Speeding up the CMS track reconstruction with a parallelized and vectorized Kalman-filter-based algorithm during the LHC Run 3

S Berkman, G Cerati[1]**, P Elmer**[2]**, P Gartung**[1]**, L Giannini**[3]**, B Gravelle**[4]**, A R Hall**[5]**, M Kortelainen**[1]**, S Krutelyov**[3]**, S R Lantz**[6]**, M Masciovecchio**[3]**, K McDermott, B Norris, M Reid**[6]**, D S Riley**[6]**, M Tadel**[3]**, E Vourliotis**[3a]**, B Wang, P Wittich**[6]**, A Yagil**[3]
on behalf of the CMS Collaboration

[1] Fermilab, IL, US
[2] Princeton University, NJ, US
[3] University of California San Diego, CA, US
[4] University of Oregon, OR, US
[5] USNA Annapolis, MD, US
[6] Cornell University, NY, US

E-mail: [a] emmanouil.vourliotis@cern.ch

Abstract. One of the most challenging computational problems in the Run 3 of the Large Hadron Collider (LHC) and more so in the High-Luminosity LHC (HL-LHC) is expected to be finding and fitting charged-particle tracks during event reconstruction. The methods used so far at the LHC and in particular at the CMS experiment are based on the Kalman filter technique. Such methods have shown to be robust and to provide good physics performance, both in the trigger and offline. In order to improve computational performance, we explored Kalman-filter-based methods for track finding and fitting, adapted for many-core SIMD architectures. This adapted Kalman-filter-based software, called "MKFIT", was shown to provide a significant speedup compared to the traditional algorithm, thanks to its parallelized and vectorized implementation. The MKFIT software was recently integrated into the offline CMS software framework, in view of its exploitation during the Run 3 of the LHC. At the start of the LHC Run 3, MKFIT will be used for track finding in a subset of the CMS offline track reconstruction iterations, allowing for significant improvements over the existing framework in terms of computational performance, while retaining comparable physics performance. The performance of the CMS track reconstruction using MKFIT at the start of the LHC Run 3 is presented, together with prospects of further improvement in the upcoming years of data taking.

1. Motivation and the mkFit Algorithm

As the Large Hadron Collider (LHC) experiments prepare for the Phase-2 upgrade of the particle accelerator with the ultimate goal of gathering about $3000\,\mathrm{fb}^{-1}$ of integrated luminosity, studies are performed to estimate the computational resources needed for the reconstruction of collision events. These studies indicate a superlinear growth for the total reconstruction time [1]. Among the different aspects of the reconstruction process, the reconstruction of charged particle trajectories, simply called "tracks", takes almost half of the total time of the current Run-3

reconstruction [2]. To complement single-threaded runtime performance, it is clear parallelized and vectorized tracking algorithms need to be developed. These will be crucial for Phase-2 but they can make a difference even in Run-3 of the LHC.

The MATRIPLEX Kalman-fitter algorithm, "MKFIT" for short, is a parallelized and vectorized version of the combinatorial Kalman-filter (CKF) algorithm used for the track reconstruction at the Compact Muon Solenoid (CMS) experiment at the LHC [3, 4]. Being in development for more than five years, it has been integrated during the first quarter of 2022 in the production release of CMS and used for the entire first year of Run 3 data taking. It has been applied to a subset of the CMS track reconstruction iterations, as illustrated in figure 1, and achieves similar physics performance as the CKF algorithm, while providing a significant speed up, as shown in section 2.

| Iteration	Seeding	Target track
Initial	pixel quadruplets	prompt, high p_T
LowPtQuad	pixel quadruplets	prompt, low p_T
HighPtTriplet	pixel triplets	prompt, high p_T recovery
LowPtTriplet	pixel triplets	prompt, low p_T recovery
DetachedQuad	pixel quadruplets	displaced--
DetachedTriplet	pixel triplets	displaced-- recovery
MixedTriplet	pixel+strip triplets	displaced-
PixelLess	inner strip triplets	displaced+
TobTec	outer strip triplets	displaced++
JetCore	pixel pairs in jets	high-p_T jets
Muon inside-out	muon-tagged tracks	muon
Muon outside-in	standalone muon	muon

Figure 1. Iterations of the CMS track reconstruction sequence. The first column indicates the name of the iteration, the second column the kind of tracks used for the seeding of the full track reconstruction and the third column the track type that each iteration targets. On the left, the iterations for which the MKFIT algorithm is used are marked.

In a nutshell, the CKF algorithm starts from track seeds and iteratively accumulates compatible hits to build full tracks. The MKFIT algorithm speeds up this procedure by vectorizing some of its aspects and making them multithreaded. For this to happen, some requirements need to be fulfilled: the branching points of the algorithm need to be minimized, the workload needs to be equally distributed to different threads and the memory accesses need to be minimized and optimized. To reduce the branching points in the code, the MKFIT algorithm cleverly parallelizes the track building over multiple levels (different events, different η regions and different z-/r- and ϕ-sorted groups of seeds). The balancing of the workloads is achieved with the usage of the Intel® Threading Building Blocks (TBB) library. In terms of memory usage, this is greatly improved by utilizing a custom matrix library, MATRIPLEX, specially designed to optimize the memory accesses for the 6×6 track candidate covariance matrices used during Kalman filter operations [5]. Finally, the memory needs are vastly reduced by dropping the detailed information on individual tracker modules in favor of a simplified tracker geometry, in which the tracker details are stored in a 2D (r or z, ϕ) map.

2. Physics and Timing Performance

The MKFIT algorithm is used by a subset of the CMS tracking iterations that reconstruct almost 90% of the hard scattering event tracks with $p_T > 0.5$ GeV. The physics performance achieved by the usage of the MKFIT algorithm for the reconstruction of tracks, identified by the "high purity" quality flag [4] and measured in a $t\bar{t}$ sample with event pileup (PU), i.e. simultaneous collisions, following a Poisson distribution with a mean value uniformly distributed between 55 and 75, is illustrated in figures 2-5. Starting from figures 2 and 3, these show the tracking efficiency,

defined as the fraction of simulated tracks matched to at least one reconstructed track, where the matching requires 75% common hits between the simulated and the reconstructed track, as a function of p_T and η respectively. The efficiency with and without the MKFIT algorithm for track building is comparable, with small gains in the pseudorapidity range $2.4 < |\eta| < 2.8$ for the MKFIT case. In figures 4 and 5, the tracking fake rate, i.e. the fraction of misidentified reconstructed tracks, can be seen as a function of η and event PU. When the MKFIT algorithm is used, the fake rate tends to be lower for increasing η and the scaling with PU is better. Finally, the tracking duplicate rate, which is defined as the fraction of reconstructed tracks associated multiple times to the same simulated track, is marginally increased ($\sim 0.5\%$) for the MKFIT track reconstruction, due to the parallel nature of the algorithm [2]. Duplicate tracks produced by the MKFIT algorithm are mitigated by a dedicated duplicate removal, tuned as a function of p_T and η.
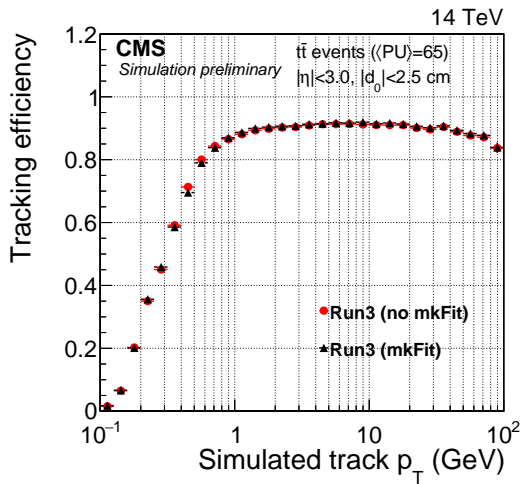

Figure 2. Tracking efficiency as a function of the simulated track p_T for CKF tracking (red) and MKFIT tracking (black), for simulated tracks with $|\eta| < 3$ and $|d_0| < 2.5$ cm [2].
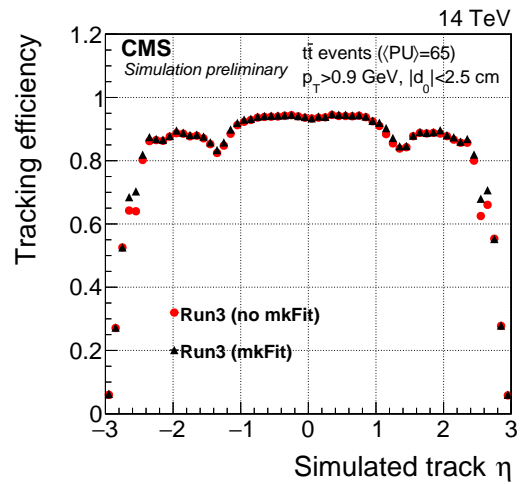
Figure 3. Tracking efficiency as a function of the simulated track η for CKF tracking (red) and MKFIT tracking (black), for simulated tracks with $p_T > 0.9$ GeV and $|d_0| < 2.5$ cm [2].

The great advantage of the MKFIT algorithm is evident when the timing performance is measured. In figure 6, the speedup achieved by vectorizing the code is shown as a function of the vector unit width of the MATRIPLEX library, i.e. the number of matrices operated on concurrently, while figure 7 shows the speedup coming from making the code multithreaded as function of the number of threads. The measurements were performed in such a way that the vectorization and the multithreading effects were factorized and on two separate machines: the "KNL" machine (64 cores: Intel® Xeon Phi™ processor 7210 @ 1.30 GHz) and the "SKL-SP" machine (dual socket × 16 cores: Intel® Xeon® Gold 6130 processor @ 2.10 GHz). A comparison with the Amdahl's Law indicates that almost 70% of the MKFIT operations are effectively vectorized and more than 95% are effectively parallelized, as detailed in Ref. [5]. The timing performance within the context of the CMS track reconstruction is shown in figures 8 and 9. These indicate that, when using the MKFIT algorithm, individual MKFIT iterations get a building time reduction up to ×6.7 and the sum of MKFIT iterations get a ×3.5 building time reduction (figure 8), while the sum of all iterations gets approximately ×1.7 speedup (figure 9). As a result, the total Run-3 tracking time is reduced by 25%, which translates in an increase of event throughput of 10–15% [2].
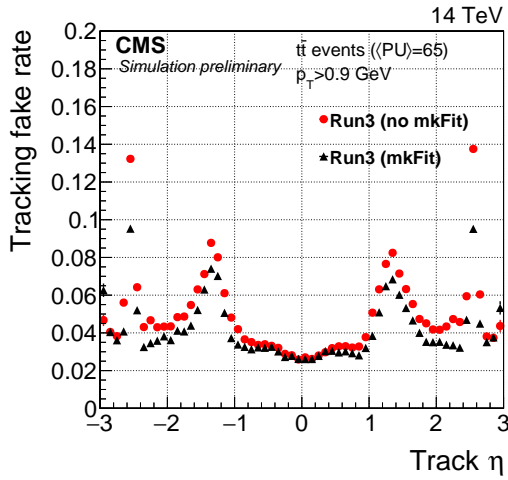
Figure 4. Tracking fake rate as a function of the reconstructed track η for CKF tracking (red) and MKFIT tracking (black), for reconstructed tracks with $p_T > 0.9$ GeV [2].
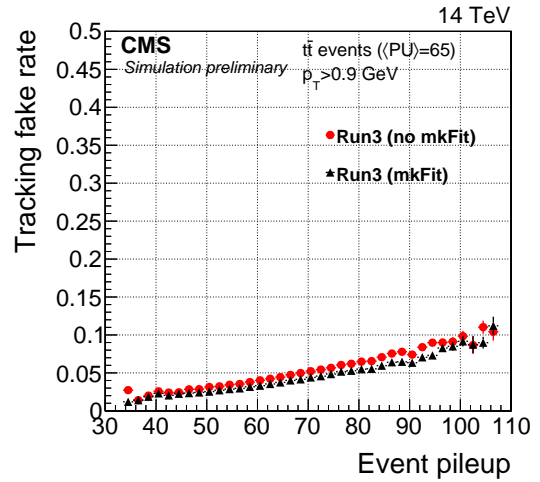
Figure 5. Tracking fake rate as a function of the event pileup for CKF tracking (red) and MKFIT tracking (black), for reconstructed tracks with $p_T > 0.9$ GeV [2].
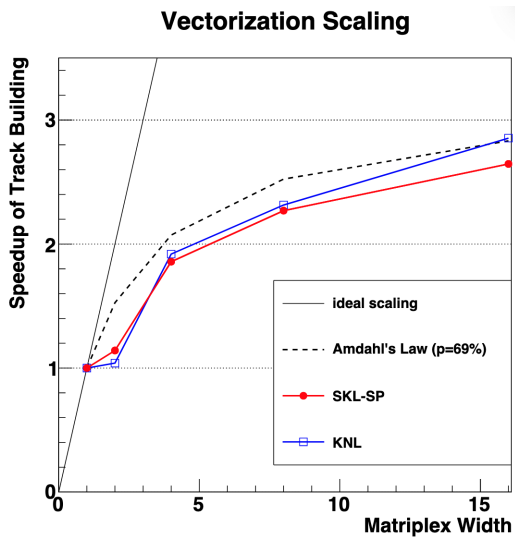

Figure 6. Speedup due to vectorization as a function of the MATRIPLEX width for MKFIT track building on the KNL and SKL-SP machines. The ideal speedup (solid line) and the speedup based on Amdahl's Law (dashed lines) are also shown [5].
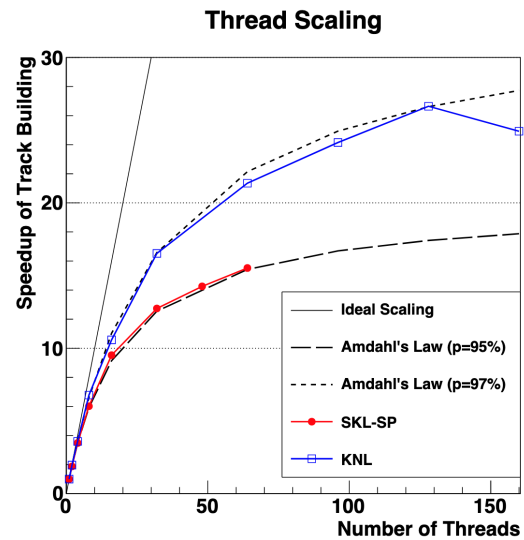

Figure 7. Speedup due to multithreading as a function of the number of threads for MKFIT track building on the KNL and SKL-SP machines. The ideal speedup (solid line) and the speedup based on Amdahl's Law (dashed lines) are also shown [5].

3. Summary and Outlook

MKFIT is a Kalman-filter algorithm for track pattern recognition, successfully following the paradigm shift towards code vectorization and parallelization. It has recently been integrated
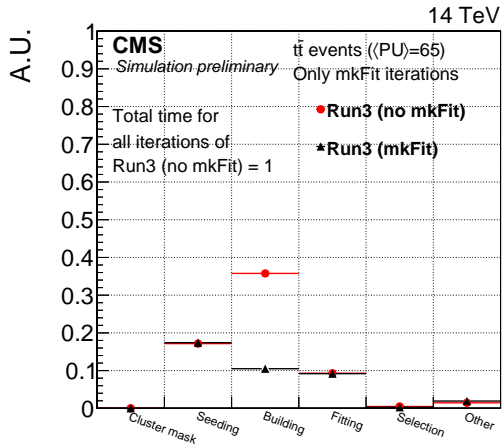
Figure 8. Tracking time as a function of the tracking steps for CKF tracking (red) and MKFIT tracking (black), for the subset of tracking iterations using the MKFIT algorithm. The vertical axis is normalized to have the total time without MKFIT equal to unity. [2].
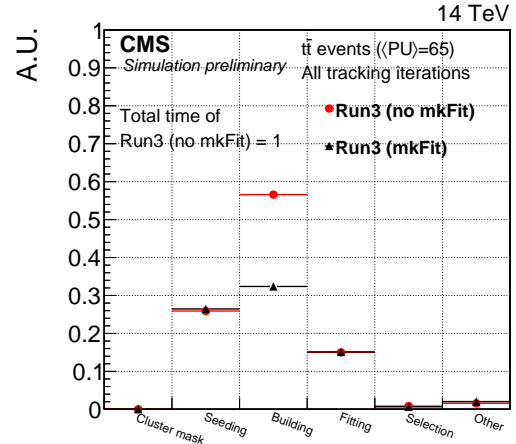
Figure 9. Tracking time as a function of the tracking steps for CKF tracking (red) and MKFIT tracking (black), for all tracking iterations. The vertical axis is normalized to have the total time without MKFIT equal to unity. [2].

to the CMS central software, replacing the CKF-based tracking in a subset of the CMS tracking iterations for Run-3. The usage of the MKFIT algorithm leads to significant improvements in terms of computational performance, while retaining a comparable physics performance.

Looking to the future, more MKFIT-related developments are foreseen to further enhance the CMS tracking. Ideas for further application of the MKFIT algorithm include its extension to more track building iterations, its implementation to the track fitting procedure, since it has now become almost as time-consuming as the track building procedure (figures 8 and 9), and its application to the High Level Trigger reconstruction of CMS. Finally, the MKFIT algorithm is being modified to accommodate the Phase-2 CMS geometry and configuration, while also exploring synergies with other tracking algorithms developed for Phase-2.

Acknowledgements

References

[1] Cerati G B 2014 Tracking and vertexing algorithms at high pileup CMS-CR-2014-345 URL https://cds.cern.ch/record/1966040
[2] CMS Collaboration 2022 Performance of Run 3 track reconstruction with the mkFit algorithm CMS-DP-2022-018 URL https://cds.cern.ch/record/2814000
[3] CMS Collaboration 2008 *Journal of Instrumentation* **3** S08004
[4] CMS Collaboration 2014 *Journal of Instrumentation* **9** P10009
[5] Lantz S, McDermott K, Reid M, Riley D, Wittich P, Berkman S, Cerati G, Kortelainen M, Reinsvold A H, Elmer P, Wang B, Giannini L, Krutelyov V, Masciovecchio M, Tadel M, Würthwein F, Yagil A, Gravelle B, Norris B 2020 *Journal of Instrumentation* **15** P09030