Automatic Differentiation in CMS Combine: From Integration to Physics-Grade Fits

Jonas Rembser*, Vassil Vassilev+
{ *CERN, +Princeton/compiler-research.org }

This work is partially supported by National Science Foundation under Grant OAC-2311471

Motivation

Likelihoods are central for High Energy Physics

$$L(\vec{n},\vec{a}|\vec{\eta},\vec{\chi}) = \prod_{c\in unbinned\ ch}\prod_{i\in obs}\frac{f_c(\vec{x}_{ci}|\vec{\eta},\vec{\chi})}{\int f_c(\vec{x}_{ci}|\vec{\eta},\vec{\chi})\,d\vec{x}_c}\cdot \prod_{c\in binned\ ch(analytical)}\prod_{b\in obs} Pois(n_{cb}|\nu(\vec{\eta},\vec{\chi}))\cdot\prod_{\chi\in\vec{\chi}} c_\chi(a_\chi|\chi)$$

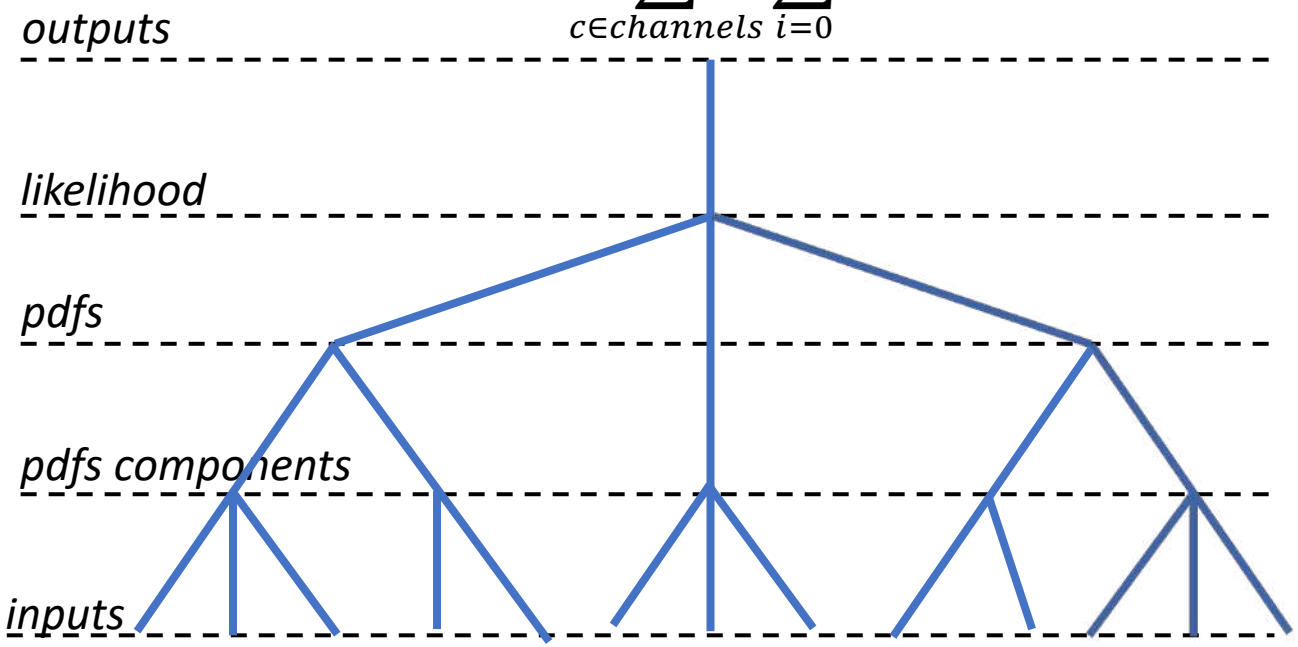
Numerical and analytic integrals

$\vec{n}: data, \vec{a}: auxilary\ data, \vec{\eta}: unconstrained\ parameters, \vec{\chi}: constrained\ parameters$

CMS Combine Paper https://arxiv.org/pdf/2404.06614

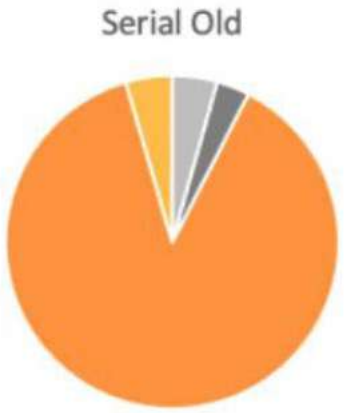
Object Oriented Math. Compute Cost

$$NLL = -\sum_{c \in channels} \sum_{i=0}^{N_c} \log\left(Model(x_i)\right)$$



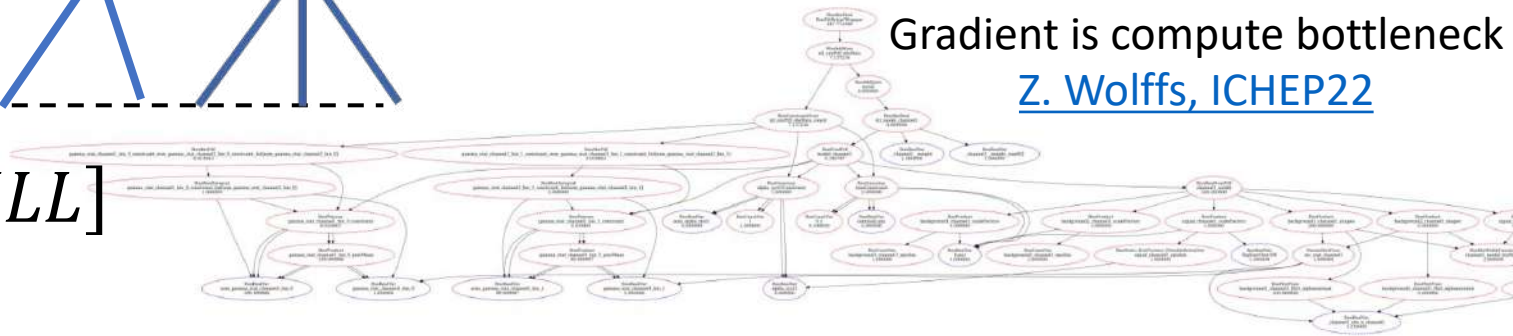
$$(\hat{\eta}, \hat{\chi}) = \arg\min_{\eta,\chi}[NLL]$$

serial old	
roofit_setup	313
migrad_seed	230
migrad_gradient	6289
migrad_descent	323



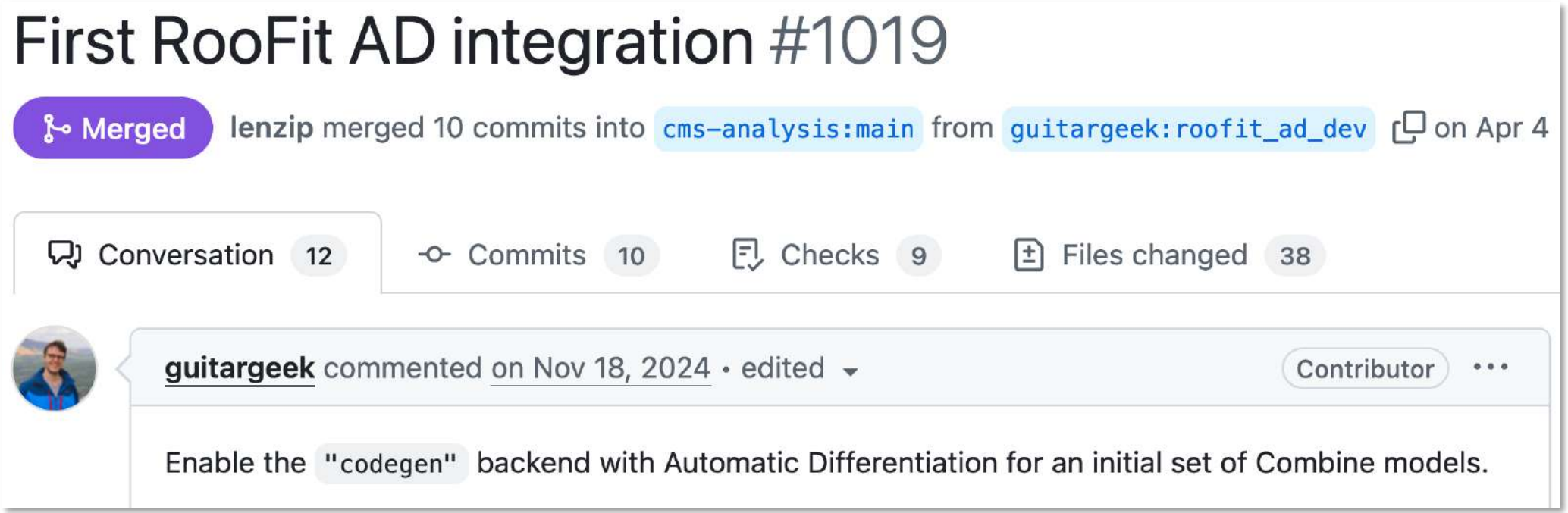
Gradient is compute bottleneck

Z. Wolffs, ICHEP22

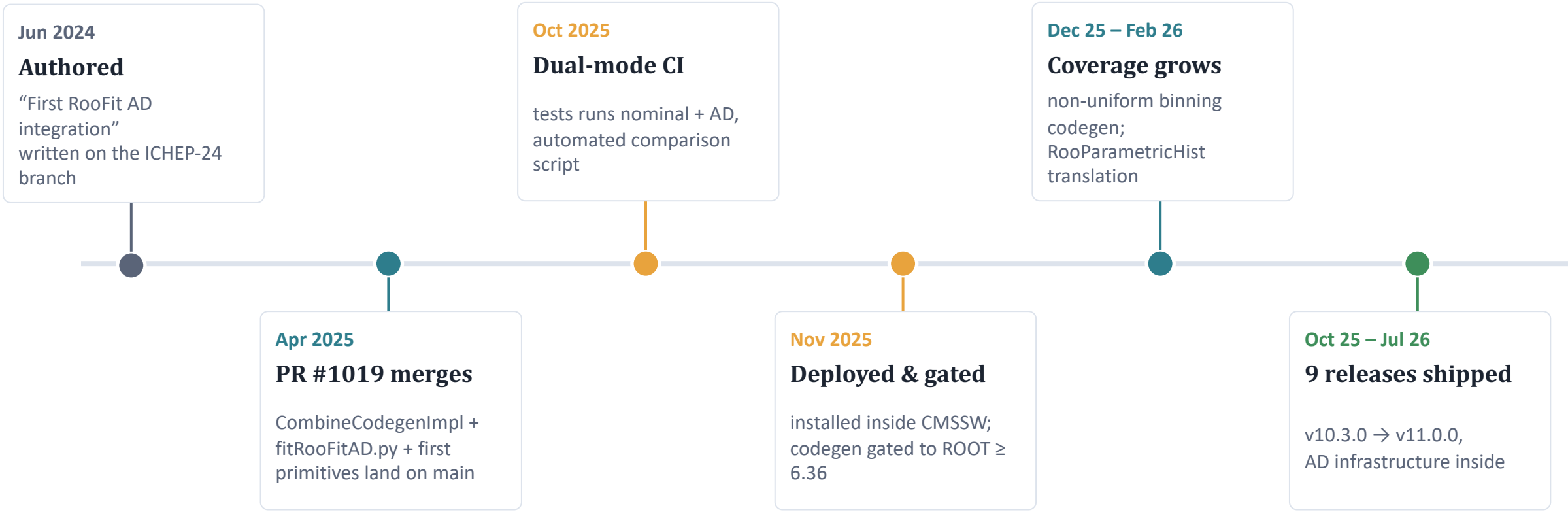


Integration in CMS Combine

Work steered mostly via CAT hackathons. Thank you Aliya Nigamova and Piergiulio Lenzi!



From prototype to deployment



Authored Jun 2024 · mainlined Apr 2025 · institutionalized through 2026. The prototype was never rewritten — it was progressively merged, installed, gated, tested, and released.

Small Things That Matter

Generated gradients are only as trustworthy as the machinery that checks them:

- Dual-mode CI
  Reference tests executes twice: numeric-diff and AD backends with an automated result-comparison script. Divergence fails the build, not the physicist.
- Test suite modernization
  Makefile → CMake; gtest/CTest unit suite; programmatic NLL construction (combineCreateNLL exposed to Python) so likelihoods are testable without the CLI.
- Compiler-matrix coverage
  LCG views spanning ROOT 6.26 → 6.36 → master, alongside CMSSW (incl. gcc13 / CMSSW 16). AD baseline pinned; regressions localized to a toolchain.
- CodeCoverage
  The compiler side instruments the physics tool's tests. Cross-checking runs in both directions.

Combine Supported Primitives Today

Each class was first refactored into stateless free functions (CombineMathFuncs), then given a codegen translation. **The unmerged row is where that refactor is harder.**

Primitive	Role in CMS models	Codegen / AD status
`VerticalInterpPdf`	template morphing (pdf form)	● **merged Oct 2025**
`FastVerticalInterpHistPdf2(+2D)`	fast vertical morphing; non-uniform binning added	● **merged Dec 2025**
`ProcessNormalization / AsymPow`	yield systematics (log-normal, asymmetric)	● **merged Feb 2026**
`RooParametricHist`	data-driven backgrounds; refactored into CombineMathFuncs	● **merged Feb 2026**
`CMSHistFunc / CMSHistSum`	the core binned-template classes of nearly every CMS analysis	◐ **open PR chain #1162 → #1191 → #1252**

Stateful Ops: CMSHistSum

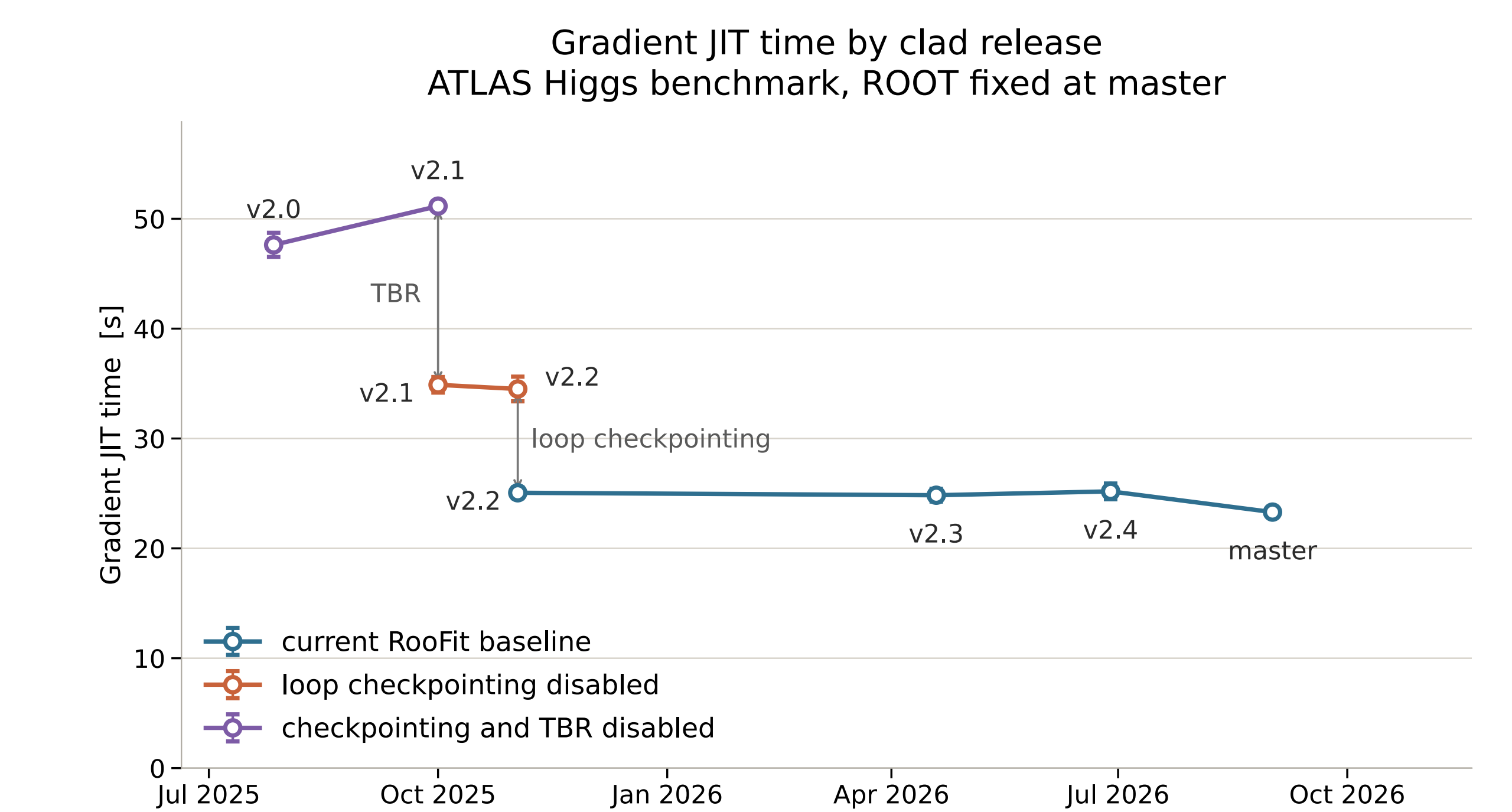
AD resists statefulness:

- Morphed templates look constant but are not: every evaluation re-morphs nominal/up/down over the vertical-morph nuisances (smooth quad-interp / lin-extrap; log-space + exp + renorm for LQL)
- Evaluation is incremental, not pure: mutable caches + dirty-flag sentries; delta-updates assume history, codegen assumes a function
- Barlow-Beeston bin parameters hide in the cache: per-bin stat nuisances (explicit / Poisson / Gaussian) baked in via getVal() at codegen time -> their gradients silently come out zero
- Naive snapshot codegen (PR #1252, ~90 lines) differentiates only the process yields; integrals are literal dummies ("return 2.0") -- wrong answers, not crashes: only dual-mode CI catches this

This is an engineering problem not a research one: rethink the implicit math relationships through state by reuse FastVerticalInterpHistPdf2's stateless kernel (same morph math, already in CombineMathFuncs) + symbolic BB terms; analytic-BB freezing is exact for gradients (envelope theorem)

A Year of RooFit Codegen: Infrastructure

- Single-call JIT of all generated code (Nov 2025) + loop checkpointing: −25% gradient JIT time on Higgs combination models
- Checkpointing selectively removed for flexibleInterp (Aug 2026) — benchmarks showed recomputation costlier than tape there. Measured engineering, not dogma
- setData() for codegen likelihoods (Feb 2026): one generated function reused across datasets — the toy-fitting workflow Combine needs
- Graceful degradation: EvalBackend fallback without Clad (Oct 2025), R__HAS_CLAD (Dec 2025); categorical parameters in codegen (Aug 2025)


Gradient JIT time by clad release
ATLAS Higgs benchmark, ROOT fixed at master
Gradient JIT time [s]
0
10
20
30
40
50
Jul 2025
Oct 2025
Jan 2026
Apr 2026
Jul 2026
Oct 2026
v2.0
v2.1
TBR
v2.1
v2.2
loop checkpointing
v2.2
v2.3
v2.4
master
current RooFit baseline
loop checkpointing disabled
checkpointing and TBR disabled

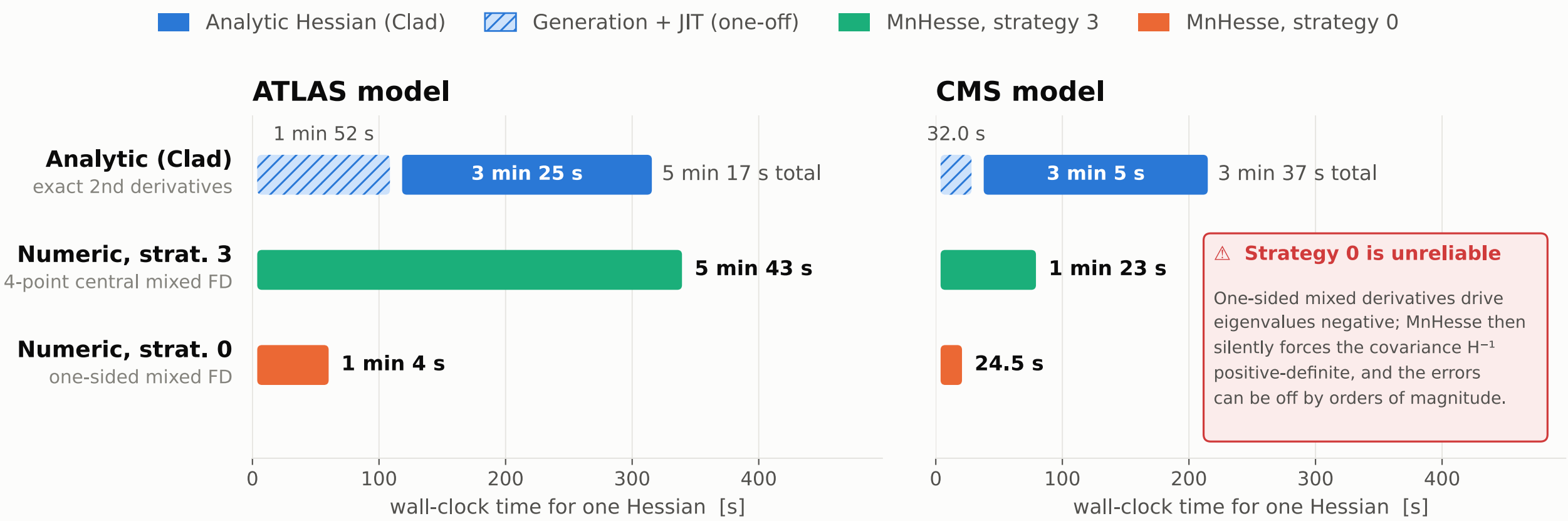
A Year of RooFit Codegen: Capability

- Analytic Hessians via Clad, forwarded to Minuit 2 (Feb 2026)
- **RooONNXFunction (Apr 2026): ONNX networks JIT-compiled via SOFIE inside RooFit, differentiable through Clad — neural surrogates in the likelihood, natively**
- SOFIE inference code refactored for AD-friendliness: explicit primitives with custom pullbacks, no tapes in NN gradients (Apr 2026)
- Special-function derivatives: custom pushforwards for inc_gamma & Landau, closed-form Landau pullbacks (Aug 2026)
- Five Clad releases integrated in the window: v2.0 (Jul 2025) → v2.4 (Jun 2026)

Higher-order: Clad Hessians in RooFit

- Infrastructure merged in ROOT Feb 2026: Clad-generated analytic Hessians via codegen, forwarded to Minuit 2, with validation in the codegen debug macro
- Merged before it could run — available in the upcoming Clad v2.6 release
- Targets: covariance / HESSE-equivalent errors. Status in Combine: not yet wired into the fit routines
- Analytic Barlow–Beeston profiles bin parameters in closed form before each NLL evaluation
- **Gradients: freezing profiled parameters is exact (envelope theorem) — this is why the Oct 2025 restructure enables AD**
- **Hessians: not exact — second derivatives through profiled parameters need the implicit-function correction**
- Hessians are exact for models without analytic BB; with it, correction term is future work

Physics-grade fits: Run-3 Higgs measurements



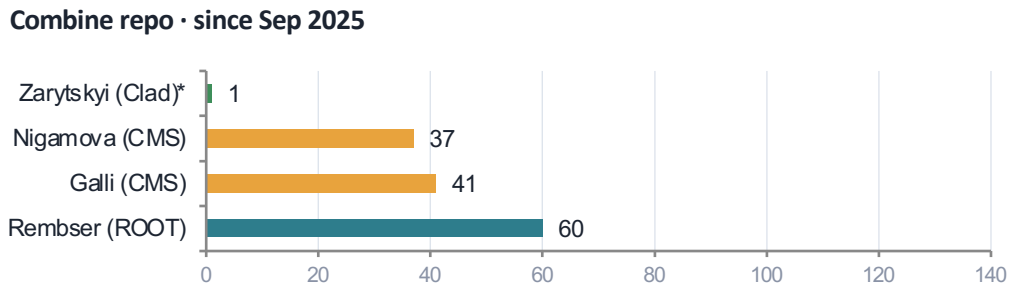
Lessons Learnt

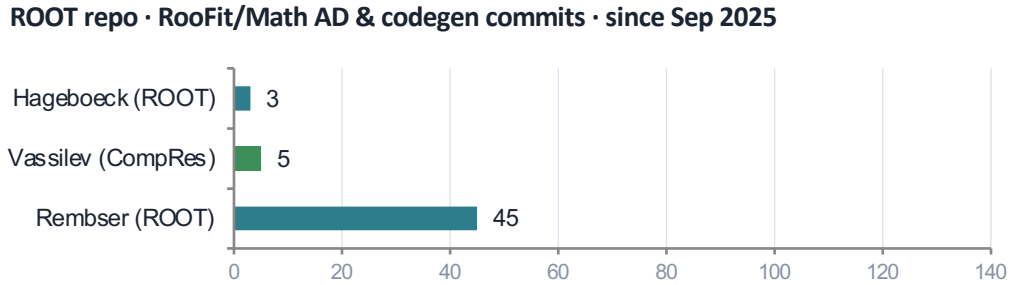
- Stateful evaluation patterns
  Mutable caches, dirty-flag sentries, Δ-update loops: 15 years of CPU optimization is the exact inverse of what codegen needs. The fix is refactor-to-free-functions
- Snapshot-style codegen prototypes
  Freezing cached values silently zeroes nuisance gradients: the failure mode is wrong answers, not crashes. Only dual-mode CI catches it.
- Debugging generated gradients by hand
  Does not scale past toy models. The answer was machinery (comparison scripts, dual-mode CI), not heroics. Maybe AI-assisted codegenImpl/pullback scaffolding — acceptable because dual-mode CI validates the output

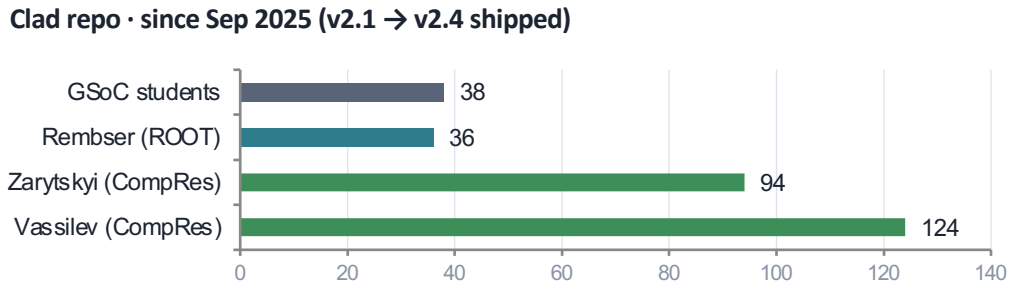
AD became a forcing function for software quality. The refactors it demanded (CMake, unit tests, stateless kernels) leave Combine better even where AD is off.

A Stark Example of Bottom Up Collaboration

- **No single participant could have justified this alone; none paid more than a marginal cost**
- Compiler Research / IRIS-HEP — compiler expertise + the student pipeline (GSoC & fellows shipping to a flagship CMS tool, incl. the CMSHist codegen prototype)
- ROOT — made the codegen/AD framework general upstream; CMS consumed the capability downstream
- CMS / Combine maintainers — carried the domain refactor (CombineMathFuncs, dual-mode CI); no new headcount
- The coalition commits across ALL THREE repositories: the RooFit maintainer is a top-3 Clad committer; the Clad lead integrates in ROOT; the Clad team commits to Combine
- The multiplier: a shared technical vision held across funding cycles and personnel churn
- **All started with our talk at MODE and with the help of Pietro!**







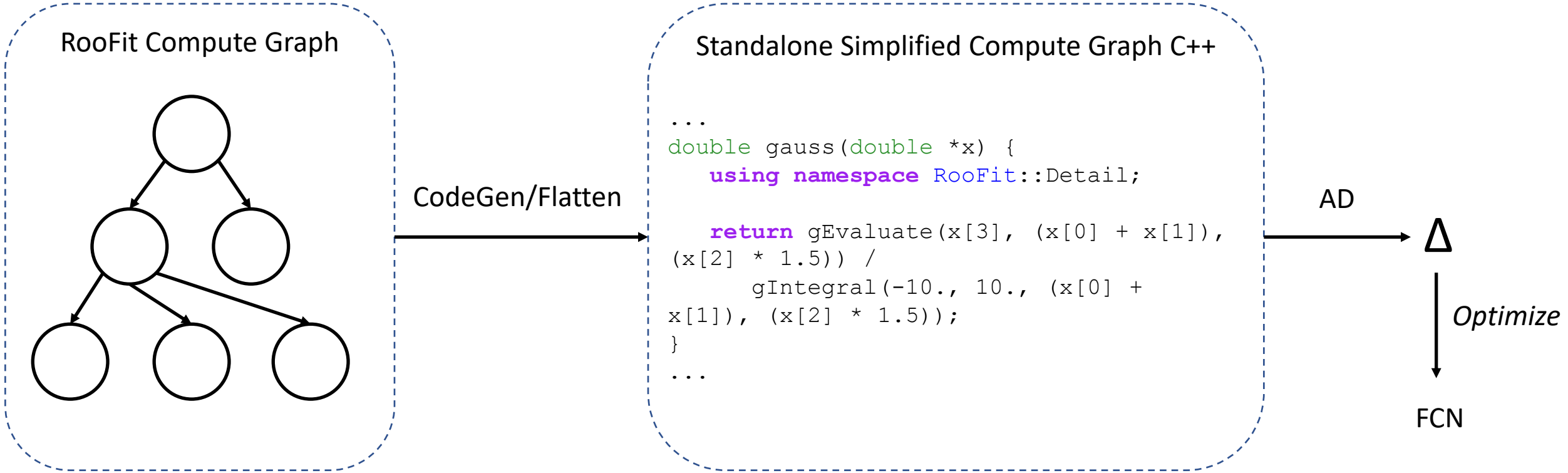
**plus the CMSHist codegen prototype PR & coverage CI. Same axis scale on all three; teal = ROOT, amber = CMS, green = Compiler Research, gray = GSoC (Goyal 26 + Elrawy 12).*

Conclusion & Future Work

- Release-gated, CI-validated AD path shipped across nine Combine releases
- Primitive coverage complete except the CMSHist frontier — whose anatomy we now understand precisely
- Clad scaling work + RooFit Hessians, with scope stated exactly
- Merge the scoped CMSHistSum codegen (vertical + explicit BB) — the path is drawn, the kernel exists
- Wire Clad Hessians into Combine's error estimation; implicit-function correction for analytic BB
- Carry the recipe to other RooFit-based frameworks — talk to us this week

Thank you!

Backup: Clad as RooFit's AD Engine




```
pdf.fitTo(data, RooFit::EvalBackend("codegen"))
pdf.createNLL(data, RooFit::EvalBackend("codegen"))
```

Most of HistFactory RooFit primitives are supported. Please reach out if you need additional primitive support