

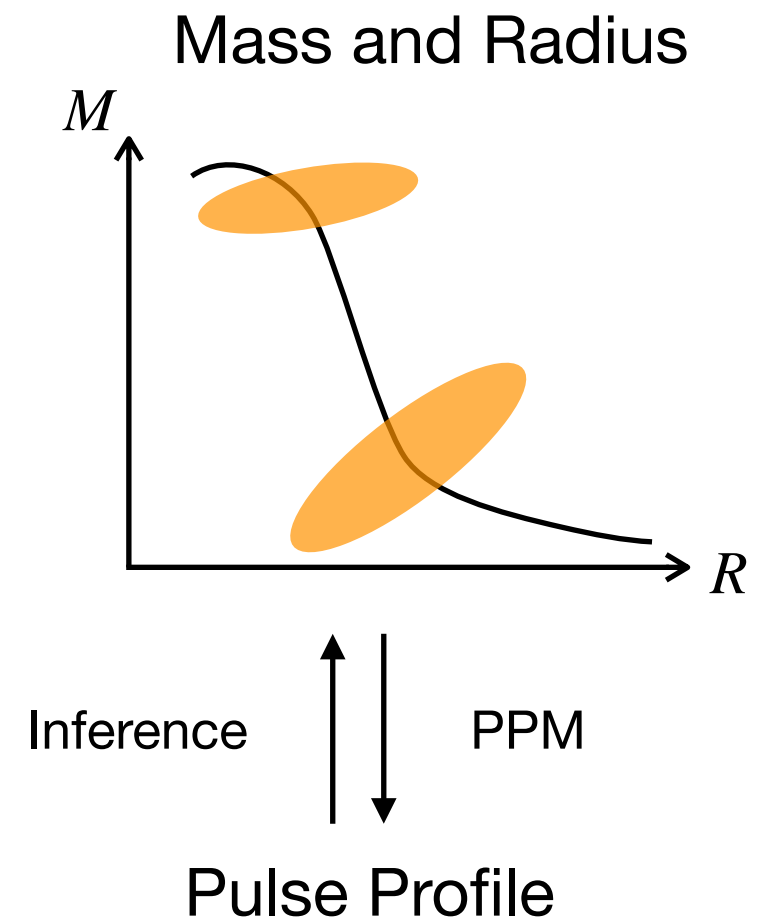
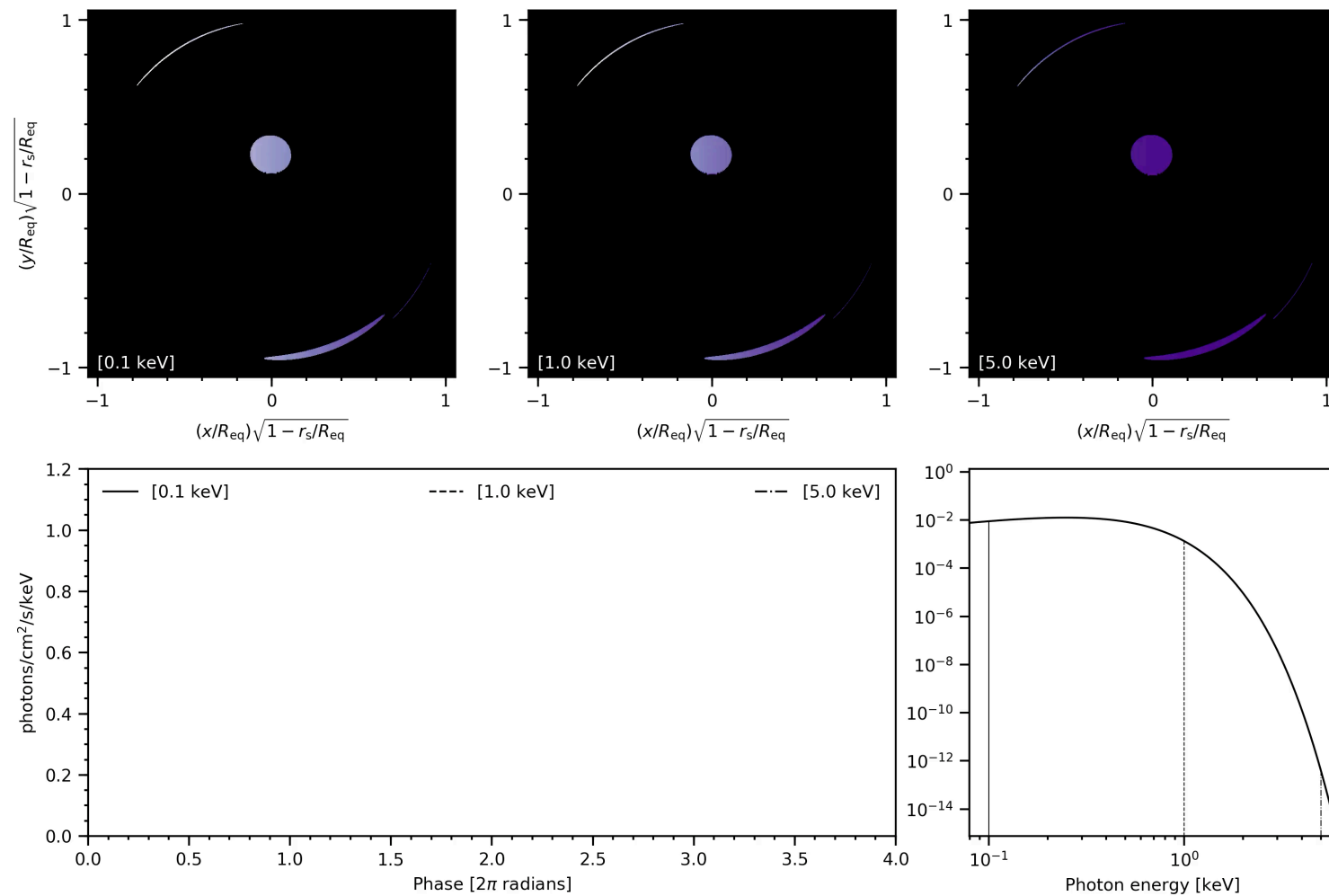
GPU-Accelerated X-ray Pulse Profile Modeling

Tianzhe Zhou, Department of Physics, Tsinghua University

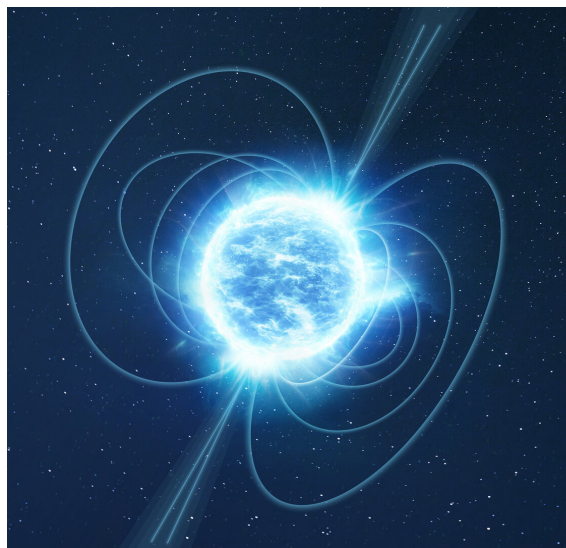
In collaboration with: Chun Huang

Based on: A&A, 709, A111 (2026), arXiv:2510.07764

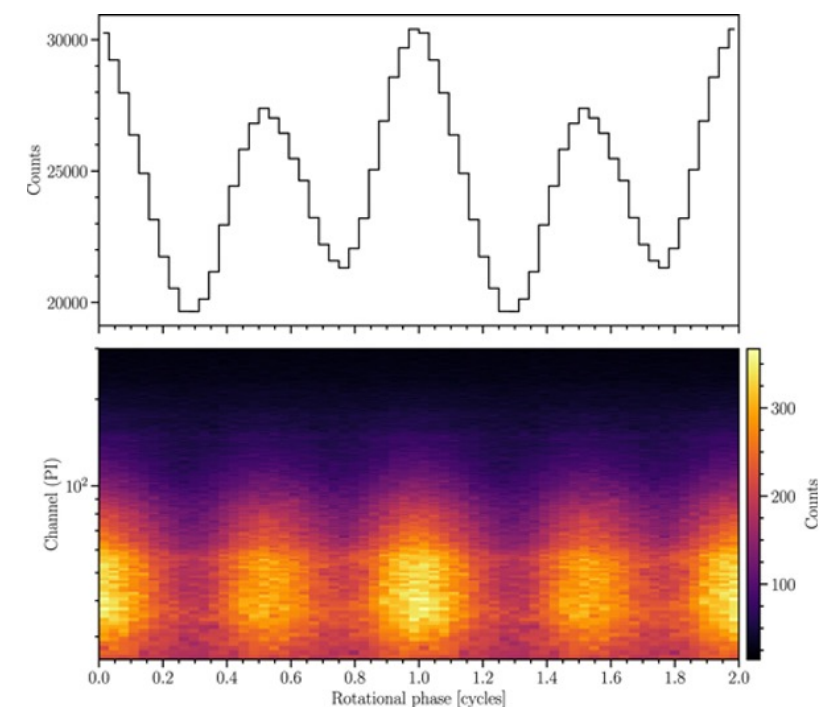
X-ray Data Analysis Pipeline



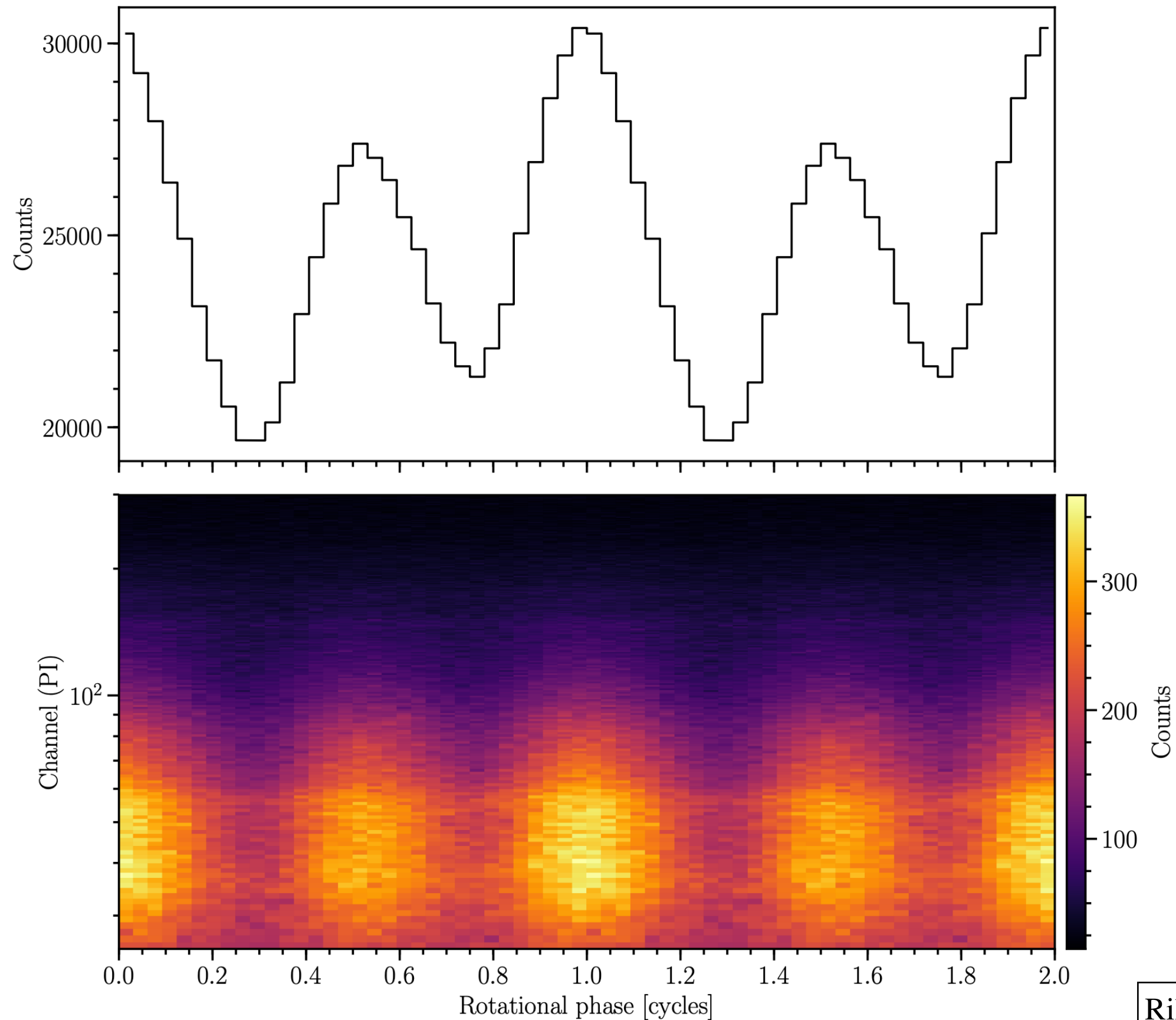
Neutron Star



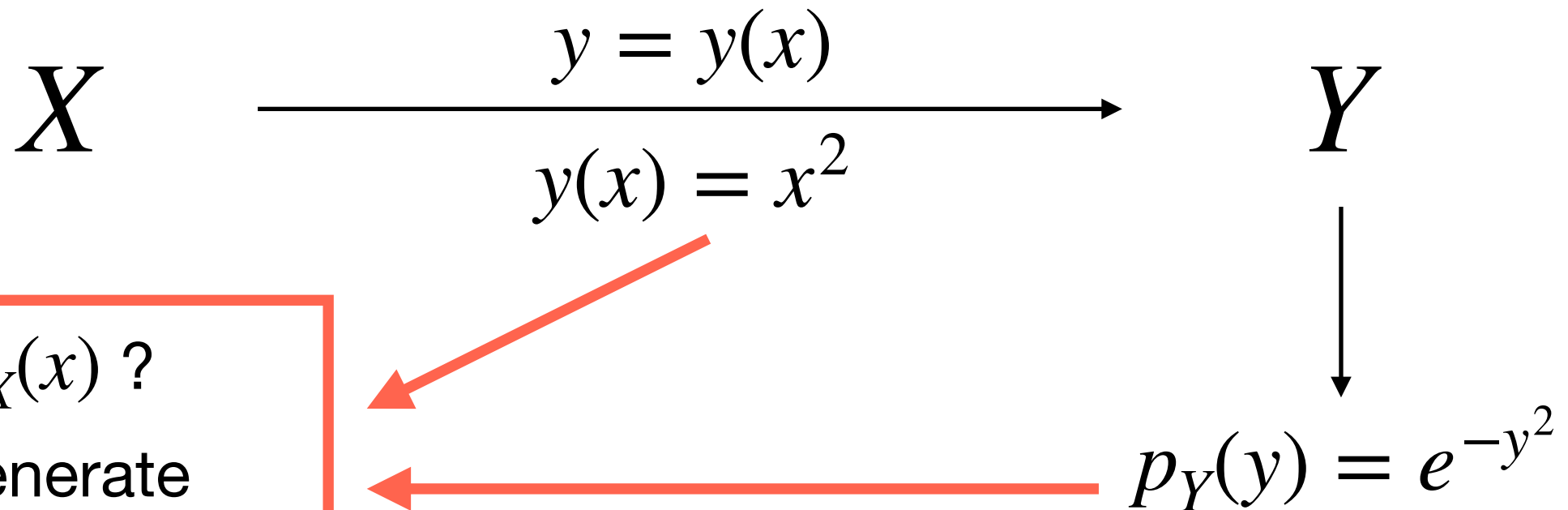
Detectors



Phase-resolved pulse profile



Inference: Sampling techniques



- What is $p_X(x)$?
- How to generate sample from $p_X(x)$?

- Weighted or equal weight sample?
- Dimension curse?

Various sampling algorithms try to answer these question:

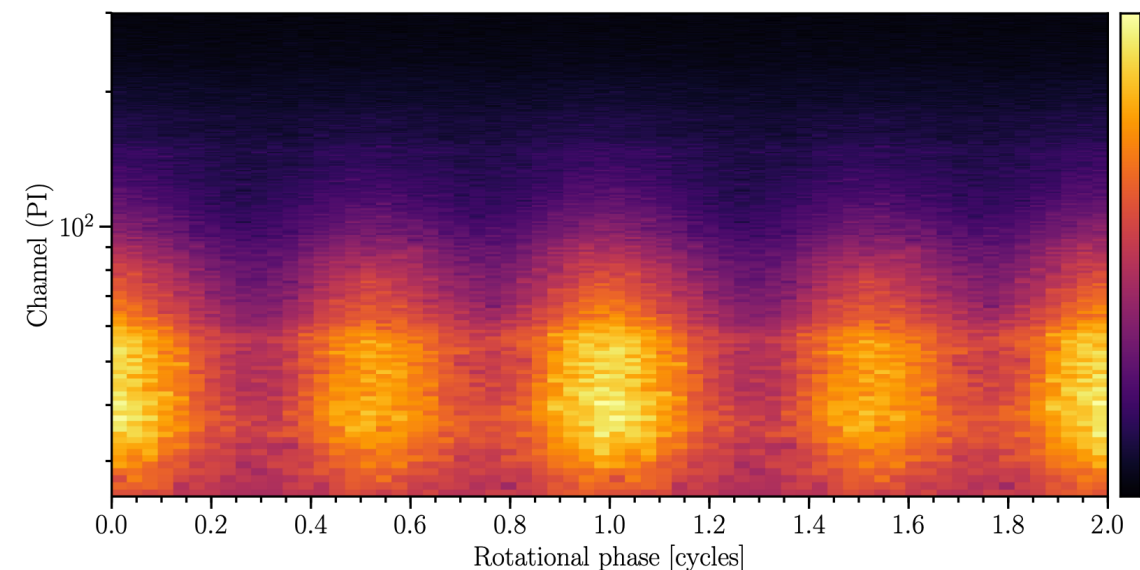
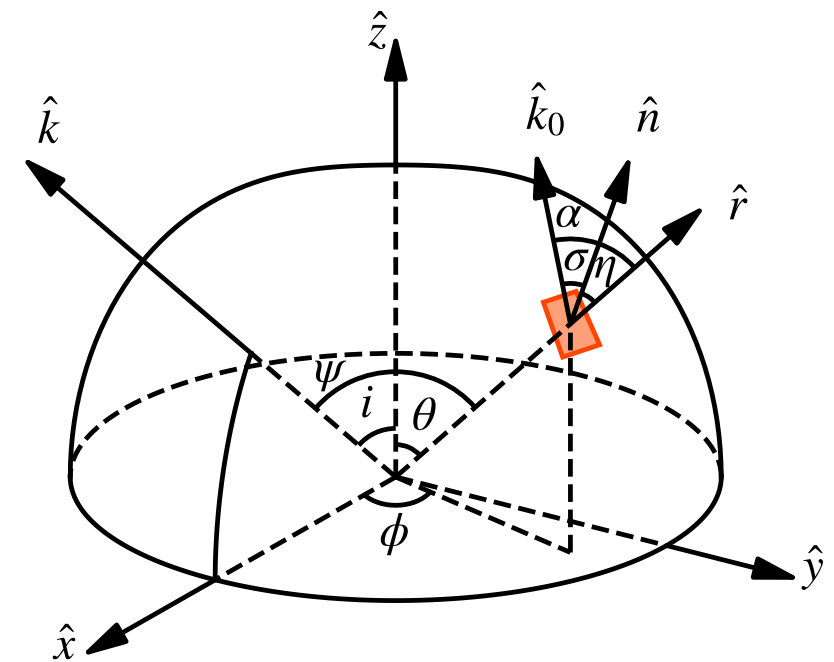
- MCMC: HMC/NUTS, Affine Invariant Ensemble sampler, Parallel-Tempering...
- Nested Sampling: ellipsoidal decomposition, slice sampler...

A simple pulsar model

- Spherical star & slow rotation
- Schwarzschild spacetime
- Blackbody spectrum Given star's mass M , radius R , frequency ν , inclination i , hotspots(location, shape, temperature)
- How to calculate phase-resolved spectrum for a distant observer?

```
# First divide NS surface
# into small patches
for each rotational_phase:
    for each energy_bin:
        for each surface_patch:
            calc_flux()
# This computation paradigm is
# naturally parallelized.
```

$$\frac{dN}{dt dA dE} = \sqrt{1-u} \delta^3 \gamma \cos \sigma' \frac{d \cos \alpha}{d \cos \psi} \frac{I(E', \sigma')}{E} \frac{dS'}{D^2}$$



A realistic millisecond-pulsar model

Often include:

1. Atmosphere(chemical composition, spectrum, beaming pattern)
2. Temperature distribution(spot location/size/shape)
3. Spin-induced oblate shape
4. Doppler effect induced by rotation
5. GR gravitational lensing
6. GR multi-image effect
7. GR redshift and time dilation
8. Interstellar medium(ISM) caused energy-dependent attenuation
9. Optic efficiency of telescope/detector

**Things
to
tweak**

Accuracy-Speed Tradeoff

NICER-only

Model	Mass [$M_{\odot}$]	Radius [km]	$\log(\mathcal{E})$	$\max(\log(\mathcal{L}))$	$\max(\log(\mathcal{L}_N))$	$\max(\log(\mathcal{L}_{N\&XMM}))$	Core hours
ST-U	$1.12^{+0.13}_{-0.08}$	$10.53^{+1.15}_{-0.89}$	-35788	-35735	-	-42950	11462
ST+PST	$1.37^{+0.17}_{-0.17}$	$13.11^{+1.30}_{-1.30}$	-35778	-35722	-	-42942	133970
ST+PDT	$1.20^{+0.14}_{-0.11}$	$11.16^{+0.90}_{-0.80}$	-35779	-35729	-	-42705	33069
PDT-U	$1.41^{+0.20}_{-0.19}$	$13.12^{+1.35}_{-1.21}$	-35777	-35721	-	-42659	74333

NICER and XMM-Newton

Model	Mass [$M_{\odot}$]	Radius [km]	$\log(\mathcal{E})$	$\max(\log(\mathcal{L}))$	$\max(\log(\mathcal{L}_N))$	$\max(\log(\mathcal{L}_{N\&XMM}))$	Core hours
ST-U	$1.88^{+0.13}_{-0.19}$	$15.12^{+0.64}_{-1.31}$	-42714	-42661	-35760	-	41219
ST-U*	1.35 ± 0.16	11.44 ± 1.14	-42718	-42666	-35769	-	-
ST+PST	$1.93^{+0.10}_{-0.13}$	$15.12^{+0.64}_{-1.31}$	-42711	-42662	-35761	-	84620
ST+PDT	$1.40^{+0.13}_{-0.12}$	$11.71^{+0.88}_{-0.83}$	-42684	-42625	-35730	-	3261
PDT-U	$1.70^{+0.18}_{-0.19}$	$14.44^{+0.88}_{-1.05}$	-42677	-42617	-35723	-	12421

Vinciguerra et al. (2024)

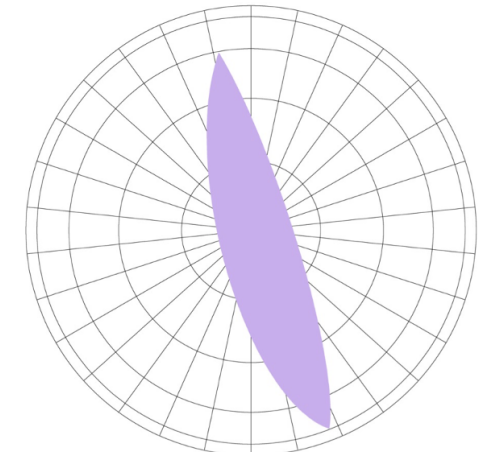
For many models, accurate computations are **time-consuming**.

Accuracy-Speed Tradeoff

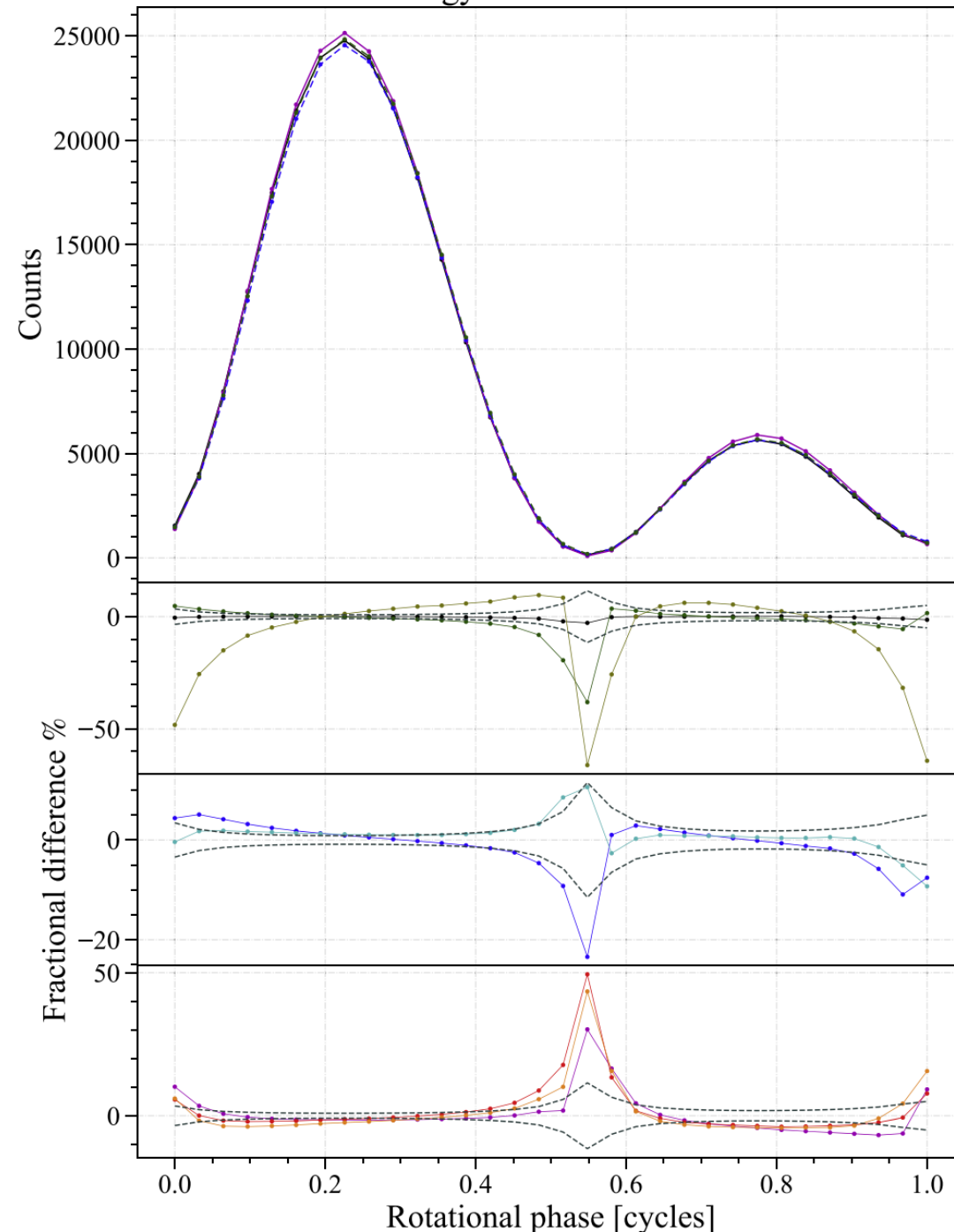
Extreme hotspots **can mismatch** by **~50%** without high-resolution computations.

This creates a seemingly impossible **accuracy-speed tradeoff** for pulse profile modeling.

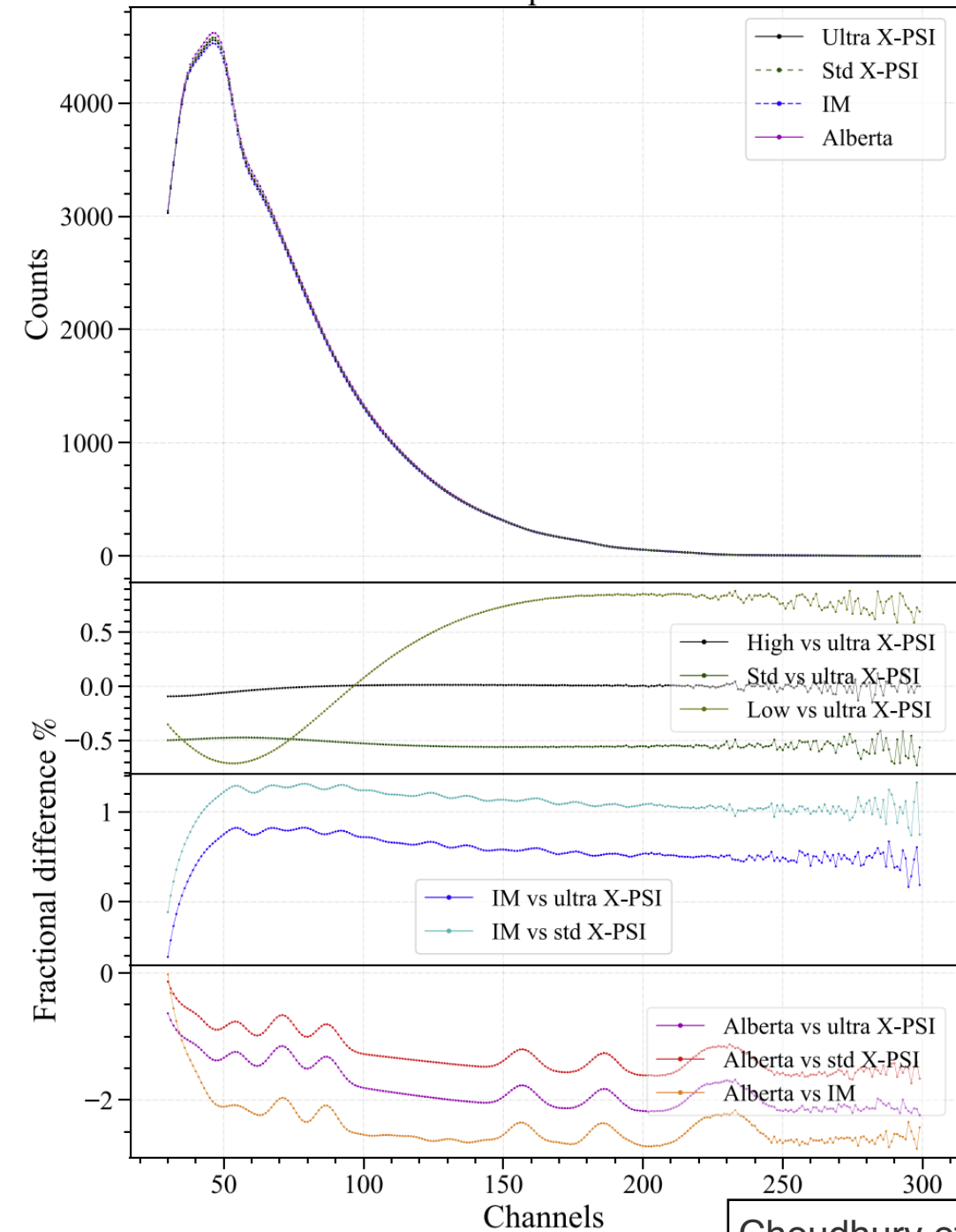
Crescent-Polar



Crescent-Polar energy-summed waveform & frac. diff.

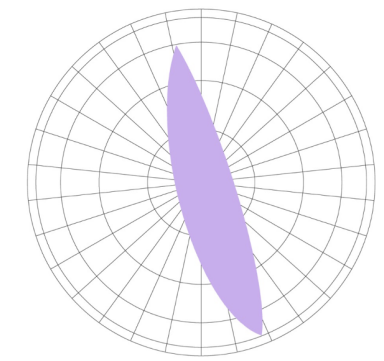


Crescent-Polar spectra & frac. diff.

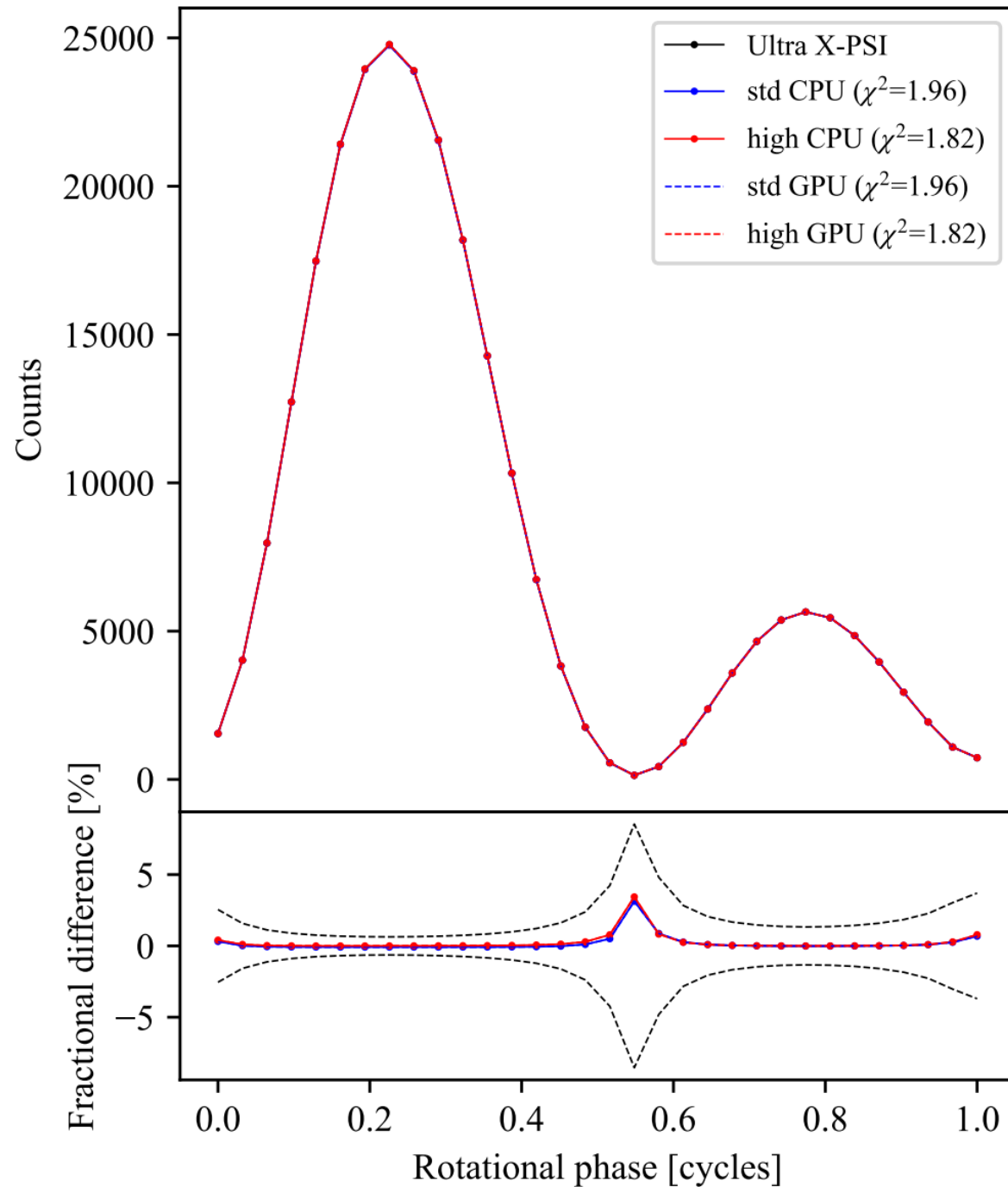


Accuracy-Speed Tradeoff

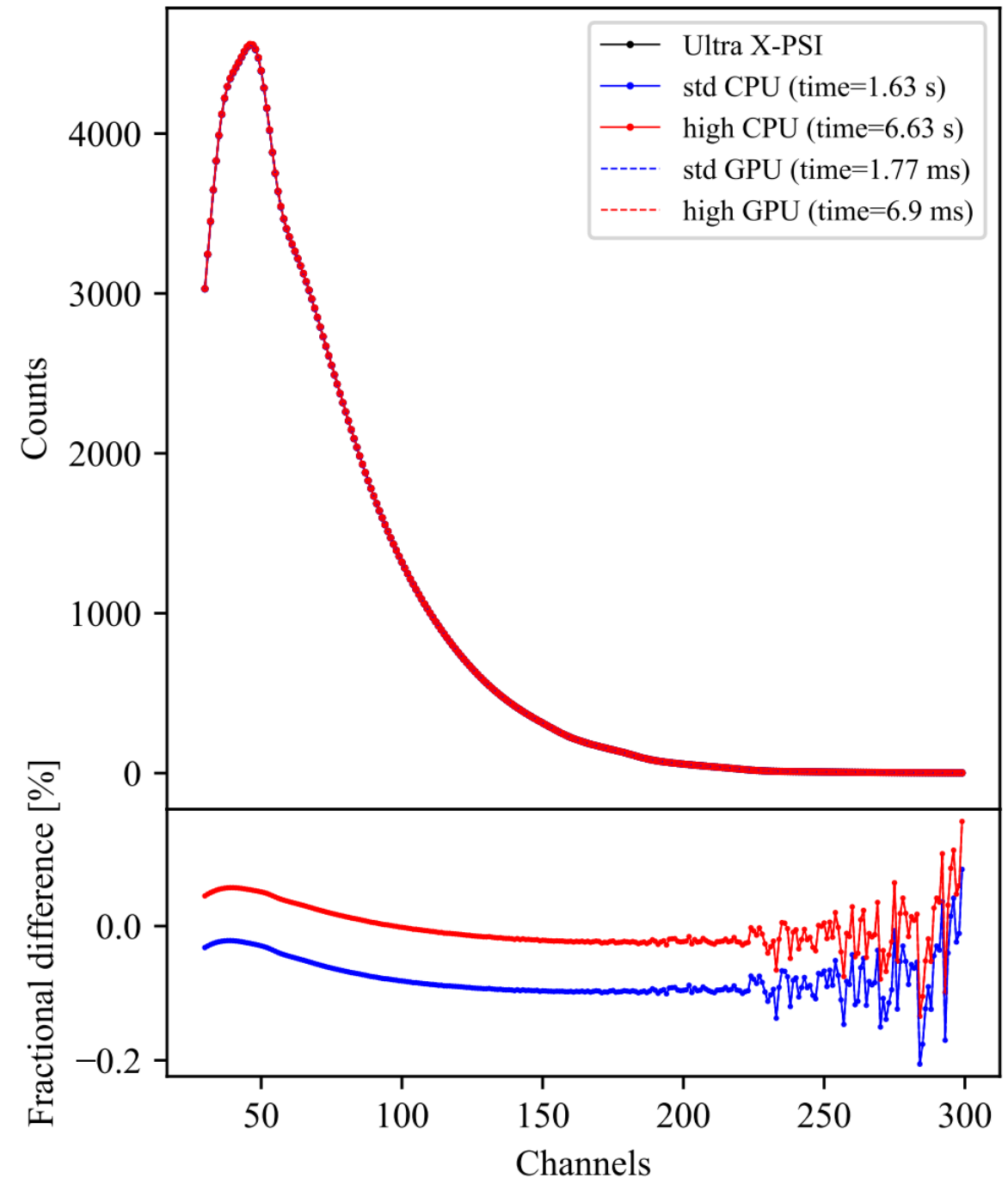
Crescent-Polar



Crescent_Polar energy-summed waveform & diff.



Crescent_Polar spectra & diff.



THIS WORK

Accuracy-Speed Tradeoff

Table 3

Approximate Pulse Profile Computation Time

Code	Computation Time
X-PSI low	$\lesssim 0.1$ s
X-PSI std	$\sim 0.1\text{--}0.5$ s
X-PSI high	$\sim 0.3\text{--}5$ s
X-PSI ultra	$\sim 1\text{--}5$ minutes
IM	$\sim 5\text{--}10$ s
Alberta	$\sim 5\text{--}30$ s

Note. The exact times depend on the model chosen and the CPU architecture of the system the code is being run on, and therefore we only provide approximate times.

Choudhury et al. (2024)

This work:

CPU-std $\sim 1\text{--}5$ s

CPU-high $\sim 6\text{--}20$ s

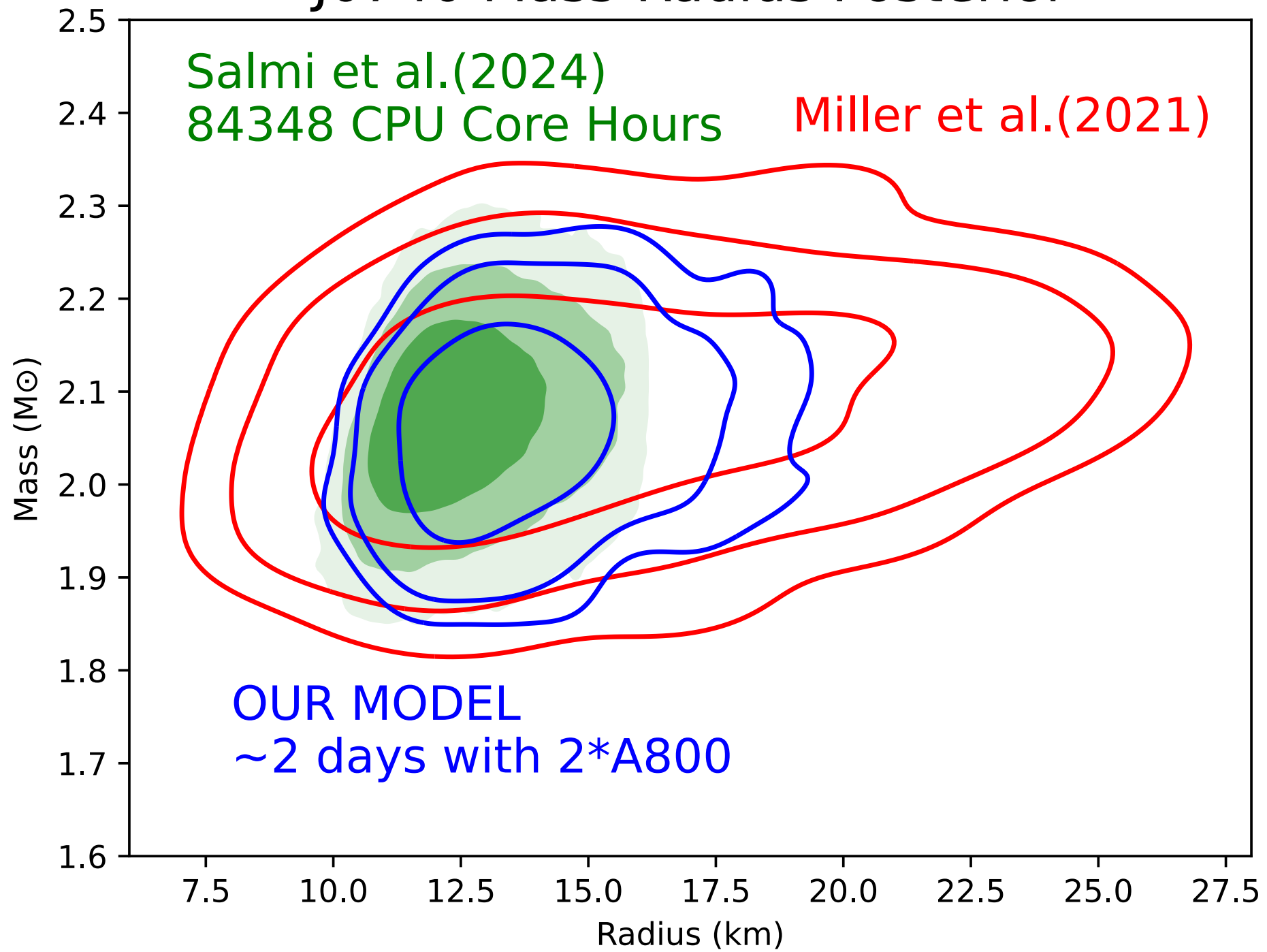
GPU-std $\sim 1\text{--}3$ ms

GPU-high $\sim 6\text{--}12$ ms

Test on CPU i7-13700KF

and GPU RTX4080

J0740 Mass-Radius Posterior



Welcome visit our GitHub repo:
https://github.com/zhoutz/gpu_ppm

The screenshot shows the GitHub repository page for `zhoutz/gpu_ppm`. The repository is licensed under MIT and contains four files: `p_devarshi24st.py`, `p_interp.py`, `p_oblate_geom.py`, and `p_os1.py`, all committed 10 months ago. The README describes the project as "gpu_ppm: GPU-Accelerated Pulsar / Neutron Star Pulse Profile Modeling".

gpu_ppm: GPU-Accelerated Pulsar / Neutron Star Pulse Profile Modeling

High-performance C++20 / CUDA code for generating phase-resolved pulse profiles and energy-phase counts for rotating neutron star surface emission with relativistic light bending, time delays, Doppler / aberration effects, oblate star shape, atmosphere beaming (NSX tables), and instrument response folding. Includes CPU reference implementations for validation and comparison.

This repository is intended to reproduce the results of our work ([arXiv:2510.07764](https://arxiv.org/abs/2510.07764)). Please consider citing the paper if you use this code in academic work.

Key Features

- CUDA implementations (`devarshi24st_gpu` , `os1_gpu`) with heavy shared memory use and warp-efficient interpolation / reduction.
- CPU counterparts (`devarshi24st_cpu` , `os1_cpu`) for correctness cross-checks and baseline timing.
- General relativistic lensing pre-tabulated (`tables/lensing/*`) with bilinear interpolation on the device.
- NSX hydrogen atmosphere model (`ext/model_data`) loaded once and cached in GPU global memory; cubic Lagrange and linear edge interpolation in energy / angle.
- HEALPix-like spherical tiling on the GPU for surface integration with dynamic active patch selection per ring.
- Time-of-flight corrections and rotational phase shifts (including oblate geometry corrections) handled per surface element.
- Instrument response folding (`tables/ism_inst/*`) to produce channel counts vs phase (simulating detector observation) on the GPU via a matrix multiply.

Contributors 2

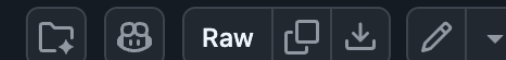
- zhoutz zhoutz
- ChunHuangPhy

Languages

Language	Percentage
C++	45.3%
Cuda	38.8%
Python	15.3%
Makefile	0.6%

Summary

- We developed a GPU-based simulation program to compute millisecond-pulsar pulse profile signals, achieving roughly a 1000× speedup compared to single-core CPU calculations.
- In future, we hope to explore more complex surface temperature distributions and consider more detailed physical processes.

Code **Blame** 400 lines (316 loc) · 15 KB

 Older ■■■■■■■■■■ Newer

 Contributors **6**

		366	
		367	<code>return erg / 4.135667662e-18</code>
		368	
		369	<code>cdef double eval_hot_Num2D_I(size_t THREAD,</code>
		370	<code>double E,</code>
		371	<code>double mu,</code>
		372	<code>const double *const VEC,</code>
2 years ago	Making X-PSI support Cython...	373	<code>void *const data) noexcept nogil:</code>
2 years ago	X-ray polarimetry - release v2....	374	<code># Arguments:</code>
		375	<code># E = photon energy in keV</code>
		376	<code># mu = cosine of ray zenith angle (i.e., angle to surface normal)</code>
		377	<code># VEC = variables such as temperature, effective gravity, ...</code>
		378	<code># data = numerical model data required for intensity evaluation</code>
		379	
		380	<code>cdef double I = eval_hot_Num2D(THREAD,E,mu,VEC,data)</code>
		381	
		382	<code>if I < 0.0:</code>
		383	<code>return 0.0</code>
		384	
		385	<code>return I</code>
		386	
		387	
		388	<code>cdef double eval_hot_Num2D_Q(size_t THREAD,</code>
		389	<code>double E,</code>
		390	<code>double mu,</code>
		391	<code>const double *const VEC,</code>
2 years ago	Making X-PSI support Cython...	392	<code>void *const data) noexcept nogil:</code>
2 years ago	X-ray polarimetry - release v2....	393	<code># Arguments:</code>
		394	<code># E = photon energy in keV</code>
		395	<code># mu = cosine of ray zenith angle (i.e., angle to surface normal)</code>
		396	<code># VEC = variables such as temperature, effective gravity, ...</code>
		397	<code># data = numerical model data required for intensity evaluation</code>
		398	
		399	<code>return eval_hot_Num2D(THREAD,E,mu,VEC,data)</code>

