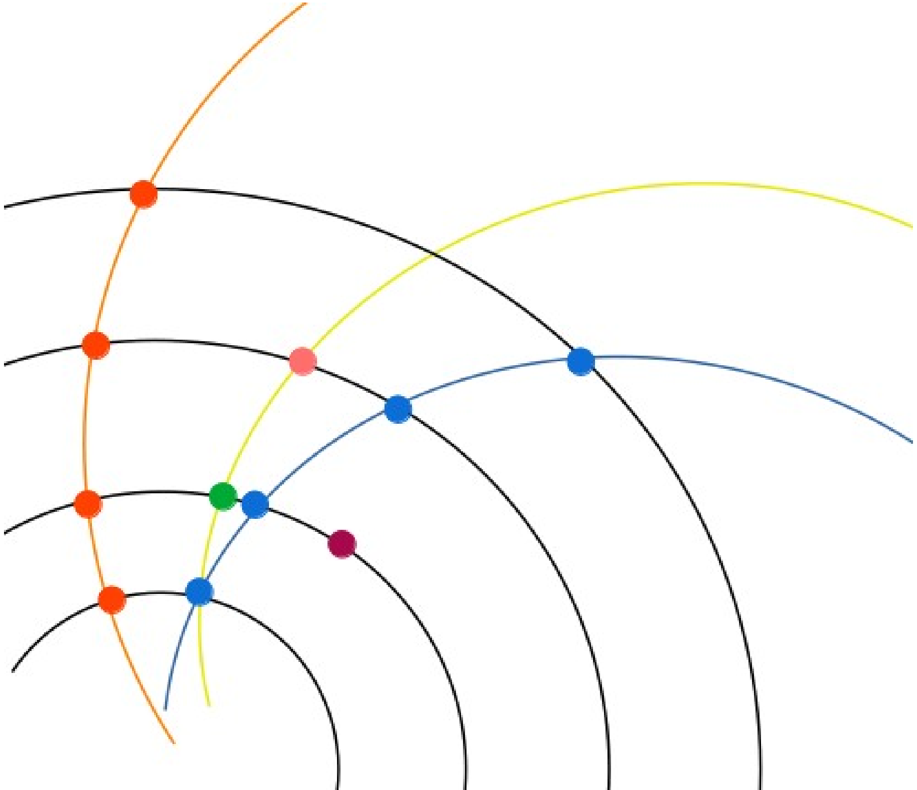
(Combinatorial) KalmanFilter in acts

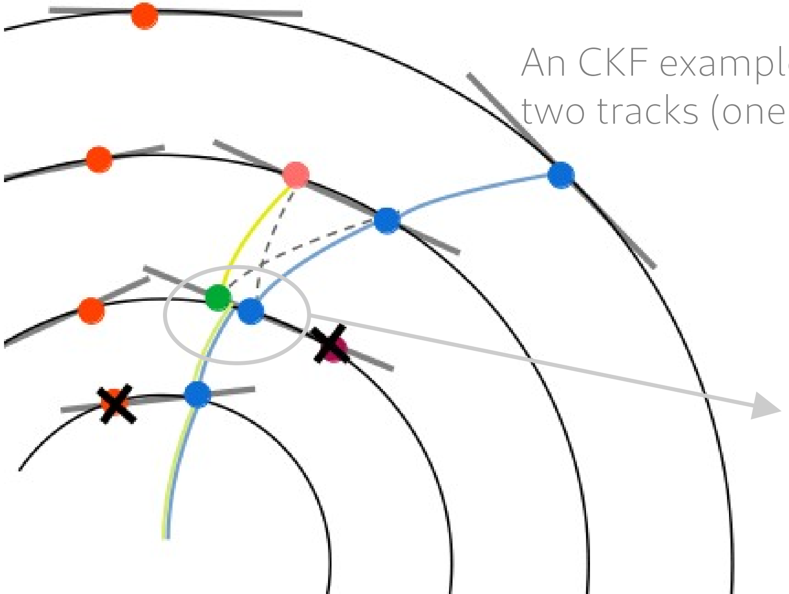


Xiaocong Ai

ACTS Developers Workshop, Sept 26, 2022

KF and CKF

- KF: track fitting for a found track
  - One seed results in 1 track if fitting succeeds
- CKF: track fitting + track finding simultaneously
  - Selection of measurements and propagation branching are involved
  - One seed might result in >=1 track(s) if track finding succeeds



An CKF example: one seed results in two tracks (one track is fake)

✕: measurement is not compatible with track parameters

One branch is splitted into two branches, propagation will go on with one branch first

Propagation will be reset to the unprocessed branches

KF invocation

```
template <typename source_link_iterator_t, typename start_parameters_t,
          typename parameters_t = BoundTrackParameters,
          bool _isdn = isDirectNavigator>
auto fit(source_link_iterator_t it, source_link_iterator_t end,
         const start_parameters_t& sParameters,
         const KalmanFitterOptions<traj_t>& kfOptions,
         std::shared_ptr<traj_t> trajectory = {}) const
    -> std::enable_if_t<!_isdn, Result<KalmanFitterResult<traj_t>>> {
```

Compile-time option to use direct navigation based on a sequence of surfaces

One “fit” takes:

- Begin and End Iterator for the input measurements (sourcelinks) of the track
- Starting parameters for the track
- KF options
- Optional existing MultiTrajectory to append the new track
  - This allows all tracks in one event to share a single MultiTrajectory (see #1507)

and returns:

- The MultiTrajectory for either only this track or prefitted tracks in this event
- The fitted track parameters for each track

CKF invocation

```
template <typename source_link_iterator_t,
          typename start_parameters_container_t,
          typename parameters_t = BoundTrackParameters>
std::vector<Result<CombinatorialKalmanFilterResult<traj_t>>> findTracks(
    const start_parameters_container_t& initialParameters,
    const CombinatorialKalmanFilterOptions<source_link_iterator_t, traj_t>&
        tfOptions) const {
```

One “findTracks” takes:

- A set of starting parameters, e.g. the found seeds for one event
- CKF options (including sourcelink accessor, measurement selector…)

and returns:

- A single MultiTrajectory for all found tracks using the given seeds
- A vector of fitted track parameters for all found tracks

KF/CKF options

Much improvement by Paul Gessinger-Befurt, Corentin Allaire and Tim Adye...

	Extensions	Other options
KF&CKF	• calibrator • (Kalman) updater • (Kalman) smoother	• Whether consider material effects • Optional target surface (e.g. beam line or tracker exit) to retrieve fitted track parameters
KF-specific	• outlierFinder • ReverseFilteringLogic (more sophisticated decision logic for smoothing approach)	• Perform "smoothing" using either smoothing formalism or backward filtering • Scaling factor for track parameters covariance at the start of backward filtering • Whether perform non-linear correction during global → local transformation
CKF-specific	• MeasurementSelector (allow eta/pt dependent selection cuts) • branchStopper	• Sourcelink Accessor to retrieve a range of measurements for a given surface (in principle, it can include measurements in neighbor surface) • Perform smoothing (can only use smoothing formalism) or not

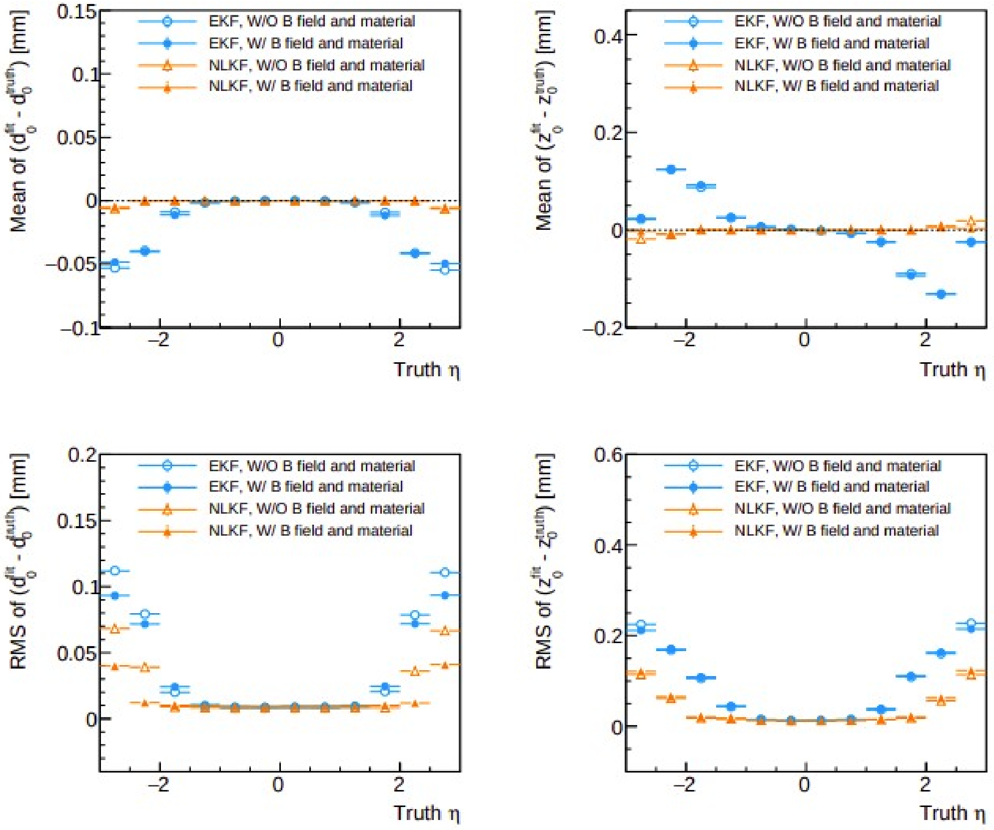
KF non-linear extension

- Track parameters covariance is transported using Jacobian ($J_{k-1 \to k}$)
  - $C_k = J_{k-1 \to k} * C_{k-1} * J^T_{k-1 \to k}$
- One-order Jacobian might be insufficient when non-linear effects are significant (e.g. large incident angle)
- Idea of non-linear correction during global track parameters → local track parameters
  - Generate a set of sample points for the global track parameters based on its covariance
    - This requires a reliable estimation of the track parameters covariance
  - Perform transformation for each sample point
  - Derive corrected local track parameters and its covariance

KF performance

Open Data Detector, ATLAS field

- Non-linear KF can correct the bias of fitted track parameters and improve their precision in a test scenario
- ~1.6X time of KF

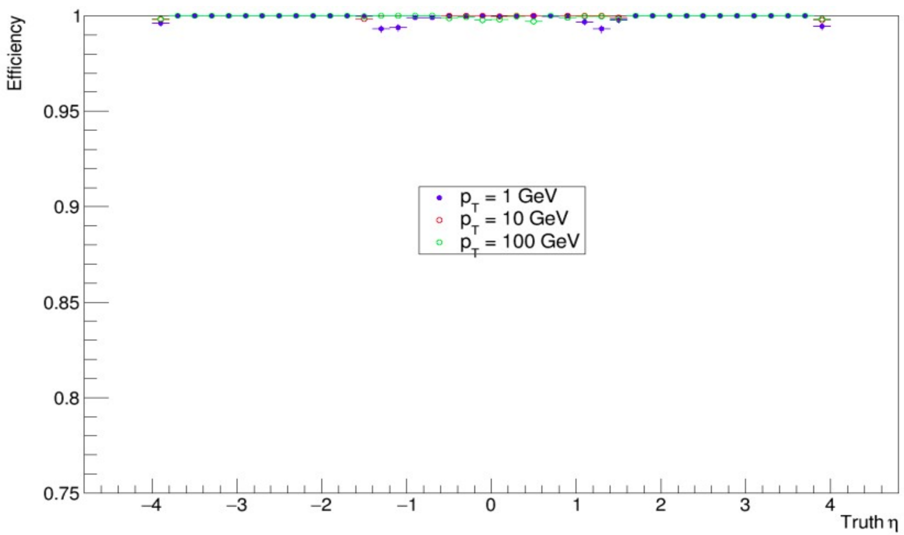


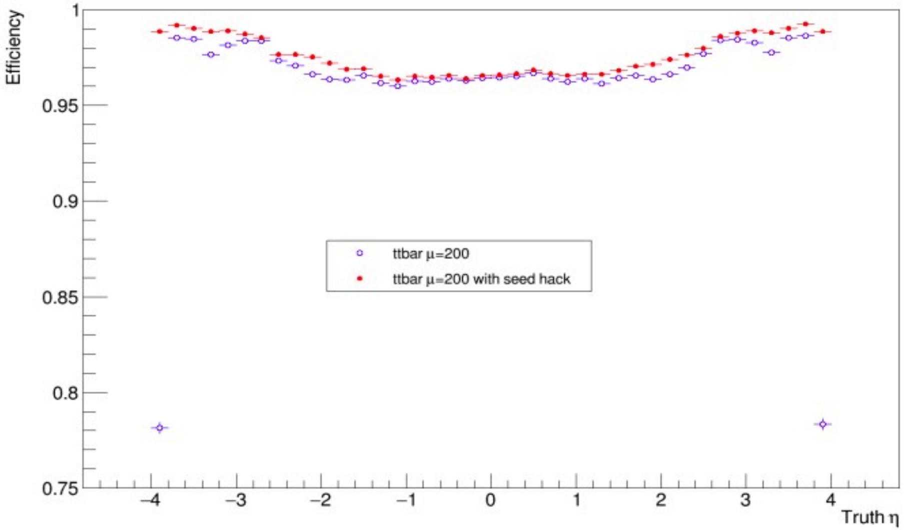
arXiv: 2112.09470

CKF performance

See Tim Adye's slides here

- For example, >95% track find efficiency for ttbar with μ = 200 for ATLAS ITk





- ATLAS ITk geometry, B-field map, material
- smeared hits digi (no HGTD)
- ITk seeding **with hack**, CKF
- 1k 100-$\mu^{\pm}$ events
- $p_T = 1, 10, 100$ GeV/c
- $|\eta| < 4$

- ATLAS ITk geometry, B-field map, material
- smeared hits digi (no HGTD)
- ITk seeding **with and without hack**, CKF
- 1k events: 100-$\mu^{\pm}$ and ttbar+μ=200
- $|\eta| < 4$

Summary

- The KF and CKF have been much refactored and extended in the last year
  - This greatly simplifies and extends the usage
- CKF is being tested with more and more experiments, e.g. ATLAS Itk, FASER…
- Further validation and optimization, in particular the time performance, are needed