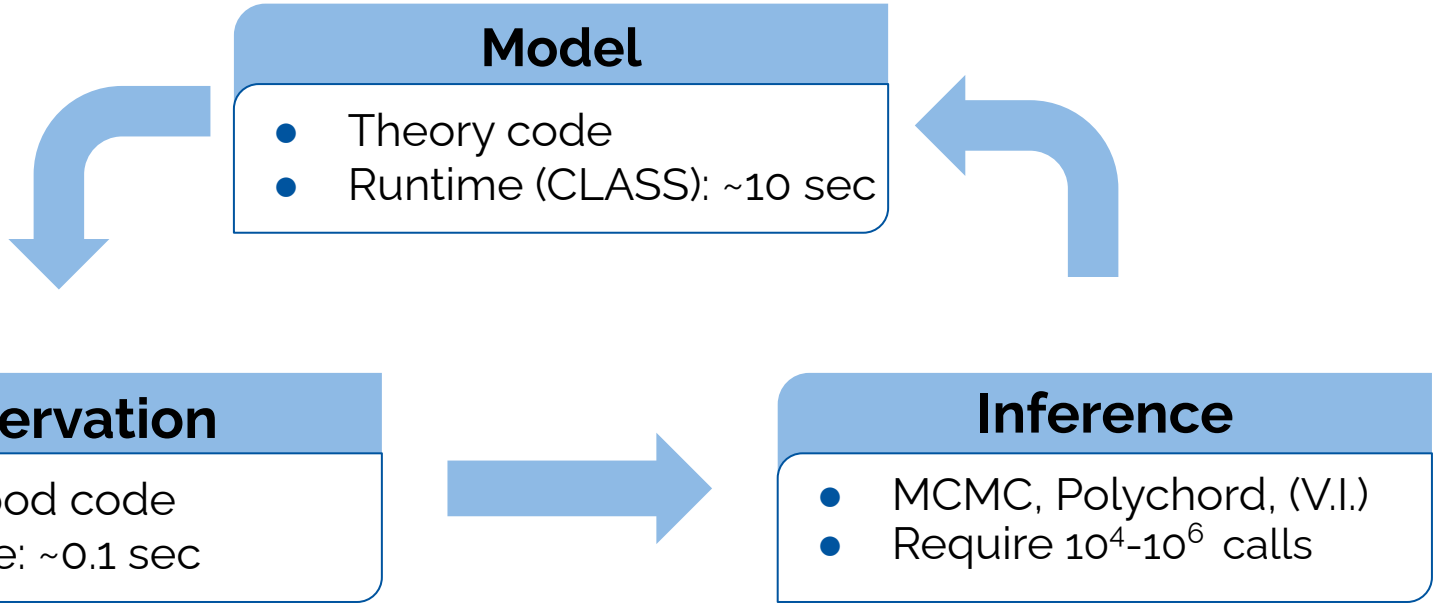
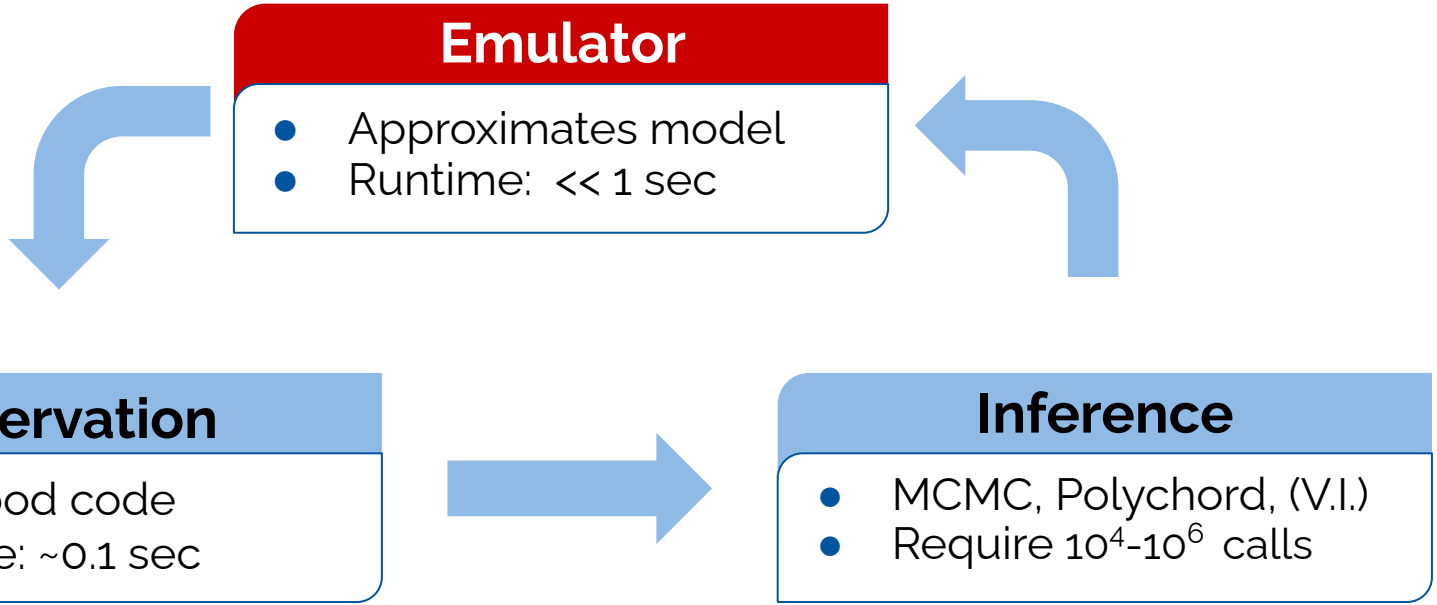
Reliable and resource preserving emulation for Bayesian model inference

(RWTH Aachen CLAIX-18 Computing Cluster)

The Inference Pipeline



The Inference Pipeline



The Emulator - Our Approach

Active learning

- gather training data during inference
- efficient data collection

Accuracy qualification

- do we have a sufficient number of samples?
- estimate accuracy

The Emulator - Our Approach

Active learning

- gather training data during inference
- efficient data collection

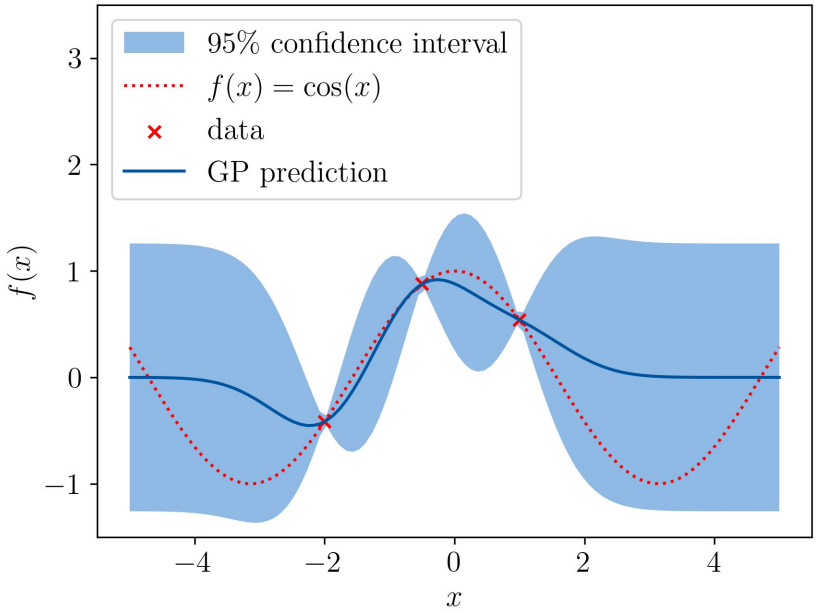
Accuracy qualification

- do we have a sufficient number of samples?
- estimate accuracy

Acquisition function

- add sample when error exceeds threshold
- retrain emulator each new sample

The Backbone - Gaussian Processes



Assume a dataset A

$$X_A \sim f(x)$$

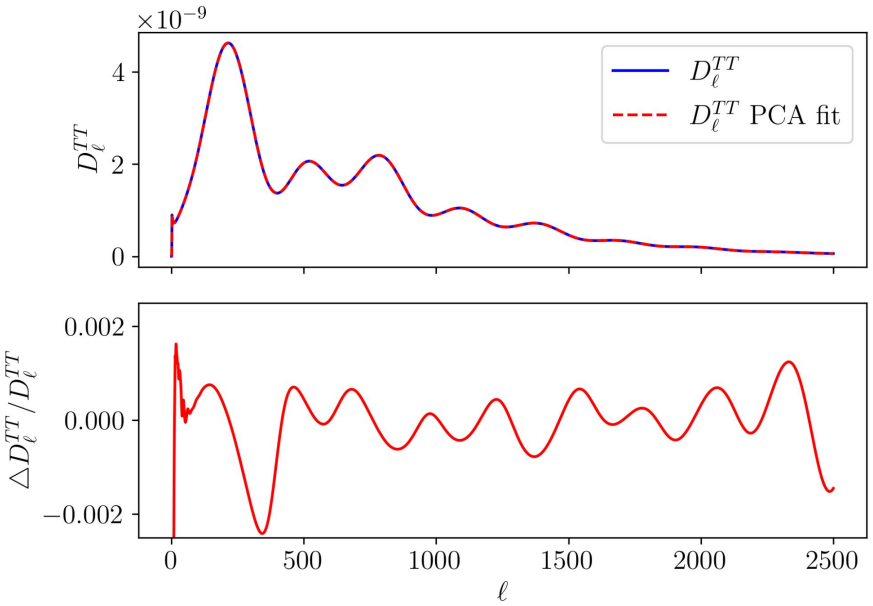
Predict new data assuming a Gaussian model

$$X_B \sim \mathcal{N}(\mu, \Sigma)$$

Condition the distribution by known dataset

$$\Sigma = \Sigma_{BB} - \Sigma_{BA}\Sigma_{AA}^{-1}\Sigma_{AB}$$

The Compression - PCA



Assume a dataset A of size N and dimension M

$$X \in \mathbb{R}^{(N \times M)}$$

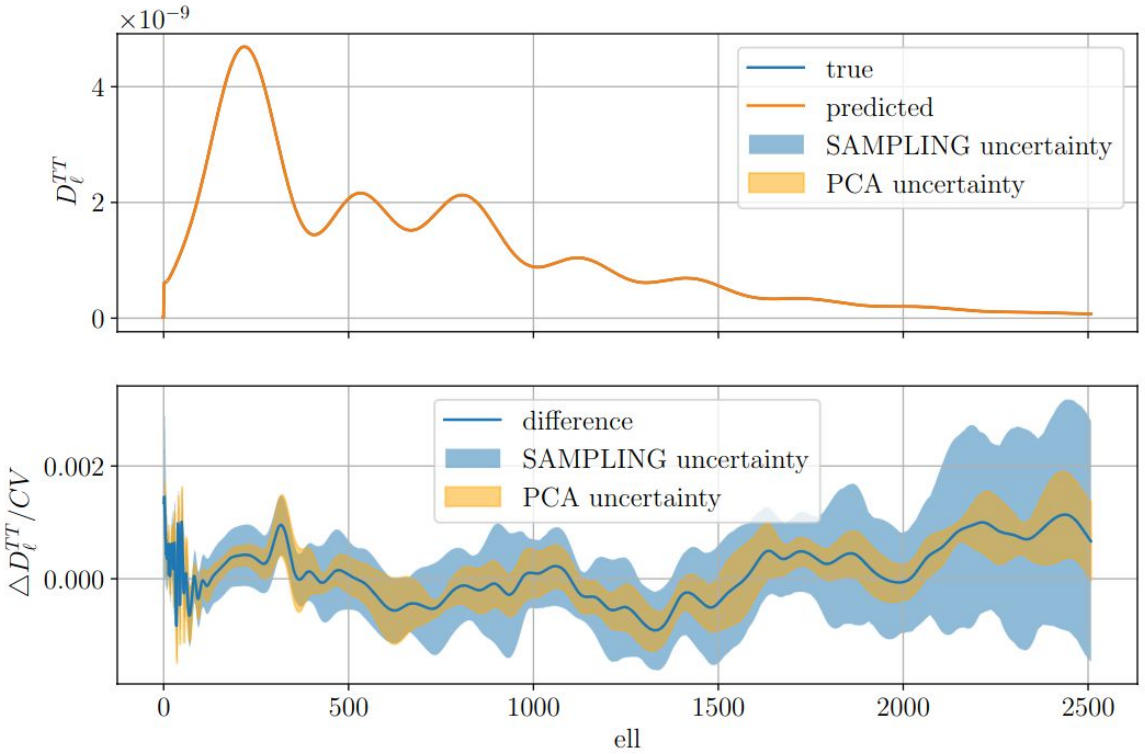
Find Eigenvalues and Eigenvectors

$$w_i = \lambda_i (X^T X) w_i$$

Transform data into Eigenspace of J largest components

$$\tilde{X}_{\mathrm{PCA}} = WX \quad with\ W = (w_0, ..., w_J)$$

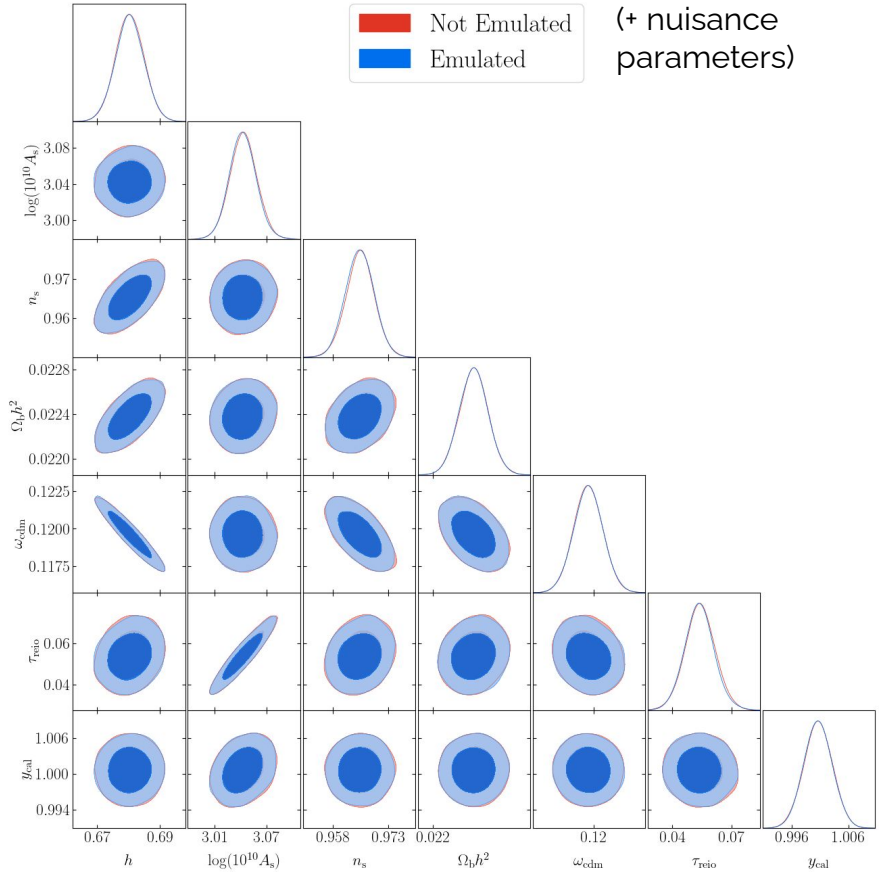
The Emulation - Uncertainty Aware!



Backtransform PCA + unc.

- uncor. PCA unc. gain **correlation**
- GP allows for easy sampling
- evaluating multiple samples with likelihood code
- provide loglike uncertainty

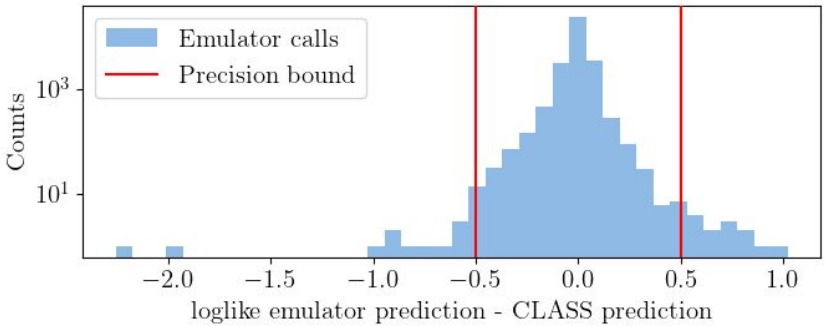
The Emulation - Does it work?



(+ nuisance parameters)

MCMC - Testcase

- LCDM on Planck TTTEEE + BAO
  - (6 cosm. + 21 nui.) parameters
- required `CLASS` calls:
  - Emulated: **126**
  - Not emulated: **58698**
- overall speedup **~10** for LCDM
- goal for precision (0.5) achieved
- expect increase in performance



The Takeaway

- combination of Gaussian processes and PCA allow for **efficient, fast, flexible** and **accurate** emulators
- accuracy estimate allows for **reliable** performance
- speed-up of **~10** for MCMC of LCDM+Planck, possibly more for computationally more expensive models!
- paper and full code (+MontePython) release in preparation
- test current code as easy to use cobaya plugin: github.com/svenguenther/cobaya

BONUS

PCA base vector