Parametrizing the parametrized NSBI

Statistical Analysis Software Meeting

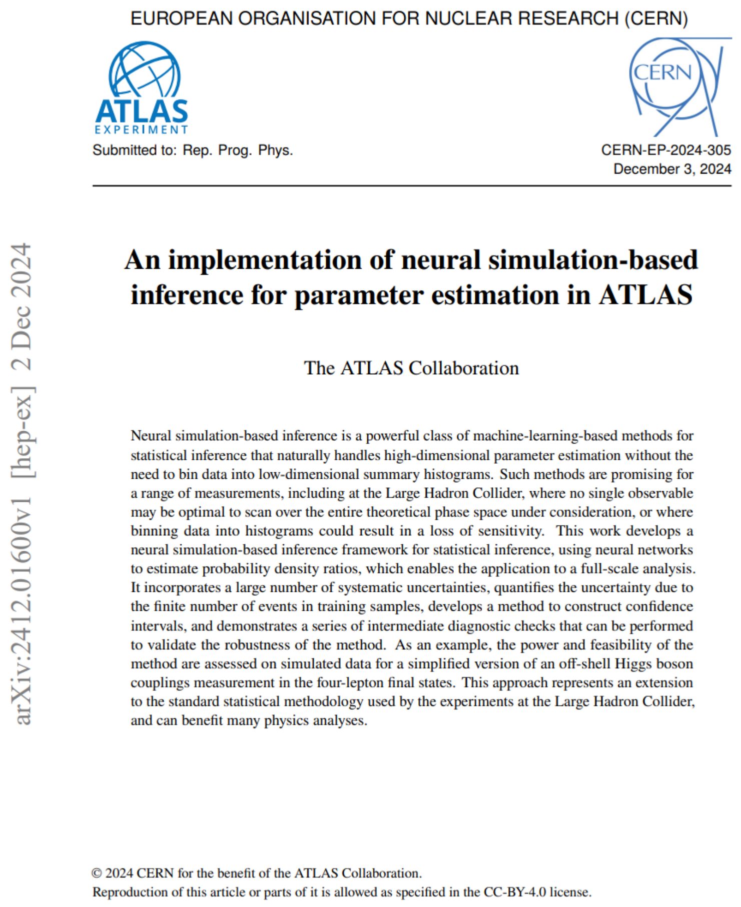
February 2025

Rafael Coelho Lopes de Sá (UMass Amherst)

(based on collaboration and/or discussion with Jay Sandesara, Matt Maroun, Will Buttinger, Aishik Ghosh, and RD Schaffer)

Introduction

ATLAS recently published an implementation of neural simulation-based inference (NSBI)



EUROPEAN ORGANISATION FOR NUCLEAR RESEARCH (CERN)

Submitted to: Rep. Prog. Phys.

CERN-EP-2024-305
December 3, 2024

arXiv:2412.01600v1 [hep-ex] 2 Dec 2024

An implementation of neural simulation-based inference for parameter estimation in ATLAS

The ATLAS Collaboration

Neural simulation-based inference is a powerful class of machine-learning-based methods for statistical inference that naturally handles high-dimensional parameter estimation without the need to bin data into low-dimensional summary histograms. Such methods are promising for a range of measurements, including at the Large Hadron Collider, where no single observable may be optimal to scan over the entire theoretical phase space under consideration, or where binning data into histograms could result in a loss of sensitivity. This work develops a neural simulation-based inference framework for statistical inference, using neural networks to estimate probability density ratios, which enables the application to a full-scale analysis. It incorporates a large number of systematic uncertainties, quantifies the uncertainty due to the finite number of events in training samples, develops a method to construct confidence intervals, and demonstrates a series of intermediate diagnostic checks that can be performed to validate the robustness of the method. As an example, the power and feasibility of the method are assessed on simulated data for a simplified version of an off-shell Higgs boson couplings measurement in the four-lepton final states. This approach represents an extension to the standard statistical methodology used by the experiments at the Large Hadron Collider, and can benefit many physics analyses.

© 2024 CERN for the benefit of the ATLAS Collaboration.
Reproduction of this article or parts of it is allowed as specified in the CC-BY-4.0 license.

The goal of this presentation is:

- Very briefly review what was published
- Very briefly review software developments
- Discuss computational challenges
- Discuss an alternative (somehow proposed in the paper)

- Ideas always welcome.

Neural simulation-based inference

- The idea of simulation-based inference is to create a NN approximation for the probability density

$$p(x|\mu, \alpha)$$

observables

nuisance parameter

parameter of interest

- Most LHC analysis perform statistical data analysis based on the profile likelihood ratio test statistic:

$$t_\mu = \sum_{\text{events}} -2 \ln \frac{p(x|\mu, \hat{\hat{\alpha}})}{p(x|\hat{\mu}, \hat{\alpha})}$$

Why doing it?

Because you can get more sensitivity if your NN approximation is better than the binned-Poisson approximation:

1. By modeling the (non-trivial) dependency with respect to a high-dimensional observable space
2. By modeling the (non-trivial) dependency with respect to the parameters (μ, α)
3. By performing an unbinned analysis.

The three factors contribute with different relative importance depending on the specific analysis being performed.

Sometimes all three points above are relevant (as in the off-shell Higgs production example we gave in paper), sometimes none of them are (in which case you probably don't want to use NSBI).

NSBI with density ratios

- Our specific implementation starts from the observation that the profile likelihood ratio does not depend on the probability density $p(x|\mu, \alpha)$ but solely on the ratio of probability densities.
- We seek instead for NN approximations of the ratio

$$\frac{p(x|\mu, \alpha)}{p_{\mathrm{ref}}(x)}$$

Where the reference density must satisfy:

- $p_{\mathrm{ref}}(x) > 0$ in the observable space probed by the analysis
- Independent of (μ, α)

but is otherwise arbitrary.

- We provided a specific proposal for the reference sample called "search-oriented" NSBI which satisfy the two criteria above.

$$t_\mu = -\sum_{\mathrm{events}} 2\ln\frac{p(x|\mu,\hat{\hat{\alpha}})}{p_{\mathrm{ref}}(x)} + \sum_{\mathrm{events}} 2\ln\frac{p(x|\hat{\mu},\hat{\alpha})}{p_{\mathrm{ref}}(x)}$$

Mixture models and signal strengths

- We usually write our probability models as mixtures between processes (signals and backgrounds):

$$\frac{p(x|\mu,\alpha)}{p_{\text{ref}}(x)} = \sum_{J \text{ processes}} \frac{\nu_J(\mu,\alpha)}{\nu(\mu,\alpha)} \frac{p_J(x|\mu,\alpha)}{p_{\text{ref}}(x)}$$

expected number of events

- This allows us to create NN approximations per-process. This is nice, since the distributions ("kinematics") can vary a lot for different processes.
- This formula can be further simplified in the case of "signal strength" measurements, which basically means that we assume:

$$\frac{p(x|\mu,\alpha)}{p_{\text{ref}}(x)} = \sum_{J \text{ processes}} \frac{\nu_J(\alpha)}{\nu(\mu,\alpha)} f_J(\mu) \frac{p_J(x|\alpha)}{p_{\text{ref}}(x)}$$

In most cases, $f_{\text{signal}} = \mu$, $f_{\text{background}} = 1$, but these simple "$\mu S + B$" are not so interesting for NSBI

Parametrizing the parameter dependency

- Signal strength measurements are particular simple because the POI-dependency of the density ratio is factorized and we don't have to rely on conditional ("parametrized") NN
- An additional simplification can be obtained by assuming factorization of the nuisance parameters (what sometimes is called "vertical interpolation" in the context of binned analyses)

$$\frac{p(x|\mu,\alpha)}{p_{\text{ref}}(x)} = \frac{1}{\nu(\mu,\alpha)} \sum_{J \text{ processes}} \left[f_J(\mu) \underbrace{\frac{p_J(x)}{p_{\text{ref}}(x)}}_{\text{parameter-independent NN}} \prod_{m \text{ NP}} G_{J,m}(\alpha_m) g_{J,m}(x,\alpha_m) \right]$$

our "master formula"

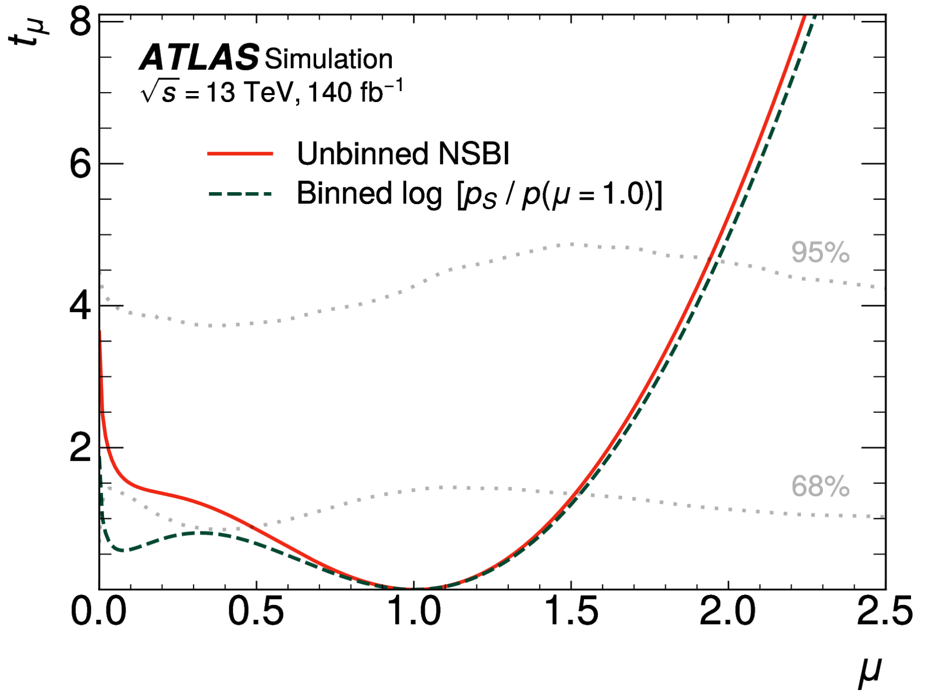
- Where $G_{J,m}(\alpha_m) = \nu_J(\alpha_m)/\nu_J(0)$ describes the normalization systematics uncertainties and

$$g_{J,m}(x,\alpha_m) = p_J(x|\alpha_m)/p_J(x|0)$$

describes the shape systematic uncertainty and can also be approximated by NN.

Is it worth the trouble?

- In some cases, yes.
- That's a comparison between a histogram analysis and NSBI for a "physics-inspired" toy analysis with $f_1 = \sqrt{\mu}$, $f_2 = (\mu - \sqrt{\mu})$, and $f_3 = \left(1 - \sqrt{\mu}\right)$.
- The many square-roots come from quantum interference effects and are universal in this kind of analysis, i.e., not just a math trick we used to sell the idea



Software development

- The papers were written based on an implementation of the master formula as JAX functions (thanks to Jay Sandesara).
- After that, we wrote a custom derived class of $\mathrm{RooAbsPdf}$ which implements our master formula in RooFit (thanks to Matt Maroun).
- More recently, we wrote an implementation the master formula using native RooFit classes, i.e., as a $\mathrm{RooRealSumPdf}$ of several $\mathrm{RooProduct}$ and $\mathrm{PiecewiseInterpolation}$ (thanks to Will Buttinger and Matt Maroun).
  - Our hope is that this native implementation may benefit from recent improvements in the evaluation and minimization of the NLL
  - Our implementation in JAX greatly benefitted from auto-differentiation and parallelized calculations.
- Both implementations work well, and Matt has done extensive validation (which will continue to be made, as this kind of analysis become more common).
- Right now, this work is critical for the ATLAS Collaboration so that combinations between NSBI and histograms-based analyses can be performed using the vetted combination tools.

Computational choices and challenges

- Our implementation does not perform NN inference on-the-fly.
- The values of the NN are pre-calculated and used as de-facto observables for the RooDataSet
- This is a trade-off:
  - Evaluation of NN on-the-fly would save memory, but it would increase CPU time.
  - Pre-evaluation of the NN saves CPU time, but it would increase memory consumption.
- In practice, the choice was also made because the infrastructure to evaluate the NN on-the-fly does not exist in RooFit.

these are the observables (instead of x)

$$\frac{p(x|\mu,\alpha)}{p_{\mathrm{ref}}(x)} = \frac{1}{\nu(\mu,\alpha)} \sum_{J\ \mathrm{processes}} \left[f_J(\mu) \boxed{\frac{p_J(x)}{p_{\mathrm{ref}}(x)}} \prod_{m\ \mathrm{NP}} G_{J,m}(\alpha_m) \boxed{g_{J,m}(x,\alpha_m)} \right]$$

- The stress on the memory can be non-negligible. Modern measurements can easily contain 100 – 500 different NP, meaning that the RooDataSet has this large number of observables (the g_J functions which describe the shape systematic uncertainties).

When is this a problem?

- For a typical “signal region” with 1,000 – 10,000 events, this is not a problem.

$$10^4 \text{events} \times 10^2 \text{observables} \times\ 8 \text{ bytes} < 1 \text{ GB} \ldots \text{ memory is under control}$$

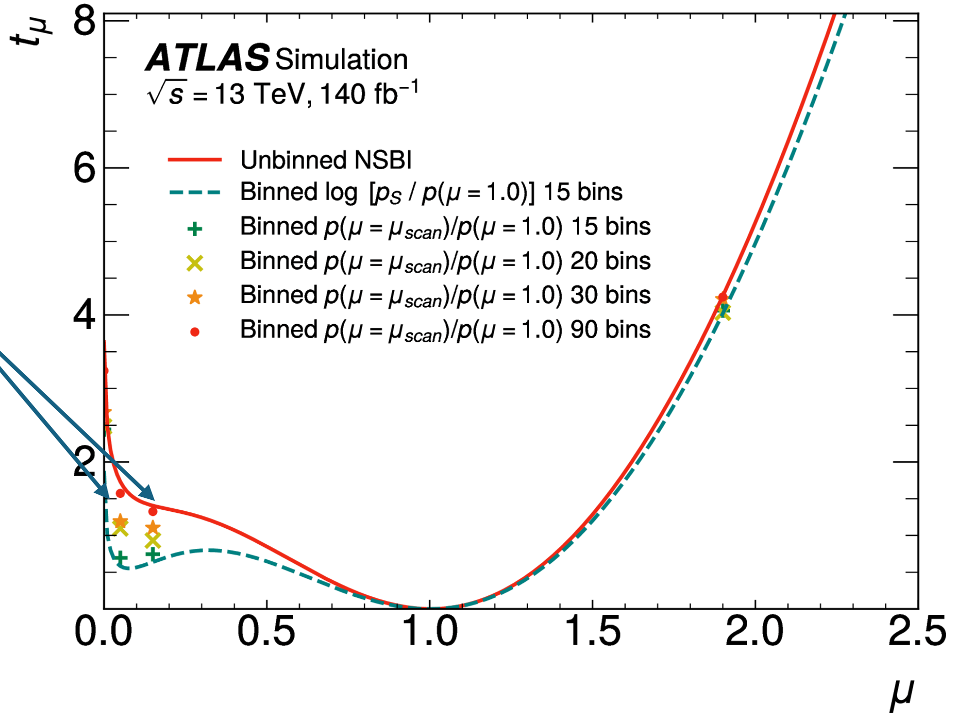
- The problem comes when using simulated events, either because the analysis is still blinded or because we are estimating the expected power of the analysis.
- In this case, the MC samples can easily have millions of events in the signal regions, what translates to 500 GB – 1 TB of RAM
- While this is not uncommon in modern machines, it is difficult to find cluster where jobs requiring this amount of memory can be submitted.

- In addition to that, RooFit is not optimized to use such large amount of memory. When having to deal with many millions of events in the signal region, we were able to evaluate and minimize the NLL using the JAX implementation, but not in RooFit.

Parametrized NSBI

- In the article, we showed that, for a given value μ', the observable $p(x|\mu', 0)/p(x|\mathrm{SM}, 0)$ is an optimal observable.
  - We can consider values of $\alpha \neq 0$ (see, for instance, the discussion about "uncertainty aware NN" developed by Aishik Ghosh), but that's not something we will explore here.
- The definition of optimal here is quite practical. If we build a Poisson-based model with this observable, we showed that, in the limit of large number of bins, the test statistic will converge to the NSBI value

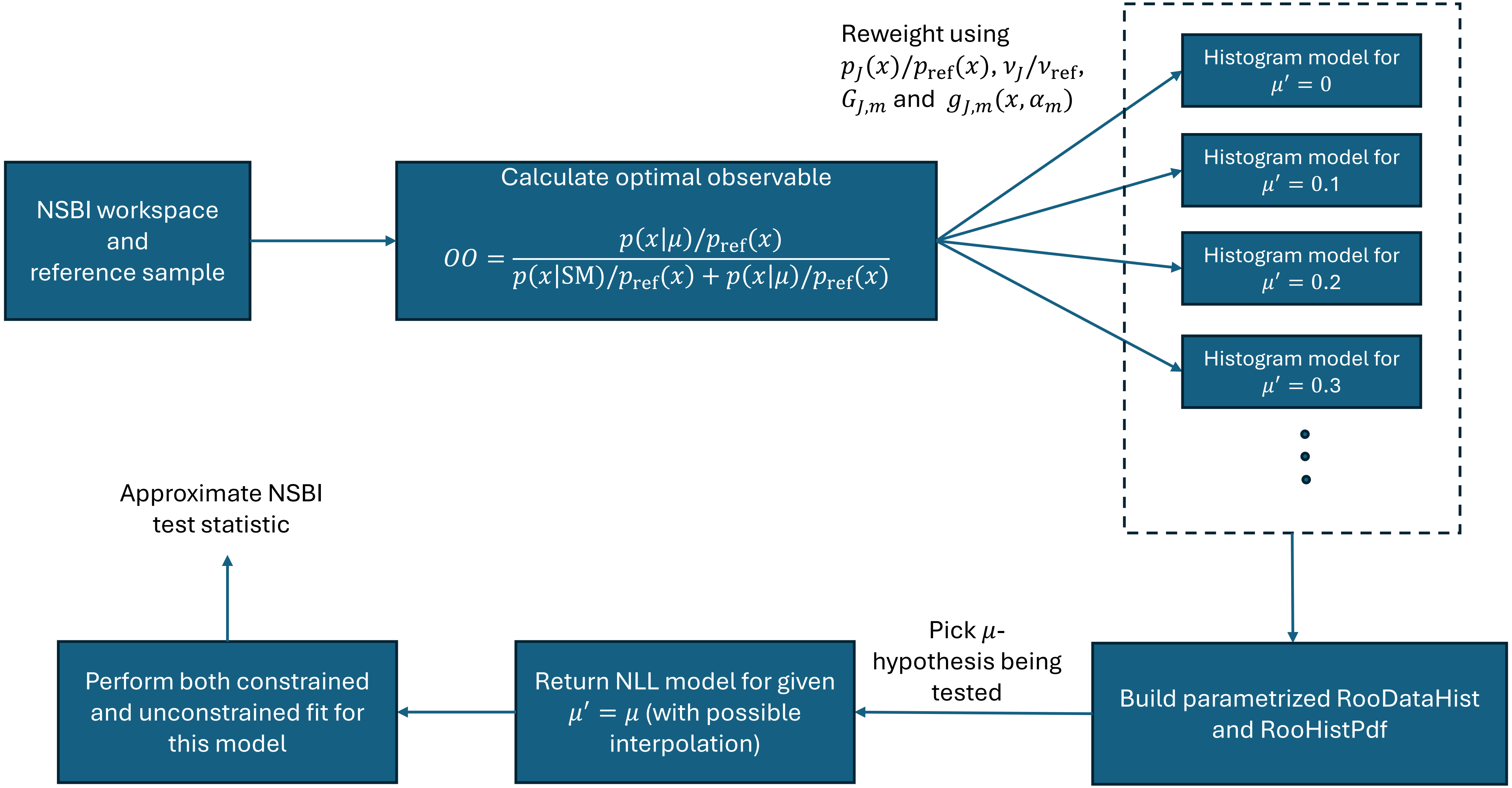
Different observables for different values of μ'. In general, there is no uniform optimal observable (à-la Karlin-Rubin). And that's when NSBI is a powerful tool.



Software challenges

- This approach can mitigate the challenges with memory consumption, because we reduced it back to RooDataSet (well RooDataHist) with ~100 observables (bins), which is pretty common in modern analysis.
- The price we pay is that we have a different observable per value of the parameter μ. This concept does not exist in RooFit.
- What we are seeking to develop is a parametrized version of RooDataHist, which takes on an additional parameter and internally uses different histograms depending on the value of the parameter.
  - Depending on how the binning is done by the user, the dependency with μ can be smooth, what means that this class could interpolate between the different histograms used to construct it.
- An associated parametrized version of RooHistPdf will be needed, but maybe this is possible to be done with existing RooFit tools (I don't know for sure)
- Data analysis codes would also need to be changed, since the value of NLL at the minimum would change (since the observable change).
  - In practice that means that the user would have to do a separate unconstrained fit per scanned value.
  - It slightly increased CPU consumption, but not by much.

Proposed workflow



NSBI workspace and reference sample
Calculate optimal observable
$OO = \frac{p(x|\mu)/p_{\text{ref}}(x)}{p(x|\text{SM})/p_{\text{ref}}(x) + p(x|\mu)/p_{\text{ref}}(x)}$
Reweight using $p_J(x)/p_{\text{ref}}(x)$, ν_J/ν_{ref}, $G_{J,m}$ and $g_{J,m}(x, \alpha_m)$
Histogram model for $\mu' = 0$
Histogram model for $\mu' = 0.1$
Histogram model for $\mu' = 0.2$
Histogram model for $\mu' = 0.3$
Build parametrized RooDataHist and RooHistPdf
Pick μ-hypothesis being tested
Return NLL model for given $\mu' = \mu$ (with possible interpolation)
Perform both constrained and unconstrained fit for this model
Approximate NSBI test statistic

Conclusions

- Most of the workflow in the previous slide has been written (and is being validated)
- But, for easy integration in combination codes, a proper RooFit implementation of the concept of parametrized RooDataHist and RooPdfHist would be needed.

Questions? Ideas?