



RABBIT

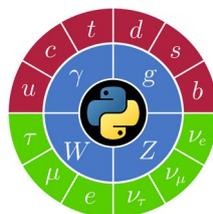
Efficient binned profile maximum likelihood estimation with Rabbit

Josh Bendavid, David Walter

October 28, 2025

PyHEP 2025

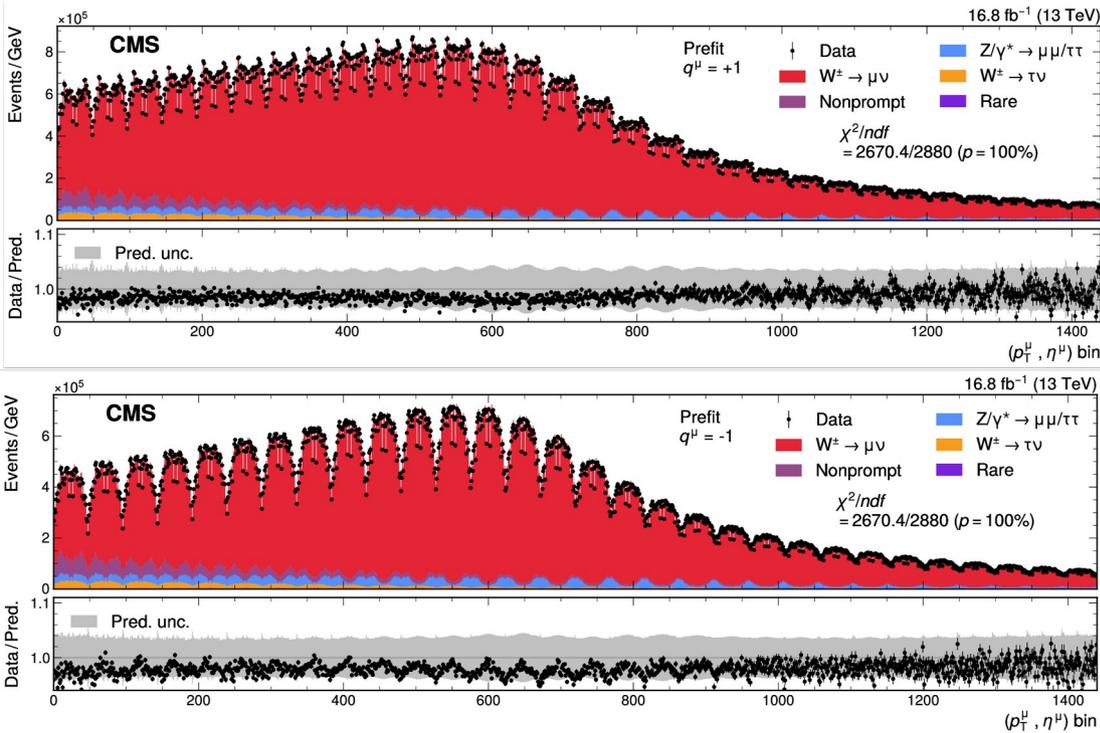
Rapid **A**utomatic **B**in **B**ased **I**nference **T**ool



Complexity

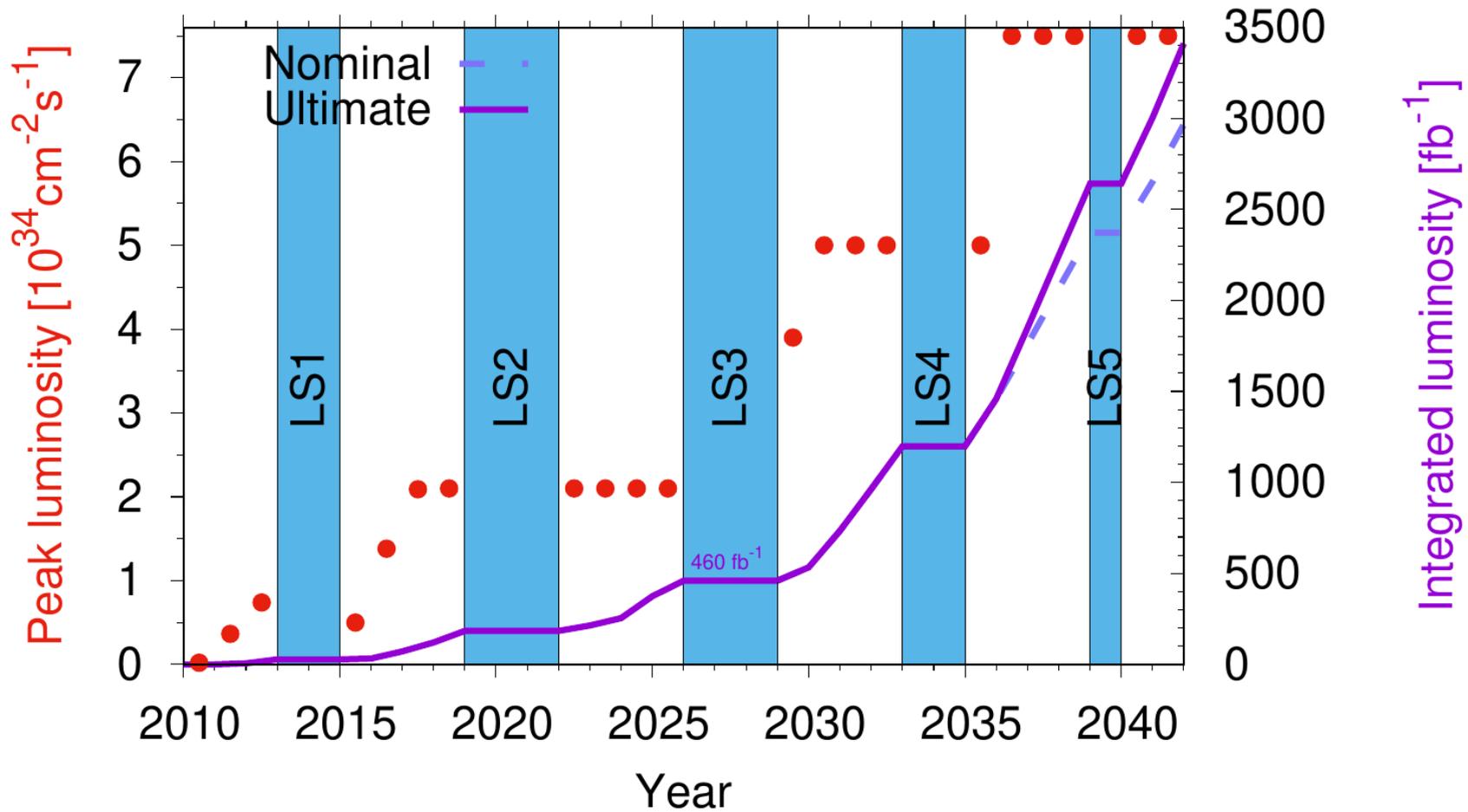
CMS extracts m_W from a template fit to a 3D histogram with 2,880 bins and $\sim 5,000$ systematic uncertainties

Only $\sim 10\%$ of Run 2 dataset



Systematic uncertainties	W-like m_Z	m_W
Muon efficiency	3127	3658
Muon eff. veto	—	531
Muon eff. syst.	343	
Muon eff. stat.	2784	
Nonprompt-muon background	—	387
Prompt-muon background	2	3
Muon momentum scale	314	
L1 prefiring	14	
Integrated luminosity	1	
PDF (CT18Z)	60	
Angular coefficients	177	353
W MINNLO _{PS} μ_F, μ_R	—	176
Z MINNLO _{PS} μ_F, μ_R	176	
PYTHIA shower k_T	1	
p_T^V modeling	22	32
Nonperturbative	4	10
Perturbative	4	8
Theory nuisance parameters	10	
c, b quark mass	4	
Higher-order EW	6	7
Z boson width	1	
Z boson mass	—	1
W boson width	—	1
$\sin^2 \theta_W$	1	
Total	3725	4833

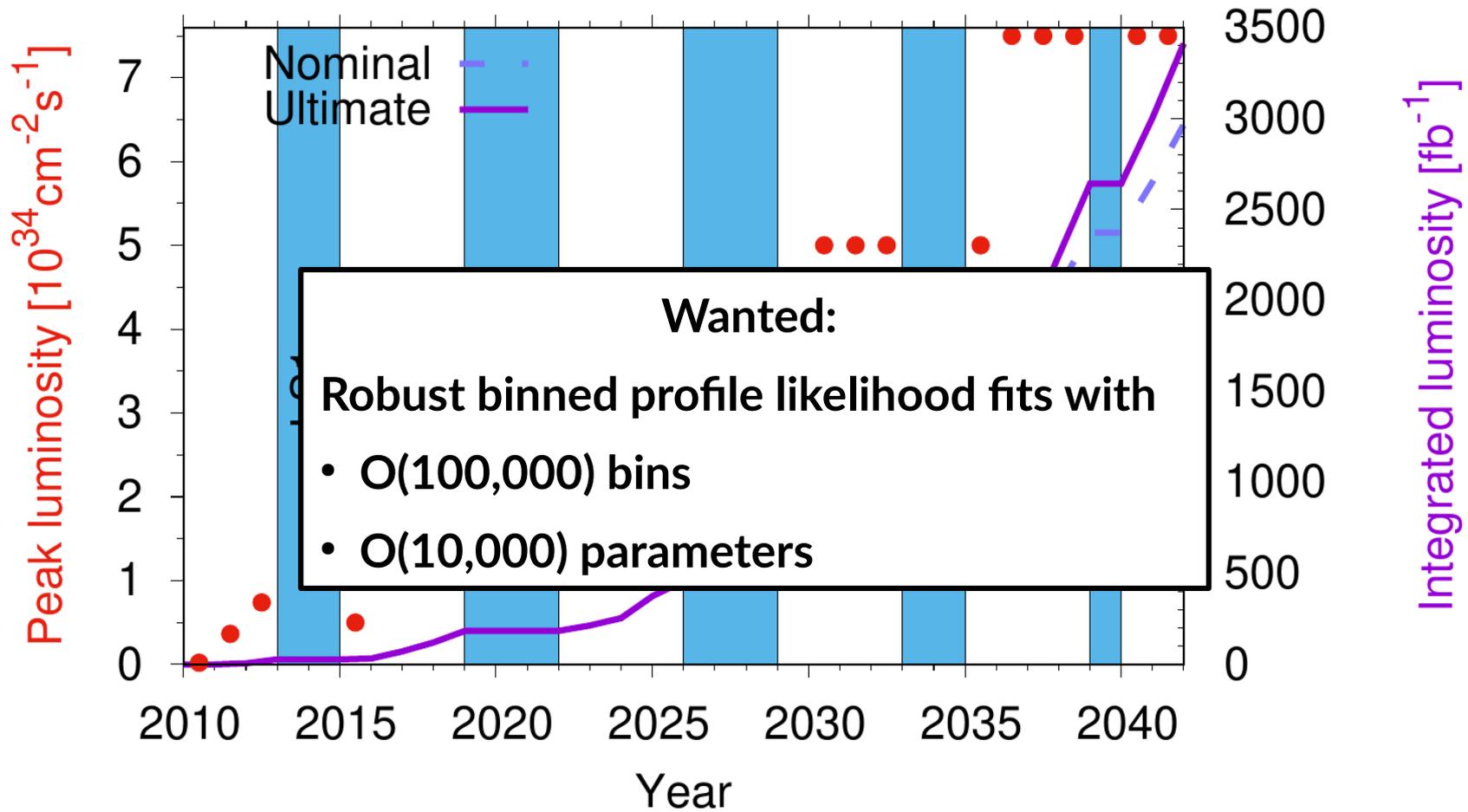
There is a lot of data, and we will get more in HL-LHC



Getting most out of the data is technically challenging

Precision measurements need extreme control over systematic effects

There is a lot of data, and we will get more in HL-LHC



Getting most out of the data is technically challenging

Precision measurements need extreme control over systematic effects



RABBIT

Prepare

- Construct model from histograms

Execute

- Minimize binned profile likelihood function

Evaluate

- Prefit & postfit plots
- Likelihood scans
- Uncertainty breakdown



RABBIT

Prepare

- Construct model from histograms

Execute

- Minimize binned profile likelihood function

Evaluate

- Prefit & postfit plots
- Likelihood scans
- Uncertainty breakdown

Improved performance due to

- Automatic differentiation using tensorflow
- Just in time compilation (JIT)
- Vectorized operations
- Advanced minimization strategy

Prepare – model

Python interface to histograms supporting UHI

Different channels (e.g. different event selection) can be added

Everything gets flattened and concatenated in the fit

Channels and histogram axes stored in meta data

- Reshape postfit distribution, etc.

Prediction stored in big 3D tensor

- $\#bins \cdot \#processes \cdot \#systematics$

```
writer = tensorwriter.TensorWriter(
    sparse=args.sparse,
    systematic_type=args.systematicType,
)

writer.add_channel(h1_data.axes, "ch0")

writer.add_data(h1_data, "ch0")

writer.add_pseudodata(h1_pseudo, "up", "ch0")

writer.add_data_covariance(cov)

writer.add_process(h1_sig, "sig", "ch0", signal=True)
writer.add_process(h1_bkg, "bkg", "ch0")

writer.add_systematic(
    h1_bkg_syst0,
    "slope_background",
    "bkg",
    "ch0",
    symmetrize="average"
    groups=["slopes", "slopes_background"],
)

writer.write(outfolder=directory, outfilename=filename)
```

Serialization

Output file written via h5py

- Efficient and flexible serialization of python objects in HDF5
- Supports boost histogram and hist objects

Custom Pickler/Unpickler classes to read/write objects inside HDF5 groups

- Large numerical buffers stored out-of-band (zero copy), compressed (LZ4)
- Pickled dictionary stores object proxies (e.g. histograms), and metadata
- On demand lazy loading of full object
- Works for all python objects which can be pickled
- Custom support for histograms to ensure zero-copy

Implemented in [wums/ioutils.py](#) package

- For standalone use: `pip install wums[pickling]`

Considering interfacing UHI serialization ([talk](#) yesterday from Henry Schreiner)

Likelihood construction

Poisson event counts

$$-\ln(\mathcal{L}_P) = \sum_{i=1}^{N^{\text{bins}}} n_i - n_i^{\text{obs}} \ln(n_i)$$

Multiplicative scaling

$$n_i = \sum_{j=1}^{N^{\text{proc}}} \mu_j \beta_i n_{i,j}^0 \prod_{k=1}^{N^{\text{syst}}} \kappa_{\pm,i,j,k}^{\theta_k} \quad \text{with} \quad \kappa_{\pm,i,j,k} = \frac{n_{i,j}^0 + v_{\pm,i,j,k}}{n_{i,j}^0}$$

Nuisances: log-normal

$$-\ln(\mathcal{L}_C) = \frac{1}{2} \sum_{k=1}^{N^{\text{syst}}} (\theta_k - \theta_k^0)^2$$

Barlow-Beeston light
uncertainties: Gamma

$$-\ln(\mathcal{L}_\Gamma) = \sum_{i=1}^{N^{\text{bins}}} k_i^{\text{stat}} \beta_i - k_i^{\text{stat}} \beta_i^0 \ln(\beta_i)$$

- Exact analytic solutions as explained in [ref.](#)
- New: Barlow-Beeston full with normal constrained uncertainties

Linearization of the likelihood

- Chi2

$$-\ln(\mathcal{L}_{\chi^2}) = \frac{1}{2} (\mathbf{n} - \mathbf{n}^{\text{obs}})^T \mathbf{C}^{-1} (\mathbf{n} - \mathbf{n}^{\text{obs}})$$

- Additive scaling

$$n_i = \sum_{j=1}^{N^{\text{proc}}} \left(\mu_j n_{i,j}^0 + \sum_{k=1}^{N^{\text{sys}}} v_{i,j,k} \theta_k \right) + \Delta n_i^0 \beta_i$$

- Nuisances: Normal

$$-\ln(\mathcal{L}_{\mathbf{C}}) = \frac{1}{2} \sum_{k=1}^{N^{\text{sys}}} (\theta_k - \theta_k^0)^2$$

- Barlow-Beeston light uncertainties: Normal

$$-\ln(\mathcal{L}_{\mathbf{N}}) = \frac{1}{2} \sum_{i=1}^{N^{\text{bins}}} (\beta_i - \beta_i^0)^2$$

Fully quadratic likelihood can be solved by computing the Hessian (\mathbf{H})

- MLE of parameters ($\bar{\mathbf{x}}$) from single step in direction of Jacobian (\mathbf{J}): $\bar{\mathbf{x}} = \mathbf{H}^{-1} \mathbf{J}$

Further simplification by absorbing uncertainties into covariance matrix \mathbf{C}

- For single POI: fully analytic solution through matrix inversion

Various combinations supported: e.g. chi2 likelihood with multiplicative scaling

Execute – Robust minimization

Tensorflow loss/grad/hess computations interfaced to SciPy minimizer

Trust region minimizer

- Looks for a minimum within a region where the function approximation is trusted

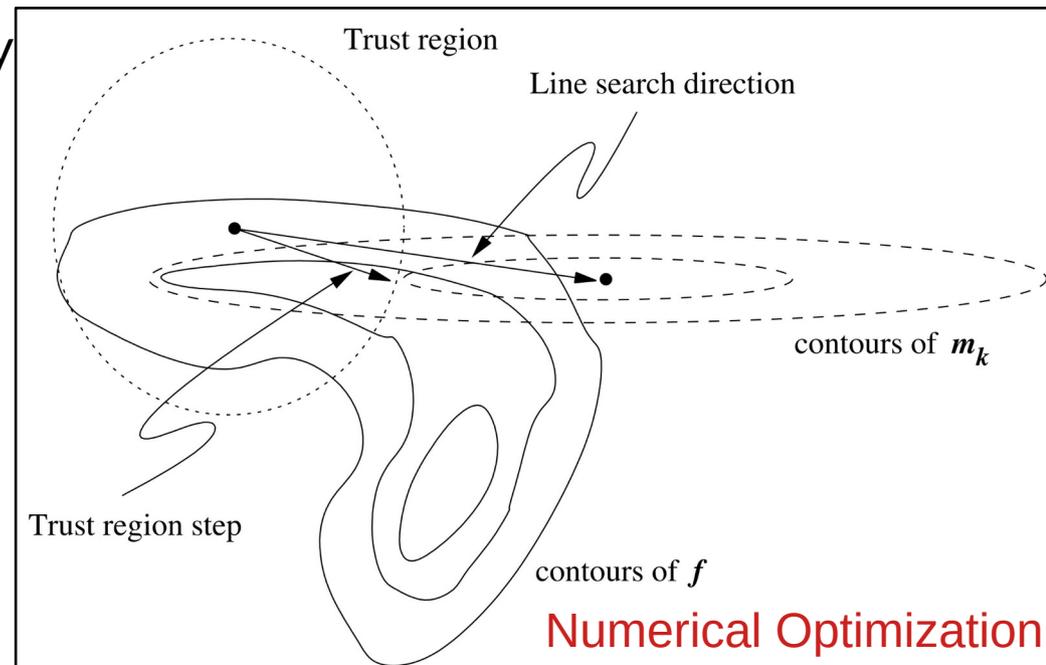
Nominal: “trust-krylov”

- Find minimum in trust region (subproblem) in Krylov subspace
 - Approximate dominant Hessian eigenvectors
 - Only requires hessian vector products (HVP)
 - Reduced memory consumption, improved computational efficiency

Backup: “trust-exact”

- Use Hessian to solve subproblem

Quantify final solution based on Hessian estimated distance to minimum



Evaluate – physics models

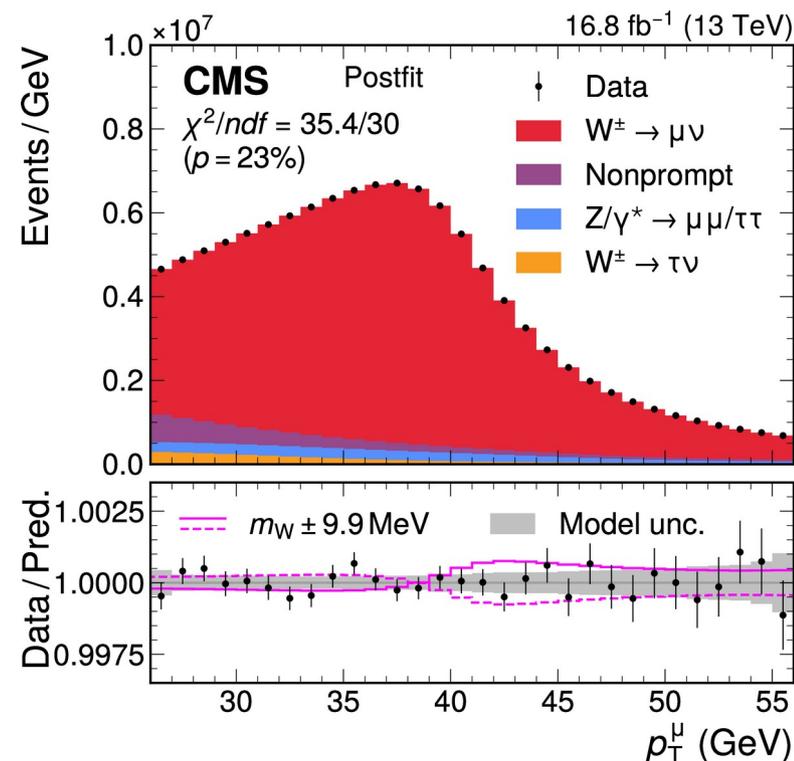
Apply transformation of parameters and observables (histogram bins)

- Error propagation via Gaussian approximation (Jacobian) → use autoDiff

Basic models implemented

- **Project** to lower dimensional distribution
- **Ratio** between channels, processes, or bins
- ...

Modular: Custom physics models may be added



Evaluate – physics models & masked channels

Apply transformation of parameters and observables (histogram bins)

- Error propagation via Gaussian approximation (Jacobian) → use autoDiff

Basic models implemented

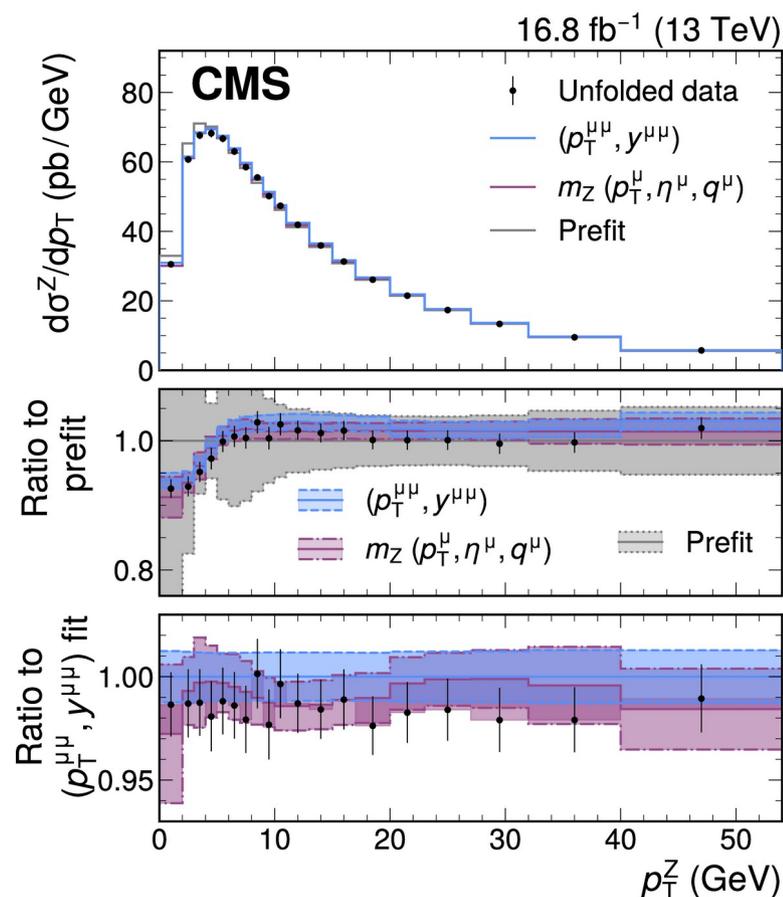
- **Project** to lower dimensional distribution
- **Ratio** between channels, processes, or bins
- ...

Modular: Custom physics models may be added

“Masked channels” to provide auxiliary data

- E.g. generator level distribution for unfolding
- Track full dependency of cross section

$$\hat{\sigma} = \hat{\mu}\sigma(\hat{\theta})$$



Nuisance parameter impacts

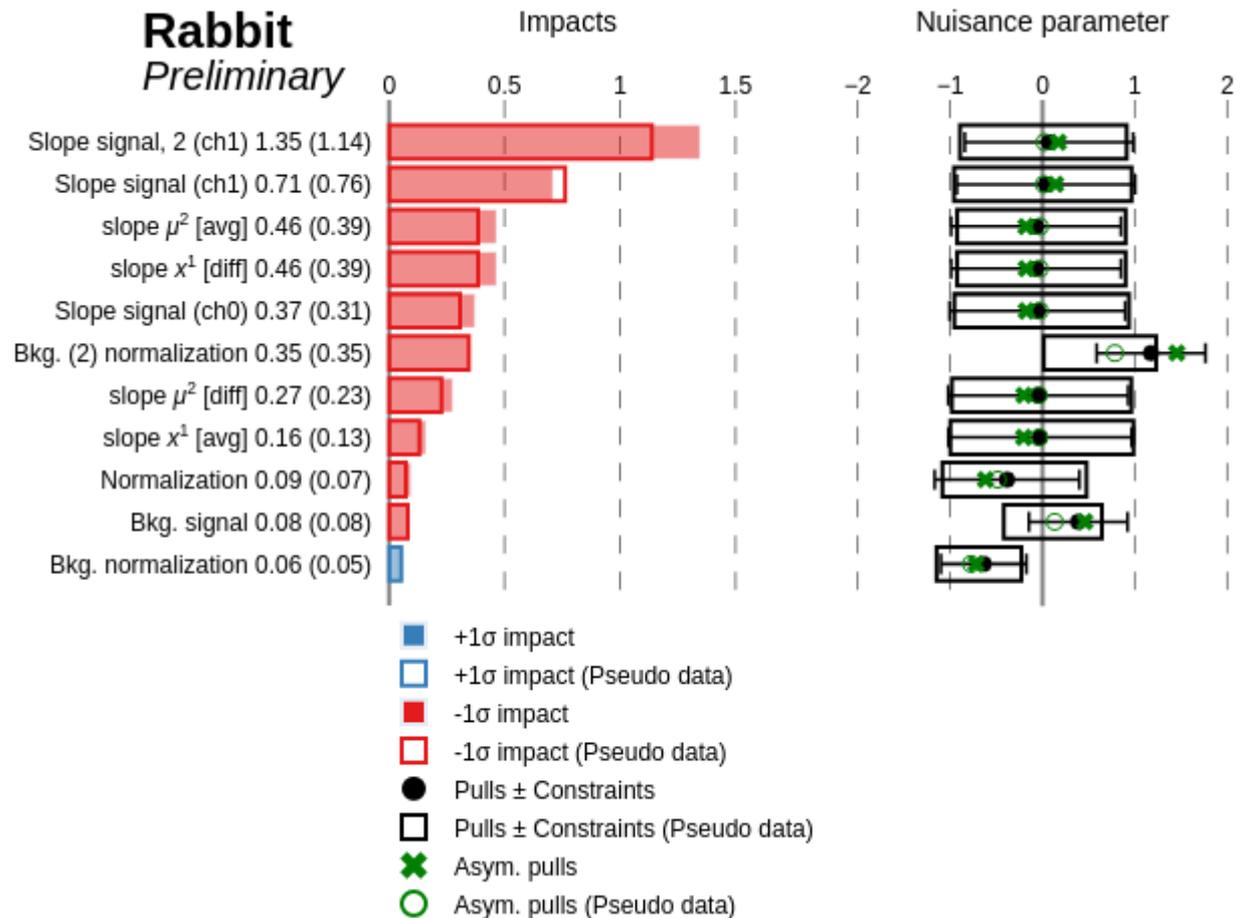
Gaussian approximation of nuisance parameter impacts

- Traditional:

$$\Delta_i x_j = \frac{C_{ij}}{\sqrt{C_{ii}}}$$

- Global (ref): Shift parameter profit value by $\pm 1\sigma$ (constrained minimum)
 - Also done via autodiff

Also support nuisance impact parameter groups



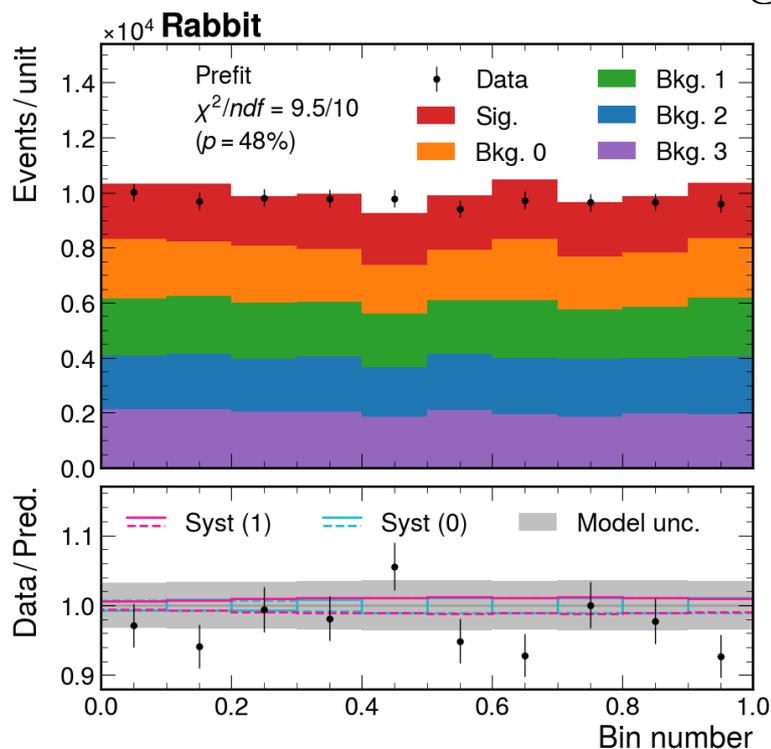
Benchmarking

A synthetic toy model

Construct a synthetic model with variable number of bins and systematic uncertainties that reflects properties of actual measurements

- 1 signal & 4 background processes, all flat and 20% each
- Shape uncertainties from

$$w_{j,k}(x_i) = 1 + \underbrace{\delta_{j,k}}_{\text{magnitude}} \cdot \underbrace{\sin(2\pi(f_k x_i + \phi_k))}_{\text{basis function}}$$



x_i : bin center of bin i

j : process

k : systematic uncertainty

$\delta \in \mathcal{N}(0, 0.02)$ within ± 0.1

phase $\phi \in \mathcal{U}(0, 1)$

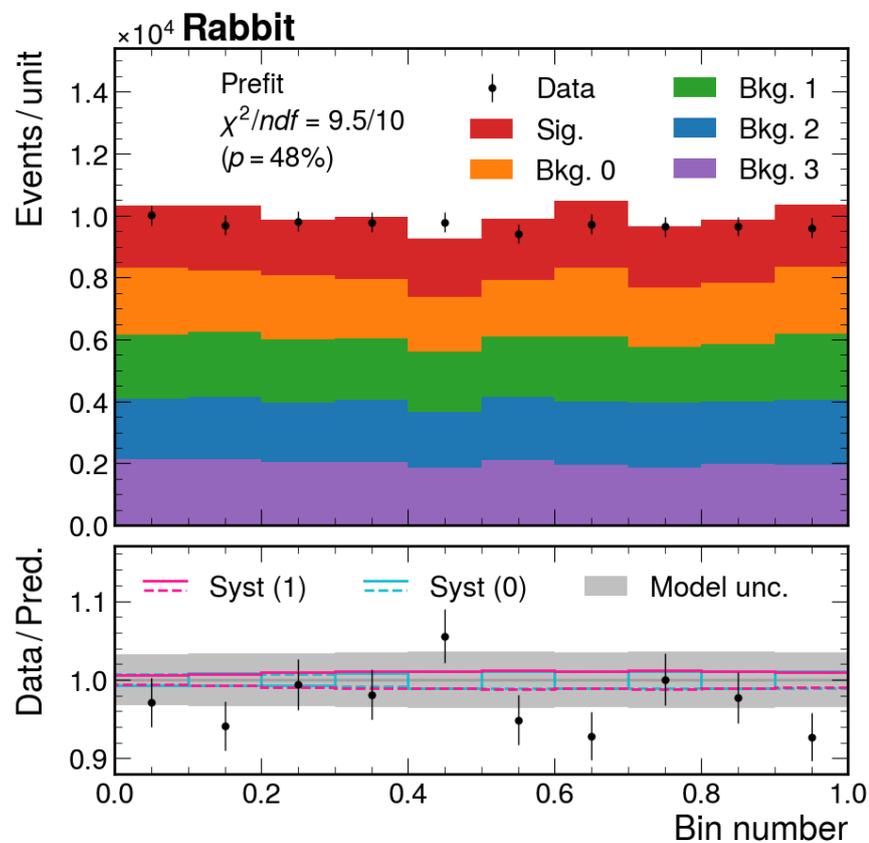
frequency $f \in \mathcal{U}(1/4, N^{\text{bins}})$

A synthetic toy model

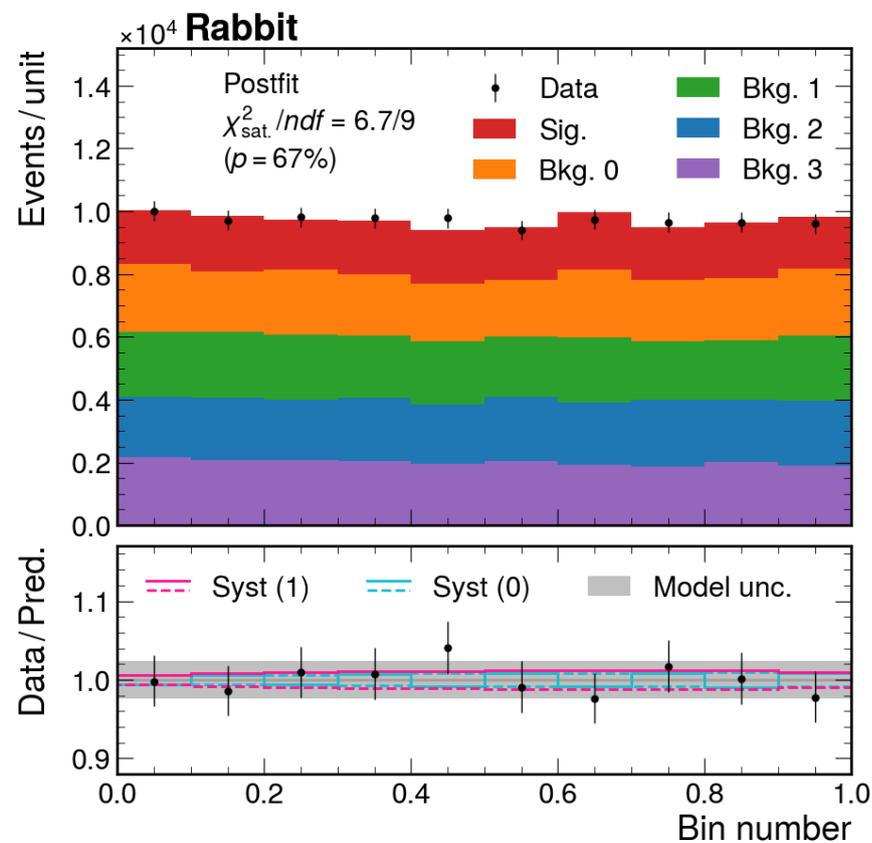
Pseudo data generated under Bayesian statistics

- Randomize signal strength $\mu \in \mathcal{N}(0, 0.05)$ within $[0.5, 1.5]$
- Randomize $\theta \in \mathcal{N}(0, 1)$
- Randomize bin content by Poisson with mean of final prediction

Independently randomize model prediction by Poisson (emulate bin by bin stat)



Fit



Performance

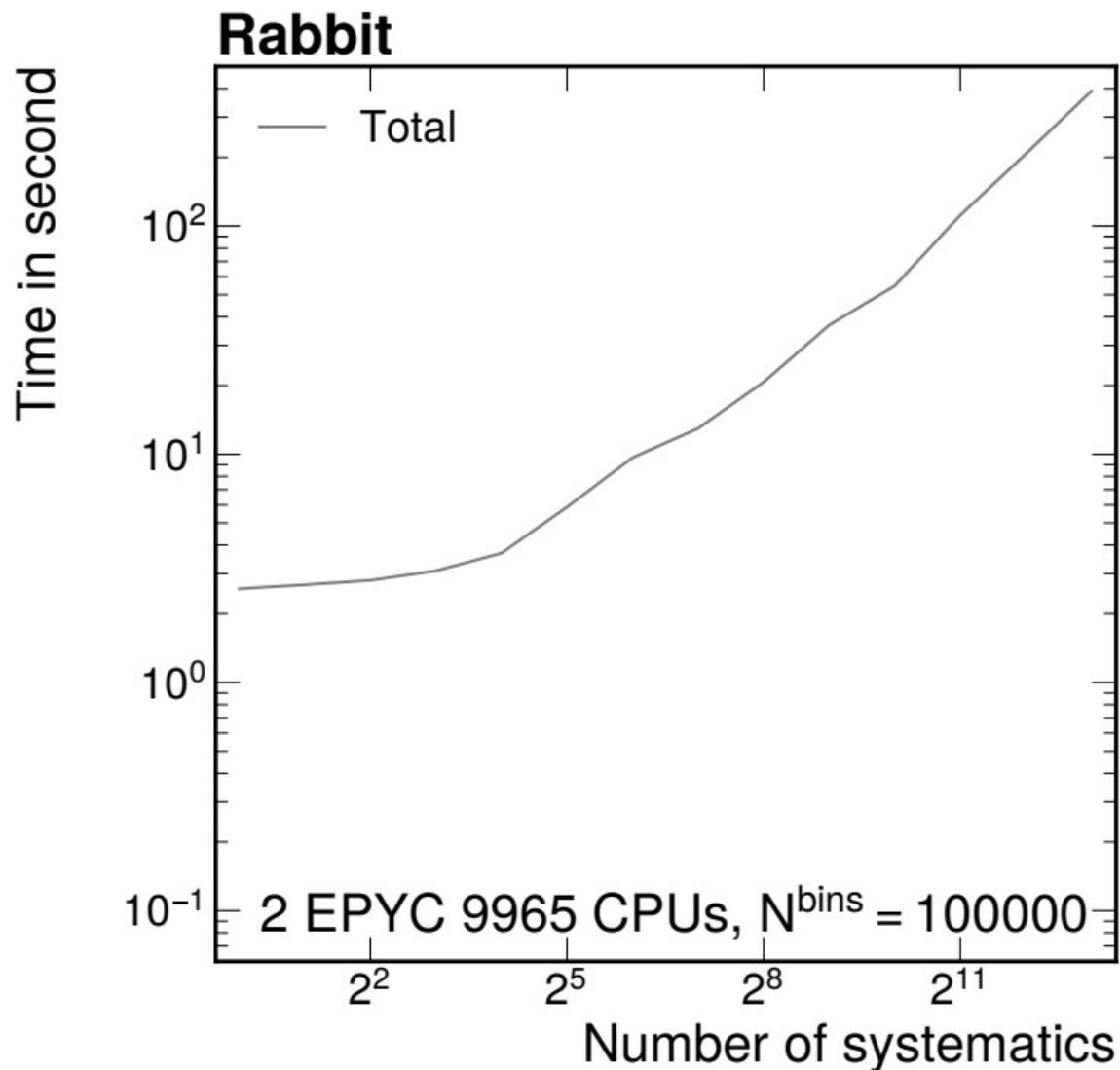
Tests performed on **SubMIT**
analysis facility



- Server with 2 EPYC 9965 CPUs with 784 CPU threads, 1.5TB memory
- Input data on fast NVME SSDs

Time needed to perform the fit and extract parameter uncertainties from inverse Hesse matrix

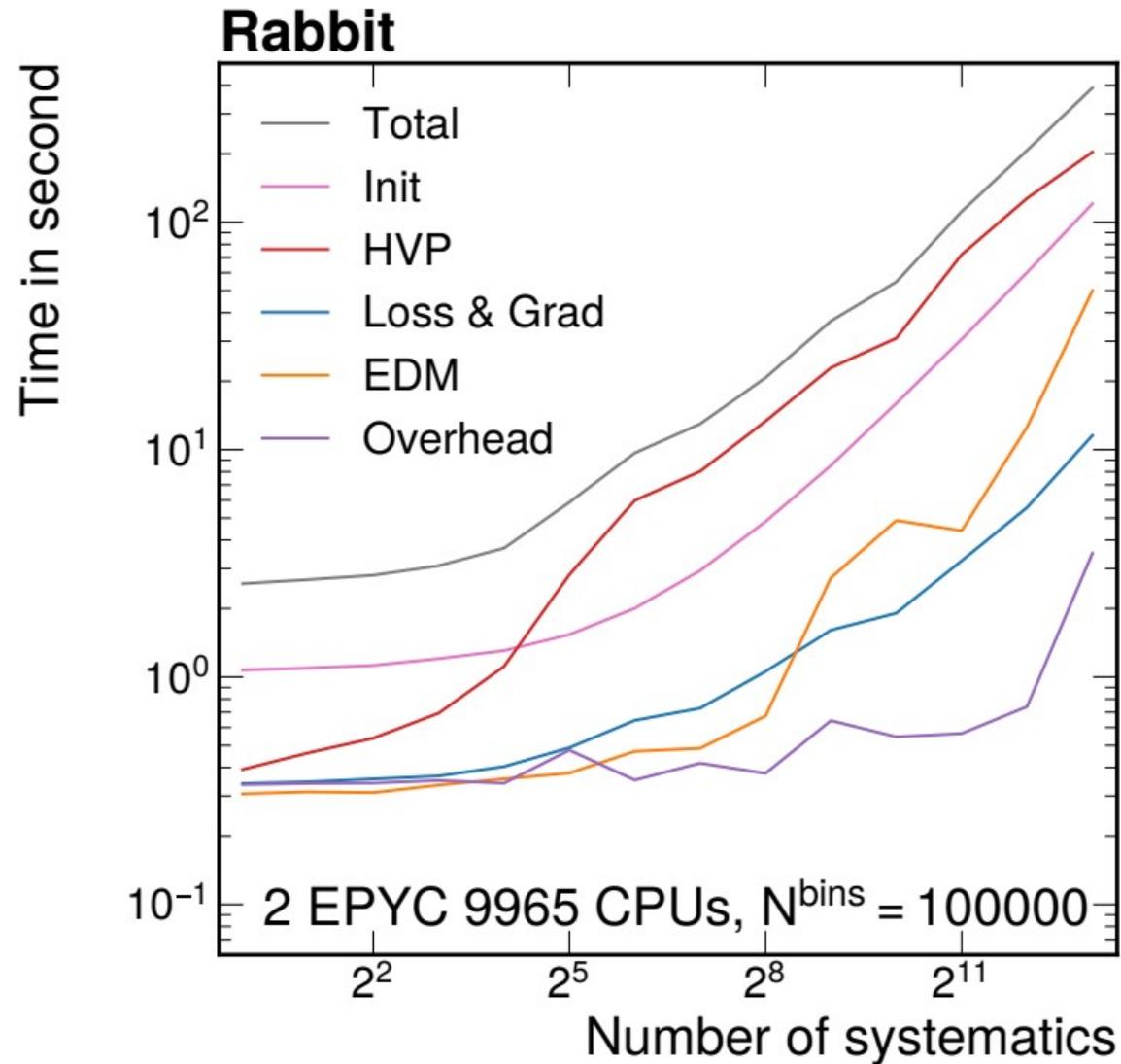
We can fit a model with 100,000 bins and 8,192 systematics in under 10 minutes!



Performance

Time to minimize dominated by

- **HVP: Hessian vector product to solve subproblem**
- **Init: Data loading/ initialization**
- **EDM: Estimated distance to minimum from hessian for fit quality assessment**
- **Loss and gradient calculation**



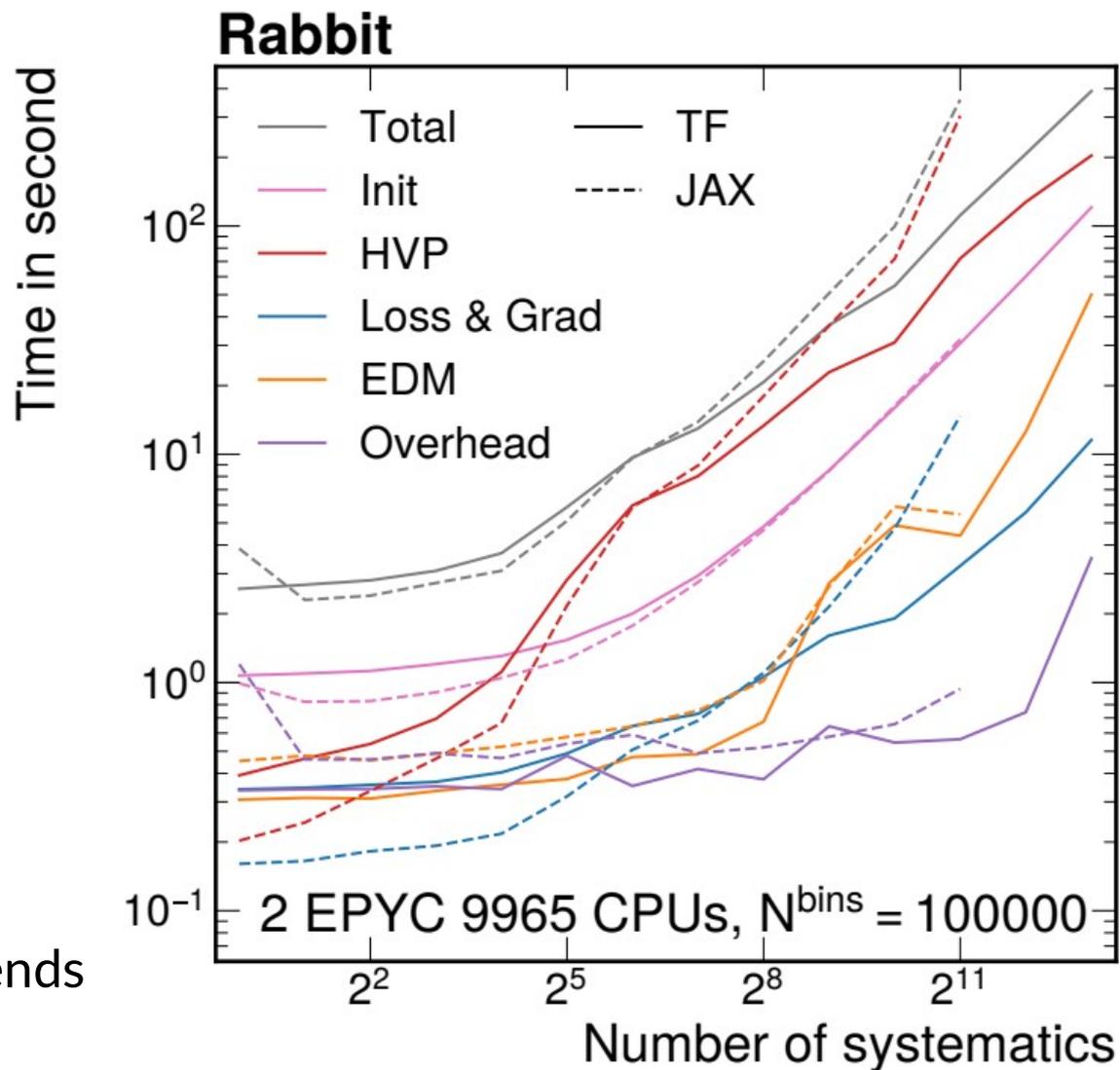
Performance

Time to minimize dominated by

- **HVP: Hessian vector product to solve subproblem**
- **Init: Data loading/ initialization**
- **EDM: Estimated distance to minimum from hessian for fit quality assessment**
- **Loss and gradient calculation**

Alternative implementation with JAX

- Similar performance
- No need to support multiple backends



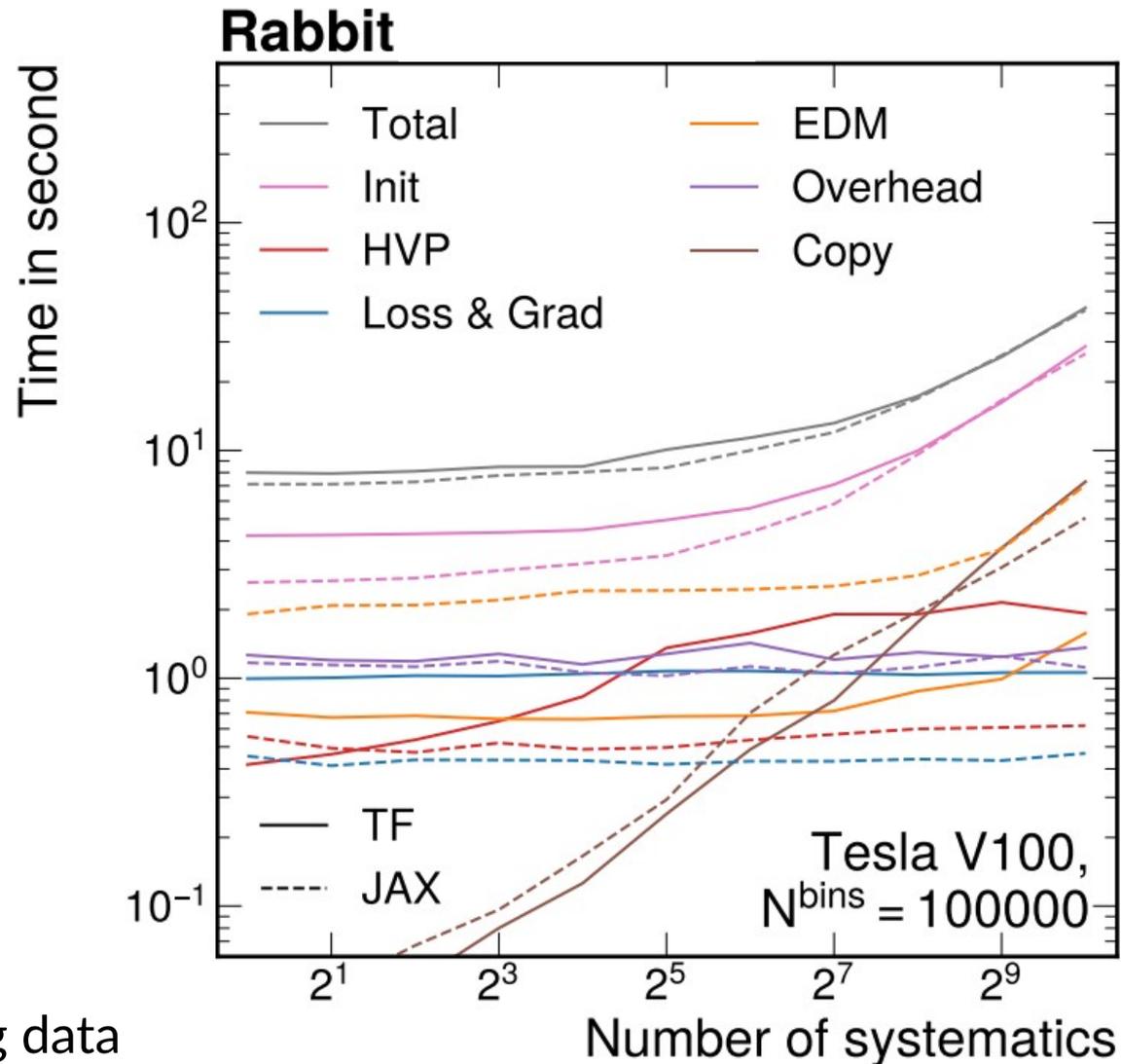
Performance: GPU

Time to minimize dominated by

- **Init: Data loading/ initialization**
- **EDM: Estimated distance to minimum from hessian for fit quality assessment**
- **Copy: Data from CPU \leftrightarrow GPU**
- **HVP: Hessian vector product to solve subproblem**
- **Loss and gradient calculation**

Can be run on GPU

- SciPy minimizer run on CPU leads to small overhead for moving data
- GPU runs out of memory before copy becomes dominant
- Can be solved by TF/JAX based minimizer in the future

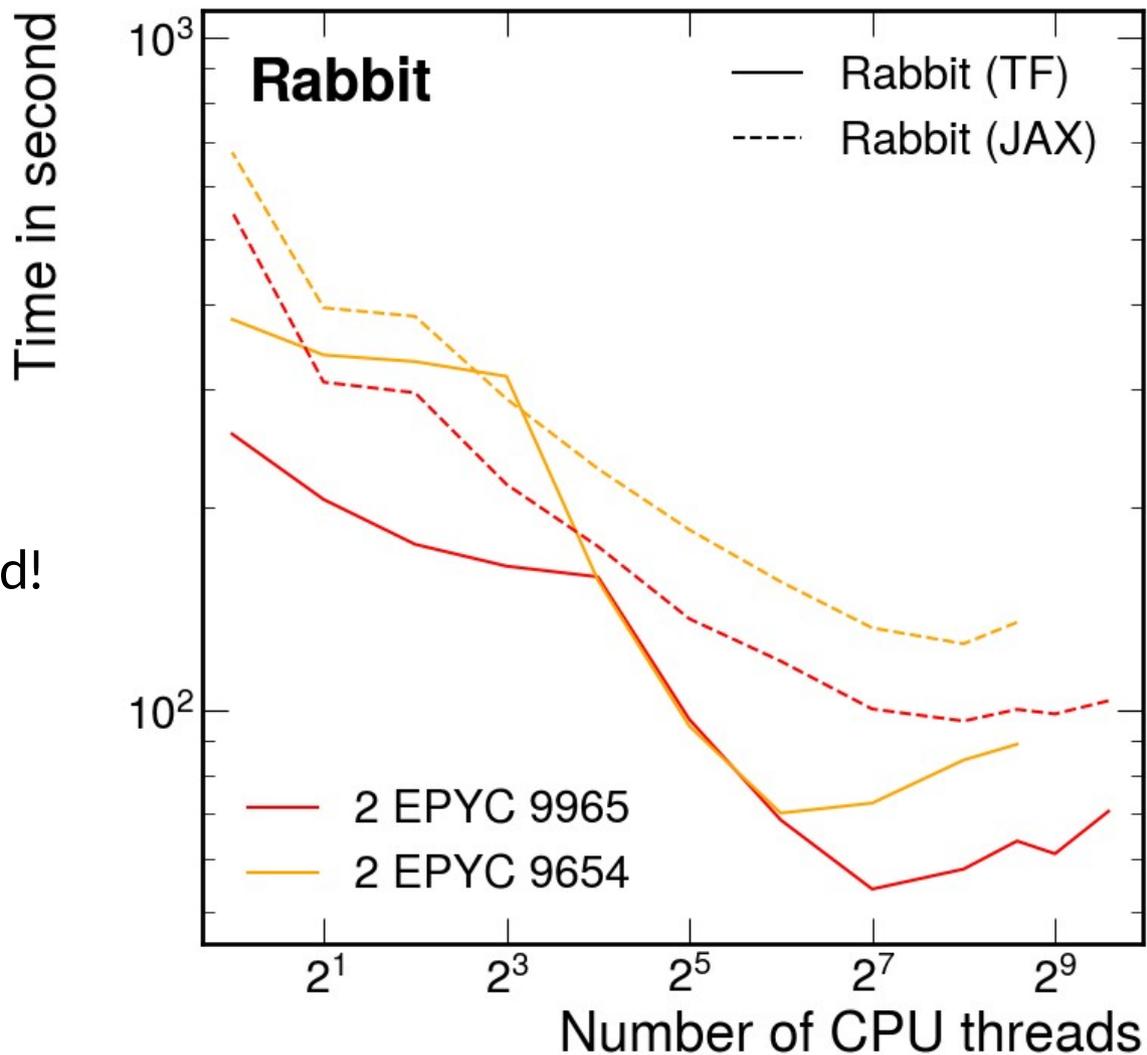


CPU thread scaling

Steering CPU threads
via cgroups v2

Testing model with 100,000 bins
and 1024 systematics

- Saturation at ~64 threads
- Rabbit can be run on a single thread!

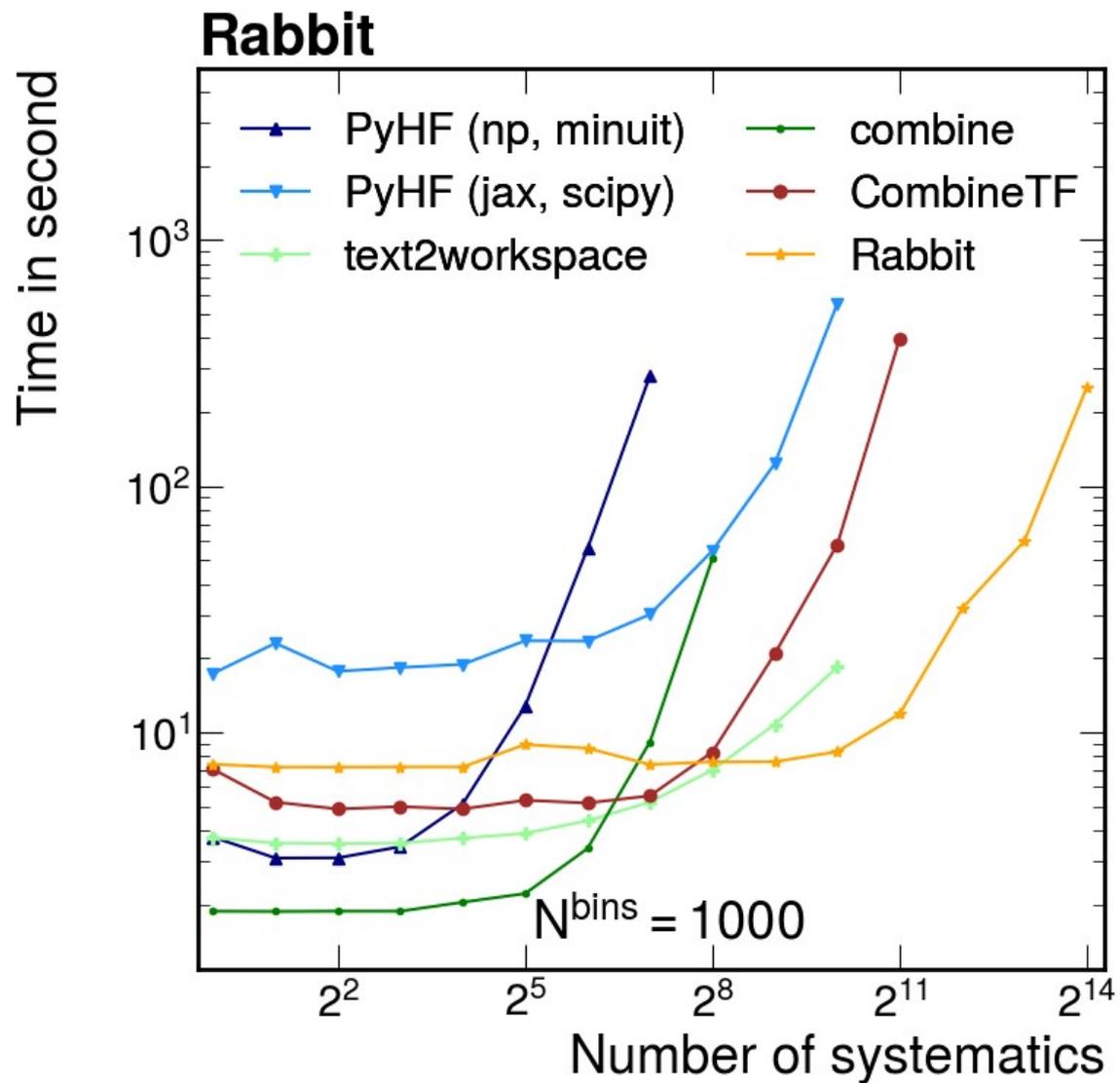


Comparison with other tools

Verified that all tools yield same likelihood values at minimum

Rabbit has increased initialization time due to JITing

- But can fit more complex models



Comparison with other tools

Verified that all tools yield same likelihood values at minimum

Rabbit has increased initialization time due to JITing

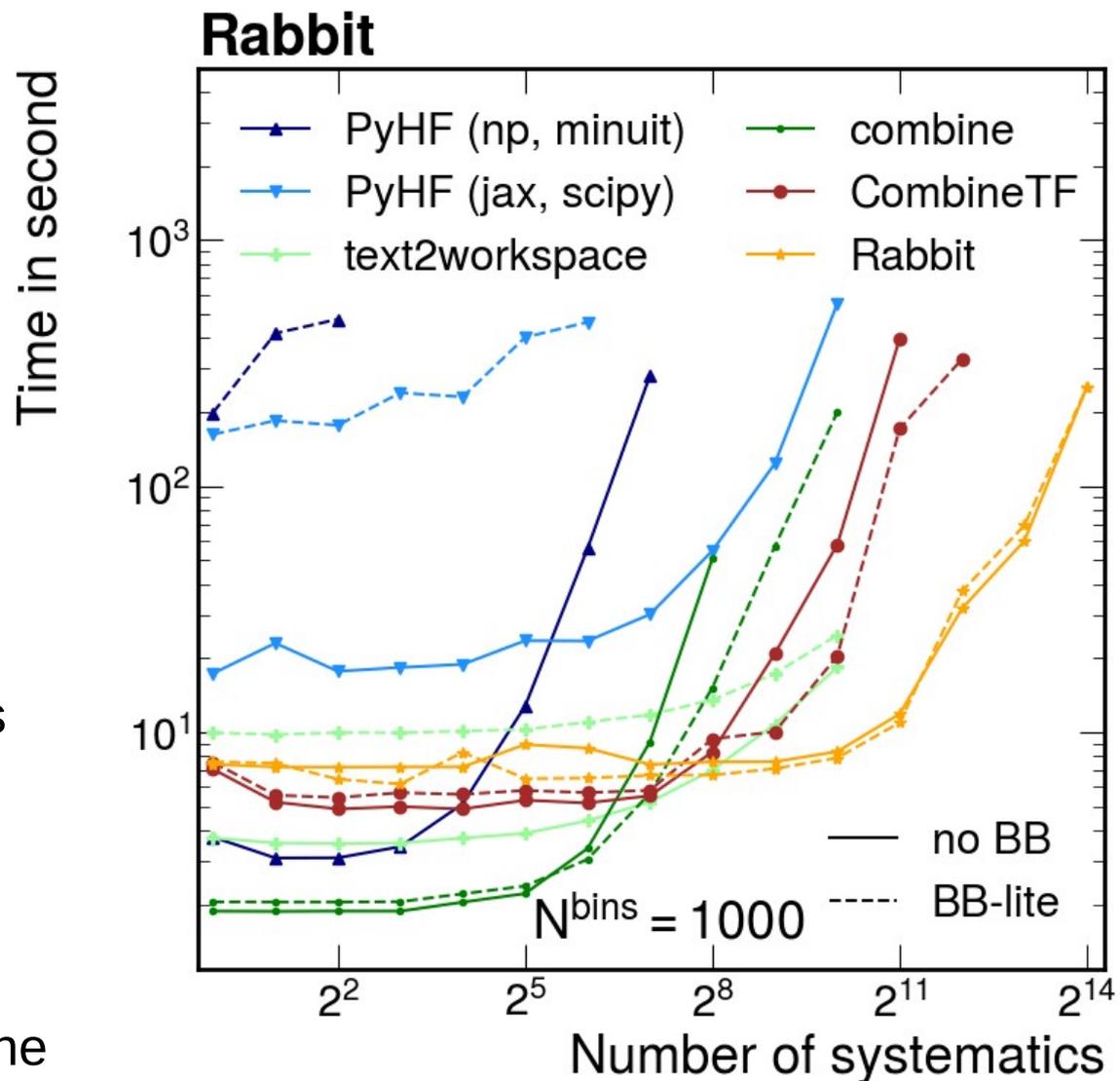
- But can fit more complex models

PyHF does not implement analytic solution to bin by bin stat uncertainties

- Explicit parameters lead to large increase in runtime

Fits with other tools similarly fast

Increased initialization time for Combine



Comparison with other tools

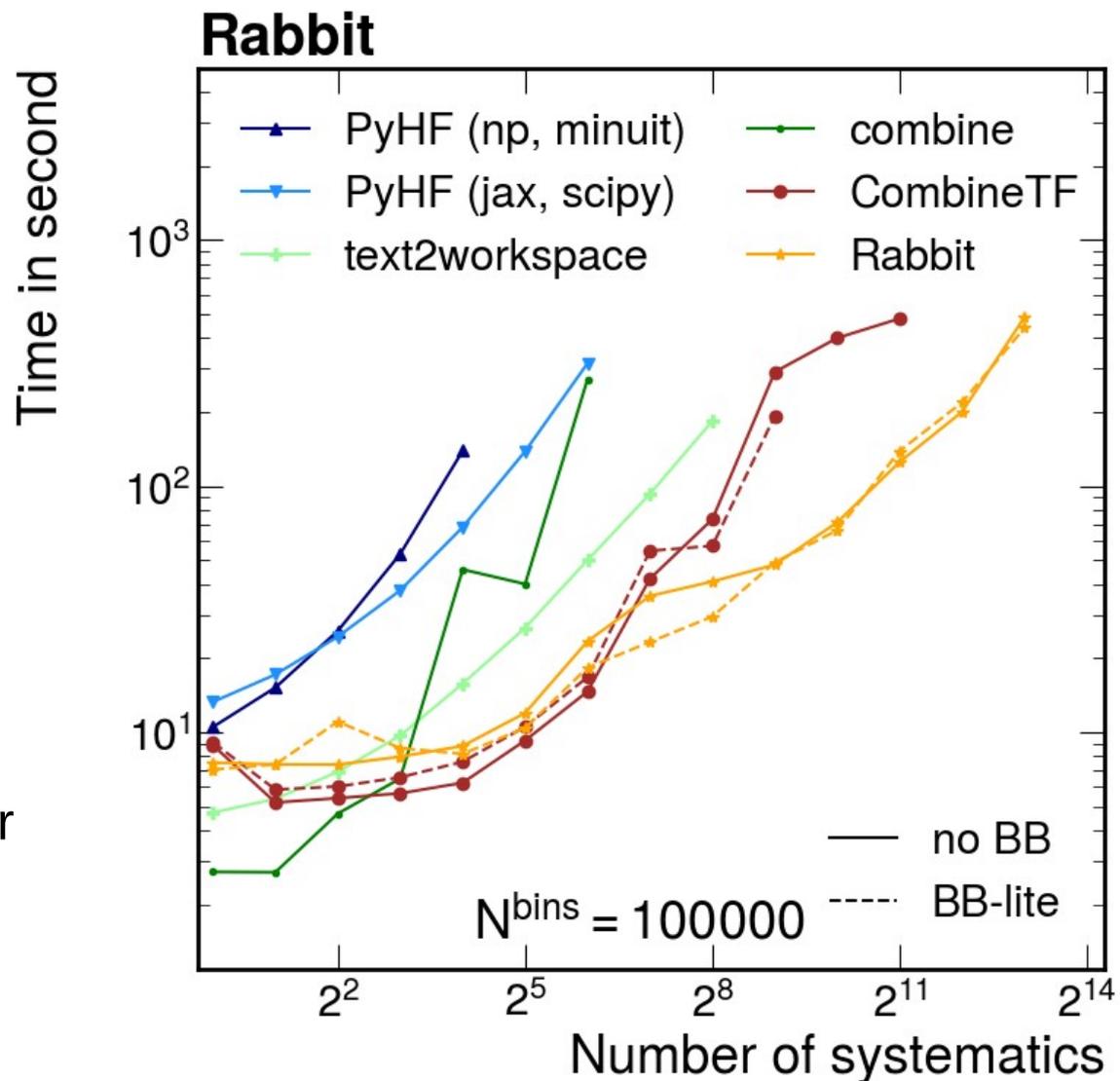
Verified that all tools yield same likelihood values at minimum

Rabbit has increased initialization time due to JITing

- But can fit more complex models

In more complex models, fits with BB

- PyHF fails due to complexity
- Combine fails due to memory error



Follow the white rabbit

Rabbit enables binned profile maximum likelihood fits with large number of bins and parameters where other tools fail

- Due to state of the art minimizer and efficient differential programming
- Linearization techniques to further reduce model complexity

Physics models and masked channels provide a generic interface

- Yet, streamlined to reduce manual customization and mitigate error rate

Feature complete for measurements where Hessian approximation holds

- Including prefit-postfit plots, pulls and impacts, ...
- Likelihood based inference may be added, similar speedup expected

Code public on github.com/WMass/rabbit

- Or: `pip install rabbit-fit`



Thank You!

Symmetrization of systematic variations

Simplify likelihood fits, improve validity of Hessian approximation

4 different symmetrization option supported

For nearly symmetric uncertainties (e.g. up to statistical fluctuations)

1) **average**: Average up and down variations per bin

2) **conservative**: Mirror maximum of up and down variation per bin

For truly asymmetric uncertainties

- Preserve covariance matrix for yield in limit of Gaussian uncertainties
- Decompose asymmetric variation into two symmetric ones

$$\mathbf{n} = \mathbf{n}^0 + \frac{1}{2}(\mathbf{v}_+ + \mathbf{v}_-)\theta_a + \frac{x}{2}(\mathbf{v}_+ - \mathbf{v}_-)\theta_d$$

3) Assume **linear** profiling: $x=1$ $\mathbf{n} = \mathbf{n}^0 + \mathbf{v}\theta$ with $v_{\pm} = \begin{cases} v_+ & \text{if } \theta \geq 0 \\ v_- & \text{if } \theta < 0 \end{cases}$

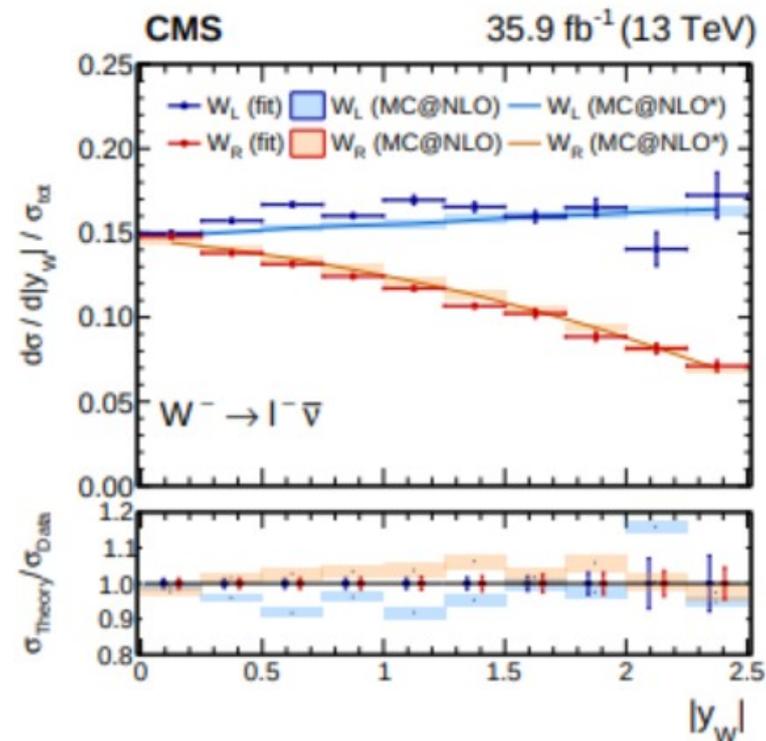
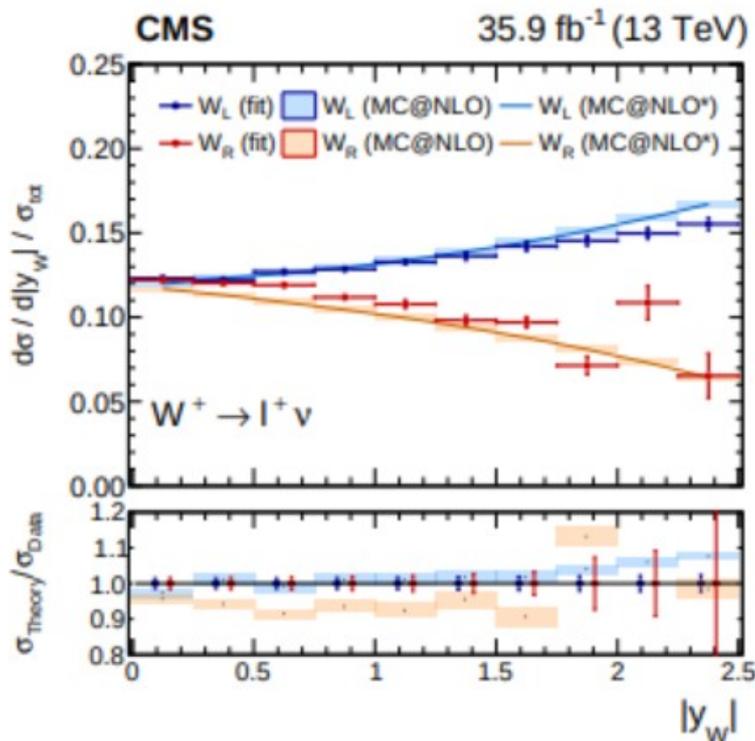
4) Assume **quadratic** profiling: $x=\sqrt{3}$

$$\mathbf{n} = \mathbf{n}^0 + \frac{1}{2}(\mathbf{v}_+ + \mathbf{v}_-)\theta + \frac{1}{2}(\mathbf{v}_+ - \mathbf{v}_-)\theta^2$$

A brief history

Work on **W helicity measurement** in 2018

- Unfolding of ~ 1000 reco bins in ~ 100 gen bins with ~ 250 nuisances,
- RooFit via minuit found to be insufficient
 - Limited numerical stability & efficiency, parallelization



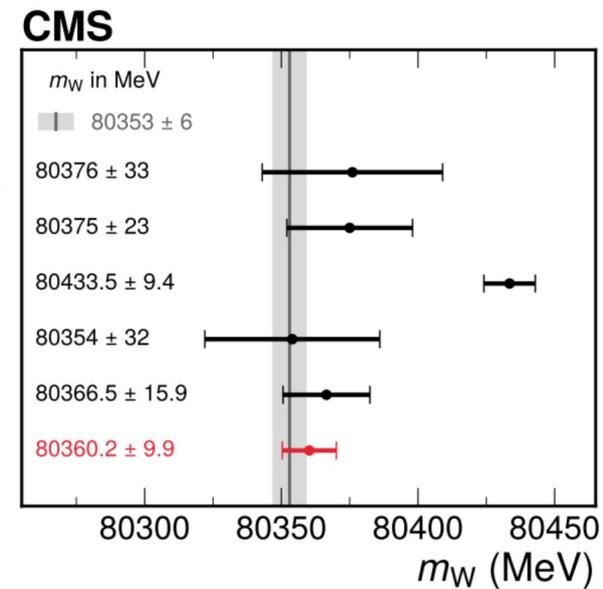
A brief history



CombineTF (see e.g. [PyHEP 2020](#))

- In [tensorflowfit](#) branch of Combine
- Based on TensorFlow (TF) 1.x
 - Make use of automatic differentiation of likelihood
 - Native TF minimizer: Trust region quasi Newton with SR1 eigenvector updates
- Input combine datacard & root histograms with text2hdf5 converter
- Also the baseline for the m_W analysis
 - ~3000 reco bins and ~4000 nuisance parameters
 - Some checks require split in >10,000 bins & additional nuisance parameters

Electroweak fit
PRD 110 (2024) 030001
LEP combination
Phys. Rep. 532 (2013) 119
D0
PRL 108 (2012) 151804
CDF
Science 376 (2022) 6589
LHCb
JHEP 01 (2022) 036
ATLAS
arXiv:2403.15085
CMS
This work



Several issues encountered

- Bottleneck in boost \rightarrow root \rightarrow hdf5 conversion
- Hard-coded “physics models” in single spaghetti code script
- Slow computation of hessian and hessian vector products (HVP)
- Has to be run in CMSSW virtual environment or similar
- More convenient and efficient libraries (e.g. tensorflow 2.x)

~~CombineTF2~~ Rabbit



Re-wrote from scratch – RABBIT: Rapid automatic bin based inference tool

- Independent **code base**
- Already used for m_W

Using TF 2 interfaced to scipy minimizer

- More efficient computation of hessian vector products
 - Motivates new minimizer: trust-krylov

Independent of CMSSW

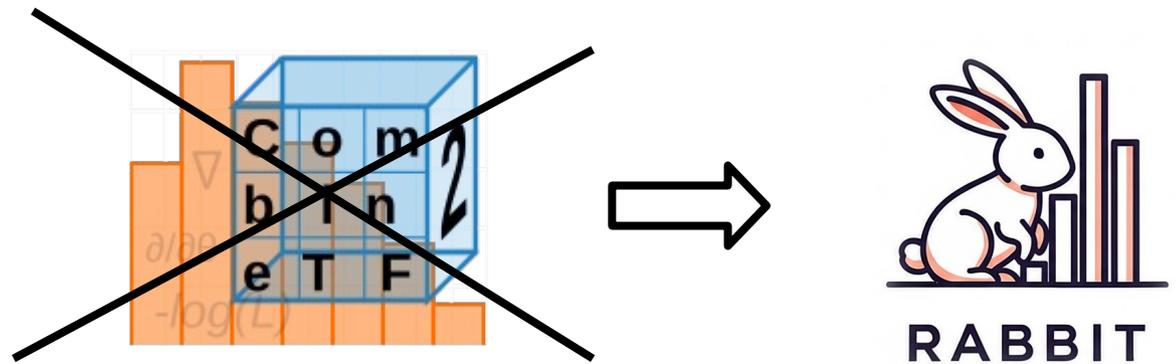
- pip install or container

Restrict to binned likelihoods

- Many simplifications

Focus on measurements in limit of large sample size

- Uncertainties/ impacts/ ... computed in Hessian approximation
- Minimal support for likelihood inference, but can/may be added



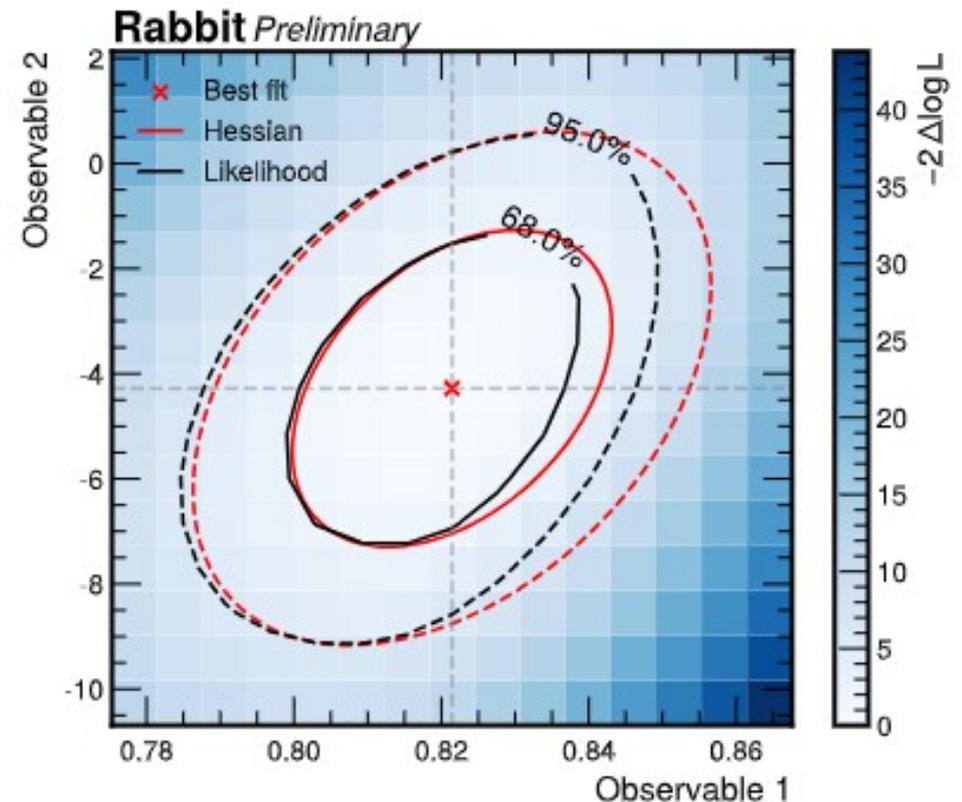
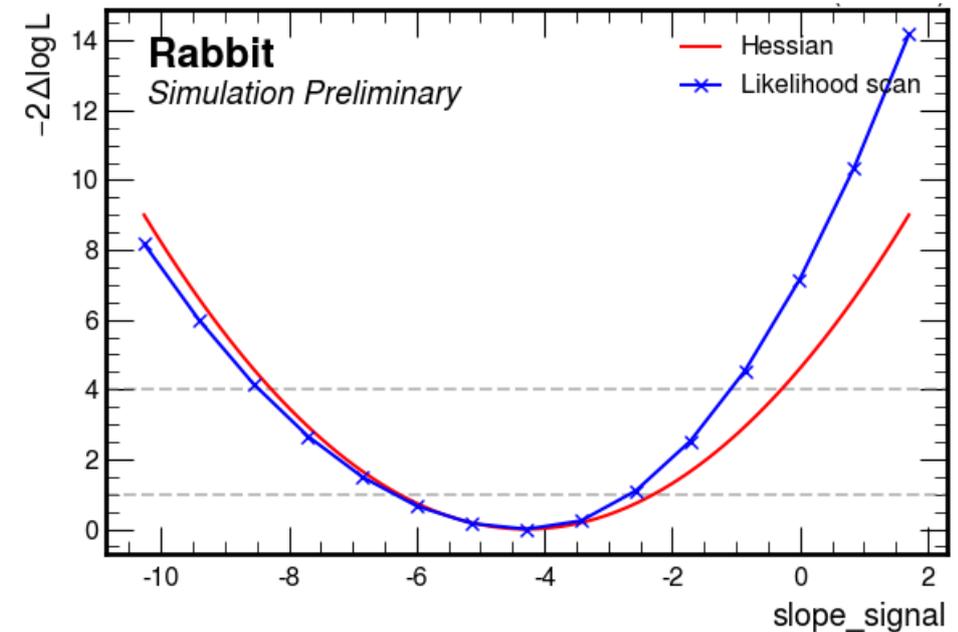
Likelihood scans

In development

- Some basic functions are written

Investigating different options:

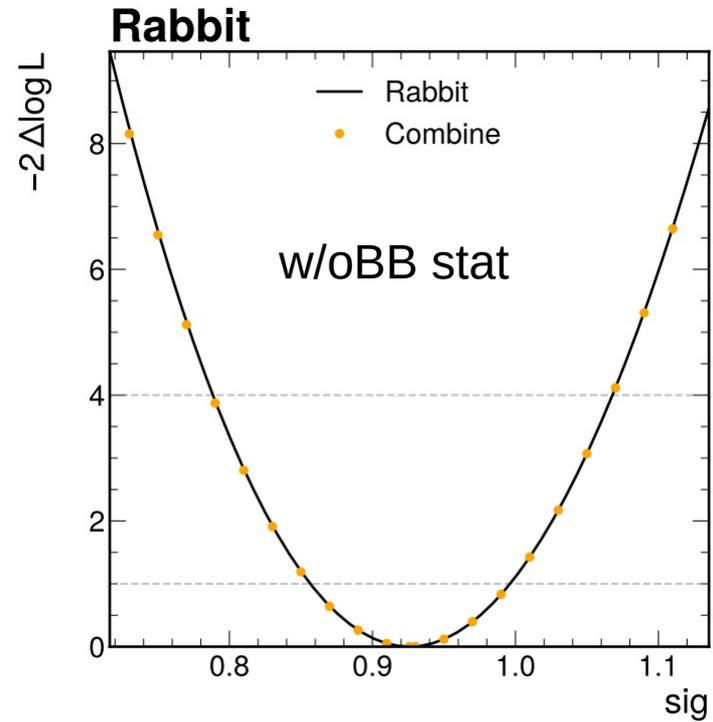
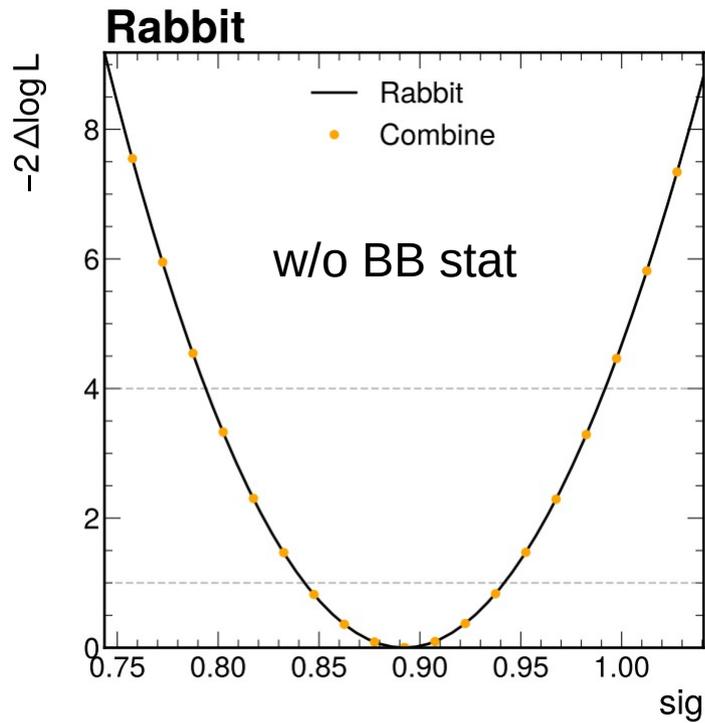
- Spiral scans
 - Gain from neighboring minimum
 - Counter gradient in scan parameter
- Contour scans
 - Constrain $f=L_{\text{sat.}} + \frac{1}{2}$ and maximize parameter value(s)



Comparison to other tools

Configuration with same likelihood implementation

- Validated by $\Delta\log(L)$ scan



Comparison with other tools – memory

Rabbit is memory hungry

- 1GB base consumption

