Kalman filter tracking library

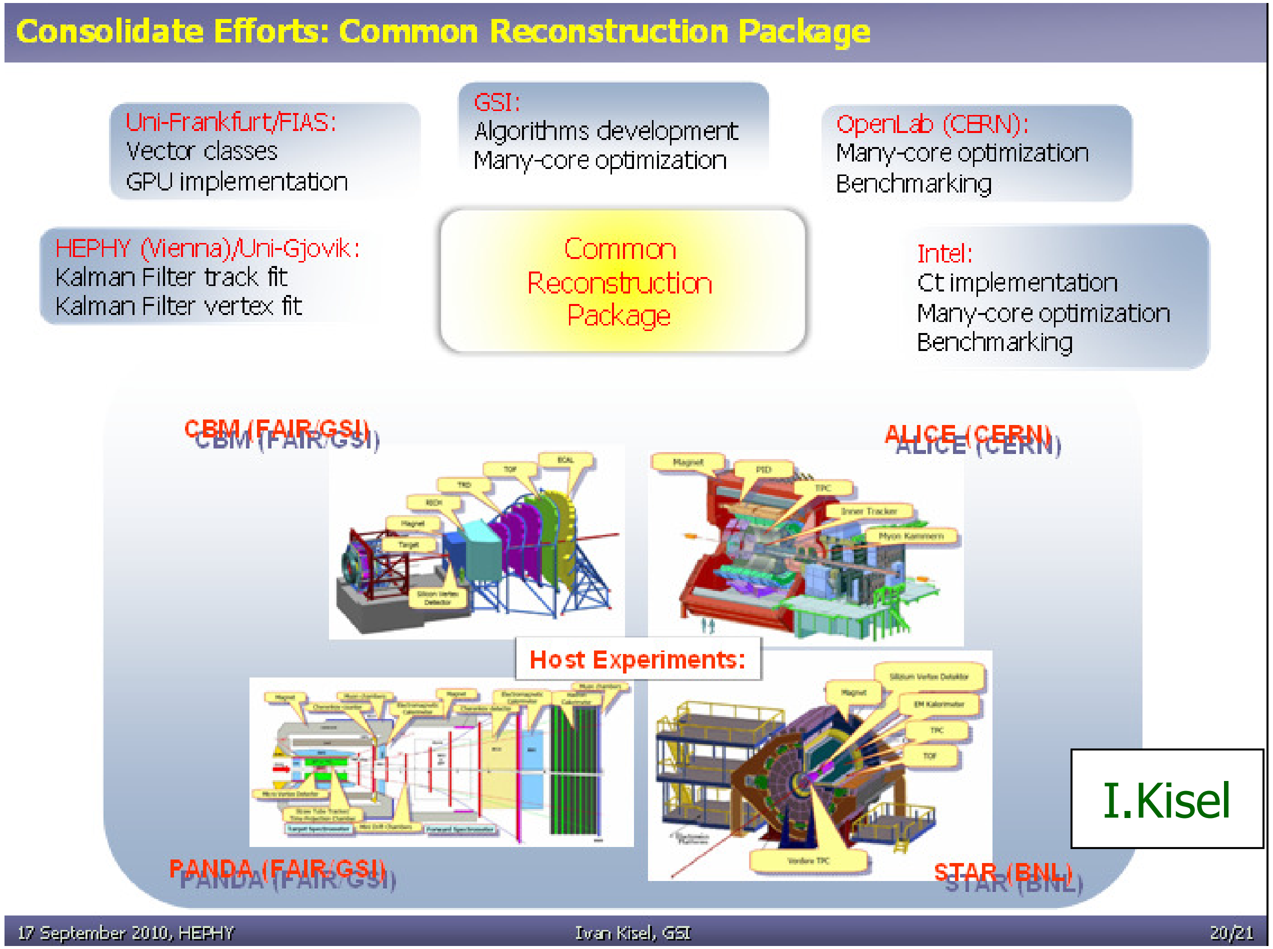
I. Kisel, I. Kulakov, H. Pabst, M. Zyzak

Tracking workshop
FIAS, Frankfurt,
February 28, 2012

Outline

- Kalman Filter (KF)
- KF Library
  - Intel Array Building Blocks (ArBB)
  - KF approaches
  - Propagation methods
  - Smoother
  - Deterministic Annealing Filter (DAF)
- Summary

Kalman Filter (KF) Based Track Fit

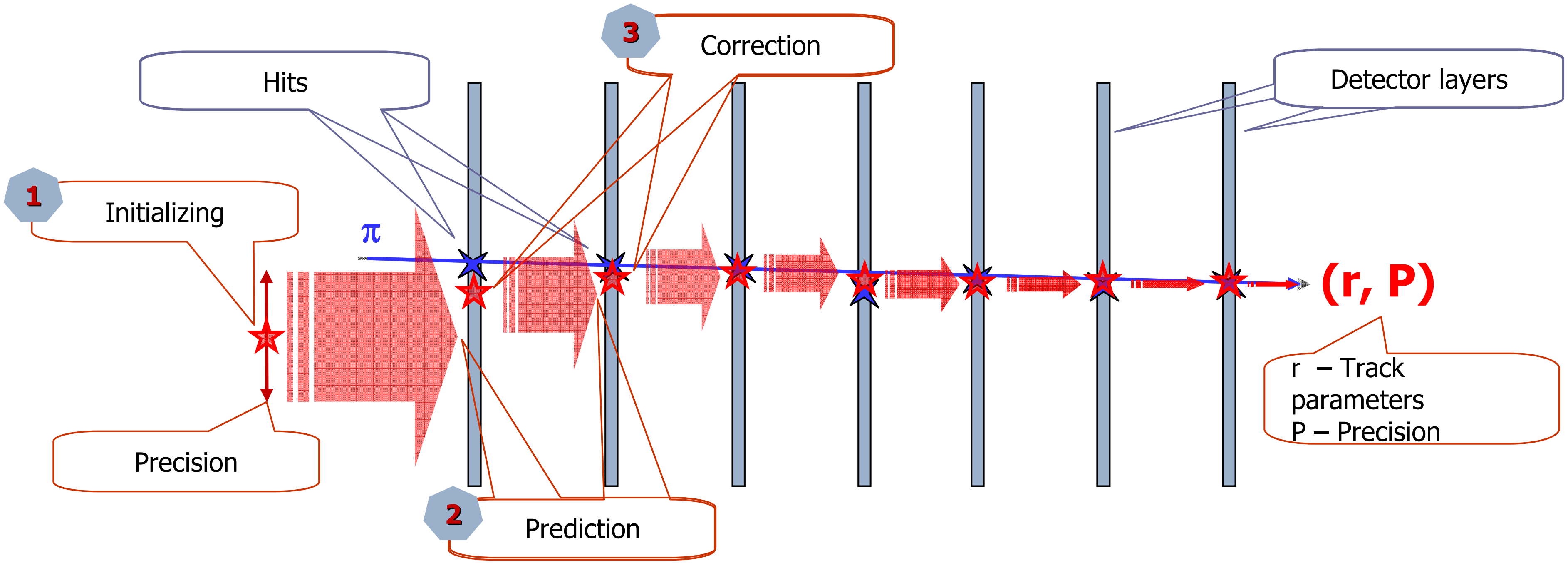


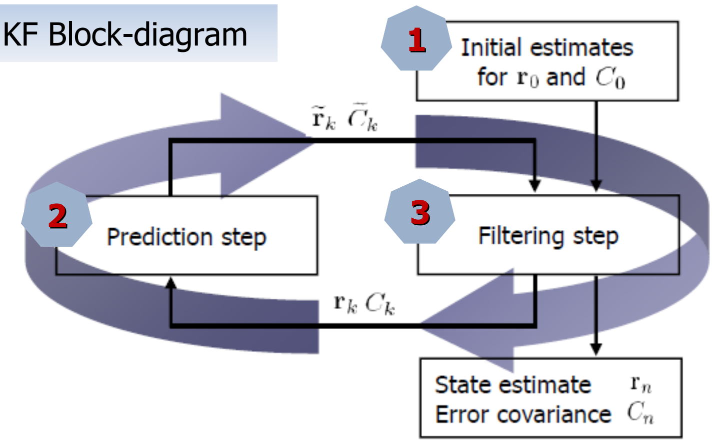
Nowadays the Kalman Filter is used
in almost all HEP experiments
at almost all stages of reconstruction
(track finding, track fitting, particle finding).

Common reconstruction package will be KF based.

Kalman Filter (KF) Based Track Fit

Track fit: Optimal estimation of the track parameters according to hits – Kalman Filter (KF)





KF as a recursive least squares method

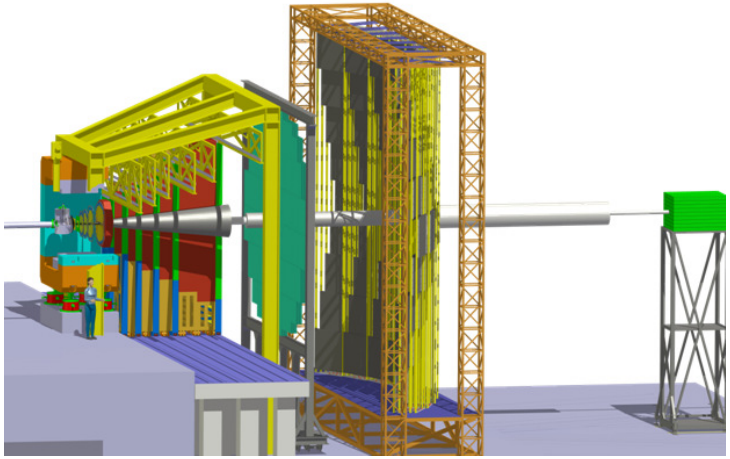
State vector

Position, direction and momentum

$\mathbf{r} = \{ x, y, z, p_x, p_y, p_z \}$

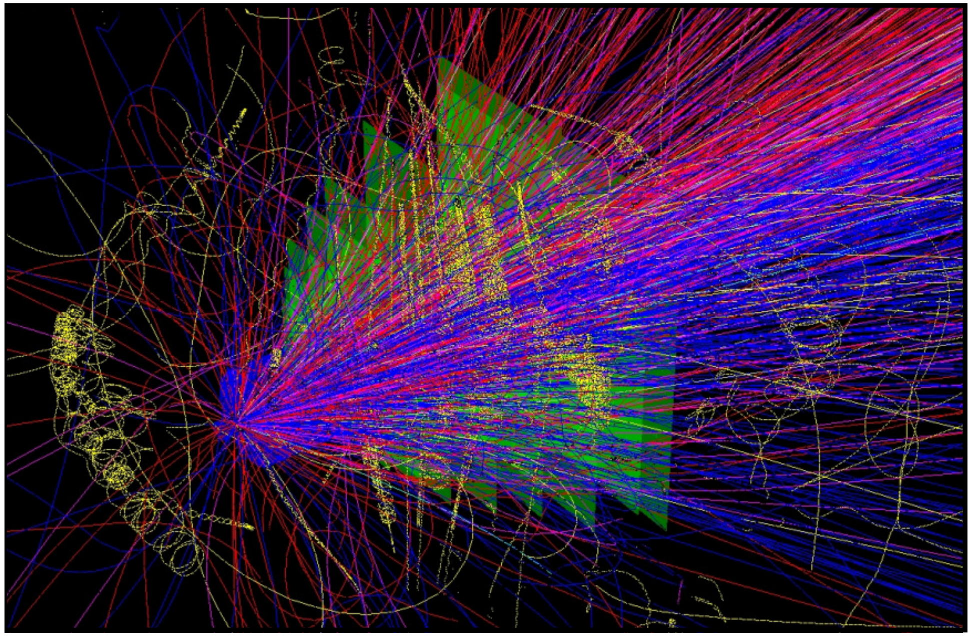
Kalman Filter:
1. Start with an arbitrary initialization.
2. Add one hit after another.
3. Improve the state vector.
4. Get the optimal parameters after the last hit.

Tracking Challenge in CBM

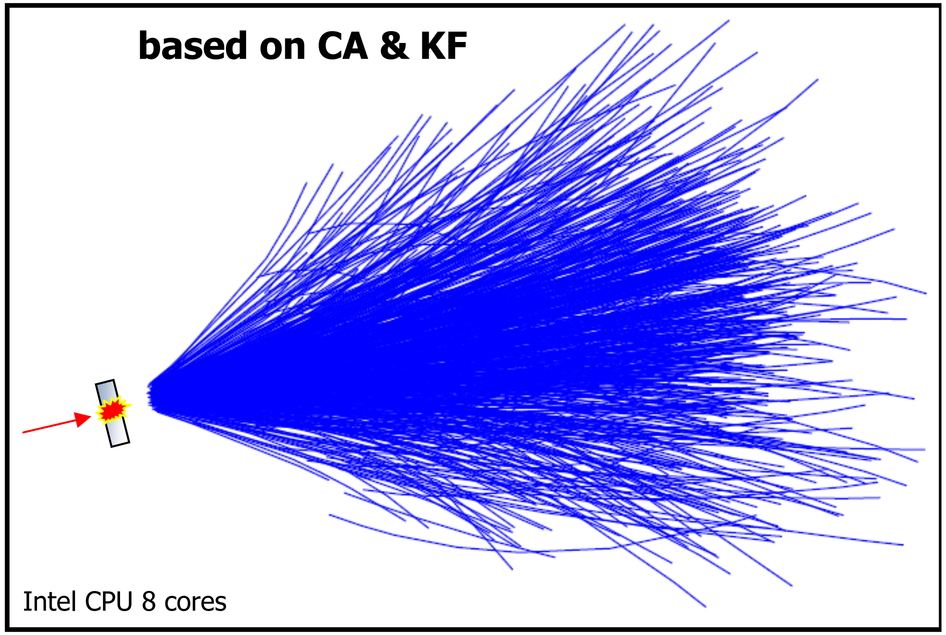


- 1000 charged particles/collision
- Double-sided strip detectors (85% fake space points)
- Non-homogeneous magnetic field
- 10^7 AuAu collisions/sec
- Track reconstruction in STS/MVD and displaced vertex search are required in the first level trigger

Simulation



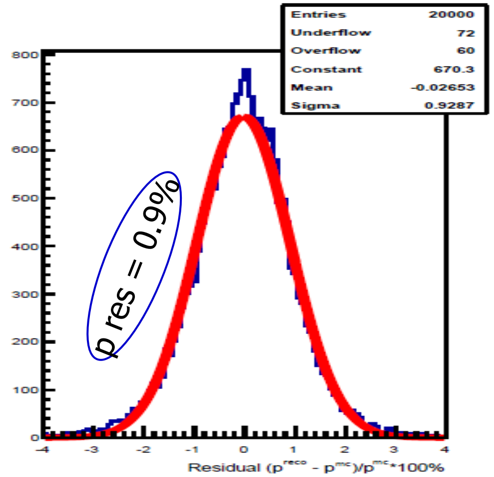
Reconstruction



A precise, fast and stable realization of the KF algorithm is required.

SIMD KF benchmark

optimal fit quality



fast

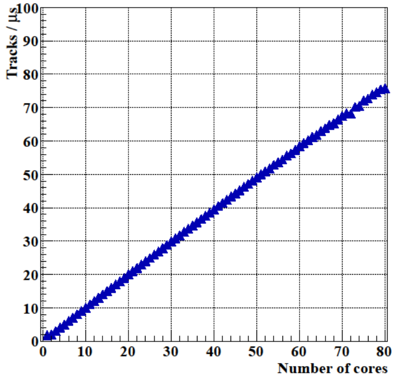
Fitting speed of 13 ns/track/node

parallel

SIMD

multithreading

scalable



alterable

single/double precision

ITBB/OpenMP

SIMD KF benchmark → **KF Library**

Algorithms

Track tools:

- KF track fitter
- KF track smoother
- Deterministic Annealing Filter

KF approaches:

- Conventional KF
- Double precision KF
- Square root KF (2 implementations)
- U-D-Filtering

Track propagation:

- Runge-Kutta
- Analytic formula

Hardware support

Parallelization:

Data level:

- Header
- Vc
- ArBB

Task level:

- ITBB
- ArBB
- Open MP

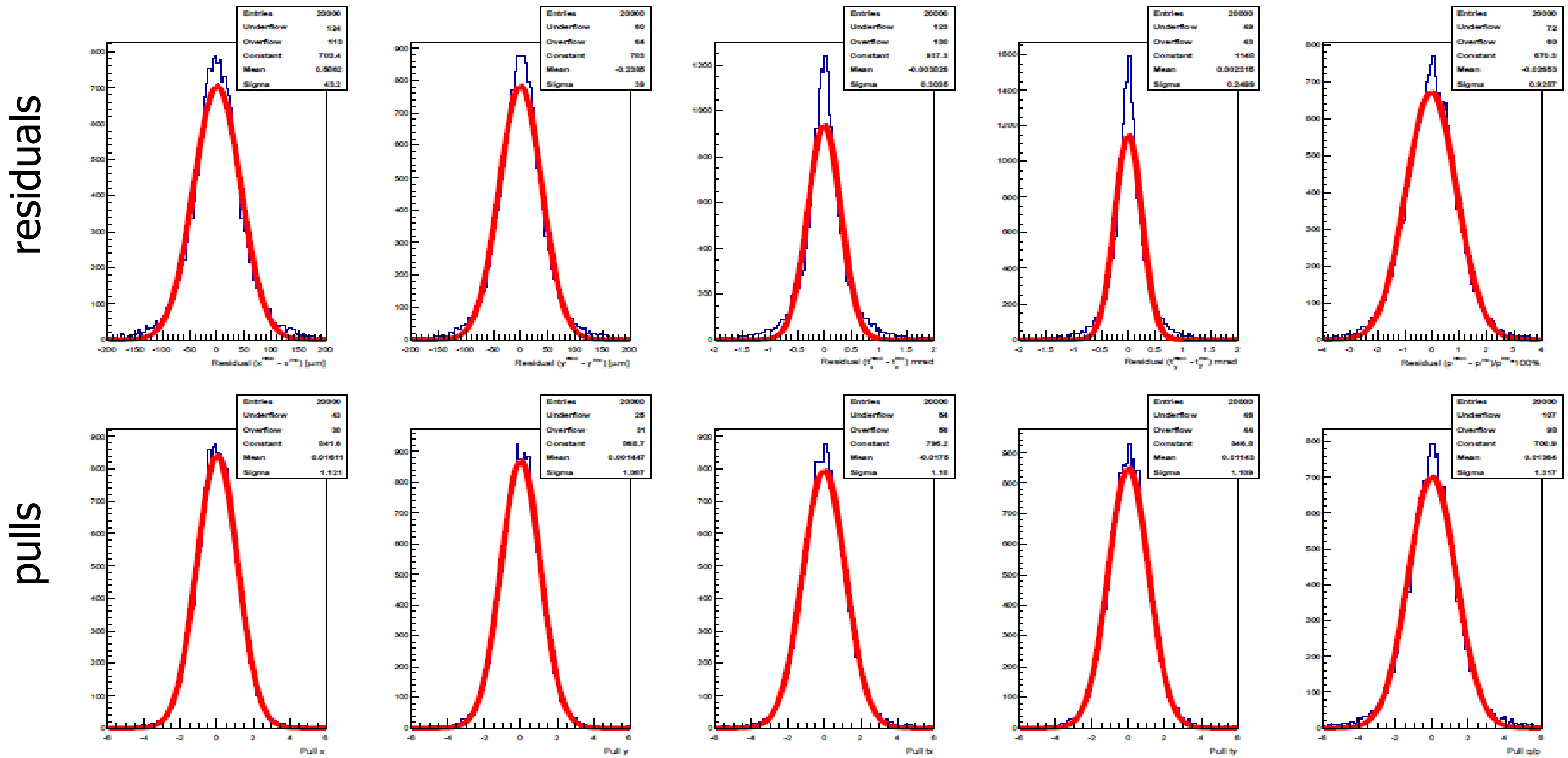
Precision:

- single
- double

Track Fit Quality

CBM state vector: $\mathbf{r} = \{\, x, y, t_x, t_y, q/p \,\}$

position, tg of slopes and charge over momentum



residuals					pulls				
x, μm	**y, μm**	**t_x, 10^{-3}**	**t_y, 10^{-3}**	**q/p, %**	**x**	**y**	**t_x**	**t_y**	**q/p**
43	39	0.3	0.25	0.93	1.1	1.1	1.2	1.1	1.3

conventional, header, geometry with 0 & 90 degree strips

KF track fit based on ArBB has been implemented by Intel.

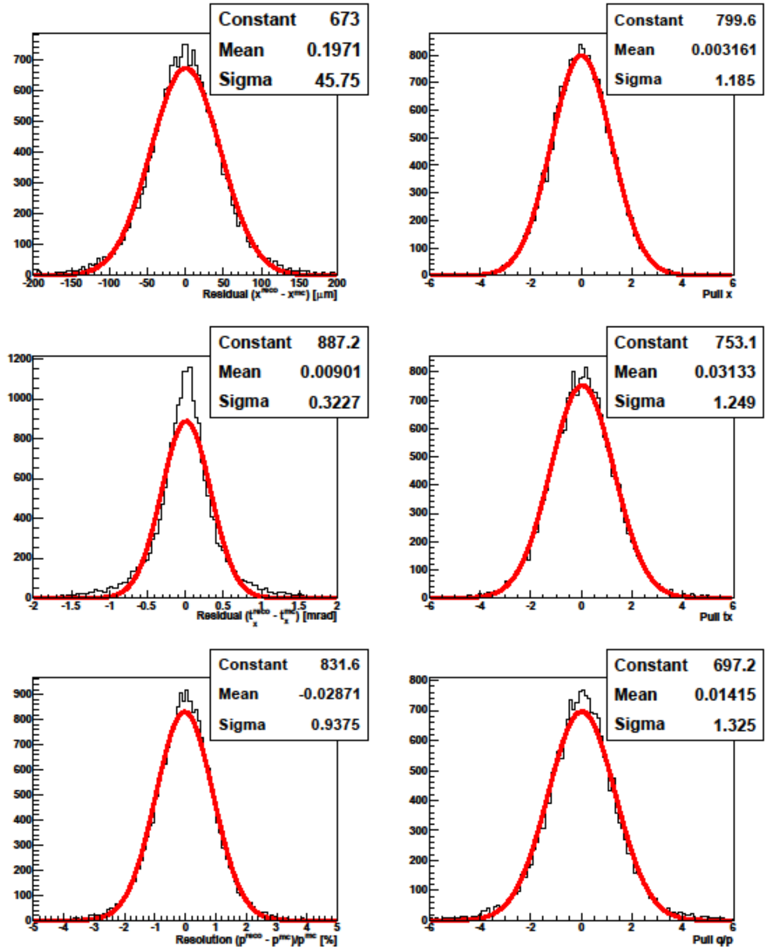
CBM Kalman Filter (KF) Track Fit Benchmark with ArBB

Array Building Blocks (ArBB) allows to avoid a lot of inconveniencies of parallel programming. It should be very useful for the event reconstruction.

Implementation of KF based on ArBB was the first step for the track finders ArBB-zation.

SIMD KF fit benchmark with ArBB has been implemented by Intel.
Comparison with SIMD KF fit benchmark based on Vector classes (Vc) was done.

	Vc		ArBB	
Cores	1	16	1	16
Time, μs	0.42	0.05	0.43	0.06



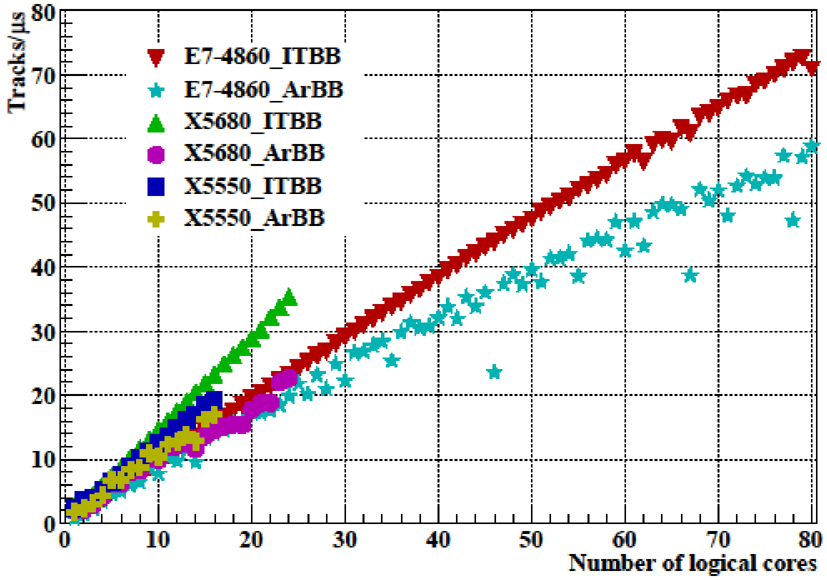
Tests were performed on the lxir039 computer with 2 Xeon X5550 processors having 8 cores in total at 2.7 GHz

conventional, geometry with 0 & 90 degree strips

With H. Pabst

KF track fit based on ArBB has been implemented by Intel.

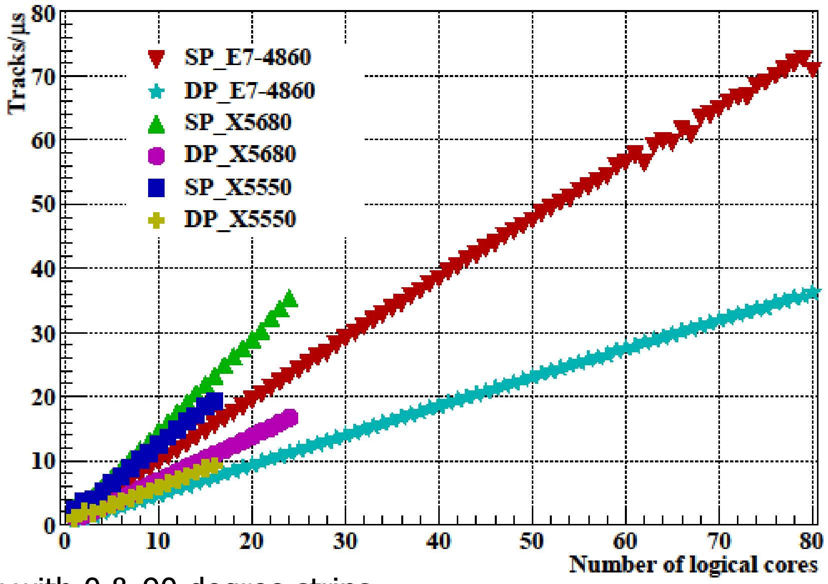
Scalability of Track Fit with Conventional Approach



Measure tracks throughput rather than time per track.

ITBB: Given n threads each filled with 1000 events, run them on specific n logical cores with 1 thread per 1 core.
Use "header" for data level parallelization.

ArBB: auto parallelization on both task and data level



Scalabilities for single and double using ITBB precision have been measured as well.

geometry with 0 & 90 degree strips

Conventional KF Implementation

Prediction step $\hat{x}_{k-1}^{+} \quad P_{k-1}^{+} \longrightarrow \hat{x}_{k}^{-} \quad P_{k}^{-}$

$$P_k^- = F_{k-1} P_{k-1}^+ F_{k-1}^T + Q_{k-1}$$

$$\hat{x}_k^- = F_{k-1} \hat{x}_{k-1}^+$$

Filtering step $\hat{x}_{k}^{-} \quad P_{k}^{-} \longrightarrow \hat{x}_{k}^{+} \quad P_{k}^{+}$

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}$$

$$P_k^+ = (I - K_k H_k) P_k^-$$

$$\hat{x}_k^+ = \hat{x}_k^- + K_k (y_k - H_k \hat{x}_k^-)$$

Square Root KF Implementation

$$\mathbf{P} \longrightarrow \mathbf{SS^T}$$

Twice a precision in comparison with conventional,
but has more complicated computations = slower.

Prediction step

$$\hat{x}^+_{k-1} \quad S^+_{k-1} \longrightarrow \hat{x}^-_k \quad S^-_k$$

$$\begin{bmatrix} (S^-_k)^T \\ 0 \end{bmatrix} = T \begin{bmatrix} (S^+_{k-1})^T F^T_{k-1} \\ Q^{T/2}_{k-1} \end{bmatrix}$$

$$\hat{x}^-_k = F_{k-1}\hat{x}^+_{k-1}$$

orthogon...

MGS Algorithm: For $k = 1, n$

$$\sigma_k = \sqrt{A_k{}^{(k)T} A_k{}^{(k)}}$$

$$W_{kj} = \begin{cases} 0, & j = 1, k-1 \\ \sigma_k, & j = k \\ \dfrac{1}{\sigma_k} A_k{}^{(k)T} A_j{}^{(k)}, & j = k+1, n \end{cases}$$

$$b_k{}' = \frac{1}{\sigma_k} A_k{}^{(k)T} b^{(k)}$$

$$A_j{}^{(k+1)} = A_j{}^{(k)} - \frac{W_{kj}}{\sigma_k} A_k{}^{(k)}, \qquad j = k+1, n$$

$$b^{(k+1)} = b^{(k)} - \frac{b_k{}'}{\sigma_k} A_k{}^{(k)}.$$

orthogonalization

Filtering step

$$\hat{x}^-_k \quad S^-_k \longrightarrow \hat{x}^+_k \quad S^+_k$$

Potter Implementation
(1. with square and 2. with triangular S matrix)

$$\phi_i = S^{+T}_{i-1,k} H^T_{ik}$$

$$a_i = \frac{1}{\phi^T_i \phi_i + R_{ik}}$$

$$\gamma_i = \frac{1}{1 \pm \sqrt{a_i R_{ik}}}$$

$$S^+_{ik} = S^+_{i-1,k}(I - a_i \gamma_i \phi_i \phi^T_i)$$

$$\hat{x}^+_{ik} = \hat{x}^+_{i-1,k} + K_{ik}(y_{ik} - H_{ik}\hat{x}^+_{i-1,k})$$

Carlson Implementation

orthogonalization

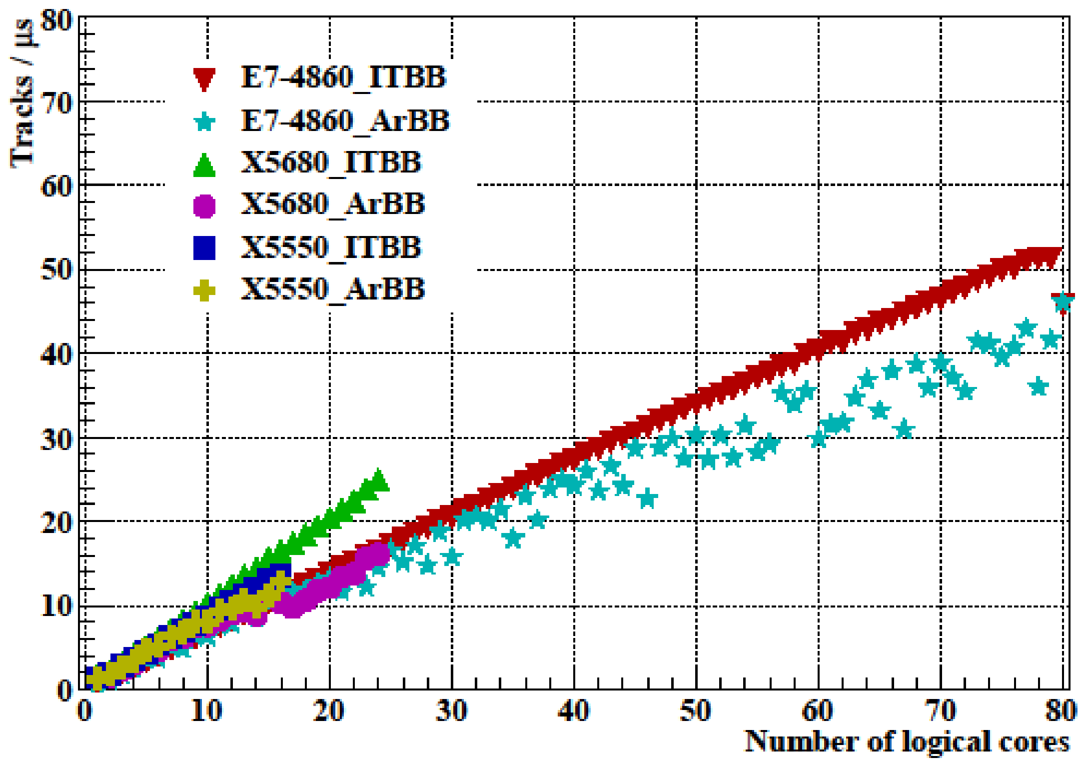
$$K_k = P^-_k H^T_k (H_k P^-_k H^T_k + R_k)^{-1}$$

$$\tilde{K}_k = K_k (R_k + H_k P^-_k H^T_k)^{T/2}$$

$$\begin{bmatrix} (R_k + H_k P^-_k H^T_k)^{T/2} & \tilde{K}^T_k \\ 0 & (S^+_k)^T \end{bmatrix} = \tilde{T} \begin{bmatrix} R^{T/2}_k & 0 \\ (S^-_k)^T H^T_k & (S^-_k)^T \end{bmatrix}$$

$$\hat{x}^+_k = \hat{x}^-_k + K_k(y_k - H_k \hat{x}^-_k)$$

Scalability of Track Fit with Square Root (Potter) Approach



geometry with 0 & 90 degree strips; the analytic formula for propagation

U-D-Filtering Implementation

$$P = UDU^T$$

$$\begin{bmatrix} 1 & u_{12} & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d_{11} & 0 & 0 \\ 0 & d_{22} & 0 \\ 0 & 0 & d_{33} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ u_{12} & 1 & 0 \\ u_{13} & u_{23} & 1 \end{bmatrix}$$

Increase precision in comparison with conventional.
Less number of computations than with square root.

Prediction step $U_{i-1}\ D_{i-1} \longrightarrow U_i\ D_i$

$$W = \begin{bmatrix} FU^+ & I \end{bmatrix}$$

$$\hat{D} = \begin{bmatrix} D^+ & 0 \\ 0 & Q \end{bmatrix}$$

$$W = U^- V \xrightarrow{\text{orthogonalization}}$$

$$v_n = w_n$$

$$v_k = w_k - \sum_{j=k+1}^{n} \frac{w_k \hat{D} v_j^T}{v_j \hat{D} v_j^T} v_j \qquad k = n-1, \cdots, 1$$

$$D^- = V \hat{D} V^T$$

Filtering step $U^-\ \ D^- \longrightarrow U^+\ D^+$

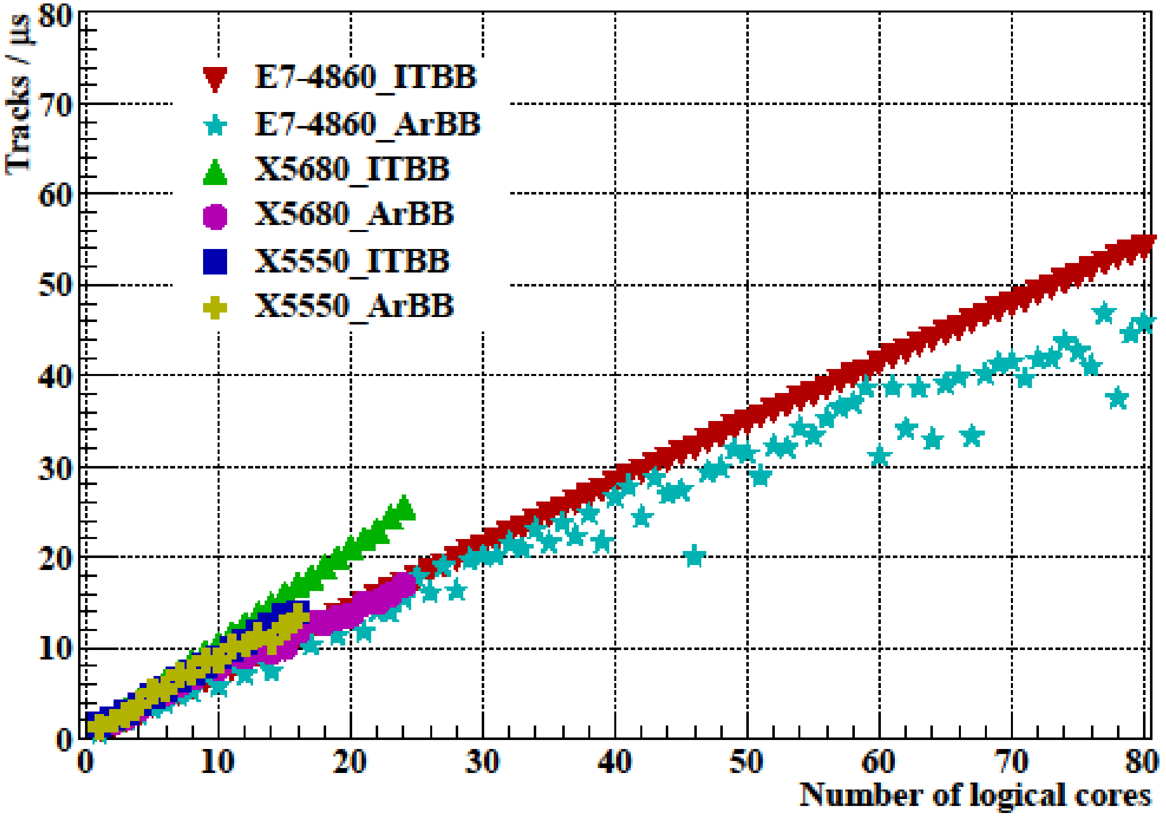
$$\alpha_i \equiv H_i P_{i-1} H_i^T + R_i$$

$$\bar{U}\bar{D}\bar{U}^T = \left[D_{i-1} - \frac{1}{\alpha_i}(D_{i-1}U_{i-1}^T H_i^T)(D_{i-1}U_{i-1}^T H_i^T)^T \right]$$

$$U_i = U_{i-1}\bar{U}$$

$$D_i = \bar{D}$$

Scalability of Track Fit with UD-filtering Approach



geometry with 0 & 90 degree strips; the analytic formula for propagation

Runge-Kutta Propagation Method

General method.

$$\frac{\mathrm{d}\mathbf{r}(z)}{\mathrm{d}z} = \begin{pmatrix} t_x \\ t_y \\ \kappa \cdot (q/p) \cdot \sqrt{1+t_x^2+t_y^2} \cdot \left(t_x t_y \cdot B_x - (1+t_x^2) \cdot B_y + t_y \cdot B_z\right) \\ \kappa \cdot (q/p) \cdot \sqrt{1+t_x^2+t_y^2} \cdot \left(\left(1+t_y^2\right) \cdot B_x - t_x t_y \cdot B_y - t_x \cdot B_z\right) \\ 0 \end{pmatrix} \equiv \mathbf{f}(z, \mathbf{r})$$

$$\begin{aligned} \Delta\mathbf{r}_1 &= \mathbf{f}(z_0, \mathbf{r}_0) \cdot \Delta z \ , \\ \Delta\mathbf{r}_2 &= \mathbf{f}(z_0 + \tfrac{\Delta z}{2}, \mathbf{r}_0 + \tfrac{\Delta\mathbf{r}_1}{2}) \cdot \Delta z \ , \\ \Delta\mathbf{r}_3 &= \mathbf{f}(z_0 + \tfrac{\Delta z}{2}, \mathbf{r}_0 + \tfrac{\Delta\mathbf{r}_2}{2}) \cdot \Delta z \ , \\ \Delta\mathbf{r}_4 &= \mathbf{f}(z_0 + \Delta z, \mathbf{r}_0 + \Delta\mathbf{r}_3) \cdot \Delta z \ . \end{aligned}$$

$$\mathbf{r}(z_e) = \mathbf{r}_0 + \left(\frac{1}{6}\Delta\mathbf{r}_1 + \frac{1}{3}\Delta\mathbf{r}_2 + \frac{1}{3}\Delta\mathbf{r}_3 + \frac{1}{6}\Delta\mathbf{r}_4\right) + O((\Delta z)^5)$$

Analytic Formula for Track Propagation

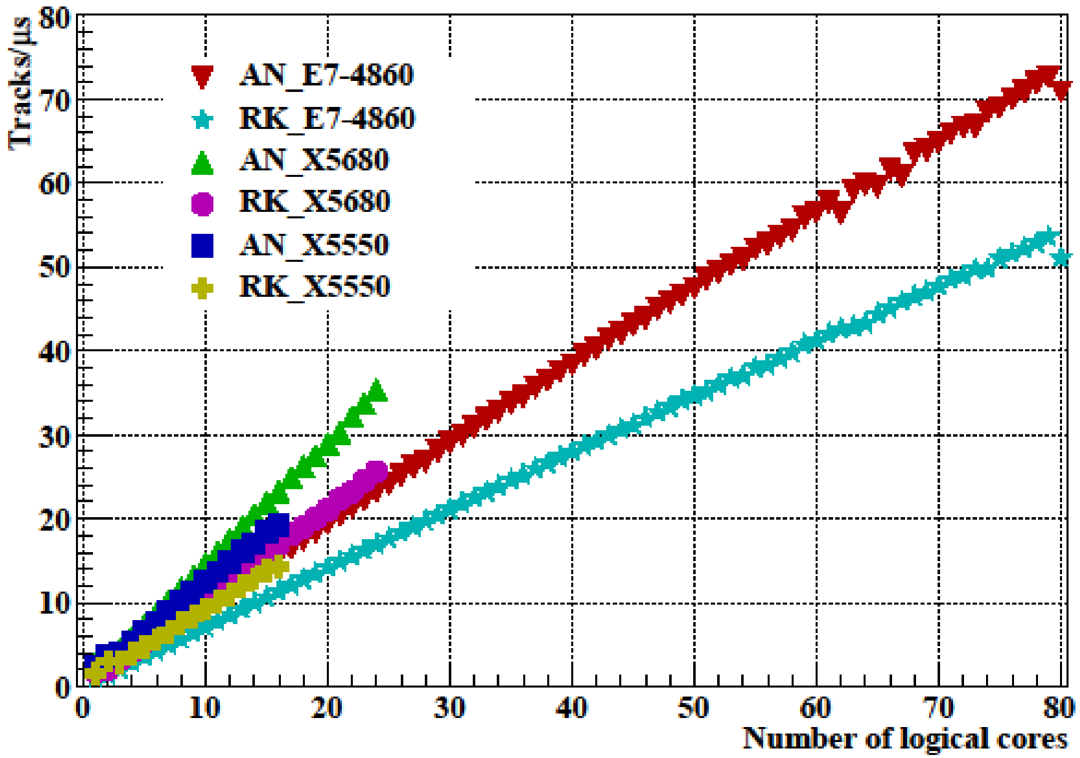
Allows to control precision and time consumptions.

$$
\begin{aligned}
x' &\equiv t_x \\
y' &\equiv t_y \\
t'_x &= \kappa \cdot (q/p) \cdot \sqrt{1 + t_x^2 + t_y^2} \cdot \Big(t_x t_y \cdot B_x - (1 + t_x^2) \cdot B_y + t_y \cdot B_z \Big) \\
t'_y &= \kappa \cdot (q/p) \cdot \sqrt{1 + t_x^2 + t_y^2} \cdot \Big(\big(1 + t_y^2\big) \cdot B_x - t_x t_y \cdot B_y - t_x \cdot B_z \Big) \\
(q/p)' &= 0
\end{aligned}
$$

Taylor expansion

$$
\begin{aligned}
t_x(z_e) &= t_x(z_0) + \sum_{k=1}^{n} \sum_{i_1,\ldots,i_k = x,y,z} t_{x i_1 \ldots i_k}(z_0) \cdot \left(\int_{z_0}^{z_e} B_{i_1}(z_1) \ldots \int_{z_0}^{z_{k-1}} B_{i_k}(z_k) \mathrm{d}z_k \ldots \mathrm{d}z_1 \right) , \\
t_y(z_e) &= t_y(z_0) + \sum_{k=1}^{n} \sum_{i_1,\ldots,i_k = x,y,z} t_{y i_1 \ldots i_k}(z_0) \cdot \left(\int_{z_0}^{z_e} B_{i_1}(z_1) \ldots \int_{z_0}^{z_{k-1}} B_{i_k}(z_k) \mathrm{d}z_k \ldots \mathrm{d}z_1 \right) , \\
x(z_e) &= x(z_0) + \int_{z_0}^{z_e} t_x(z) \mathrm{d}z , \\
y(z_e) &= y(z_0) + \int_{z_0}^{z_e} t_y(z) \mathrm{d}z ,
\end{aligned}
$$

Scalability of Track Fit with the Analytic and Runge-Kutta Propagation

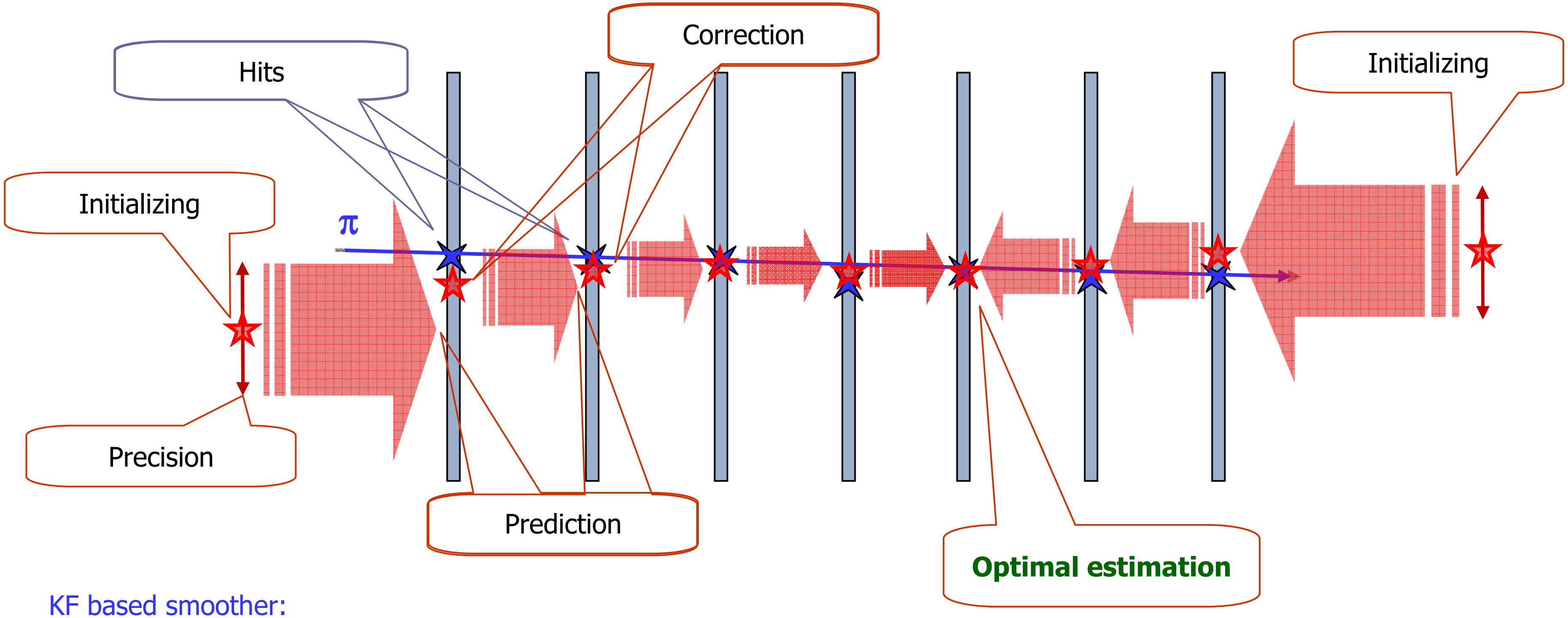


geometry with 0 & 90 degree strips; conventional approach; ITBB; header

Track Smoother

Optimal estimation of the track parameters at any station.

Use: for alignment of station positions
for fake hits rejection (DAF)
for track quality estimation (ghost tracks rejection)



KF based smoother:

1. Take 2 state vectors.
2. Start with an arbitrary initializations from begin and end of track.
3. Add one hit after another moving downstream and upstream.
4. Improve the state vectors.
5. Get two sets of parameters at the given station.
6. Merge them into one and get the optimal parameters.

Track Smooth Quality

$r = \{ x, y, t_x, t_y, q/p \}$

station		residuals					pulls				
		x, µm	y, µm	t_x, 10^{-3}	t_y, 10^{-3}	q/p, %	x	y	t_x	t_y	q/p
MVD	1	24	27	0.63	0.62	0.90	1.0	1.0	1.1	1.1	1.3
	2	19	22	0.39	0.43	0.89	1.0	1.0	1.0	1.0	1.3
STS	1	20	46	0.19	0.25	0.88	1.1	1.1	1.4	1.2	1.3
	2	15	49	0.25	0.35	0.88	1.1	1.1	1.2	1.2	1.3
	3	18	50	0.26	0.36	0.88	1.1	1.1	1.2	1.2	1.3
	4	21	56	0.22	0.34	0.88	1.1	1.1	1.2	1.2	1.3
	5	24	62	0.24	0.33	0.88	1.1	1.1	1.2	1.2	1.3
	6	30	67	0.23	0.30	0.89	1.1	1.1	1.2	1.2	1.2
	7	23	77	0.21	0.37	0.87	1.1	1.1	1.4	1.3	1.2
	8	30	101	0.50	0.60	0.90	1.1	1.1	1.1	1.2	1.2

conventional, header, geometry with 0 & 15 degree strips

KF track smoother has been implemented in KF Library.

Deterministic Annealing Filter (DAF)

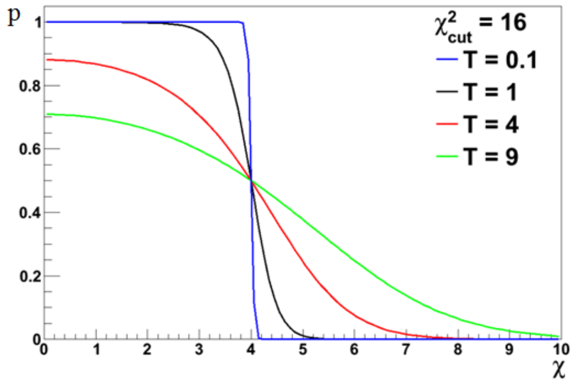
Task: reduce an influence of attached distorted or noise hits on the reconstructed track parameters.

- DAF has been implemented within SIMD KF track fit package
- The KF mathematics has been modified to include weights

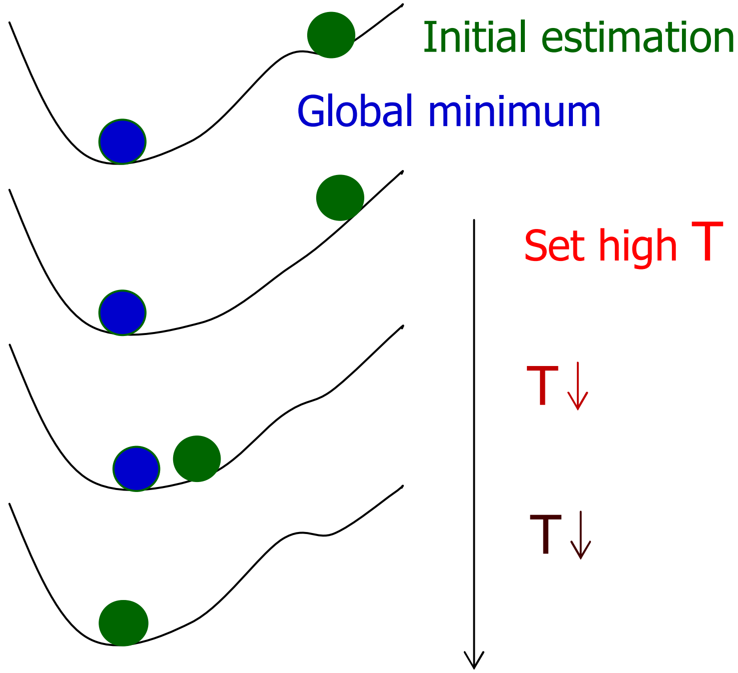
DAF algorithm:

- A weight is introduced to each hit

$$p = \frac{1}{1 + \exp\left((\chi^2 - \chi^2_{\rm cut})/(2T)\right)}$$



- Algorithm is iterative, with each iteration T is decreasing, weight is recalculated using smoothed track parameters from the previous iteration



R. Frühwirth and A. Strandlie, Track Fitting with ambiguities and noise: a study of elastic tracking and nonlinear filters. Comp. Phys. Comm. 120 (1999) 197-214.

DAF and Noise Hits Rejection

- The hit on the 4th STS station was displaced by a certain amount of the hit error (σ_{hit} = 17 μm) from the MC position

- The percentage of rejected hits was calculated. In ideal case for the 4th station it should be 100%, for other – 0%

Rejection probability, %					
station		unshifted	5 σ_{hit}	10 σ_{hit}	20 σ_{hit}
MVD	1	0.4	0.4	0.4	0.4
	2	0.7	0.7	0.7	0.7
STS	1	0.3	0.3	0.3	0.3
	2	0.4	0.4	0.4	0.4
	3	0.4	0.7	0.8	0.5
	4	**0.5**	**43.9**	**85.0**	**98.7**
	5	0.5	1.6	1.6	0.8
	6	0.6	0.6	0.6	0.6
	7	0.6	0.6	0.6	0.6
	8	0.1	0.1	0.1	0.1

In collaboration with R. Frühwirth (HEPHY, Austria) and A. Strandlie (Uni-Oslo, Gjøvik University College, Norway)

Progress

		Conventional	Square root	UD-filtering
ITBB	Fit	+	+	+
	Smooth	+	+	+
	DAF	+	+	+
ArBB	Fit	+	+	+
	Smooth			
	DAF			

Summary

- ✓ KF library includes track fit, track smooth and DAF algorithms; it is fast, scalable and allows to choose between different KF approaches, two propagation methods, different parallelization libraries, as well as between single and double precision calculations.
- ✓ KF library has been fully implemented with ITBB and tested. ArBB implementation is in progress.

- Back up

covariance matrix -> square root of covariance matrix

$$\boxed{\mathbf{P = SS^T}}$$

Transport example

$$\mathbf{P' = F\,P\,F^T}$$

$$P = \begin{pmatrix} 1 & 0 \\ 0 & 10^{10} \end{pmatrix} \quad F = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad P' = \begin{pmatrix} 10^{10}+1 & 10^{10} \\ 10^{10} & 10^{10} \end{pmatrix} = \begin{pmatrix} 10^{10} & 10^{10} \\ 10^{10} & 10^{10} \end{pmatrix}$$

Lose information!

$$\mathbf{S' = F\,S}$$

$$S = \begin{pmatrix} 1 & 0 \\ 0 & 10^{5} \end{pmatrix} \quad F = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad S' = \begin{pmatrix} 1 & 10^{5} \\ 0 & 10^{5} \end{pmatrix}$$

No problem with precision