Track reconstruction with mkFit and developments towards HL-LHC

mkFit team for CMS Collaboration

A.R. Hall[2], A. Yagil[1], D.S. Riley[5], E. Vourliotis[1], G. Cerati[3], L. Giannini[1], M. Kortelainen[3], M. Masciovecchio[1], M. Tadel[1], P. Gartung[3], P. Elmer[4], P. Wittich[5], **S. Krutelyov**[1], S.R. Lantz[5], T. Reid[5]

1. UCSD, 2. USNA Annapolis, 3. Fermilab, 4. Princeton, 5. Cornell

Connecting the Dots 2023

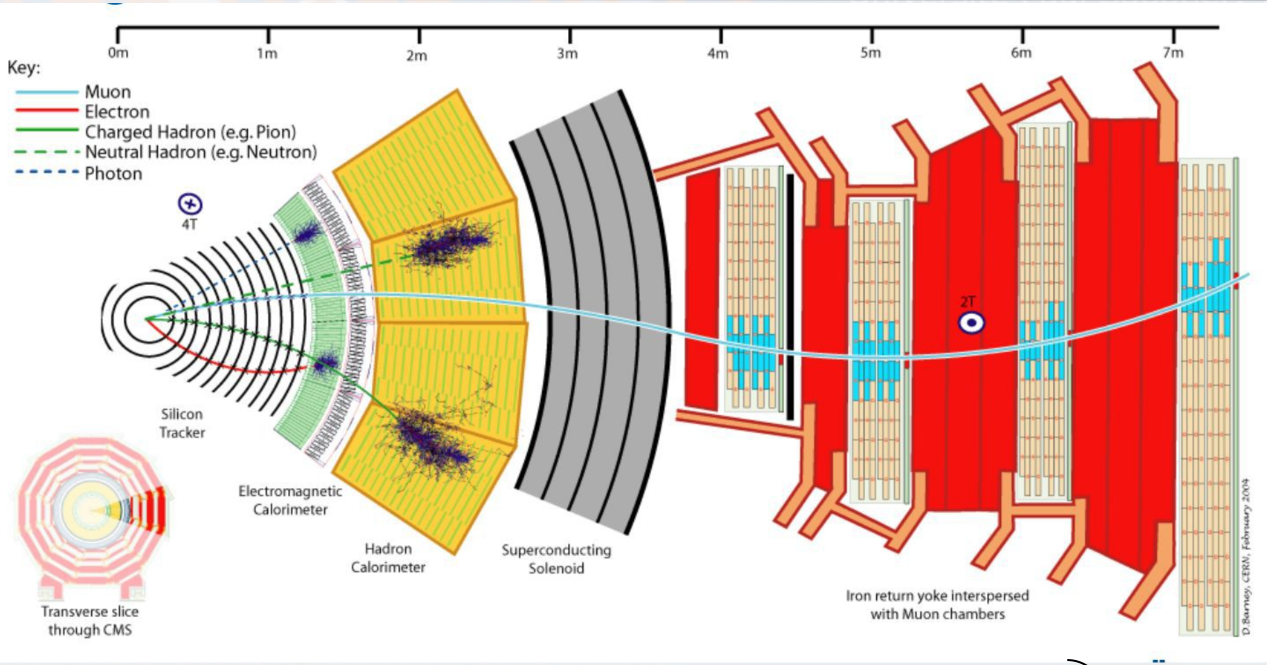
https://indico.cern.ch/event/1252748

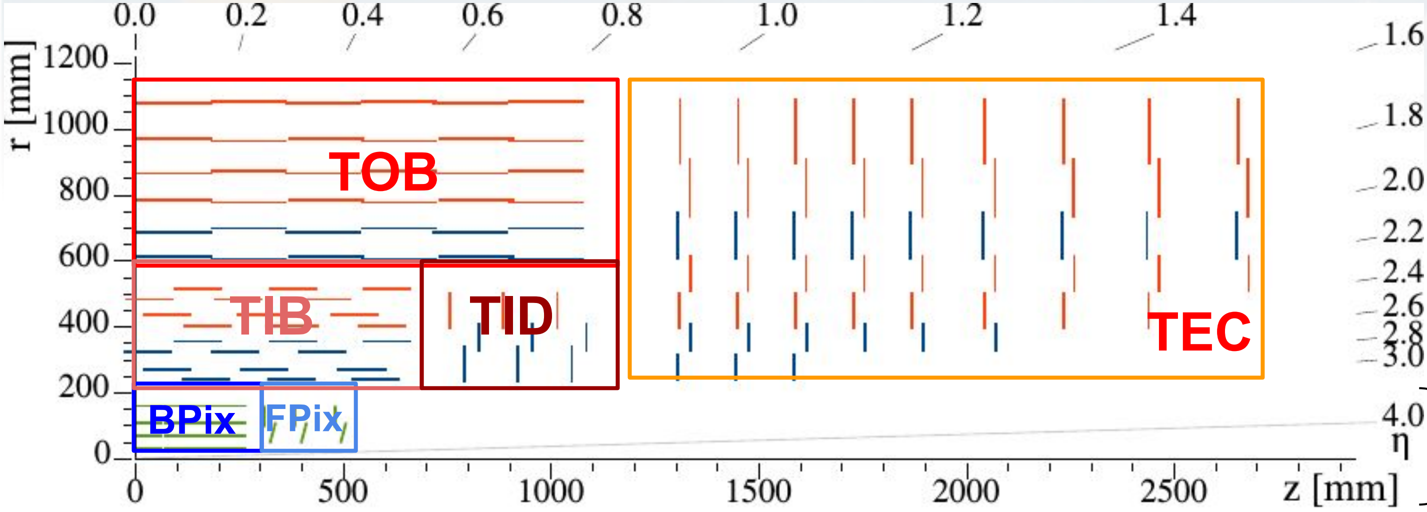
Toulouse, 10th Oct 2023

Outline

- Introduction to CMS tracking and mkFit
- mkFit in CMS: usage & performance in Phase-1/Run 3
- Work towards HL-LHC

CMS tracking: detector

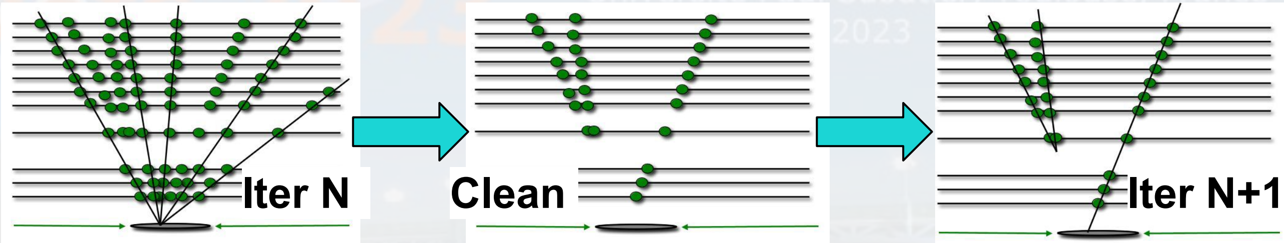




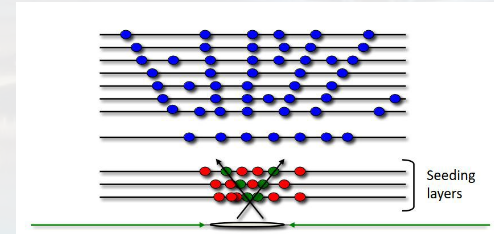
- Micro-strip sensors
- Stereo modules (two components with a 0.1 rad stereo angle)
- Analog readout

- Pixel sensors
- ~124M channels
- Digital readout

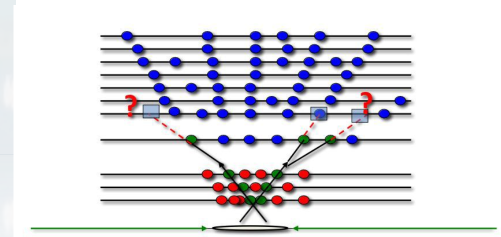
CMS tracking: iterative tracking



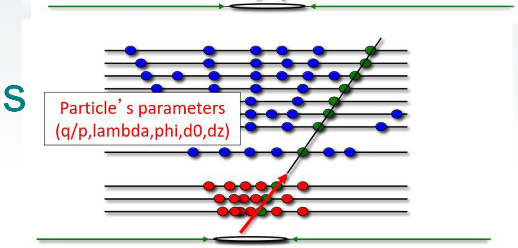
- Iterative steps
- Seeding:
  - provides track candidates, with an initial estimate of the trajectory parameters and their uncertainties (use combination of pixel, strip or mixed hits)



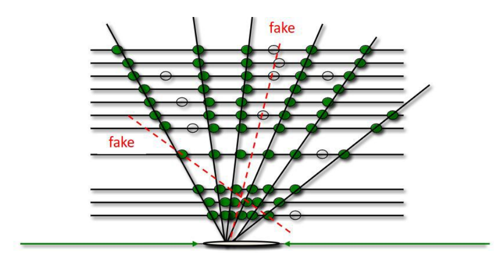
- Pattern recognition:
  - hits compatible with the predicted track position are added (Kalman update) to the trajectory and track parameters are updated
  - using Combinatorial Kalman Filter (CKF) or mkFit



- Final fit:
  - take into account non-uniform B-field and material details
  - get the best estimate of (5) parameters of the smoothed trajectory combining all hits (outlier hits are rejected)



- Selection:
  - sets quality flags using ML/MVA (more than 20 inputs)
  - aims to reject fake tracks; tracks sharing too many hits are also cleaned as duplicates
  - hits on high quality tracks are removed for next iterations



Introduction to mkFit ⇒ Matriplex Kalman trajectory Fitter

- Parallelized and vectorized track finding and fitting
  - Parallelization through Intel TBB
  - Vectorization via SIMD pragmas (mostly in propagation) and Matriplex (Kalman operations)
    - Made possible by generalizing detector geometry and its traversal so that sets of track candidates undergo the same operations
- Matriplex: classes for vectorized operations on a set of matrices / vectors
  - Includes code generator for optimized matrix multiplication code:
    - fixed element 0 or 1 values – can reduce number of operations by 50%
    - inline transpose
    - generates regular matrix calculation C++ code or intrinsics (FMA supported)
- A three line history
  - 2014 – explore vectorized fitting (Xeon Phi) → success → track finding for high-PU environments
    - Goal: Attempt to keep mkFit core experiment-independent
  - 2018 – decent CMS prototype → improve precision, low-p_T performance → configurability
    - accompanying paper JINST 15 (2020) 09, P09030
  - 2022 – inclusion into CMSSW (CMS software) → start preparing for HL-LHC / Phase-2
    - stand-alone mode of operation is still supported

mkFit in CMS iterative tracking

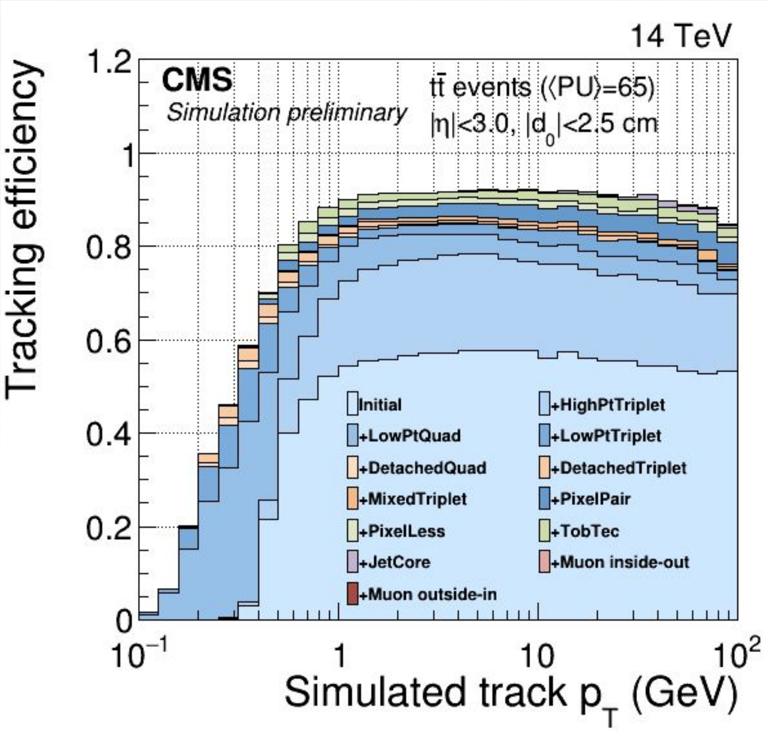
- CMS tracking has 11 general tracking iterations (+3 regional for jets and muons), starting from prompt, pixel-based seeds, then swiping up the rest
- mkFit is introduced since Run 3 and currently used for 5 general iterations (≈90% of all reconstructed tracks with $p_T > 0.5$ GeV)

	Iteration	Seeding	Target track
mkFit	Initial, pre-splitting	Pixel quadruplets (before cluster splitting)	Prompt, high-p_T (JetCore tracking regions)
mkFit	Initial	Pixel quadruplets	Prompt, high-p_T
	LowPtQuad	Pixel quadruplets	Prompt, low-p_T
mkFit	HighPtTriplet	Pixel triplets	Prompt, high-p_T recovery
	LowPtTriplet	Pixel triplets	Prompt, low-p_T recovery
mkFit	DetachedQuad	Pixel quadruplets	Displaced--
mkFit	DetachedTriplet	Pixel triplets	Displaced-- recovery
	MixedTriplet	Pixel+strip triplets	Displaced-
	PixelPair	Pixel pairs	Displaced- recovery
	PixelLess	Inner strip triplets	Displaced+
	TobTec	Outer strip triplets	Displaced++
	JetCore	Pixel pairs within jets	Within high-p_T jets
	Muon inside-out	Muon-tagged tracks	Muons
	Muon outside-in	Standalone muons	Muons

Tracker-only track candidates (Initial, pre-splitting through JetCore)

All track candidates (all iterations)

From CMS-DP-2022-018

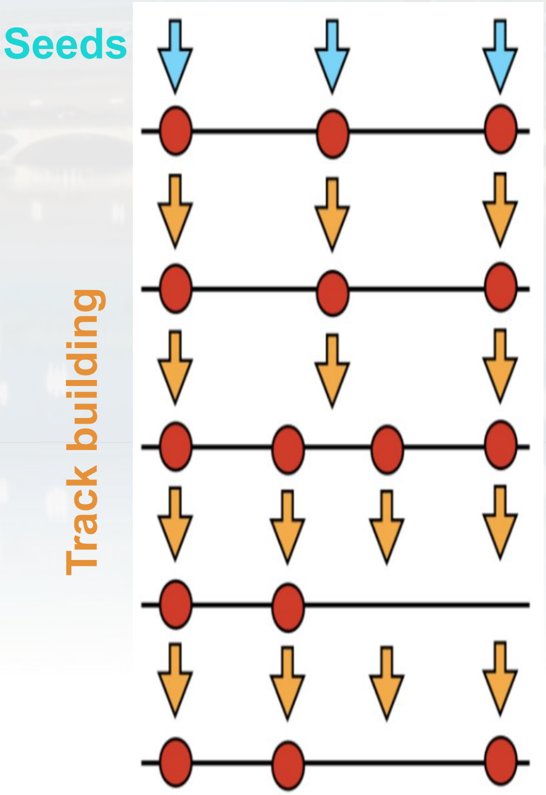


* In CMS-DP-2022-018, mkFit is also used in PixelLess iteration

mkFit in CMS - the tracking workflow

In iterations using mkFit, the tracking workflow consists of the following tasks:

- pre-mkFit: seed finding
- mkFit: track building
  - Seed cleaning (if needed):
    - mkFit processes seeds in parallel
      - can not rely on claimed hits to discard seeds
  - Seed partitioning and sorting:
    - 5 partitions in η:
      - barrel /|η|~0.8/ transition$^{\pm}$ /|η|~1.6/ endcap$^{\pm}$
    - sorting in { η, φ } with `Binnor<>`
  - Forward search with quality filtering (optional)
  - Backward fit / search with quality filtering
  - Duplicate removal
- post-mkFit: final-fit, and track quality flagging/selection

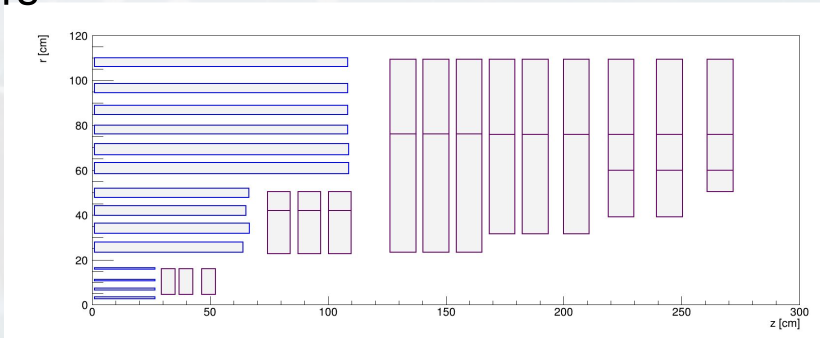


Sorting with Binnor<>

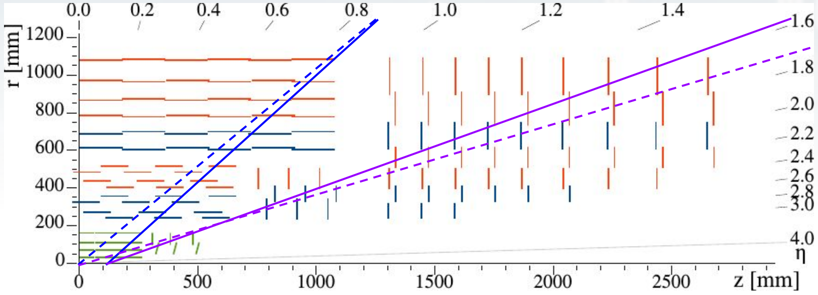
- Fast 2D nearest neighbor search on a grid
  - Generalization of algorithm initially developed for pre-selecting hits.
  - Now also used for seed cleaning, seed partitioning, and duplicate removal.
- Specify two axes (like histogram: N_{bins}, min, max)
  - U(1) type (cyclic) supported → used for ordering in φ
  - Uses bit packing to minimize memory usage (and cache misses)
- Lookup structures created by sorting of registered entries
  - { start, size } pairs are stored for each bin
  - Uses Radix sort

Geometry description and traversal

- Detectors split into mkFit layers (72 layers in phase-1/Run 3 CMS)
  - Potentially finer granularity than readout / construction
    - E.g., mono/stereo treated as separate layers



- Layer as an mkFit tracking concept:
  - Track search proceeds through a sequence of layers → called a LayerPlan
    - Plans differ for barrel / transition / endcap
  - This allows parallel processing of multiple tracks as we do not deal with individual modules



- Recent updates in mkFit
  - On-the-fly extraction of layer envelopes/gaps
  - Add module-id information to hits to allow for overlap hit collection

Single block memory allocation

- Memory for all track candidates, including hit-on-track information is acquired in a single allocation and distributed sequentially (dealloc is a no-op).
  - Reduce allocation and deallocation overhead while still using `std::vector`s.
  - Vector-gather (vgather) instruction is used to fill Matriplex's with input data

mkFit Configuration system & classes

- Each tracking iteration needs to be separately configurable.
  - class `IterationConfig` → top-level configuration → which tasks to perform
    - parameters for seed & duplicate cleaning
    - includes `LayerPlan` and the following classes
  - class `IterationParams` → tracking parameters, e.g., max # of holes, χ^2 cuts; quality filter params
    - can be different for forward / backward search
  - class `IterationLayerConfig` → parameters specific to layers, hit search windows; one per layer (72 layers in phase-1/Run 3)!
- In CMSSW (or any other multi-threaded framework) configuration is required to be completely separable → instantiated and managed independently
  - Tracking iterations are configured via Python.
  - This works well for relatively small number of parameters. mkFit full configuration as described above is more complex.
- As a compromise, all mkFit configuration can be loaded (and saved) into JSON
  - Reading of partial JSON overrides is fully supported – patch mode:
    - read full configuration from CMSSW release
    - override desired parameters with a simple additional JSON file
  - Frequently used parameters can also be set via Python (in particular, for heavy-ion operations)
  - Plugin-style configuration is still supported in stand-alone mode

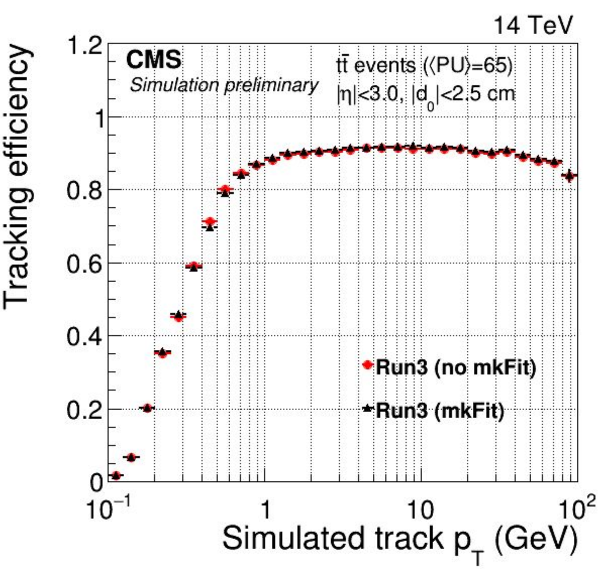
"Standard" functions

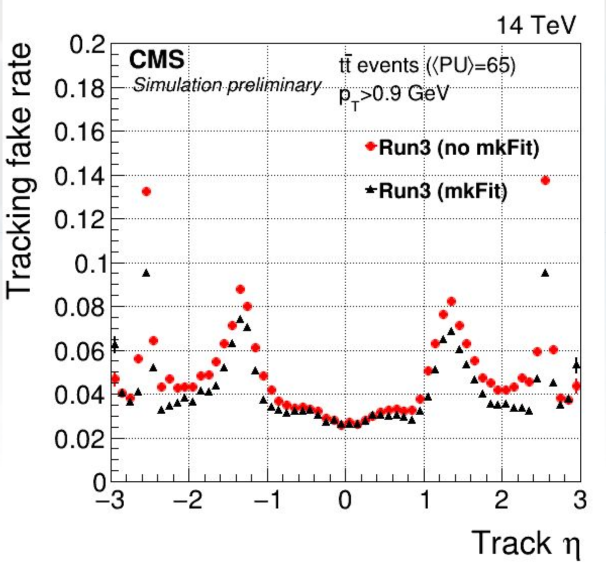
- To support multiple iterations and Phase-2 geometry it became obvious we would need to introduce a more flexible configuration mechanism for the following tasks:
    - seed cleaning & partitioning – per iteration
    - candidate filters: pre- and post-backward fit – per iteration
    - duplicate cleaning – per iteration
    - candidate scoring – per iteration with possible per region override
  - Stuffing extra parameters into `IterationConfig` & friends can not scale
- Solution: use `std::function<task_func_type>` catalogs with string keys
  - Populate the catalogs via static object initializers in source files that contain the task code
    - can all be hidden in anonymous namespaces
    - function templates can be used to inject compile-time parameters
    - can even be lambdas for simple cases
  - JSON files specify the names / strings for the functions to be picked
  - After configuration loading / setup is complete the names get resolved into `std::functions<>` and become available through `IterationConfig`

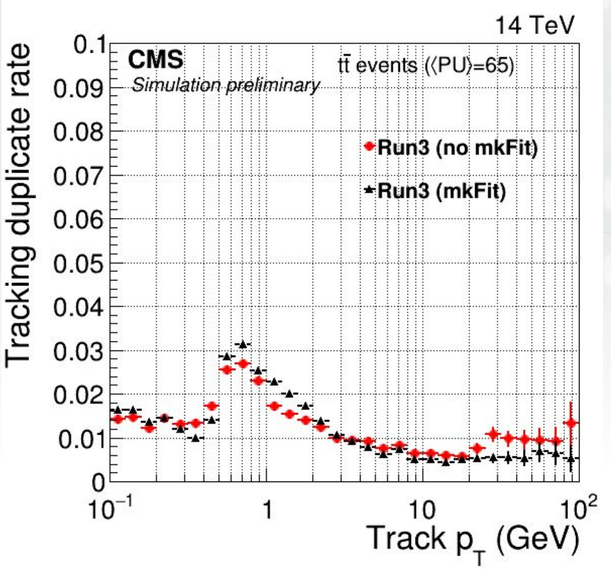
mkFit in CMS - physics performance

From CMS-DP-2022-018 (*where mkFit is also used in PixelLess iteration)

- Tracking **efficiency comparable**: Small gains in endcap (2.4 < |η| < 2.8)
- Tracking **fake rate better overall**: Fake rate reduction with increasing |η|
- Tracking **duplicate rate slightly increased**: Can be mitigated by dedicated duplicate removal.







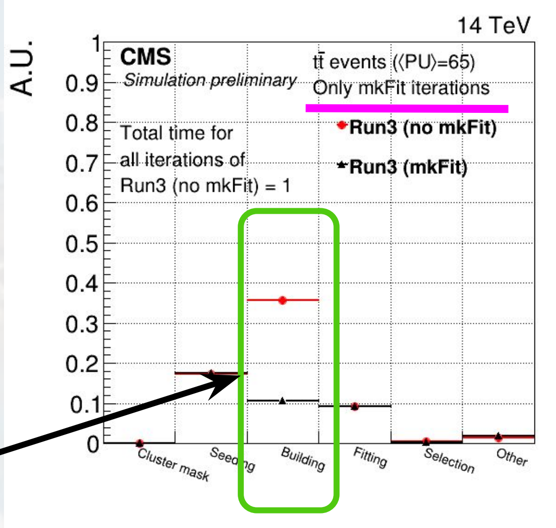
- (*) pixelLess iteration was switched back to CKF in 2022 after inefficiencies were found for low-p_T very displaced tracks as in Λ and Ξ decays

mkFit in CMS - computational performance

From [CMS-DP-2022-018](CMS-DP-2022-018) (*where mkFit is also used in PixelLess iteration)

- Vectorization and threading scaling tests for initial iteration show (according to Amdahl's Law)
  - ~70% of operations effectively vectorized.
  - >95% of code effectively parallelized.
- Computational speedups when using mkFit:
  - Individual mkFit iterations: 2.7x to 6.7x building time reduction
    - varies depending on quality of seeds/candidates
    - seed cleaning and duplicate merging are not vectorized and some iterations need more seed/duplicate cleaning
    - CKF sequential processing of seeds can skip building a seed if its hits were already used. mkFit needs to process all seeds (after cleaning) independently for effective vectorization

☝ Hidden cost of vectorization



Single-threaded measurements on
1 Intel® Xeon® Gold 6130 CPU @ 2.10GHz,
local access to inputs

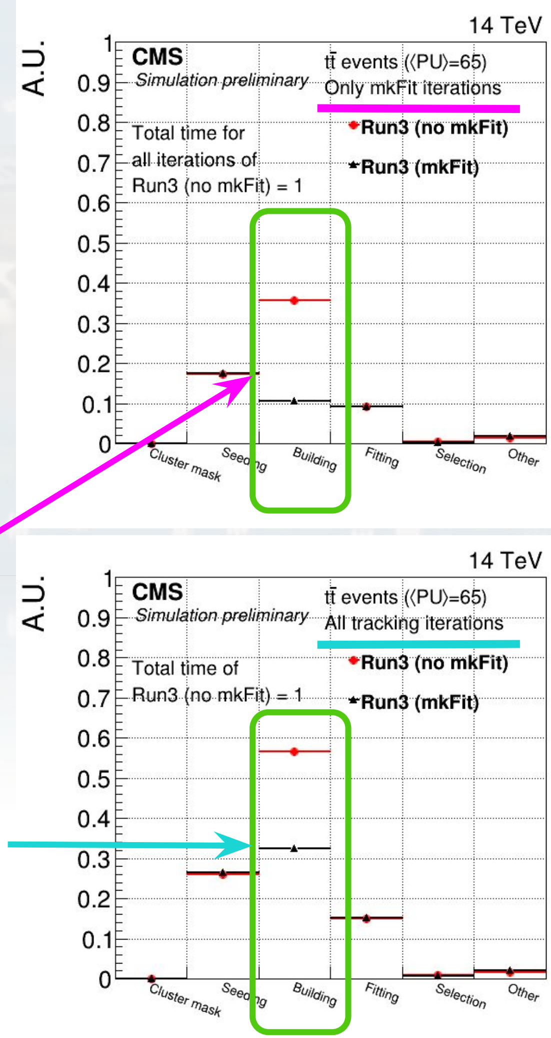
mkFit in CMS - computational performance

From CMS-DP-2022-018 (*where mkFit is also used in PixelLess iteration)

- Vectorization and threading scaling tests for initial iteration show (according to Amdahl's Law)
  - ~70% of operations effectively vectorized.
  - >95% of code effectively parallelized.
- Computational speedups when using mkFit:
  - Individual mkFit iterations: **Up to 6.7x** building time reduction
  - Sum of mkFit iterations: **~3.5x** building time reduction
    - **Track building with mkFit costs less than seeding, ≈ fitting**
  - Sum of all iterations: **~1.7x** building time reduction

⇒ 25% reduction of total tracking time

⇒ Event throughput increase by 10-15% in Run-3



Single-threaded measurements on 1 Intel® Xeon® Gold 6130 CPU @ 2.10GHz, local access to inputs

Ongoing & Future work: Phase-1/Run 3

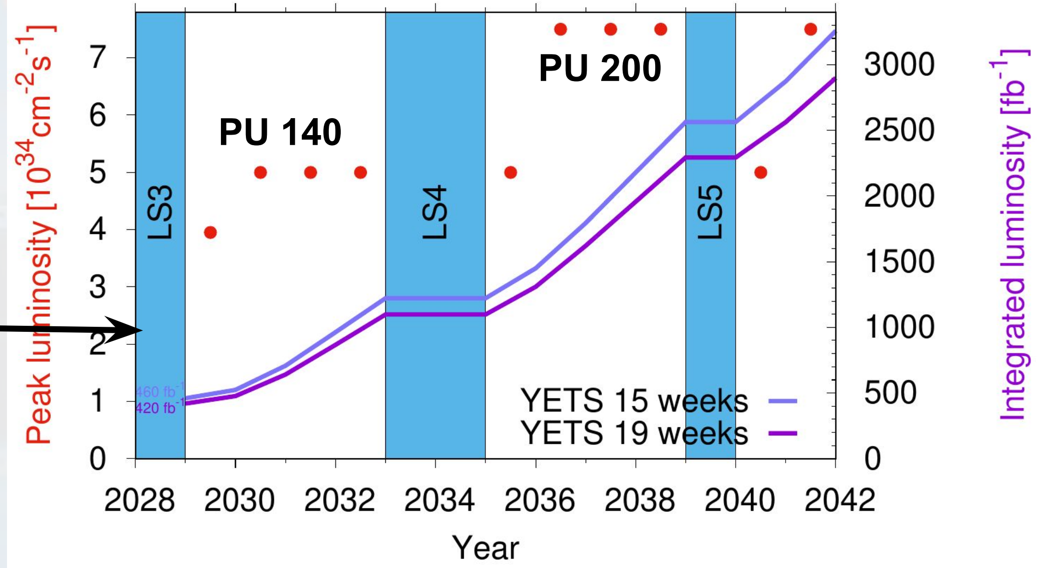
- Use the described changes to further tune Phase-1 CMS iterations
  - Especially track scoring ⇒ use mkFit for more than 5 current iterations
- Final-fit now the most time-consuming tracking task in iterations using mkFit
  - ⇒ Explore how mkFit could be used effectively in this area
    - In parallel, this can also improve backward-fit and backward-search in mkFit

Towards HL-LHC

http://lhc-commissioning.web.cern.ch/schedule/HL-LHC-plots.htm

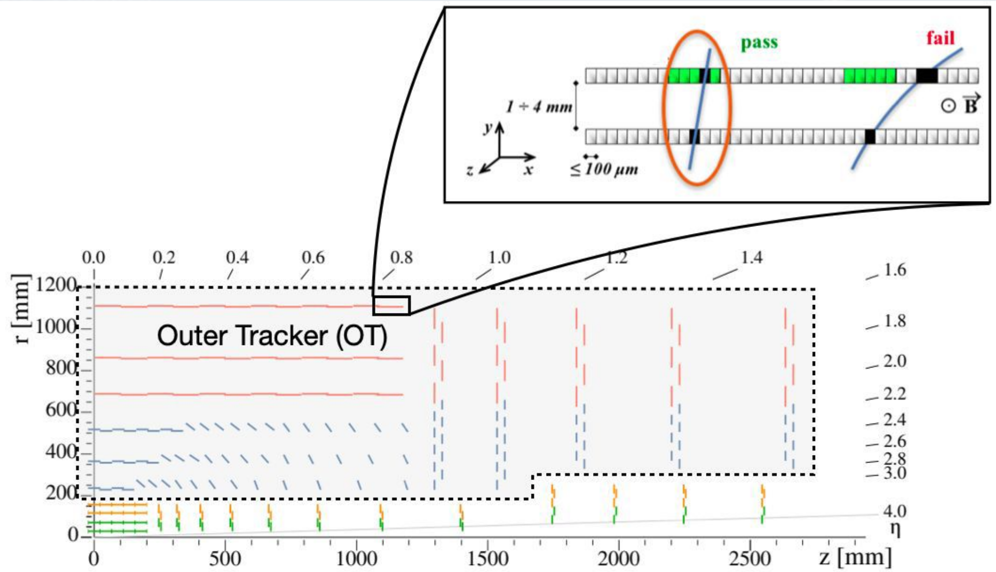
More data and higher pileup

Run 3 level



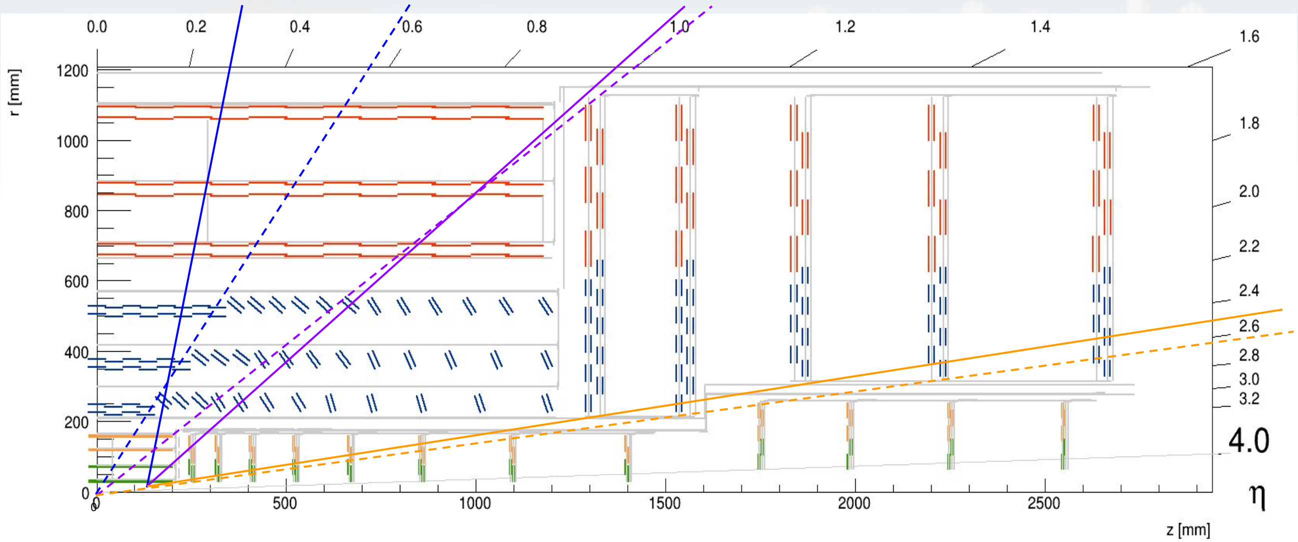
New tracker

- Inner Tracker:
  - 4 barrel layers, 12 endcap disks
  - Extended coverage up to η = 4.0
- Outer Tracker:
  - 6 barrel layers, 5 endcap disks
  - Each module consists of two closely spaced sensors allowing for an L1 track trigger ("p_T modules")



Geometry Phase-2 support

- CMS Phase-2 geometry has tilted modules
  - Tilted module extent in η also drives the mkFit barrel/transition/endcap partitioning
    - implemented in phase-2 specific layer plan support
  - Propagation and Kalman operations requires module position, normal and strip direction to be known to mkFit
    - module info was recently added with Phase-2 in mind
  - Propagation to tilted modules requires proper propagation to plane
    - implementation is a work in progress



Ongoing & Future work: Phase-2

- For Phase-2 we have a proof-of-life minimal configuration
  - Geometry, LayerPlan's and seed-partitioning are correct
    - Phase-1 functions still used for others
  - ⇒ Continue Phase-2 developments, focus on the first (Initial) iteration
- Explore synergy of mkFit with Line Segment Tracking (LST)
  - LST is a highly parallelizable algorithm that runs efficiently on GPUs
    - Uses Alpaka portability library to run on GPUs and CPUs
    - Can run in CMSSW
    - more details on LST in a talk on Thursday
  - LST track candidates can be extended/refined by mkFit to leverage more complete knowledge of hits/uncertainties/material using Kalman technique

Conclusion

- mkFit is in production mode in CMS since start of Run-3 (2021)
  - As drop-in replacement for CKF (*), used in 5 out of 11 general (+3 regional/special) iterations with equivalent physics
    - With time reduction for overall tracking of ~25% → for full reconstruction of >10%
    - With event throughput increase by ~10-15%

(*) CKF = Combinatorial Kalman Filter, default/legacy for track building when mkFit is not used

- Work has started to support Phase-2 tracking
  - Done: generalizations of geometry description, configuration, and standard functions
  - In progress: further modularization to support final fit.
  - This will also help us in tuning mkFit for additional CMS iterations (already for Run-3) …
  - … and makes mkFit easier to tune for potential other uses.

Related presentation in CTD2023:

- J. Guiang : Improving tracking algorithms with machine learning (Thu 9.30AM)

Backup slides

CMS data-MC comparison in 2022

From CMS-DP-2022-064 to highlight simulation quality compared to data

- tracks in ZeroBias (inclusive collision events) and Z→μμ
  - represent broad range of kinematics and production modes
- fairly good agreement
  - residual discrepancies are not attributed to mkFit specifics