Speeding Up Tracking

Simon Taylor/JLAB

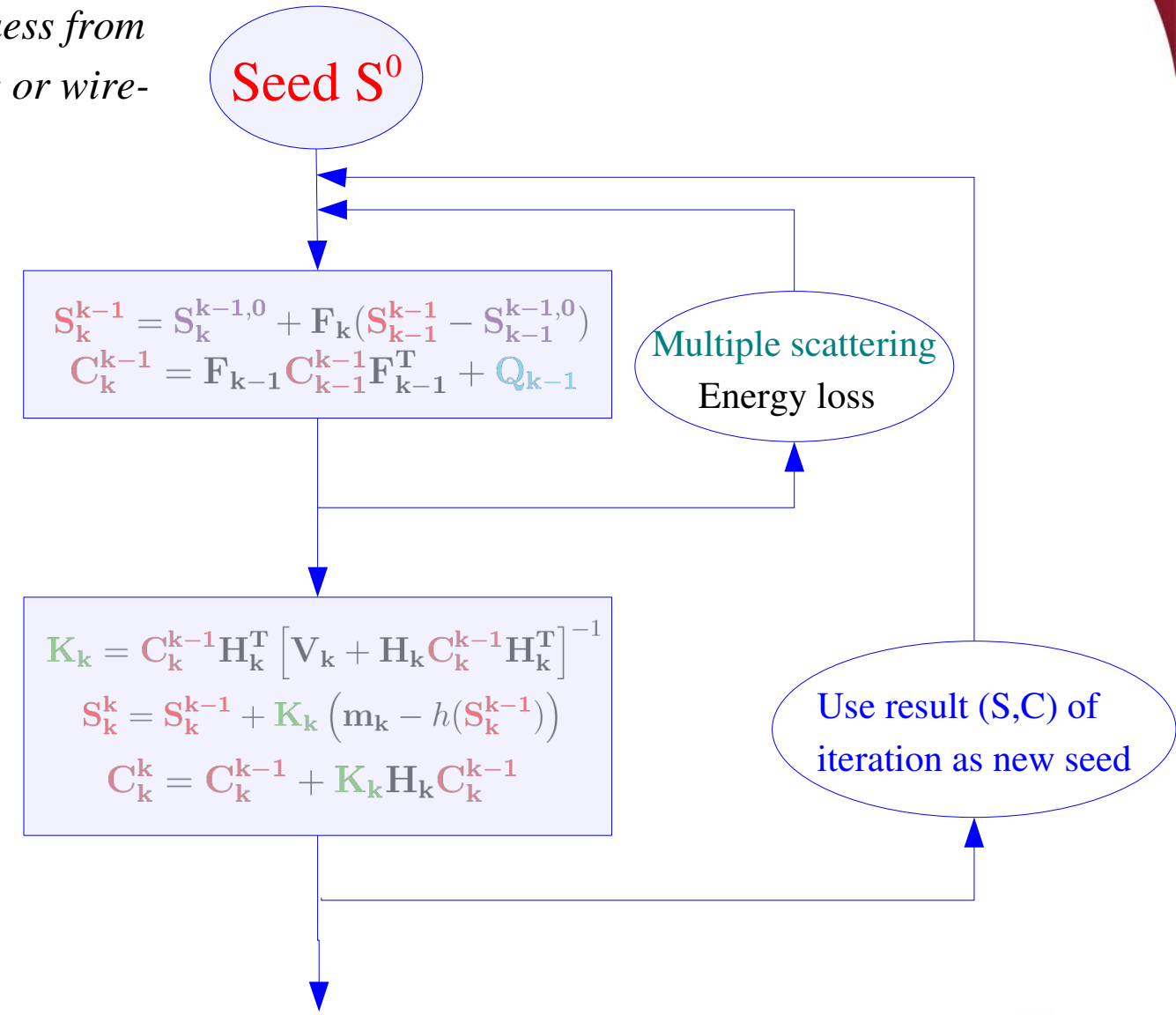
- Kalman Filter review
- Places to look for speed-up
- Vectorization
- Results

Kalman Filter

- Track described by 5 parameter state vector at each point along its path
  - Forward-going tracks: $\{x,y,t_x,t_y,q/p\}$
  - Central tracks: $\{q/p_T,\phi,\tan\lambda,D,z\}$
- State vector propagated step-by-step toward target
  - Multiple scattering and energy loss taken care of at each step
- State vector treated as a perturbation to an initial guess
  - Generate reference trajectory by swimming with initial parameters from the target
- Perform filter in multiple passes
  - Can iterate up to 10 times per pass, checking for χ^2 convergence
  - Regenerate reference trajectory based on results of previous pass

Kalman Filter Algorithm

Start with guess from track finding or wire-based stages



Propagate state S and covariance C through magnetic field

Use measurements to update state vector

Where can we speed up the code?

- Code obtains magnetic field multiple times per step
  - Implemented *FineMesh* lookup table – no interpolation...

Reconstruction rates with 4 threads on ifarml1

Particle	Interpolation(Hz)	FineMesh(Hz)	Note
p	24.6	34.9	two mass hypotheses
π^-	44.1	62.3	one mass hypothesis

- Code performs copious amounts of matrix operations per iteration
  - Old code relied on TMatrix, a generic matrix package in ROOT
  - Largest matrix in Kalman Filter is 5×5 → could benefit from custom matrix classes
    - Implemented new matrix classes

    Dmatrix2x1,Dmatrix3x1,DMatrix3x2,...,DMatrix5x5

Vectorization

- Modern CPUs have special registers that support Single Instruction, Multiple Data operations ("vectorization")
  - Operations on 4 ints, 4 floats, or 2 doubles at a time can be done in parallel
  - "Streaming SIMD Extensions"
- SSE2 instructions
  - ADDPD 

  - SUBPD 

  - MULPD 

- SSE3 instructions
  - HADDPD = horizontal add 

Matrix Operation Benchmarks

- Compare Root-based matrices (TMatrixD) to custom SIMD-ized matrices (DMatrix)
  - 10,000,000 “events” on ifarml6 (2.8 GHz Nehalem)

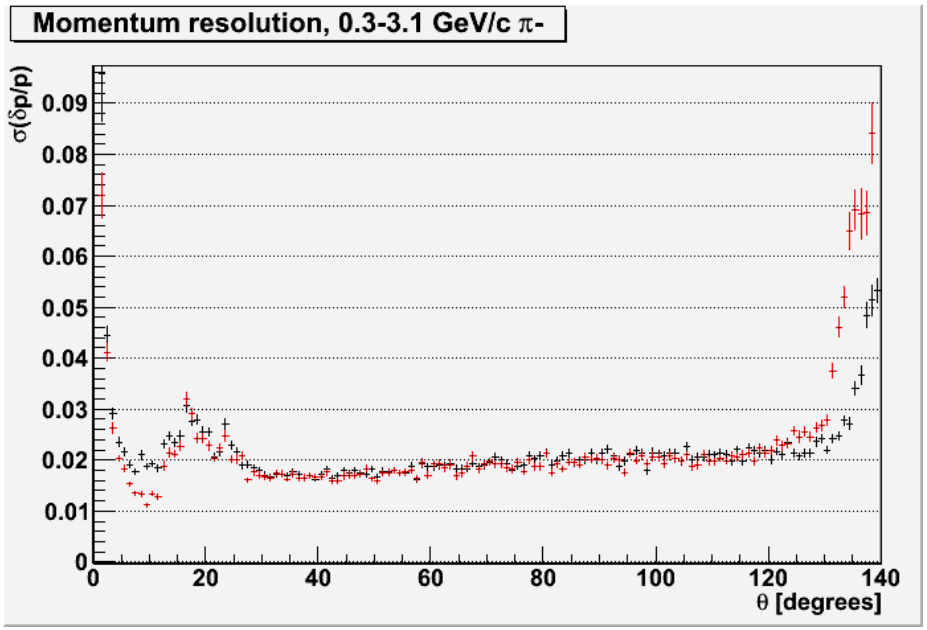
Operation	TMatrixD(s)	DMatrix(s)	T/D
$A_{2\times2}^{-1}$	1.493	0.090	16.59
$A_{5\times5}^{-1}$	6.274	2.274	2.76
$A_{5\times5}+B_{5\times5}$	1.470	0.609	2.41
$A_{5\times5}B_{5\times5}$	3.821	0.974	3.82
$A_{5\times5}^{T}$	1.178	0.393	3.00
$A_{5\times5}^{T}B_{5\times5}A_{5\times5}$	7.892	1.857	4.25
$C_{1\times5}A_{5\times5}D_{5\times1}$	2.466	0.520	4.74
$E_{2\times5}A_{5\times5}F_{5\times2}$	3.630	0.651	5.58

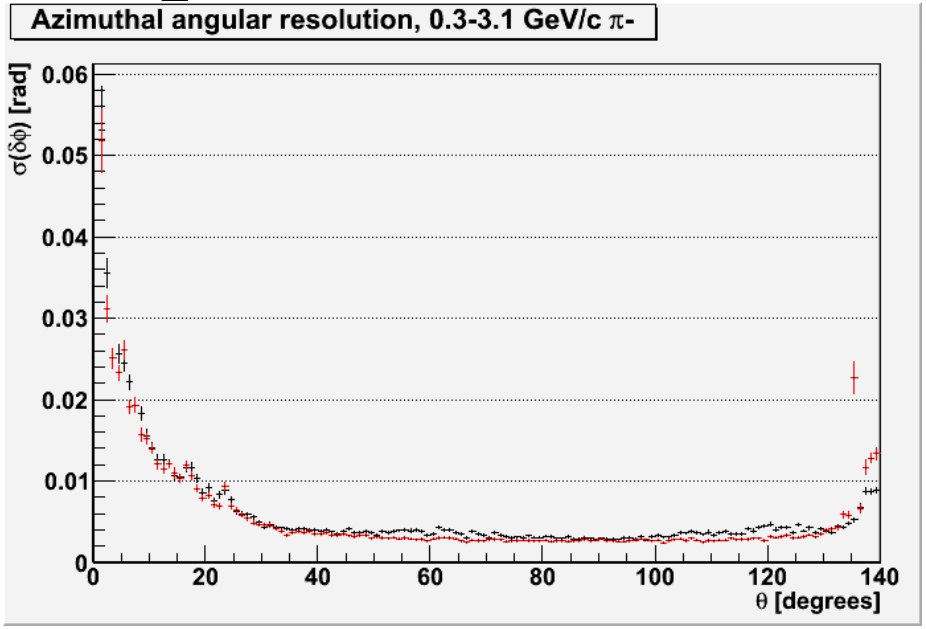
Reconstruction Rates

- Generated 50000 events → HDGeant → MCSmear → hd_ana
- 4 threads on ifarml1(2.8 GHz Nehalem)

Topology	ALT1(Hz)	Kalman(Hz)	KalmanSIMD(Hz)
π^+ (particle gun)	56.0	32.8	141.7
π^- (particle gun)	127.7	62.3	269.6
p (particle gun)	85.9	34.9	156.8
$p\rho$	14.9	7.4	46.5
$n\pi^+\pi^+\pi^-$	7.5	6.4	31.8
$pb_1{}^+\pi^-$	3.8	4.7	18.2

Resolution comparison

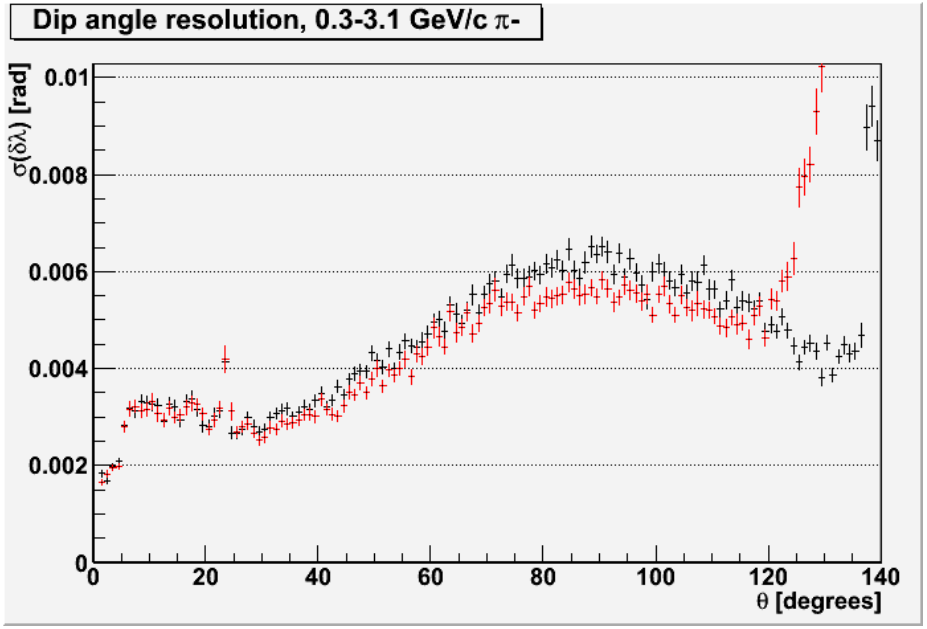




ALT1

KalmanSIMD

Particle gun



Conclusion

- FineMesh option for magnetic field speeds up Kalman filter by ~40%
- SIMD-ized version of the Kalman Filter runs ~2-3× faster than ALT1
  - Except for back angles, no degradation in track parameter resolution

Plan to switch to KalmanSIMD as default on 64bit machines in next software release