Adding Automatic Differentiation to RooFit

The Road to Differentiable and Probabilistic Programming in Fundamental Physics

Munich, June 2023

Garima Singh (Princeton University), Jonas Rembser (CERN), Lorenzo Moneta (CERN), Vassil Vassilev (Princeton University)

compiler-research.org
Add automatic differentiation (AD) to RooFit, a statistical modelling library packed in ROOT.
Methods of Automatic Differentiation

The Two Techniques

Source Code Transformation AD

- Synthesize derivative code from the input program automatically.
- Faster - allows for easier compiler optimization.
- Eg. Tapenade, Enzyme, Clad

Operator Overloading AD

- Use a new data type and operator overloading to keep track of derivatives as the original program executes.
- Slower and requires hand writing annotations and changing data types.
- Eg. PyTorch/TensorFlow, CoDiPack, etc.

[1]: https://en.wikipedia.org/wiki/Automatic_differentiation
An Efficient Method of Differentiation

Compiler-Based Source Transformation AD: Clad

Clad\[^{[1]}\], a source code transformation AD tool, implemented as a plugin to the clang compiler. Clad inspects the internal compiler representation of the target function to generates its derivative.

```cpp
double absFunc(double x) {
    if (x < 0) return -x;
    else return x;
}
```

```cpp
double absFunc_darg0(double x) {
    double _d_x = 1;
    if (x < 0) return -_d_x;
    else return _d_x;
}
```

- Proximity to compiler allows for more control over code generation.
- Support for a good subset of modern C++ constructs.

---

\[^{[1]}\]: https://github.com/vgvassilev/clad

Adding Automatic Differentiation to RooFit - Garima Singh | Differentiable and Probabilistic Programming for Fundamental Physics TWS 2023
An Efficient Method of Differentiation

*Compiler-Based Source Transformation AD: Clad*

Clad also can be used within Cling[^3], the C++ interpreter used with ROOT.

```
[2]: double fn(double x, double y) {
    return x*x*y + y*y;
}

[3]: auto fn_dx = clad::differentiate(fn, "x");

[4]: fn_dx.execute(5, 3)

[4]: 30.000000
```

[^3]: https://github.com/root-project/cling

Binder Tutorial
# Motivation

**Why AD?**

- One goal - Make RooFit Faster. Results from a Higgs-combination fit:

<table>
<thead>
<tr>
<th>Serial old</th>
<th>Parallel N=1</th>
<th>Parallel N=2</th>
<th>Parallel N=4</th>
<th>Parallel N=8</th>
<th>Parallel N=16</th>
</tr>
</thead>
<tbody>
<tr>
<td>setup_roofit</td>
<td>setup_roofit</td>
<td>setup_roofit</td>
<td>setup_roofit</td>
<td>setup_roofit</td>
<td>setup_roofit</td>
</tr>
<tr>
<td>minute_init</td>
<td>minute_init</td>
<td>minute_init</td>
<td>minute_init</td>
<td>minute_init</td>
<td>minute_init</td>
</tr>
<tr>
<td>gradient_calc</td>
<td>gradient_calc</td>
<td>gradient_calc</td>
<td>gradient_calc</td>
<td>gradient_calc</td>
<td>gradient_calc</td>
</tr>
<tr>
<td>line_search</td>
<td>line_search</td>
<td>line_search</td>
<td>line_search</td>
<td>line_search</td>
<td>line_search</td>
</tr>
<tr>
<td>time_search</td>
<td>328</td>
<td>328</td>
<td>328</td>
<td>328</td>
<td>328</td>
</tr>
</tbody>
</table>

Derivatives become bottleneck!

- Good results, but still use numerics
- Potential next step – use Automatic

---

**We have seen some promising results (in ROOT) already!**

**Performance Comparison of Generation in TFormula**

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>14.4 x</td>
</tr>
<tr>
<td>Exponential</td>
<td>9.9 x</td>
</tr>
<tr>
<td>BreitWigner</td>
<td>22.9 x</td>
</tr>
<tr>
<td>Chebyshev deg.0</td>
<td>12.4 x</td>
</tr>
<tr>
<td>Chebyshev deg.1</td>
<td>29.4 x</td>
</tr>
<tr>
<td>Chebyshev deg.2</td>
<td>26.4 x</td>
</tr>
</tbody>
</table>

**Performance Speedup of a Multi-Gaussian Fit (10000 bins)**

- Speedup of 60x!

TF1 based benchmarks. TF1 is the TFormula fitting interface for fitting histograms.
Automatic Differentiation in RooFit

Sounds easy…

What we want to differentiate

Made up of various RooFit objects

Our AD tool of choice

Differentiable RooFit Models!

Actually, not so simple…

RooFit has an object oriented model which deliberately hides the differential properties of the nodes in favor of ease of use.

Adding Automatic Differentiation to RooFit - Garima Singh | Differentiable and Probabilistic Programming for Fundamental Physics TWS 2023
Automatic Differentiation in RooFit

Challenges

RooFit represents all mathematical formulae as RooFit objects which are then brought together into a compute graph. This compute graph makes up a model on which further data analysis is run.

<table>
<thead>
<tr>
<th>Math Notations</th>
<th>RooFit Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable</td>
<td>x</td>
</tr>
<tr>
<td>function</td>
<td>f(x)</td>
</tr>
<tr>
<td>PDF</td>
<td>f(x)</td>
</tr>
<tr>
<td>space point</td>
<td>( \hat{x} )</td>
</tr>
<tr>
<td>integral</td>
<td>( \int_a^b f(x) )</td>
</tr>
<tr>
<td>list of space points</td>
<td>( \tilde{x}_1, \tilde{x}_1, \tilde{x}_1 \ldots )</td>
</tr>
</tbody>
</table>

\[
f(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{x - \mu}{\sigma} \right)^2}
\]

//Obj represents \( f(x) \) here

RooGaussian \( \text{obj}(x, \mu, \sigma) \);

Gaussian Probability Distribution Function (pdf)

Equivalent Code in C++ with RooFit

Programmers/users know this relationship. But how do we connect these two together when a connection is not obvious in code?
Automatic Differentiation in RooFit

How Does it work?

What that we want to differentiate

Some way to expose differentiable properties of the graph as code.

C++ code the AD tool can understand

C++ code the AD tool can understand

The AD tool

Derivative code of the model!

Adding Automatic Differentiation to RooFit - Garima Singh | Differentiable and Probabilistic Programming for Fundamental Physics TWS 2023
Automatic Differentiation in RooFit

*How Does it work?*

What that we want to differentiate

C++ code the AD tool can understand

Define 2 Functions in RooFit

The “glue” function enabling graph squashing.

Stateless function enabling differentiation of each class.

```cpp
double ADDetail::gauss(double x, double mean, double sigma) {
    const double arg = x - mean;
    const double sig = sigma;
    return std::exp(-0.5 * arg * arg / (sig * sig));
}
```

```cpp
void RooGaussian::translate(...) override {
    result = "ADDetail::gauss(" +
             _x->getResult() + " ," + _mu->getResult() +
             " ," + _sigma->getResult() + "")";
}
```
Automatic Differentiation in RooFit

How Does it work?

What that we want to differentiate

RooGaussian::evaluate()
The RooFit call to evaluate a gaussian

ADDetail::gauss(x, mu, sig)
The equivalent code generated

ADDetail::gauss(x, mu, sig) / ADDetail::gaussIntegral(...)
The equivalent code generated (given the class supports analytical integrals)

ADDetail::gauss integral

C++ code the AD tool can understand

Define 2 Functions in RooFit

- Bookkeeping
  & caching
Automatic Differentiation in RooFit

The Big Picture

What that we want to differentiate

‘Squash’ the graph into code

C++ code the AD tool can understand

Roo*::translate()

C++ code the AD tool can understand

Derivative code of the model!

C++ code the AD tool can understand

The AD tool

Clad

Adding Automatic Differentiation to RooFit - Garima Singh | Differentiable and Probabilistic Programming for Fundamental Physics TWS 2023
Automatic Differentiation in RooFit

Interlude: Other Methods

What we want to differentiate

A mature AD tool that can handle object oriented programming well

Derivative code of the model!

What we want to differentiate

Differentiate each node independently and then combine it exactly like RooFit does.

Derivative of the model!
Automatic Differentiation in RooFit

The Big Picture

What we want to differentiate

‘Squash’ the graph into code

C++ code the AD tool can understand

C++ code the AD tool can understand

The AD tool

Derivative code of the model!

Adding Automatic Differentiation to RooFit - Garima Singh | Differentiable and Probabilistic Programming for Fundamental Physics TWS 2023
Results

- A HistFactory example (binned pdfs based on template histograms)
  - Out of RooFit, POC

- A basic RooFit example with binned fit of analytical shapes
  - In RooFit

- A WIP ATLAS HistFactory Benchmark
  - In RooFit
Results

- A HistFactory example (binned pdfs based on template histograms)
- A basic RooFit example with binned fit of analytical shapes
- A WIP ATLAS HistFactory Benchmark
Results

The POC HistFactory Model

An example histogram fitting model with 2 bins and 2 channels, with 3 samples per channel. Based on the hf_001 example.
Results

The POC HistFactory Model

~5.5x speedup

Tested on ROOT v6.26.

Adding Automatic Differentiation to RooFit - Garima Singh | Differentiable and Probabilistic Programming for Fundamental Physics TWS 2023
Results

- A HistFactory example (binned pdfs based on template histograms)

- A basic RooFit example with binned fit of analytical shapes

- A WIP ATLAS HistFactory Benchmark
Results
The Real RooFit Example

![Performance Comparison for Minimization Time](https://example.com/plot.png)

**Model used:** gauss + gauss + expo

- **Code-Squashing Numerical-Diff**
- **RooFit Numerical-Diff**
- **Code-Squashing AD**

**18x Faster!**

Tested on ROOT master as of May 2023.

*Excludes the seed generation time, more info - [look here](https://example.com/lookhere.github.io)*

Adding Automatic Differentiation to RooFit - Garima Singh | Differentiable and Probabilistic Programming for Fundamental Physics TWS 2023
Results

The Real RooFit Example

RooFit has clear advantages over “hand-writing” models, but can be pushed more with AD!

*Excludes the seed generation time, more info - look here

Adding Automatic Differentiation to RooFit - Garima Singh | Differentiable and Probabilistic Programming for Fundamental Physics TWS 2023
## Results

### Why??

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Time / Iteration</th>
<th>Total Iterations to Converge</th>
<th>Final FCN Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code-Squashing Numerical-Diff</td>
<td>~380 ms</td>
<td>136</td>
<td>659552.2918</td>
</tr>
<tr>
<td>RooFit Numerical-Diff</td>
<td>~86 ms</td>
<td>136</td>
<td>659552.2917</td>
</tr>
<tr>
<td>Code-Squashing AD</td>
<td>~11 ms</td>
<td>58</td>
<td>659551.9860</td>
</tr>
</tbody>
</table>

Tested on ROOT master as of May 2023.
## Results

### Why? Code-Squashing vs RooFit (Numerical)

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Time / Iteration</th>
<th>Total Iterations to Converge</th>
<th>Final FCN Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code-Squashing Numerical-Diff</td>
<td>~380 ms</td>
<td>136</td>
<td>659552.2918</td>
</tr>
<tr>
<td>RooFit Numerical-Diff</td>
<td>~86 ms</td>
<td>136</td>
<td>659552.2917</td>
</tr>
<tr>
<td>Code-Squashing AD</td>
<td>~11 ms</td>
<td>58</td>
<td>659551.9860</td>
</tr>
</tbody>
</table>

~3.5x Slower time/iteration.

*Why?* Even Though both use num-diff, RooFit uses complex caching logic, making it faster!
## Results

### Why? Code-Squashing AD vs RooFit Numerical

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Time / Iteration</th>
<th>Total Iterations to Converge</th>
<th>Final FCN Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code-Squashing Numerical-Diff</td>
<td>~380 ms</td>
<td>136</td>
<td>659552.2918</td>
</tr>
<tr>
<td>RooFit Numerical-Diff</td>
<td>~86 ms</td>
<td>136</td>
<td>659552.2917</td>
</tr>
<tr>
<td>Code-Squashing AD</td>
<td>~11 ms</td>
<td>58</td>
<td>659551.9860</td>
</tr>
</tbody>
</table>

**~ 8x Faster Derivatives**

**Why?** AD is faster than NumDiff, esp. For large number of params!

Tested on ROOT master as of May 2023.

Adding Automatic Differentiation to RooFit - Garima Singh | Differentiable and Probabilistic Programming for Fundamental Physics TWS 2023
## Results

### Why? Code-Squashing AD vs RooFit Numerical

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Time / Iteration</th>
<th>Total Iterations to Converge</th>
<th>Final FCN Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code-Squashing Numerical-Diff</td>
<td>~380 ms</td>
<td>136</td>
<td>659552.2918</td>
</tr>
<tr>
<td>RooFit Numerical-Diff</td>
<td>~86 ms</td>
<td>136</td>
<td>659552.2917</td>
</tr>
<tr>
<td>Code-Squashing AD</td>
<td>~11 ms</td>
<td>58</td>
<td>659551.9860</td>
</tr>
</tbody>
</table>

### Faster (and better) Convergence (for large fits)

**Why?** AD is faster than NumDiff, esp. For large number of params!

**Why?** AD is more numerically stable than NumDiff. *Less num error = faster convergence!*

 Tested on ROOT master as of May 2023.

*Adding Automatic Differentiation to RooFit* - *Garima Singh* | Differentiable and Probabilistic Programming for Fundamental Physics TWS 2023
Results

- A HistFactory example (binned pdfs based on template histograms)
- A basic RooFit example with binned fit of analytical shapes
- A WIP ATLAS HistFactory Benchmark

Out of RooFit, POC

In RooFit

In RooFit
## Results

*WIP: ATLAS HistFactory Benchmark*

<table>
<thead>
<tr>
<th>No. Of Channels</th>
<th>RooFit Numerical-Diff</th>
<th>Code-Squashing AD</th>
<th>AD Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.03</td>
<td>0.01</td>
<td>2x</td>
</tr>
<tr>
<td>5</td>
<td>1.19</td>
<td>0.26</td>
<td>3.5x</td>
</tr>
<tr>
<td>10</td>
<td>2.22</td>
<td>0.36</td>
<td>5.2x</td>
</tr>
<tr>
<td>20</td>
<td>7.38</td>
<td>1.17</td>
<td>5.3x</td>
</tr>
</tbody>
</table>

*Excludes the seed generation time, more info - [look here](https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PAPERS/HIGG-2018-51/)*


Adding Automatic Differentiation to RooFit - *Garima Singh* | Differentiable and Probabilistic Programming for Fundamental Physics TWS 2023
Results

**WIP: ATLAS HistFactory Benchmark**

We are still investigating issues with JIT-ing in ROOT and also working on reducing these times.

<table>
<thead>
<tr>
<th>JIT Time in ROOT (s)*</th>
<th>Compile Time (g++ 10, s)</th>
<th>Compile Time (clang-13, s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-O0</td>
<td>~16</td>
<td>1.15</td>
</tr>
<tr>
<td>-O1</td>
<td>~17</td>
<td>4.46</td>
</tr>
<tr>
<td>-O2</td>
<td>~17</td>
<td>9.24</td>
</tr>
<tr>
<td>-O3</td>
<td>~17</td>
<td>10.69</td>
</tr>
</tbody>
</table>

* For a **non optimized** channel in the benchmark, For a partly optimized one, the time taken is < 1 sec

Adding Automatic Differentiation to RooFit - *Garima Singh* | Differentiable and Probabilistic Programming for Fundamental Physics TWS 2023
Current Status

What Can I Do Right Now?*

```
root[0] RooWorkspace myWS;
root[1] myWS.factory("sum::mu_shifted(mu[0, -10, 10], shift[1.0, -10, 10])");
root[2] myWS.factory("prod::sigma_scaled(sigma[3.0, 0.01, 10], 1.5)");
root[3] myWS.factory("Gaussian::gauss(x[0, -10, 10], mu_shifted, sigma_scaled)");
root[4] RooAbsReal &x = *myWS.var("x");
root[6] RooArgSet normSet{x};
```
Current Status

What Can I Do Right Now?*

```cpp
root[6] RooFuncWrapper gaussFunc("myGauss", "myGauss", pdf, normSet);

(double (*)(double *, const double *)) Function @0x7fcfbd2f6000
at input_line_19:1:
double myGauss(double *params, double const *obs)
{
    const double sigma_scaled = params[2] * 1.5;
    const double mu_shifted = params[0] + params[1];
    const double gauss_Int_x = ADDetail::gaussianIntegral(-10, 10, mu_shifted, sigma_scaled);
    const double gauss = ADDetail::gauss(params[3], mu_shifted, sigma_scaled);
    const double normGauss = gauss / gauss_Int_x;
    return normGauss;
}
```

*In ROOT master as of May 2023.
Adding Automatic Differentiation to RooFit - Garima Singh | Differentiable and Probabilistic Programming for Fundamental Physics TWS 2023
Current Status

What Can I Do Right Now?*

root[6] RooFuncWrapper gaussFunc("myGauss", "myGauss", pdf, normSet);

(double (*)(double *, const double *)) Function @0x7fcfbd2f6000
at input_line_19:1:

double myGauss(double *params, double const *obs)
{
    const double sigma_scaled = params[2] * 1.5;  "prod::sigma_scaled(sigma[3.0, 0.01, 10], 1.5)"
    const double mu_shifted = params[0] + params[1];  "sum::mu_shifted(mu[0, -10, 10], shift[1.0, -10, 10])"
    const double gauss_Int_x = ADDetail::gaussianIntegral(-10, 10, mu_shifted, sigma_scaled);
    const double gauss = ADDetail::gauss(params[3], mu_shifted, sigma_scaled);
    const double normGauss = gauss / gauss_Int_x;  "Gaussian::gauss(x[0, -10, 10], mu_shifted, sigma_scaled)"
    return normGauss;
}

*In ROOT master as of May 2023.
Adding Automatic Differentiation to RooFit - Garima Singh | Differentiable and Probabilistic Programming for Fundamental Physics TWS 2023
Conclusion

This work presents an efficient way to translate complex models such that they can be differentiated using AD. It demonstrates that AD can be used to effectively lower the fitting time for non-trivial models.
Conclusion

This work presents an efficient way to translate complex models such that they can be differentiated using AD. It demonstrates that AD can be used to effectively lower the fitting time for non-trivial models.
Conclusion

This work presents an efficient way to translate complex models such that they can be differentiated using AD. It demonstrates that AD can be used to effectively lower the fitting time for non-trivial models.
Future Work

● Continue efforts in supporting the ATLAS HistFactory benchmark in RooFit.
● Completely avoid the use of numerical gradients in fits using MINUIT.
● Extend support to cover other parts of RooFit.
● Optimize Clad generated derivatives and further explore how they can be parallelized (OpenMP or CUDA).
The End!

Questions?

https://www.linkedin.com/in/garimasingh28/

https://github.com/grimmmyshini

garima.singh@cern.ch
Backup

Model From Benchmarks

A RooPlot of "x_1"

Plot for number of channels = 1

Adding Automatic Differentiation to RooFit - Garima Singh | Differentiable and Probabilistic Programming for Fundamental Physics TWS 2023
RooRealVar c("c", "c", -0.5, -0.8, 0.2);
RooExponential expo("expo", "expo", x, c);

// Create two Gaussian PDFs $g_1(x, mean_1, sigma)$ and $g_2(x, mean_2, sigma)$ and their parameters
RooRealVar mean1("mean1", "mean of gaussians", 3, 0, 5);
RooRealVar sigma1("sigma1", "width of gaussians", 0.8, 0.01, 3.0);
RooRealVar mean2("mean2", "mean of gaussians", 6, 5, 10);
RooRealVar sigma2("sigma2", "width of gaussians", 1.0, 0.01, 3.0);
RooGaussian sig1("sig1", "Signal component 1", x, mean1, sigma1);
RooGaussian sig2("sig2", "Signal component 2", x, mean2, sigma2);

// Sum the signal components
RooRealVar sig1frac("sig1frac", "fraction of signal 1", 0.5, 0.0, 1.0);
RooAddPdf sig("sig", "g1+g2", {sig1, sig2}, {sig1frac});

// Sum the composite signal and background
RooRealVar sigfrac("sigfrac", "fraction of signal", 0.4, 0.0, 1.0);
RooAddPdf model("model"), "g1+g2+a", {sig, expo}, {sigfrac});
Backup

Share of fitting time for 700 parameters

RooFit Num-Diff
- Seeding Time: 130 ms
- Minimization Time: 11700 ms

Code Squashing Num-Diff
- Seeding Time: 723 ms
- Minimization Time: 51762 ms

Code Squashing AD
- Seeding Time: 652 ms
- Minimization Time: 730 ms

Seeding uses numerical differentiation = Larger times for AD
Possible Fix? Use AD here too!

Seeding: initial parameter scale estimation to get the step size for the minimization.
Adding Automatic Differentiation to RooFit - Garima Singh | Differentiable and Probabilistic Programming for Fundamental Physics TWS 2023
Backup

How models are translated

```cpp
// Declare the code
gInterpreter->Declare(code.c_str());
// Get the derivatives of 'code'
gInterpreter->ProcessLine("clad:gradient(code);");
// Use code_grad in wrappers that interface with
// the minimizer.
```

The parent node queries the results from the child nodes.
Backup

Numerical error and convergence rates: EDM vs Iterations

Large number of parameters usually causes numerical issues\(^3\) with minimizations, leading to fluctuation in step sizes and eventually leading to longer or no convergence.

\(^3\) : https://root.cern.ch/root/htmldoc/guides/minuit2/Minuit2.html#convergence-in-mboxmigrad-and-positivedefiniteness

Adding Automatic Differentiation to RooFit - Garima Singh | Differentiable and Probabilistic Programming for Fundamental Physics TWS 2023
The granularity of the RooFit computation graph that represents a HistFactory model is too high. It caches the result of relatively simple operations, so the caching logic is more expensive than re-evaluating the model.

However, these results inspired us to do some optimizations in HistFactory, so by now RooFit should be again on par with code squash num-diff or even better!
Automatic Differentiation (AD) is a set of techniques to evaluate the exact derivative of a computer program.

- Faster than numerical differentiation - scales better for problems with large number of parameters.
- More accurate than numerical differentiation - fewer numerical errors!
Introduction

What is Automatic Differentiation?

Simply put, it's a way for computers to differentiate computer programs. AD applies the chain rule of differential calculus throughout the semantics of the original program.

\[
\begin{align*}
\frac{\partial z}{\partial x} &= y \\
\frac{\partial z}{\partial y} &= x \\
\frac{\partial f}{\partial z} &= 1 \\
\frac{\partial f}{\partial y} &= 1
\end{align*}
\]

\[
\begin{align*}
f'(x, y)_x &= y \\
f'(x, y)_y &= x + 1
\end{align*}
\]
Introduction

Why AD over numerical differentiation?

- Calculates exact derivatives of programs, free from numerical errors.
- More performant for functions with high number of parameters.

Difficulty in choosing step size due to numerical error

Comparison between Clad’s AD and numerical diff

https://commons.wikimedia.org/wiki/File:AbsoluteErrorNumericalDifferentiationExample.png

Adding Automatic Differentiation to RooFit - Garima Singh | Differentiable and Probabilistic Programming for Fundamental Physics TWS 2023
Introduction

Source Code Transformation Based Automatic Differentiation

Automatic Differentiation (AD) is a set of techniques to evaluate the exact derivative of a computer program.

- Faster than numerical differentiation - scales better for problems with large number of parameters.
- More accurate than numerical differentiation - fewer numerical errors!

Source code transformation based AD synthesizes derivative code from the internal representation of the target program.
void RooGaussian::translate(ADDetail::CodeSquashContext &ctx) const
{
    // Build a call to the stateless gaussian.
    std::string const& xName = ctx.getResult(&x.arg());
    std::string const& muName = ctx.getResult(&mean.arg());
    std::string const& sigName = ctx.getResult(&sigma.arg());
    std::string const& ResName = "ADDetail::gauss(" + xName + "," + muName + "," + sigName + ")";
    ctx.addResult(this, ResName);
}
Motivation

Why AD in RooFit?

Usual RooFit is performant even with numerical-diff because of its complex caching logic.

However, even if this caching would be done at a very granular level, it has lots of overhead from virtual calls and bookkeeping, which is why we expect AD to be superior.