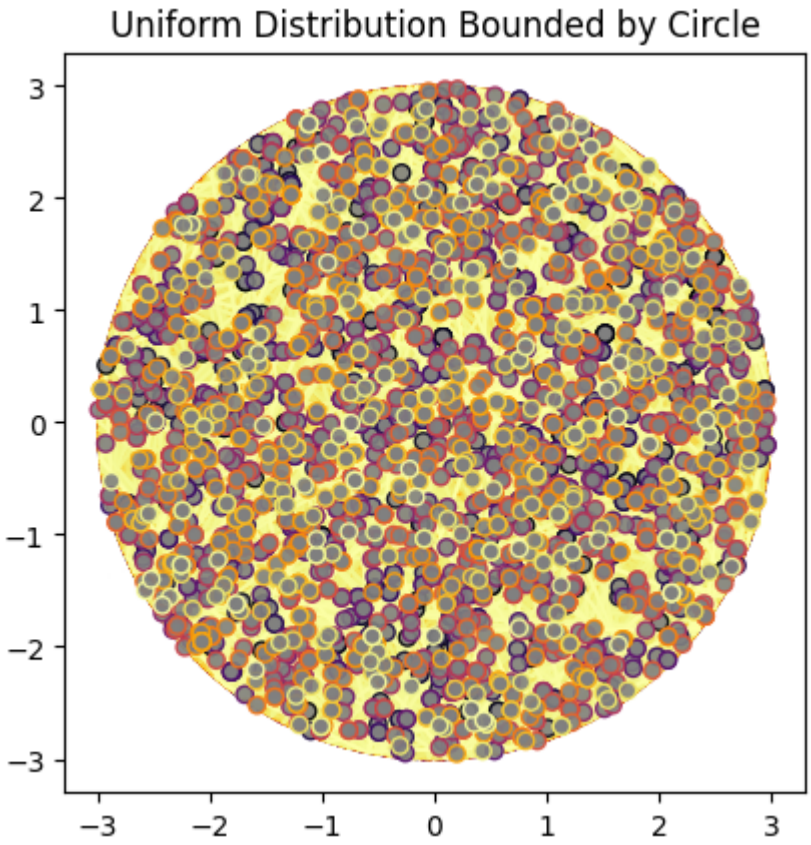
2025/02/08

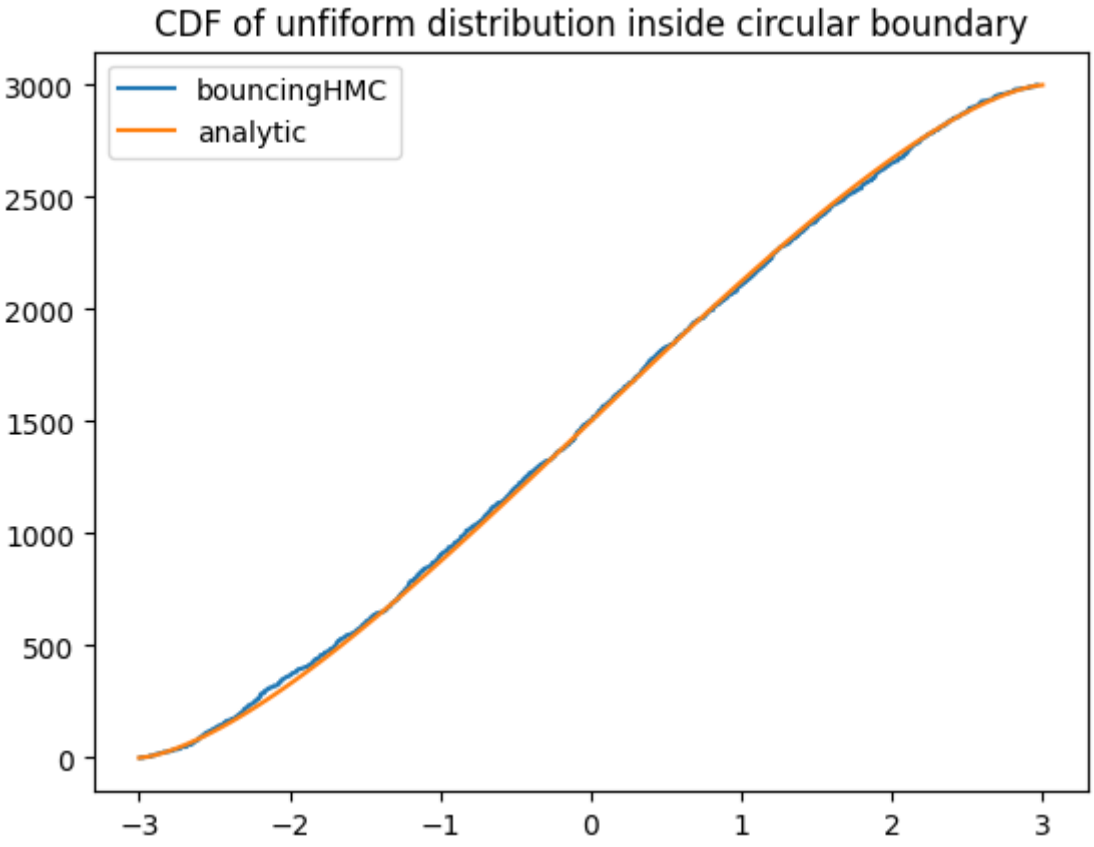
In a meeting with Will, he spotted that I'd forgetten to actually normalise the mask gradient I use to calculate reflections. After doing this, the quadratic mask on the Gaussian seems to yield good bouncing behaviour.





It also behaves well when we test it on a uniform distibution bounded by a circle:



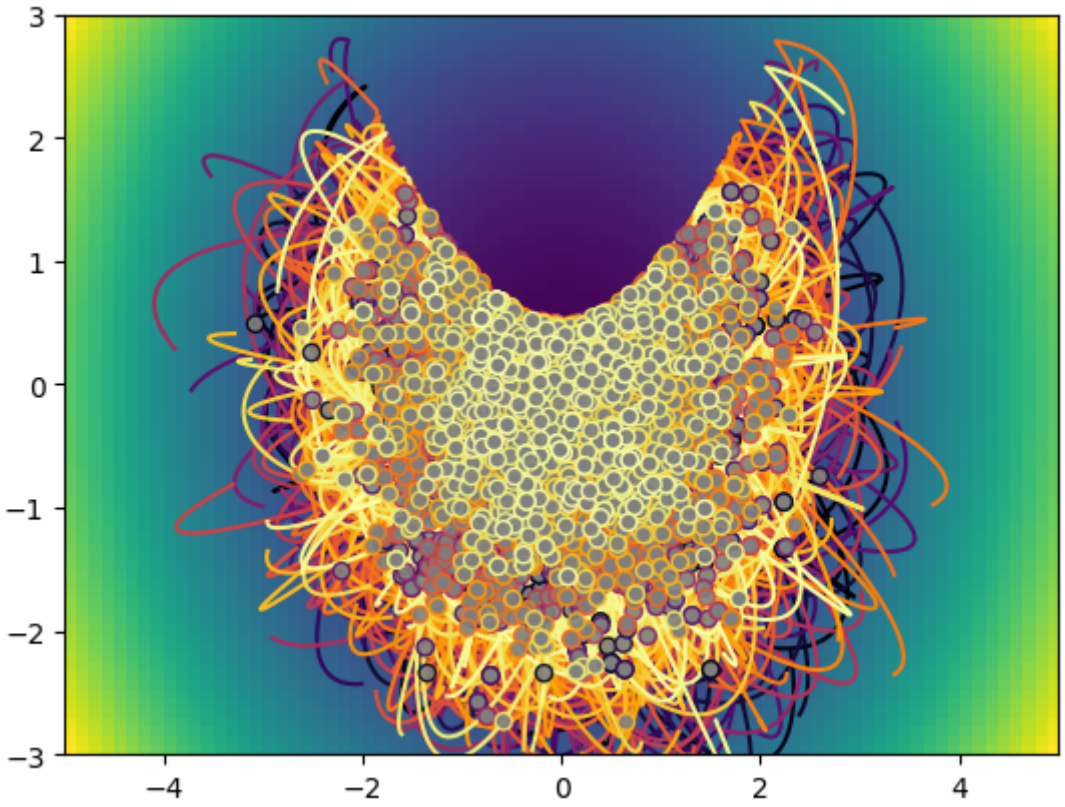


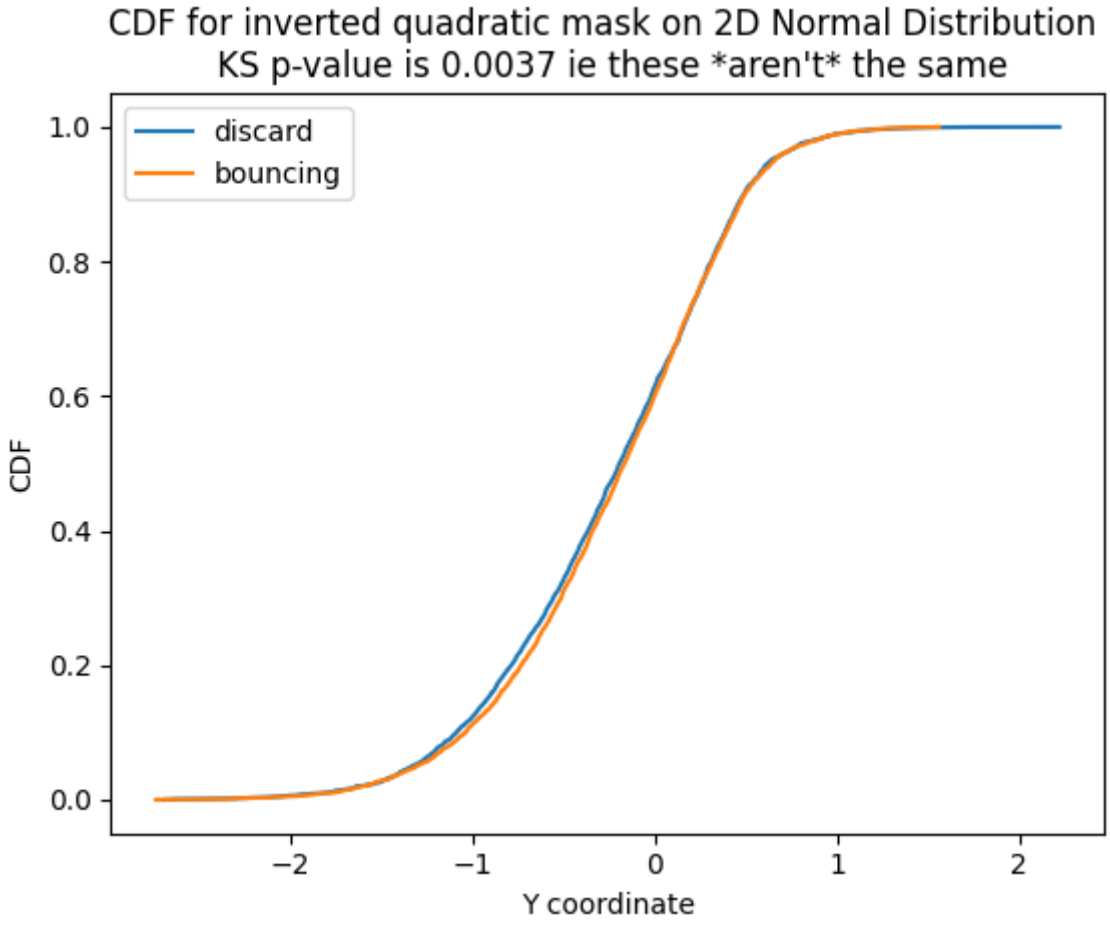
2025/02/08

Used an inverted quadratic mask as my intution is this would do a better job of shifting the sample points. This is indeed what we see - the mean y coordinate for bouncing HMC is -0.2210 and for uniform sampling with discard it's -0.2446

KS-test gives a p-value of 0.0037

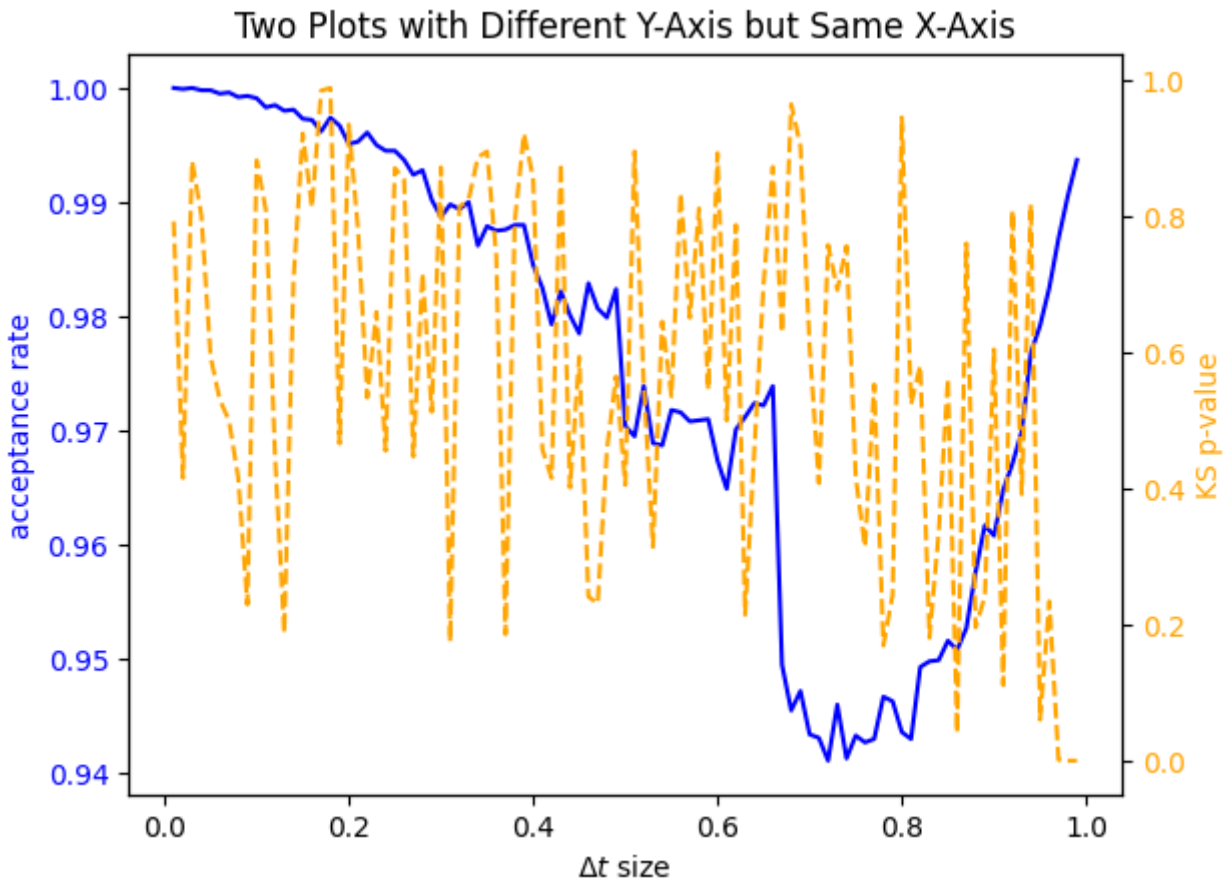
This is with 10000 samples.



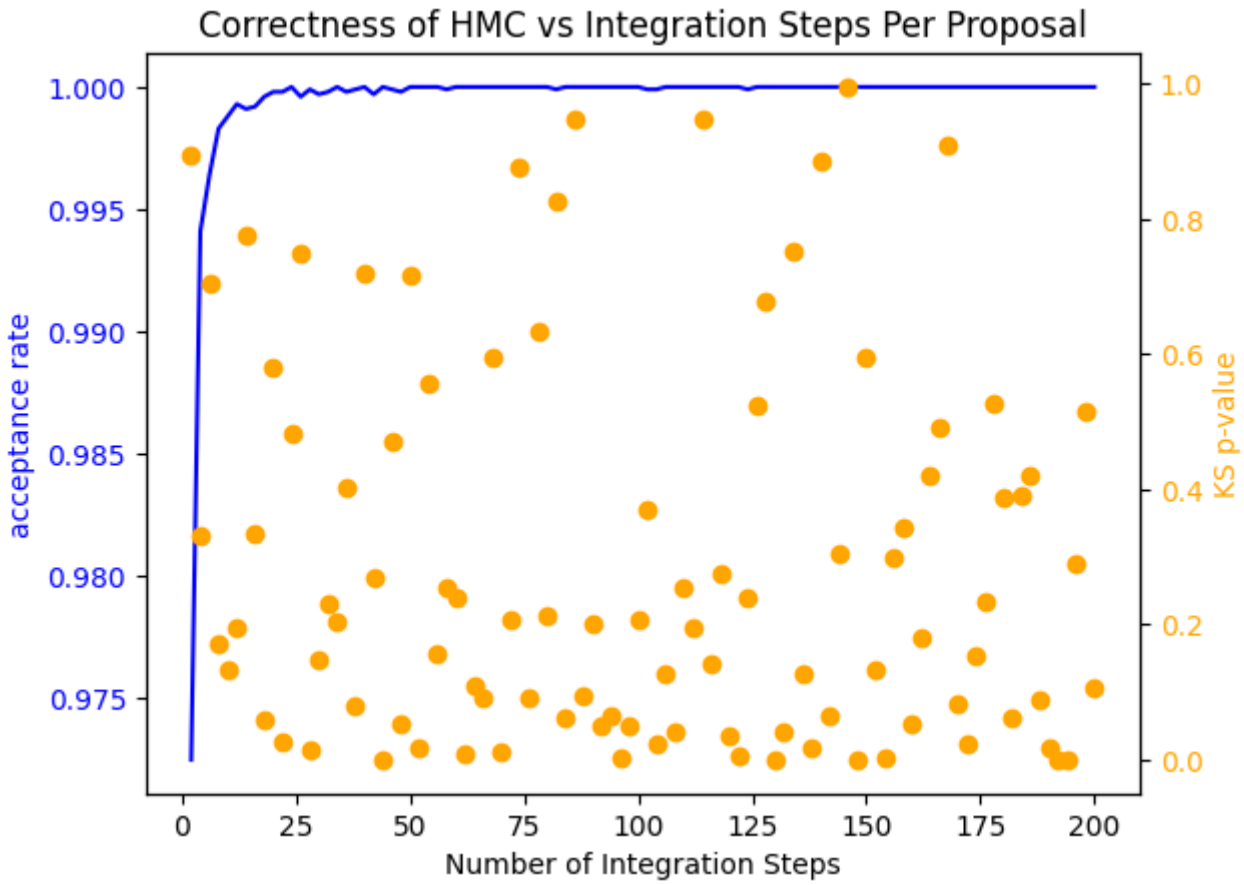


2025/02/20

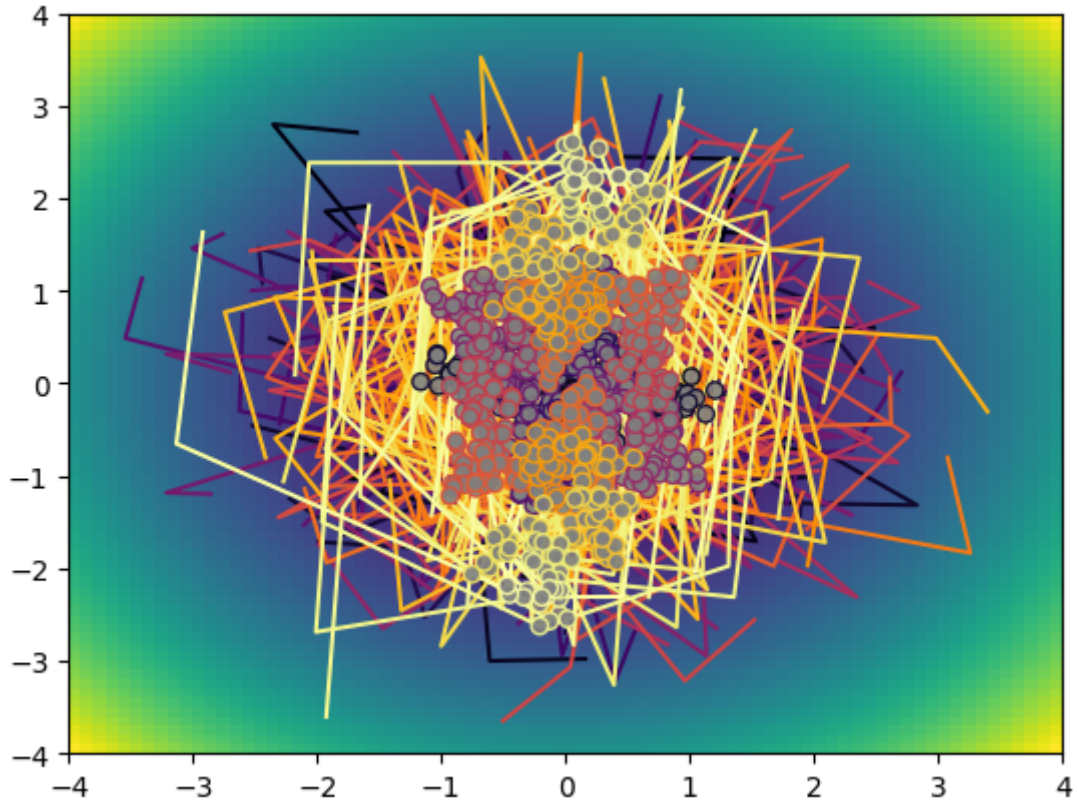
Experimented by increasing integration step size. As theoretically predicted, accuracy depends only weaky on step size whilst acceptance rate goes down steadily. I varied the number of steps inversely so that we traverse approximately the same distance in each case (ie we're getting coarser approximations to the same true path). Otherwise we would have noticed looping back on ourselves.



There is a very unexpected jump in the acceptance rate at high Δt. Presumably this is some kind of resonance behaviour. However, if we instead vary the number of integration steps (L) while varying Δt in a reciprocal fashion, we see much nicer behavior. This might be because of the fact we had to round L when varying Δt - as Δt approached 1, L was ceiling-ed from 4 to 3 to 2 while $Delta t$ varied smoothly so that total path length varied. It's much more natural to increment L uniformly since Δt can take floating point values. Some of the low-L p-values are concerningly high and suggest that the KS-test is not a useful indicator of corectness!

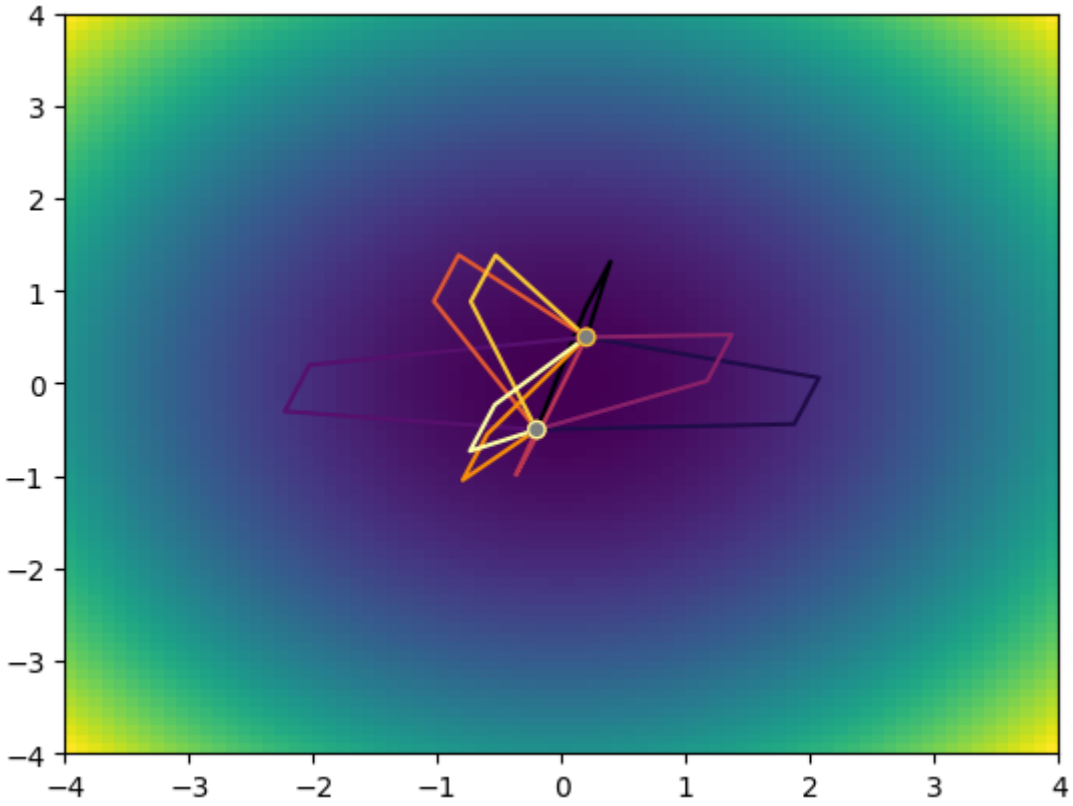


Upon checking the surprisingly high-acceptance rate but low p-value Δt - we see clear resonance effects:



The similar colours on opposite sides shows that the chain alternates opposite sides of the gaussian.

Indeed for L = 3, dt = 1 and a 2D Normal distribution, starting at (0.2,0.5) gives perfect bouncing:




```
xPositions
```

0.0s Python

```
array([[ 0.2,  0.5],
       [-0.2, -0.5],
       [ 0.2,  0.5],
       [-0.2, -0.5],
       [ 0.2,  0.5],
       [-0.2, -0.5],
       [ 0.2,  0.5],
       [-0.2, -0.5],
       [ 0.2,  0.5],
       [-0.2, -0.5],
       [ 0.2,  0.5]])
```

In fact we can take this to a silly extreme:

```
randi = bouncingHMCMC(logpdf, gradLogPDF, noMask, gradNoMask, dist.dim)
randi.dt = 1
randi.L = 3
xPositions, arcs = randi.rvs(1000, startX=np.array((0.2,0.5)))
randi.acceptances
```

[104] ✓ 0.0s Python

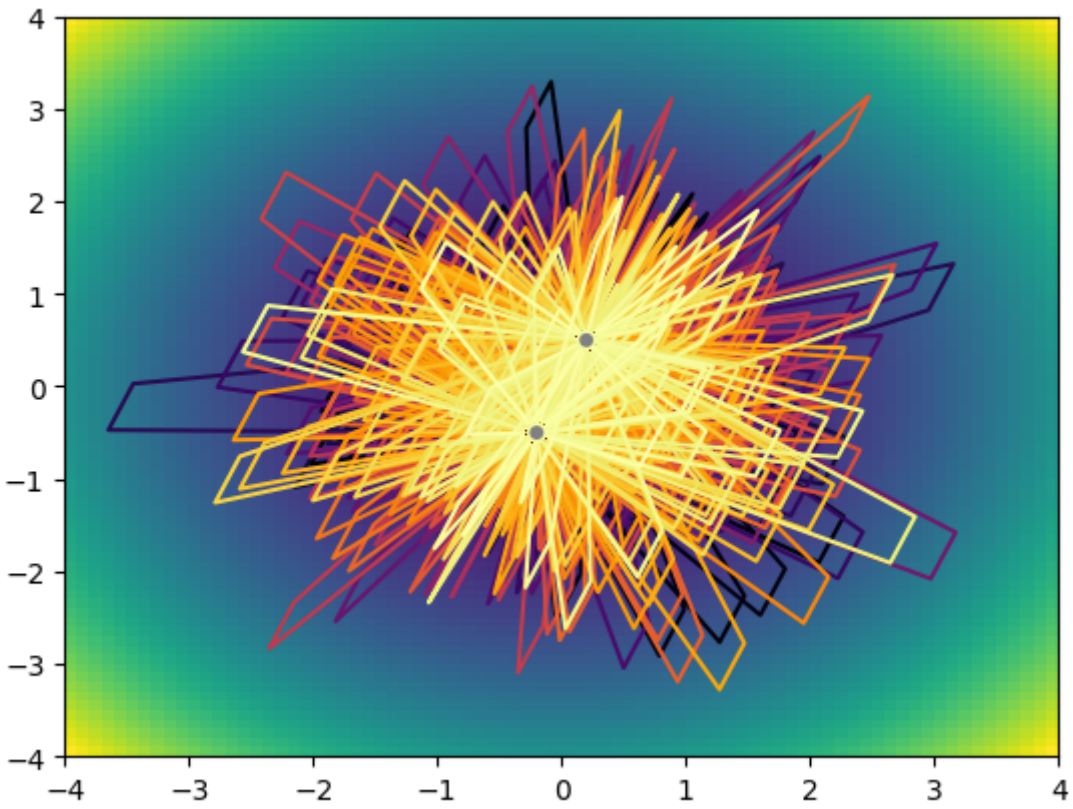
```
1000
```

+ Code + Markdown

```
xPositions
```

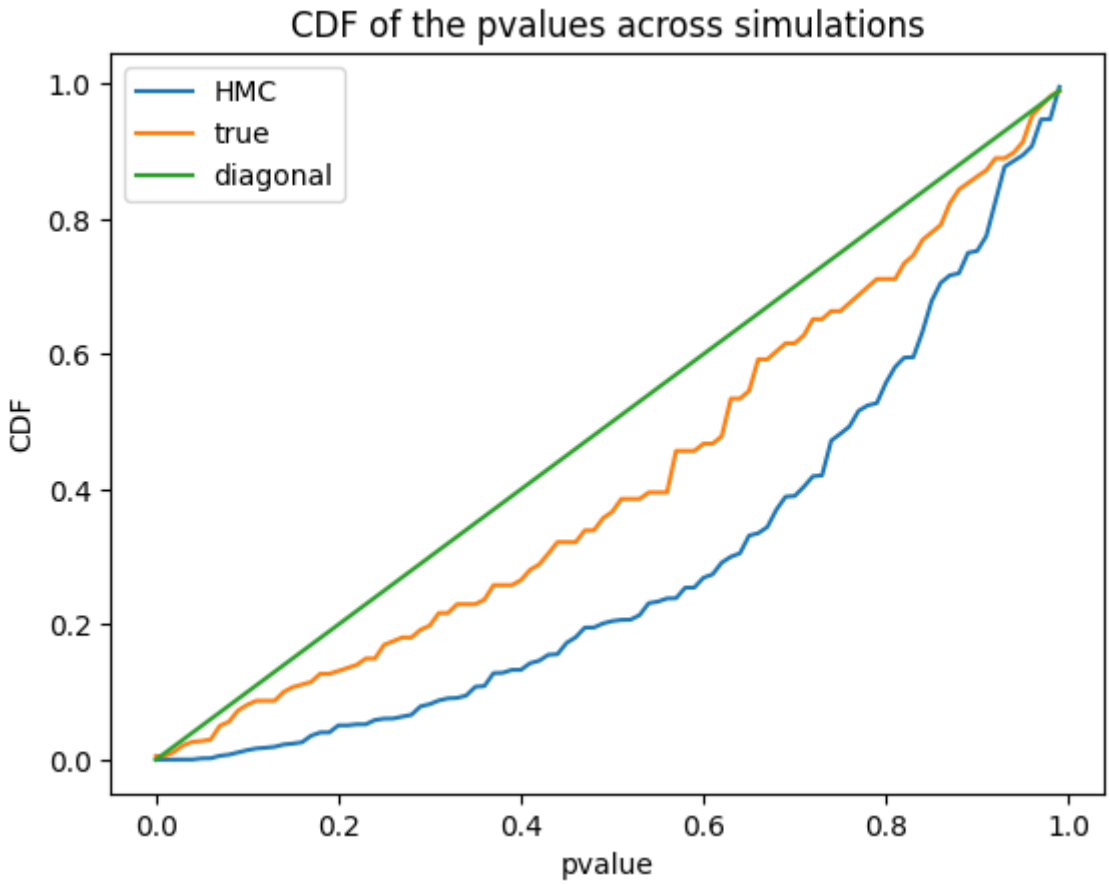
[106] ✓ 0.0s Python

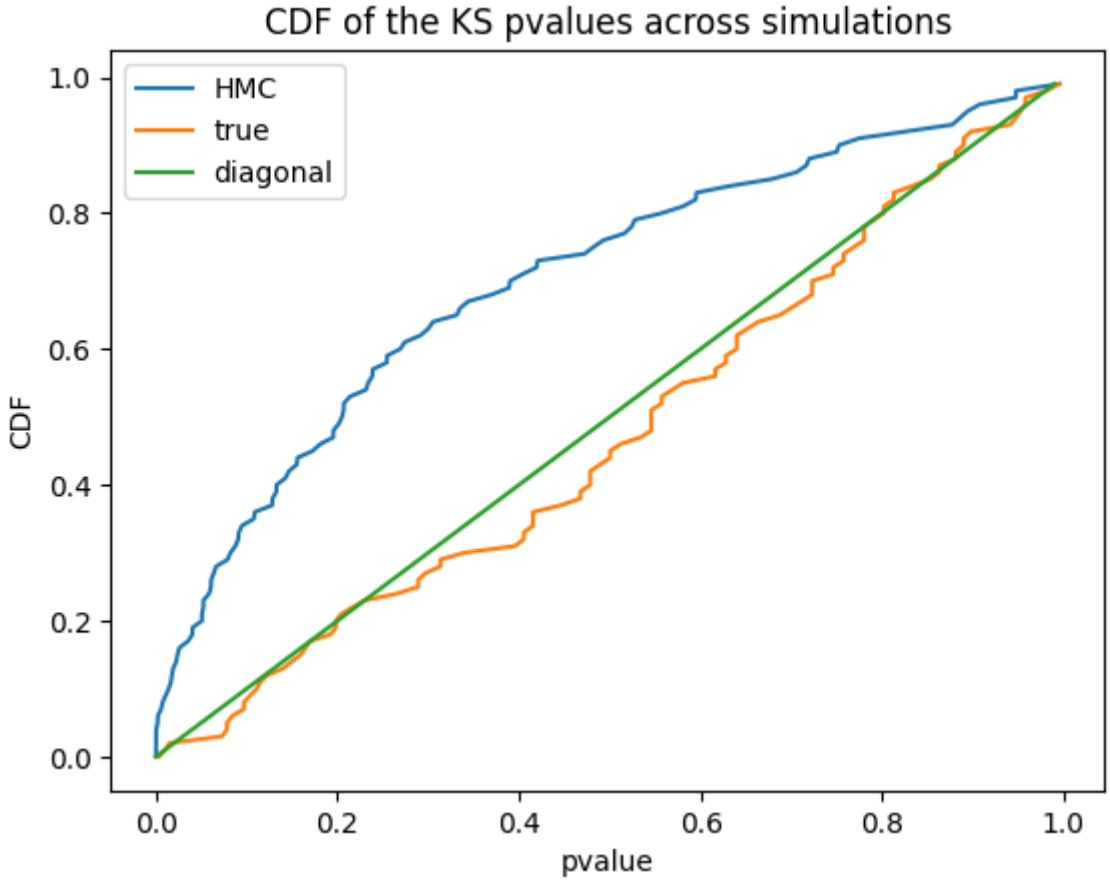
```
array([[ 0.2,  0.5],
       [-0.2, -0.5],
       [ 0.2,  0.5],
       ...,
       [ 0.2,  0.5],
       [-0.2, -0.5],
       [ 0.2,  0.5]])
```



NOTA BENE This alternating behaviour is another thing that would be fixed by no-u-turn!

Out of curiosity I checked the CDF of the KS p-values across the whole range of Ls. I compared it to the p-values of the dist.rvs with itself. I expect a straight line fit. The fact that the dist.rvs failed with itself first time around told me that I should generate a new dist.rvs each time - rather that comparing againt one "true" dist.rvs. Duh! I also realised my axes were swapped by mistake.





2025/03/08

Did a big lit review on Supernovae inference - need to type up thoughts. Bigliest problem is possible host-galaxy dust extinction evolving with age (distance/redshift). This is highlighted as substantial systematic

error in Pearlmutter et al 1999. Early indication that this is not significant is that it would induce additional spread in the hubble diagram due to random galaxy orientation - which was not seen. Presumably there is additional evidence of expansion if all the dark energy people are putting so much work in. Kaisey's papers actually do seem to address this directly. Kaisey's work uses Bayesian probability to fit everything at once and directly propagate errors forward whilst not allowing higher inferences to back-propagate to the earlier inference steps (previously "nuisance" parameters). This creates a high dimesnsional distribution hence using sampling. Some light curve models are degenerate - degeneracies are eliminated parametrically which allows us to force hard boundaries. Also, if we were to fit cosmological constants concurrently then we have the physical restriction that Omega_M > 0.

NB K-corrections come from the fact that we want to transform from observer frame filters to the rest-frame equivalent filter and this is complicated by the transmission functions of the filters, with additional effects of both Milky-Way and host-galaxy extinction. Rv is the ratio of selective extinction to total (grey) extinction which are A_x and A_v resp.

There are also camera sensor standardisations to consider (energy vs photon count vs photo-electric current etc) and atmospheric conditions. We might hope that observers have correctly standardised this for us to provide equivalent filter-band magnitudes. NB Milky Way dust is well characterised and there is a standard data set for this. There are additionally lens distortions to consider - both from the telescope and the curved atmosphere.

An additional effect is peculiar velocities - this is from local orbits and inhomegineity in the distribution of mass (voids). This has relatively less influence at higher redshifts.

Going forward:

- Blackjax is a generic sampling library using Jax (autodiff and GPU accel etc), they have example notebooks. I should write my bouncing HMC as a fork of their vanilla HMC/NUTS thing.
    - Blackjax repo
    - Examples notebooks
- Toby Lovick has a Blackjax fork for CMB stuff that isn't so useful for me
    - Toby Lovick Blackjax Fork
- Sam Leeney has an existing supernovae JAX code
    - SALT3 Model in JAX
    - Supernovae examples in notebook
- The lab has it's own set of examples notebooks (which contains Sam's one)
    - Handley-Lab Notebooks

Things I need to do:

1. Run Sam's code with the generic sampler from Blackjax
   a. This will require finding some supernovae data and being sure of the format (try Will's winter email)
2. Write report - half context!!!!!!!

3. Fork Blackjax and modify the HMC to be bouncing - possibly write from scratch if the adaptive is too complicated. Remember the M metric with reflections!
4. Maybe implement a non-SALT model?
5. Maybe follow Kaisey 2022 with the Bayesian approach
6. Maybe do a cosmologicial fit

Side quests:

1. Implement multiple masks - think over what was discussed in meeting where must hit corner and the reversed path must also hit that overlapping corner. Does this work for n overlaps (induction?).
2. Do a quick lit review on the bouncing/sampling/stat inference. Use ChatGPT and Connected Papers
3. **Presentation due 17th!!!**
4. *Still think there might be a Jacobian-like stretch factor on the probability per reflection* - **investigate!**
5. NUTS!
6. Adaptive sampling!

If I get all of the above done then that sounds like a bona fide project with maybe a first. **However don't forget that the math exams take priority** - could probably get a mediocre project grade with substantially less effort a la 80:20 principle.

- **UPDATE previous todos with yellow slips and move above notes into seperate file**

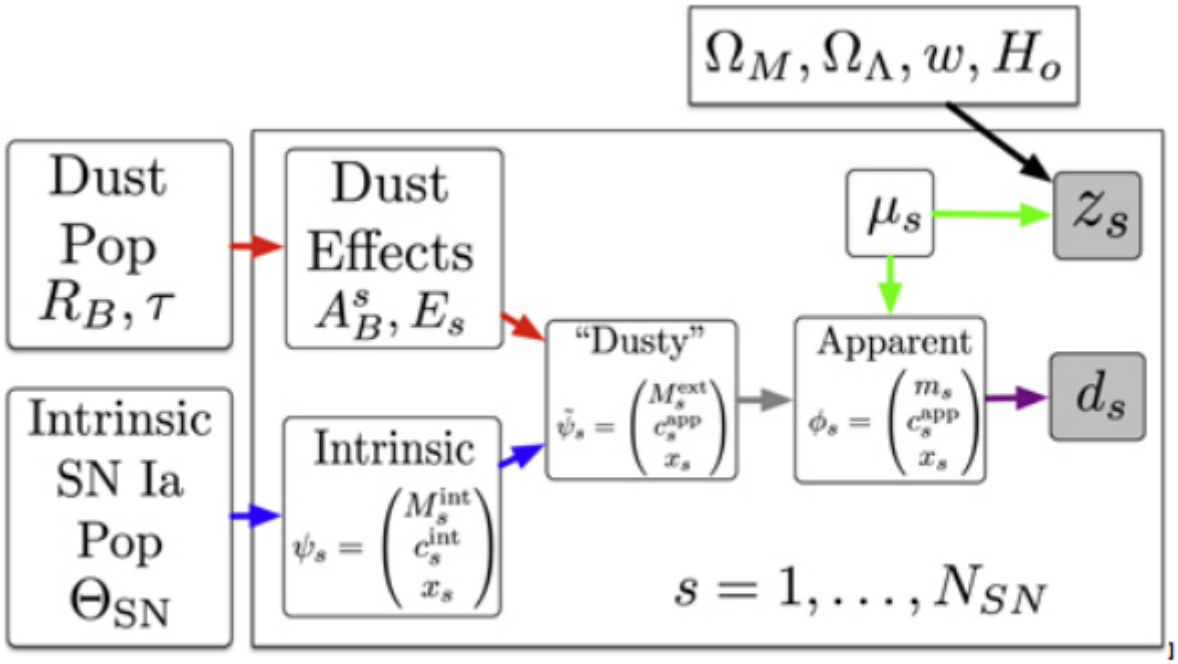


Figure 1: A Probabilistic Graphical Model describing the Simple-BayeSN hierarchical Bayesian model for Type Ia supernovae (Mandel+2017). The goal of this project is to explore distributed inference with models of this type through "nuisance-free likelihood" analysis of subsets of the full supernova dataset.

2025/03/13

Conversation with Kaisey Mandel and Matt Grayling

param bounded + tran log spaces

sample in log -> post pushes you ver far away from zero

therefore advantage to sampling in lin space with proper bounding

eg dust estimation in supernovae Av pos and Rv

uniform prior Rv 1.2 (Rayleigh) - 6

Av pos and Rv bounded on two sides

low extinction no constraints on Rv -> Under hood transformations HMC

We think transforms under hood

Ana Sofia paper variational inference -> dust problem

post usually non-gaussian -> find best gaussian though

doing log transform under hood with gaussian post assumes log-normal underneath with zero density at 0

-> had to come up with new dist : zultan dist to deal with this problem

MVZLTN Ana Sofia Uzsoy 2024 Matt and Kaisey

Crash course Kaisey

2024 Matt paper has tutorial link to tutorial

Running Sam's Code Issues:

- need to install blackjax in addition
- tries to import log_weights from blackjax.ns.utils but blackjax.ns doesn't exist
    - bug report in notebook not obvious (see image)

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[2], line 7
      4 import matplotlib.pyplot as plt
      5 import tqdm
----> 7 from blackjax.ns.utils import log_weights
      8 from jax_supernovae.salt3 import optimized_salt3_multiband_flux
      9 from jax_supernovae.data import load_and_process_data

File ~/bouncingHMCsupernovae/.venv/lib/python3.9/site-packages/blackjax/__init__.py:6
      2 from typing import Callable
      4 from blackjax._version import __version__
----> 6 from .adaptation.chees_adaptation import chees_adaptation
      7 from .adaptation.mclmc_adaptation import mclmc_find_L_and_step_size
      8 from .adaptation.meads_adaptation import meads_adaptation

File ~/bouncingHMCsupernovae/.venv/lib/python3.9/site-packages/blackjax/adaptation/__init
----> 1 from . import (
      2     chees_adaptation,
      3     mclmc_adaptation,
      4     meads_adaptation,
      5     pathfinder_adaptation,
      6     window_adaptation,
      7 )
      9 __all__ = [
...
    213
    214     """
    216     kernel = build_kernel()

TypeError: unsupported operand type(s) for |: '_UnionGenericAlias' and 'NoneType'
```

- I updated to python3.10 - chatGPT identified error as being a modern langauge feature (| as bitwise OR for type hints)
  - File "/home/harvey/bouncingHMCsupernovae/.venv/lib/python3.9/site-packages/blackjax/mcmc/barker.py", line 162, in inverse_mass_matrix: metrics.MetricTypes | None = None, TypeError: unsupported operand type(s) for |: '_UnionGenericAlias' and 'NoneType'
- I had to manually install the "requests" library since I'm working in a venv - the code/notebook really ought to handle this as a dependancy
- The first time running the sampler hadn't converged by the 11 minute mark. I had no idea of how to monitor progress so with the help of chatGPT I added:
  - pbar.set_description(f"Dead points | logZ: {(state.sampler_state.logZ_live - state.sampler_state.logZ):.2f}") # Update tqdm description
- ..which meant I could track progress by watching the difference head towards -3

2025/03/14

Actually Skilling

Meeting With Will

SALT andddddddddd STAN

Examples for pres :

Light curve fitting and blah - refer to Kaisey's Group

Also used in Galaxy evolution: force pho (force photometry)

Also in CMB parater estimation - cosmopower

Exoplanets

Strong grav lens modelling

Slides:

HMC important across astronomy! Linker anal - many aspects of real data analysis where HMC fails eg hard boundaries Now discuss existing solutions and discuss

About half should be context

Give a lot of lot of background - NOT INTERESTED IN FINE DETAILS OF WHAT'S DONE About half should be context

Blackjax -> HMC

Much easier to just remove the nss bit from Sam's code and plug in the log-likelihood into my own (Jax powered) sampler

2025/03/16

Afshar & Domke (2015) Reflection, Refraction, and Hamiltonian Monte Carlo. + 2021 paper **Important notes on volume transformations and modifications to probability etc**

2025/03/17

STAN on reparam

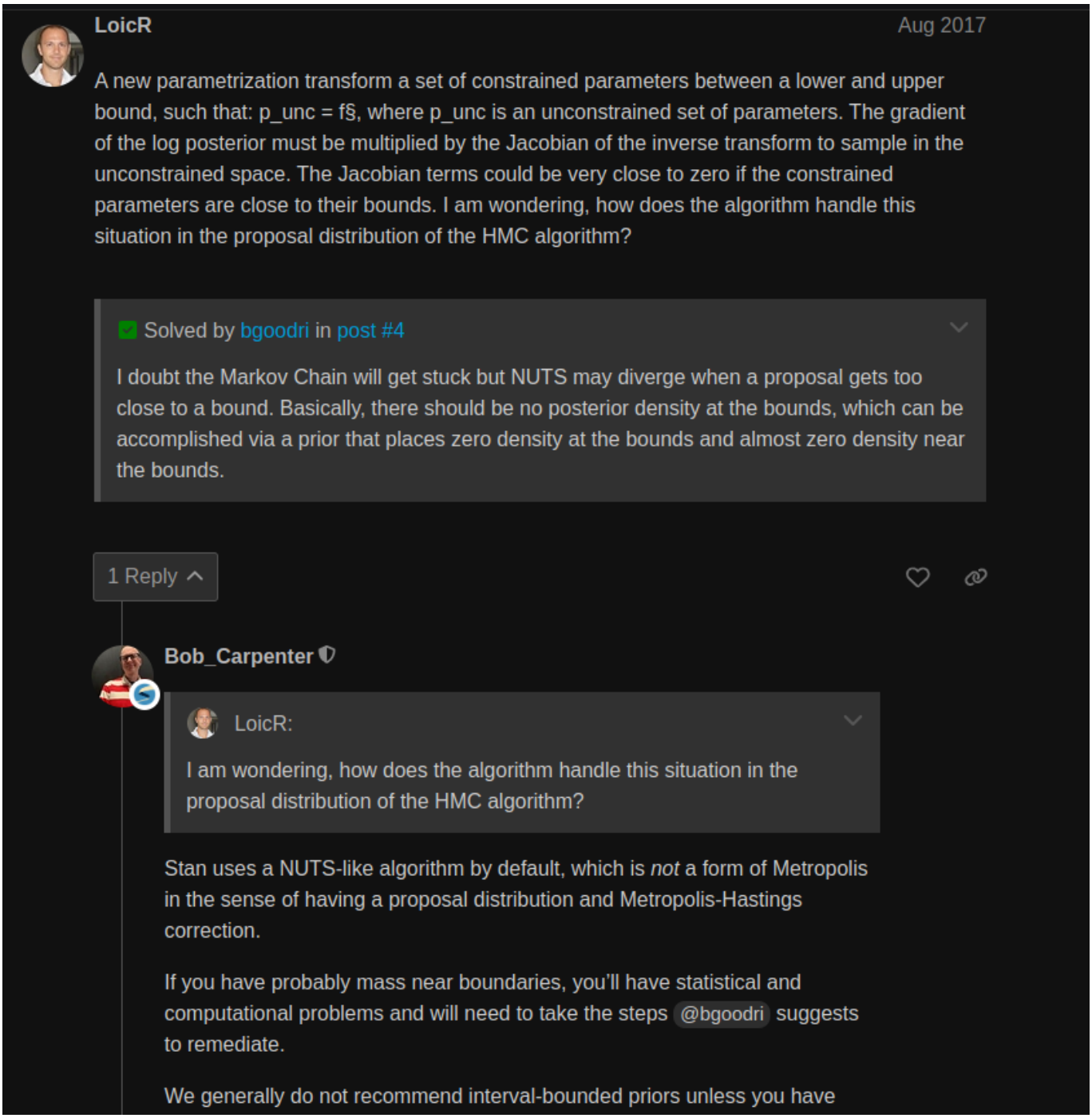
Places the algo would be good

PolyChord by Will Handley

- Nested sampling
    - Bouncing HMC makes it good for non-uniform priors

2025/03/31

ChatGPT found me a STAN forum discussion highlighting the challenge of hard boundaries:



2025/04/12

Learned about warp divergence in GPU programme design. See GPT convo for details. Issue is SIMT. Workarounds include masking and tables. Relevant to my code would be to evaluate all functions regardless and then use a switch/case structure or similar instead of the nested logic.

TODO: understand pytree stuff TODO: GPT book recs eg