How to have your cake and eat it

Inverting a sum of small matrices

Stefan Kluth
MPI für Physik
ID Performance WS
24.10.2005

The LLS Alignment Fit

General solution for alignment parameters $\mathbf{a}$, initial values $\mathbf{a}_0$, track residuals $\mathbf{r}_i$ with error matrices V_i, Jacobi (derivative) matrices $H_i = d\mathbf{r}_i/d\mathbf{a}$; index i counts events:

$$\mathbf{a}_k = \mathbf{a}_0 - (\sum_{i=1}^{k} H_i^T V_i^{-1} H_i)^{-1} \cdot \sum_{i=1}^{k} H_i V_i^{-1} \mathbf{r}_i = \mathbf{a}_0 - C_k \sum_{i=1}^{k} H_i V_i^{-1} \mathbf{r}_i$$

Correspondence to Kalman filter; consider $\mathbf{a}_k - \mathbf{a}_{k-1}$:

$\mathbf{a}_k = \mathbf{a}_{k-1} + -C_{k-1} H_{k-1} V_{k-1}^{-1} \mathbf{r}_{k-1}$ Update formula with “gain matrix”

$C_k^{-1} = C_{k-1}^{-1} + H_k^T V_k^{-1} H_k$ Error propagation

Interesting (→CMS), but not essential to matrix inversion

Iterative Matrix Inversion

Start from error propagation: $C_k^{-1} = C_{k-1}^{-1} + H_k^T V_k^{-1} H_k$ (1)

$$C_k^{-1} = C_{k-1}^{-1} (1 + C_{k-1} H_k^T V_k^{-1} H_k)$$

$C_k = (1 + C_{k-1} H_k^T V_k^{-1} H_k)^{-1} C_{k-1}$; Taylor Expansion of $(1+\varepsilon)^{-1}$

$$C_k \approx (1 - C_{k-1} H_k^T V_k^{-1} H_k) C_{k-1} \quad (2)$$

Is Taylor expansion justified?

$C_{k-1} (C_k^{-1} - C_{k-1}^{-1}) = C_{k-1} H_k^T V_k^{-1} H_k = C_{k-1} C_k^{-1} - 1 \approx 0 \Rightarrow$ yes

Now have iterative update formulas for C_k^{-1} and C_k

Bootstrapping

Need initial value for C_0^{-1} in order to start iteration

Physically correct: $C_0^{-1} = 0$, but breaks (2)

Choose $C_0^{-1} = 1 \Rightarrow$ solution is $C_k = (1 + \sum_{i=1}^{k} H_i^T V_i^{-1} H_i)^{-1}$

$$\Rightarrow C_k^{-1} = (1 + \sum_{i=1}^{k} H_i^T V_i^{-1} H_i) = 1 + C'^{-1}_k$$

$$\Rightarrow C'_k = (C^{-1}_k - 1)^{-1} = (1 - C_k)^{-1} C_k \approx C_k + C_k C_k$$

if C_k is sufficiently small. Thus correction for $C_0^{-1} = 1$ possible but vanishes asymptotically ($k \rightarrow \infty$) anyway

Proof of Principle Tests

Tests with 20x20 matrix; add 2x2 “error matrices” ($\sigma^2 = 0.1$, $\rho = 0.5$) to random row and column

Accumulate (1) and (2), correct for $C_0 = 1$

after n events, calculate $C_k^{-1} C_k$, calculate true C'_k,

events	test	1st order	2nd order
10^3	$C_k^{-1} C_k$	< 3%	< 0.1%
	$C_0 = 0$	3%	1%
10^4	$C_k^{-1} C_k$	< 1%	< 0.1%
	$C_0 = 0$	< 1%	< 0.1%

Computing/Numerics

- Computing issues
  - multiply large matrices once or twice / event
    - $\dim(C) = N = O(10^4)$, $\dim(H_i^T V_i^{-1} H_i) = n = O(100)$
    - $N^2 n \sim O(10^{10})$ flop/event $\Rightarrow$ 10 s/event on Gflop CPU $\Rightarrow$ need (embarrasingly) parallel processing?
  - job needs > 1 GB RAM
    - ~ 1 GB for matrices + Athena
  - seems possible on (large) normal farm
- Numerics
  - expect double to be sufficient due to iterations

Computing

- Split data processing and matrix calculation?
  - Athena jobs write $H_i^T V_i^{-1} H_i$ (O(10 kB/event))
  - Dedicated jobs do matrix calculations
  - basically the current “big matrix” approach
  - but with or without explicit parallel processing
  - try to fit into standard system
    - CAF or Tier-1/2
  - avoid buying + maintaining dedicated system

Parallel Computing

- Embarrasingly parallel processing
  - many (N_{job}) jobs processing same type of data
  - straight weighted averages of results
  - averaged covariances scaled by $1/N_{job}$
  - limited by need for finite # events / job
- Explicit parallel processing
  - run single job on many processors
  - needs dedicated environment
    - e.g. MPI integrated into CAF

Conclusions/Outlook

- Invert sum of small matrices iteratively
  - avoid numerical problems of direct inversion
  - fairly easy to run (embarrasingly) parallel
  - should be able to run on CAF or Tier-1/2
- Need studies with larger matrices
  - proof-of-principle and flop counting
  - real world: Athena
  - comparison with direct inversion where possible