Generalizing mkFit and its Application to HL-LHC

The mkFit team, for the CMS collaboration
CHEP-2023

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

Overview:

- One-slide introduction to mkFit
- mkFit in CMS: usage & performance
- Code generalizations and improvements in support of iterative tracking and HL-LHC
- Planned future work
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
  - 2022 – inclusion into CMSSW (CMS software) → start preparing for HL-LHC / Phase-2
    - stand-alone mode of operation is still supported
mkFit in CMS - a brief introduction

● CMS uses iterative tracking:
  ○ 12 main tracking iterations, starting from central, pixel-based seeds, then swiping up the rest
  ○ mkFit is currently used for 5 of such iterations (≈90% of all reconstructed tracks with $p_T > 0.5$ GeV)

---

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Seeding</th>
<th>Target track</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>Pixel quadruplets</td>
<td>Prompt, high-$p_T$</td>
</tr>
<tr>
<td>Pre-splitting</td>
<td>(before cluster splitting)</td>
<td>(JetCore tracking regions)</td>
</tr>
<tr>
<td>LowPtQuad</td>
<td>Pixel quadruplets</td>
<td>Prompt, low-$p_T$</td>
</tr>
<tr>
<td>HighPtTriplet</td>
<td>Pixel triplets</td>
<td>Prompt, high-$p_T$ recovery</td>
</tr>
<tr>
<td>LowPtTriplet</td>
<td>Pixel triplets</td>
<td>Prompt, low-$p_T$ recovery</td>
</tr>
<tr>
<td>DetachedQuad</td>
<td>Pixel quadruplets</td>
<td>Displaced--</td>
</tr>
<tr>
<td>DetachedTriplet</td>
<td>Pixel triplets</td>
<td>Displaced-- recovery</td>
</tr>
<tr>
<td>MixedTriplet</td>
<td>Pixel+strip triplets</td>
<td>Displaced-</td>
</tr>
<tr>
<td>PixelPair</td>
<td>Pixel pairs</td>
<td>Displaced- recovery</td>
</tr>
<tr>
<td>PixelLess</td>
<td>Inner strip triplets</td>
<td>Displaced+</td>
</tr>
<tr>
<td>TobTec</td>
<td>Outer strip triplets</td>
<td>Displaced++</td>
</tr>
<tr>
<td>JetCore</td>
<td>Pixel pairs within jets</td>
<td>Within high-$p_T$ jets</td>
</tr>
<tr>
<td>Muon inside-out</td>
<td>Muon-tagged tracks</td>
<td>Muons</td>
</tr>
<tr>
<td>Muon outside-in</td>
<td>Standalone muons</td>
<td>Muons</td>
</tr>
</tbody>
</table>

* In [CMS-DP-2022-018](#), mkFit is also used in PixelLess
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
      - cannot rely on claimed hits to discard seeds
    - Seed partitioning:
      - barrel / transition / endcap + sorting in \{ \eta, \varphi \}
    - Forward search with quality filtering (optional)
    - Backward fit / search with quality filtering
    - Duplicate removal
  - post-mkFit: final-fit, final NN quality selection

Seeds

Track building
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**: Mitigated by dedicated duplicate removal.
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
Generalizations for iterative tracking & HL-LHC:

- Geometry description & traversal
- Configuration classes / mechanisms
- Catalog approach to standard track-processing functions
Geometry description and traversal

- **Detectors split into mkFit layers**
  - Potentially finer granularity than readout / construction
    - E.g., mono/stereo treated as separate layers

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

- **Changes**
  - On-the-fly extraction of layer envelopes/gaps
  - Add module-id information to hits to allow for overlap hit collection
  - CMS Phase-2 geometry has tilted modules → requires module position, normal and strip direction to be known to mkFit
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, $\chi^2$ cuts; quality filter params
    - can be different for forward / backward search
  - class `IterationLayerConfig` → parameters specific to layers, hit search windows; one per layer!

- In CMSSW (or any other multi-threaded framework) configuration is required to be completely separable → instantiated and managed independently
  - Tracking iterations are driven by the CMS module system, typically configured via Python scripts

- 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

- With support of multiple iterations and Phase-2 geometry it became obvious we need 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

- 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
**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 supported → 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
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::vectors.
  - Vector-gather (vgather) instruction, which is used to fill Matriplex’s with input data, breaks if hit or track allocations are done from different threads (probably virtual memory segment)
Ongoing & Future work

- 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

- 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 Line Segment Tracking – mkFit hybrid
  - Highly parallelizable algorithm that can run efficiently on GPUs
  - Uses Alpaka
  - Already integrated into CMSSW
Conclusion

- **mkFit is in production mode since Run-3**
  - As drop-in replacement for CKF (*), used in 5 out of 12 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 for CMS 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 presentations:

- **L. Giannini:** *A DNN for CMS track classification and selection*
  - Poster
- **P. Chang:** *Line Segment Tracking in the High-luminosity LHC*
  - Track 2 (Online computing): Tue. May 9, 2pm