

Constraining COT Track t_0

Chris Hays, Duke University

Tracking Meeting

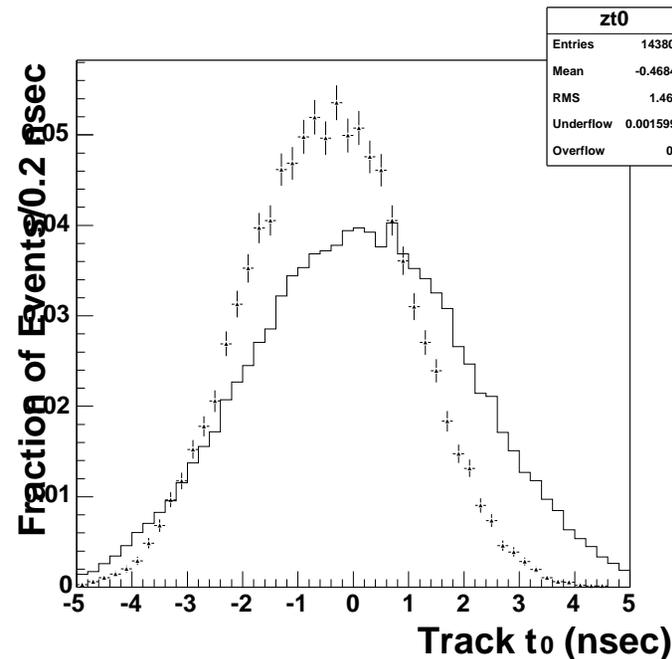
March 3, 2004

Motivation

t_0 calibrated for each run

Event vertices have ~ 1.5 ns spread in t_0

- 30 cm (=1 ns) luminous region
- Multiply by $\sqrt{2}$ for bunch-bunch overlap in time
- Z muons show 1.5 ns RMS (track t_0 resolution ~ 0.4 ns)



Z muons have 0.5 ns offset (believed due to slow particles in Stage0 t_0 fit)

1 ns \sim 50 microns

t_0 constraint could remove 75 μm contribution to resolution and 25 μm systematic

“Fast” Algorithm

Internal to COT code

Step 1: Create vertex seeds

- Loop over tracks and create new vertex if no existing vertex within 1 ns, 4 cm
- Add tracks to vertices if $\Delta t_0 < 1$ ns, $\Delta z_0 < 4$ cm, $d_0 < 4$ mm

Step 2: Merge vertices

- Combine all vertices within 1.5 ns and 6 cm
- Keep only vertices with ≥ 3 tracks

Step 3: Constrain tracks

- Constrain track to best vertex within 1.5 ns and 6 cm

“Full” Algorithm

Use best available information

Step 1: Find t_0 for each COT vertex in ZVertexColl

- Use highest p_T track to seed t_0 for each vertex
- Require tracks to be consistent with vertex:
 - $|\Delta z|/\sigma z < 3$
 - $d_0 < 3$ mm (COT-only), $d_0 < 300$ μm (SVX)
- Add tracks to t_0 if within 1.5 ns of vertex
- Add new t_0 for if track $t_0 > 1.5$ ns from vertex t_0
- Up to three t_0 's for vertex

Step 2: Merge vertex t_0 's

- Combine t_0 's for vertex if $\Delta t_0 < 1.5$ ns
- Keep only t_0 's with ≥ 2 tracks

Step 3: Constrain tracks in PadTrackMaker

- Constrain track to best t_0 with $\Delta t_0 < 1.5$ ns and $\Delta z_0 < 5$ cm (< 1 cm for OIS/OIZ)

Vertices

“Full” more likely to have vertex t_0

- 85.4% of events have vertex t_0 compared to 74.5% for “fast”
- Probably due to looser requirement on # of tracks

“Fast” more likely to have multiple t_0 's

- 42.6% of events have multiple t_0 's compared to 3.2% for “full”
- Includes overlap in z_0 vertices

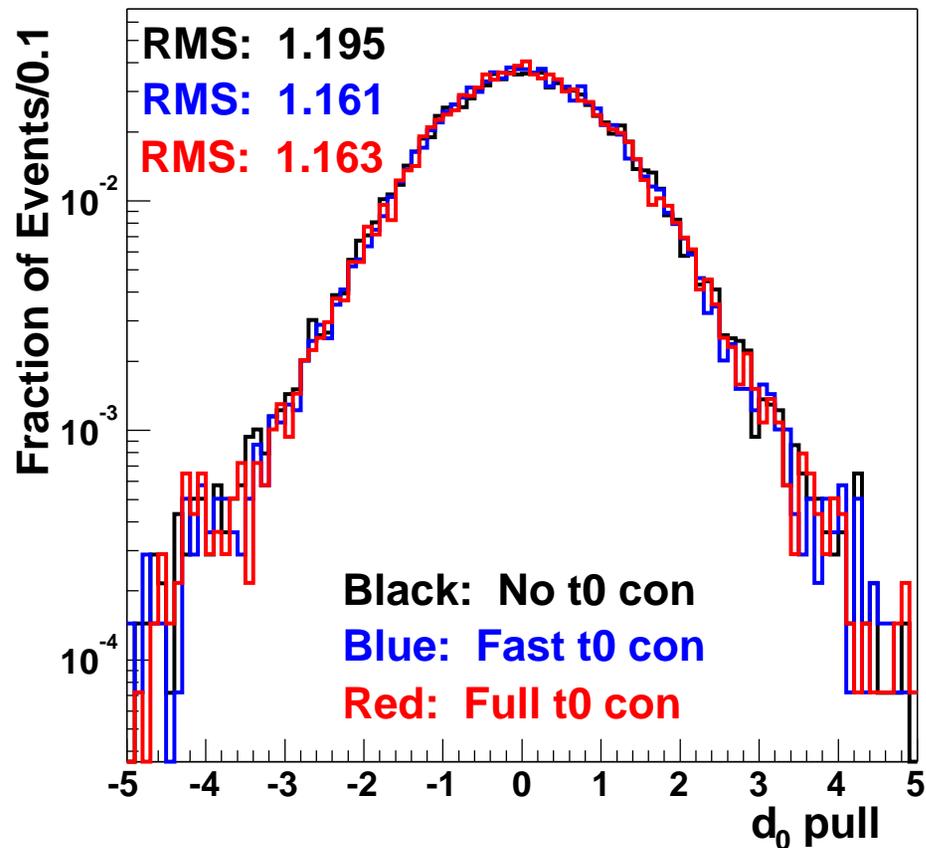
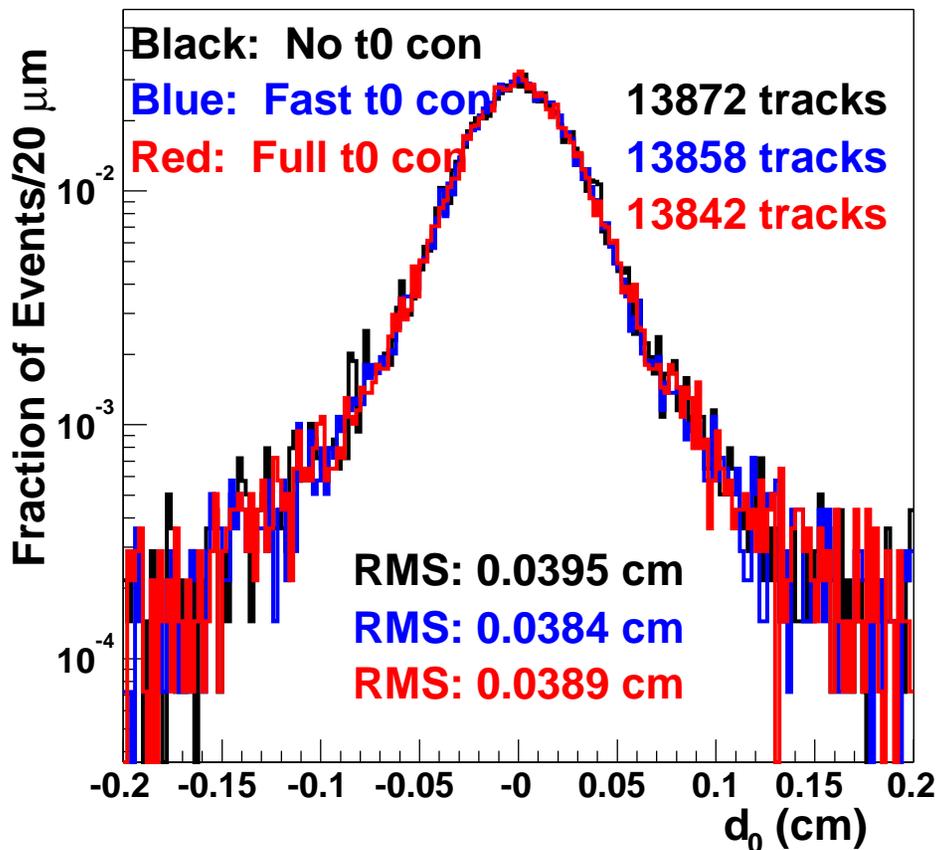
“Fast” more likely to constrain tracks

- 50.7% of tracks have constrained t_0 's compared to 41.3% for “full”

Performance: d_0

Test resolution and pull for non-BC COT tracks from Z 's with $76 < M_{\mu\mu}/\text{GeV} < 116$

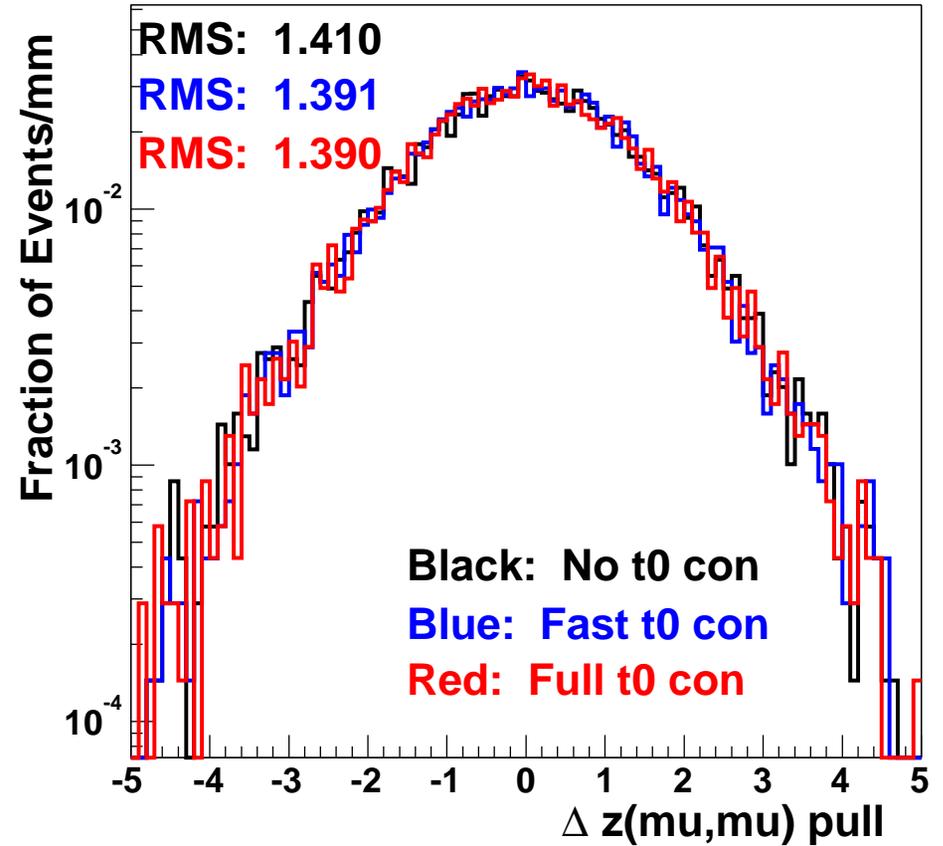
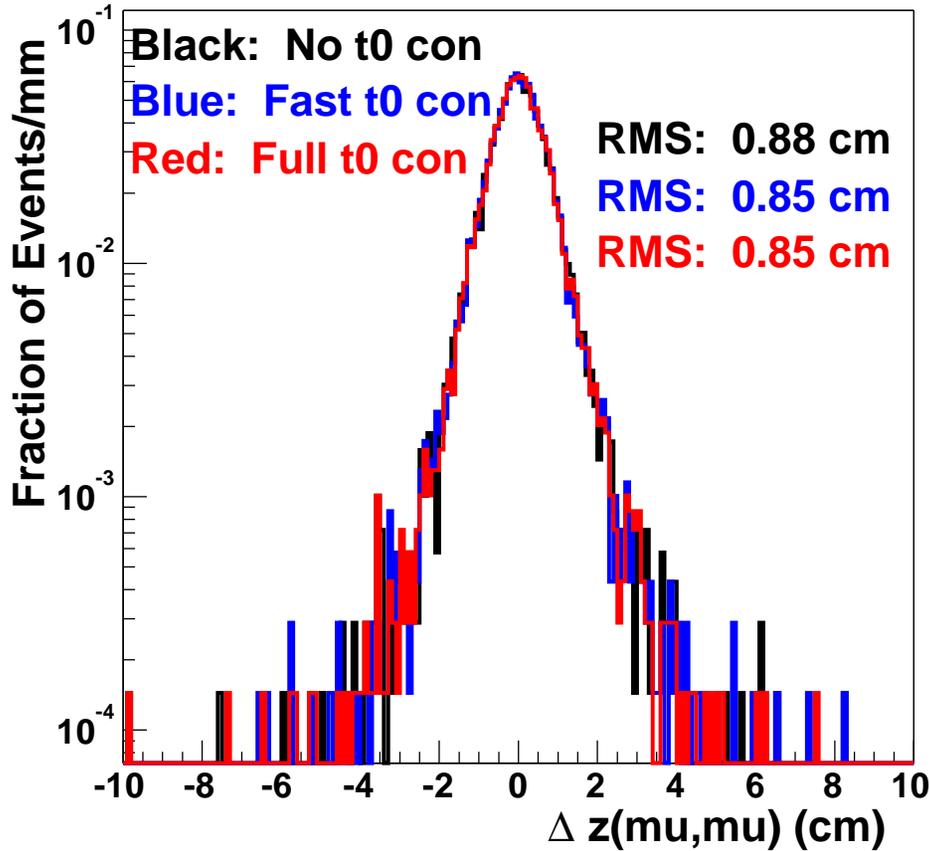
- Loosen had E cut to < 7 GeV to recover 0.5% inefficiency in “full” algorithm



t_0 constraint a definite improvement, slightly better with “fast” algorithm

Performance: z_0

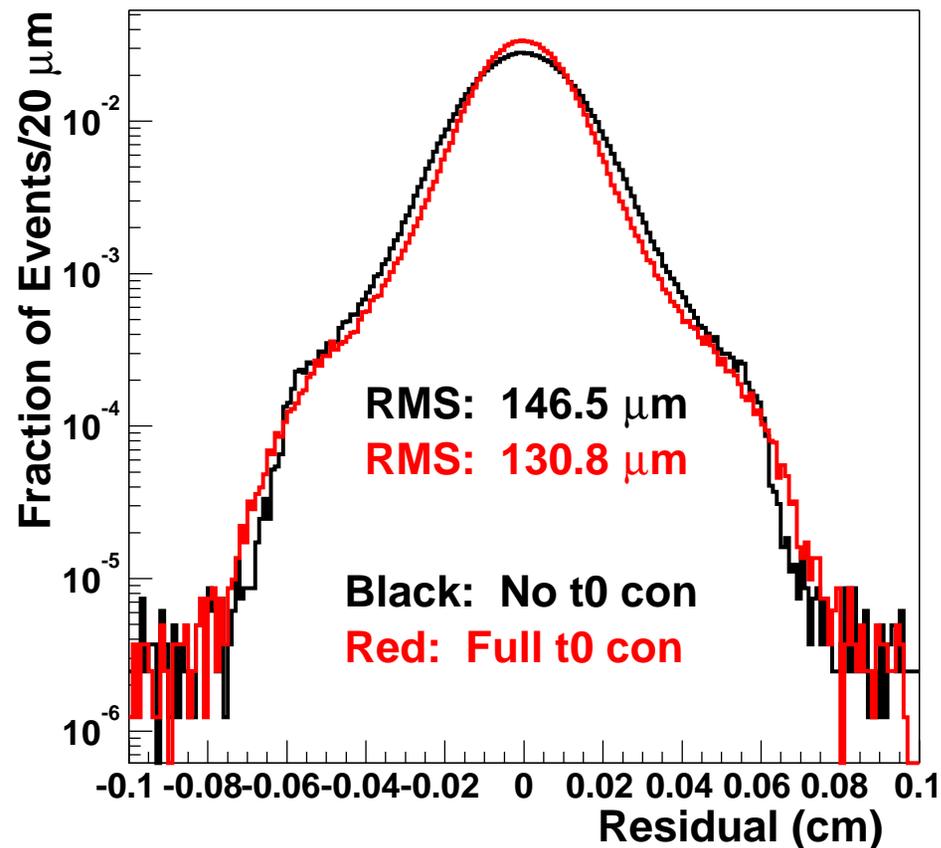
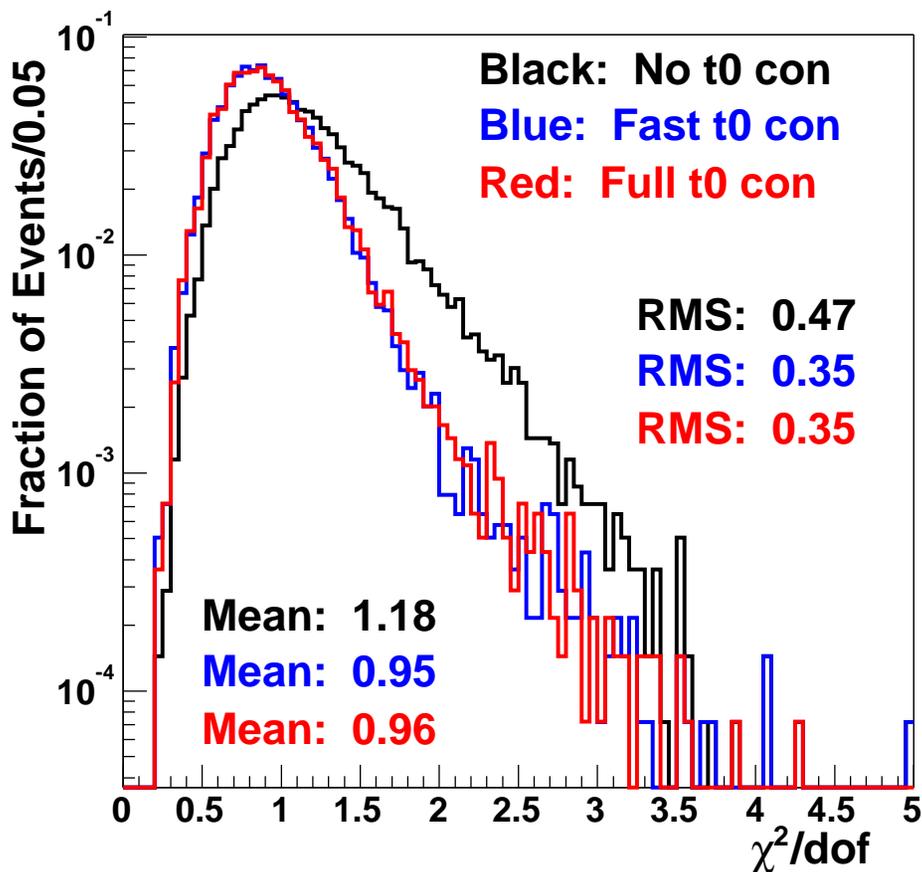
Test resolution and pull for non-BC COT tracks from Z 's with $76 < M_{\mu\mu} / \text{GeV} < 116$



t_0 constraint a definite improvement

Performance: χ^2 , Residual

Test χ^2/dof non-BC and residual for BC COT tracks from Z 's

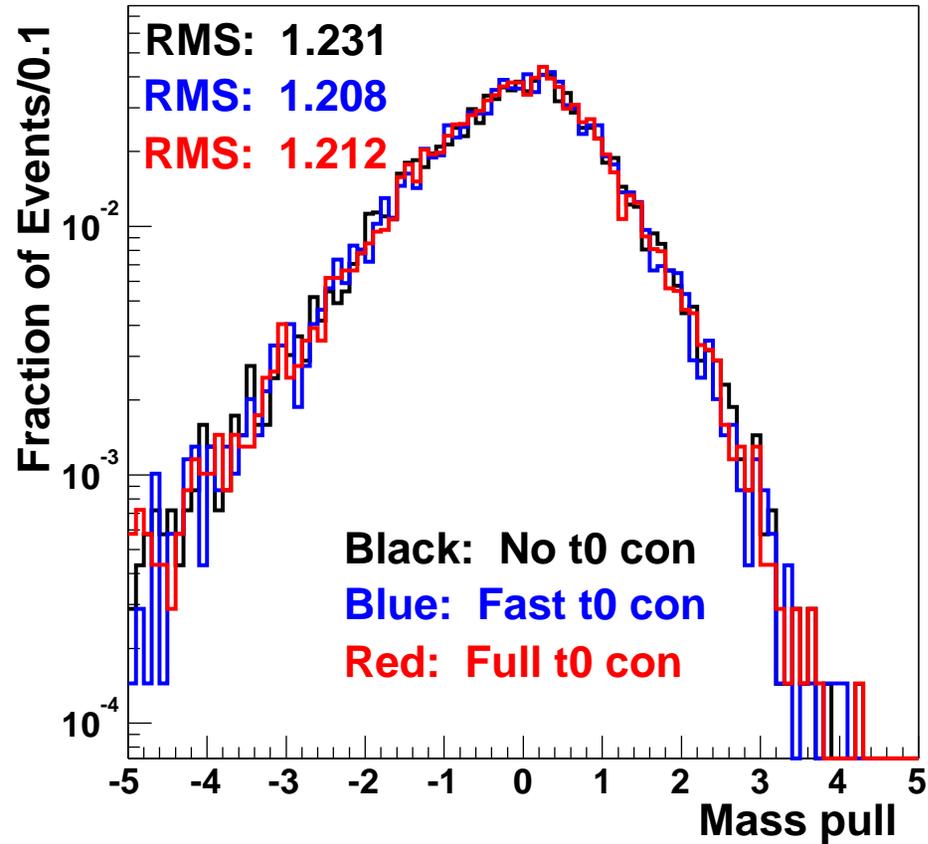


t_0 constraint a definite improvement, slightly better with “fast” algorithm

- χ^2 reduction corresponds to $152 \rightarrow 136 \mu\text{m}$ resolution ($68 \mu\text{m}$ reduction)

Performance: Mass

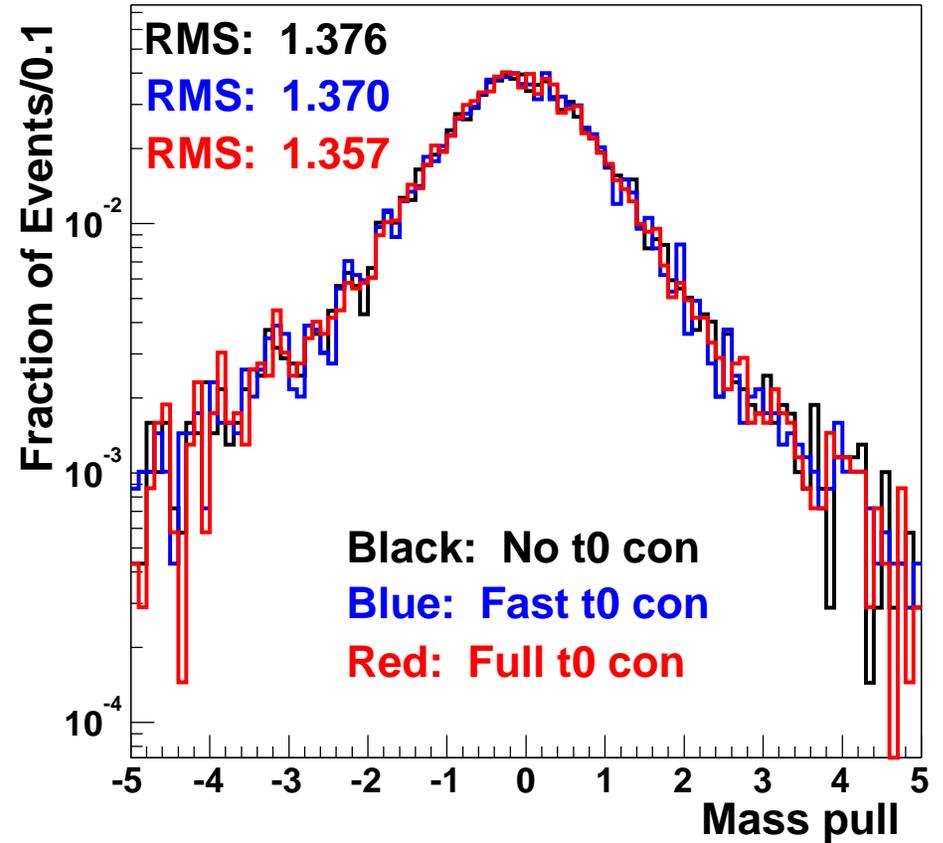
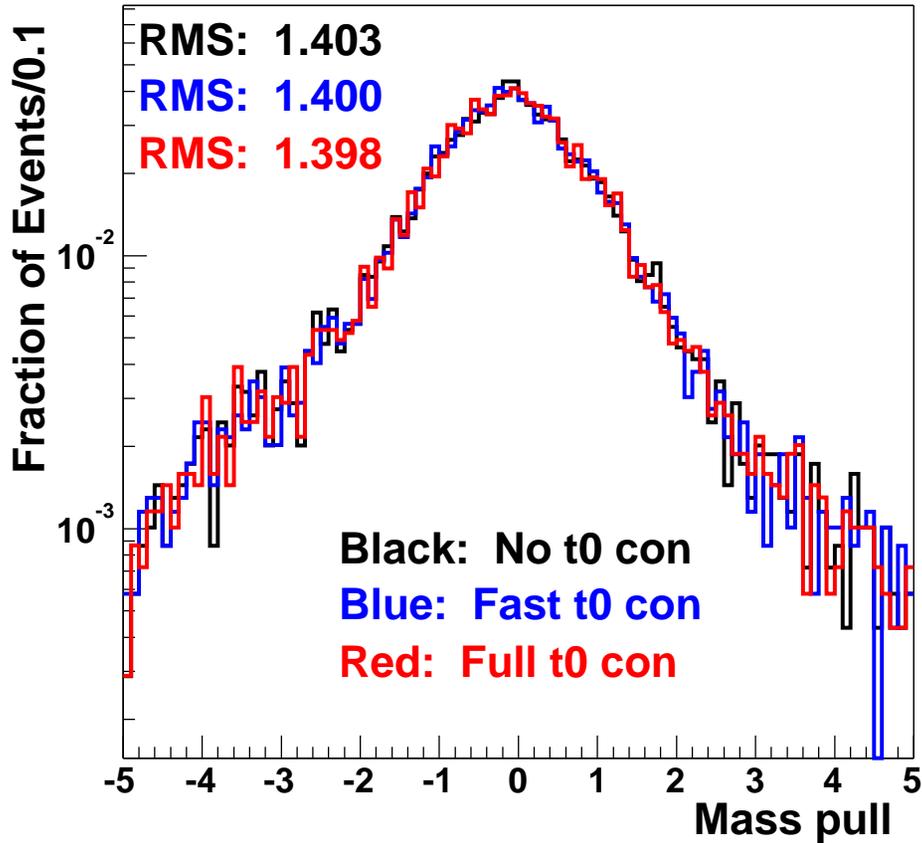
Test mass pull for COT non-BC tracks from Z 's



t_0 constraint a definite improvement, slightly better with “fast” algorithm

Performance: Mass

Test mass pull for COT BC (left) and default (right) tracks from Z 's



t_0 constraint a definite improvement, slightly better with “full” algorithm

Summary and Plan

Algorithms developed to constrain track t_0

- Significant improvement in χ^2 , hit resolution
- Definite improvement in parameter resolution
 - Alignment may be limiting gains

Sub-100 ps vertex t_0 resolution could have other applications

Need to choose algorithm

- “Fast” algorithm advantages:
 - Internal to COT code
 - More tracks constrained
 - Faster algorithm ($\sim 20\%$ slower than nominal, but CPU reduction possible)
- “Full” algorithm advantages:
 - More robust vertices
 - More information available: potential for further improvements

Given similar performance, preference to internal COT code

Plan: Commit code as soon as 5.3.1 is out, test further using integration release