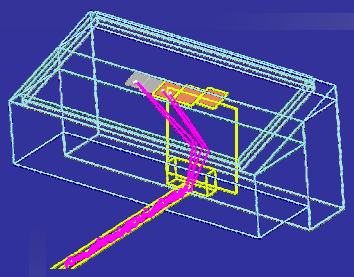


Progress on Pad Angles, Times, and Resolutions

Since last meeting:

- created position-dependent thetaC and kBar angle functions for all slots/pads from fixed lambda G4 – GetPadAngles.C
- created position-dependent function to retrieve true times in bar/block/oil/window from variable lambda G4 – GetPadTimes.C
- created position-dependent epsilon look-up functions for all slots/pads – GetEpsilons.C
- defined new position-dependent slot epsilons (and, yes, a function)
- ran full analysis for all runs and MC, compared thetaC resolutions (pixels, TOP, and kBarZ methods)
- looked at analysis using pixels close to expected ring
- ran G4 simulation with PMTs in all 7 slots

- I'll be back at SLAC in the afternoon of June 20th.

New functions for fixed lambda assignments for pad angles and paths:

```
// define mean angles and paths per pad from Geant 4 simulation (using entire pad)

  Float_t padCherTheta[7][64];  // type 0
  Float_t padkBarX[7][64];      // type 1
  Float_t padkBarY[7][64];      // type 2
  Float_t pathBlock[7][64];     // type 3
  Float_t pathOil[7][64];       // type 4
  Float_t pathWindow[7][64];    // type 5

[...]

 for(Int_t iSlot=0;iSlot<7;iSlot++)
    {
      for(Int_t iPad=0;iPad<64;iPad++)
        {
          padCherTheta[iSlot][iPad]=GetPadAnglesByPosition(Position,iSlot,iPad,0);
          padkBarX[iSlot][iPad]=GetPadAnglesByPosition(Position,iSlot,iPad,1);
          padkBarY[iSlot][iPad]=GetPadAnglesByPosition(Position,iSlot,iPad,2);
          pathBlock[iSlot][iPad]=GetPadAnglesByPosition(Position,iSlot,iPad,3);
          pathOil[iSlot][iPad]=GetPadAnglesByPosition(Position,iSlot,iPad,4);
          pathWindow[iSlot][iPad]=GetPadAnglesByPosition(Position,iSlot,iPad,5);
        }
    }
```

(alternative: GetPadAnglesByMethod to access angles from Jerry, Ivan, Jose, Joe, variable lambda)

All recent code, ntuples, and functions reside in /u3/jochen/latest on kzero
Makefile simplifies compilation and linking of functions and the main code (testbeam_new_fill.C)

New functions for variable lambda assignments for pad times:

```
// get true propagation times from variable lambda method - GetPadTimes
// used in epsilon determination

Float_t timeBar0Lambda[7][64];    // type 0
Float_t timeBar1Lambda[7][64];    // type 1
Float_t timeBlock0Lambda[7][64];  // type 2
Float_t timeBlock1Lambda[7][64];  // type 3
Float_t timeOil0Lambda[7][64];    // type 4
Float_t timeOil1Lambda[7][64];    // type 5
Float_t timeWindow0Lambda[7][64]; // type 6
Float_t timeWindow1Lambda[7][64]; // type 7

for(Int_t iSlot=0;iSlot<7;iSlot++)
  {
    for(Int_t iPad=0;iPad<64;iPad++)
      {
        timeBar0Lambda[iSlot][iPad]=GetPadTimes(Position,iSlot,iPad,0);
        timeBar1Lambda[iSlot][iPad]=GetPadTimes(Position,iSlot,iPad,1);
        timeBlock0Lambda[iSlot][iPad]=GetPadTimes(Position,iSlot,iPad,2);
        timeBlock1Lambda[iSlot][iPad]=GetPadTimes(Position,iSlot,iPad,3);
        timeOil0Lambda[iSlot][iPad]=GetPadTimes(Position,iSlot,iPad,4);
        timeOil1Lambda[iSlot][iPad]=GetPadTimes(Position,iSlot,iPad,5);
        timeWindow0Lambda[iSlot][iPad]=GetPadTimes(Position,iSlot,iPad,6);
        timeWindow1Lambda[iSlot][iPad]=GetPadTimes(Position,iSlot,iPad,7);
      }
  }
```

New functions for epsilon assignments for pads/slots:

```
Float_t offset_phillips0[7][65];
Float_t offset_phillips1[7][65];
Float_t offset_slot_phillips0[7];
Float_t offset_slot_phillips1[7];

// define the epsilons 10: pad/slot epsilons/epsilon'   20: new slot epsilons

#if __EPSILONS==10
  for(Int_t iSlot=0;iSlot<7;iSlot++)
    {
      offset_slot_phillips0[iSlot]=GetSlotEpsilonsByPosition(Position,iSlot,0);
      offset_slot_phillips1[iSlot]=GetSlotEpsilonsByPosition(Position,iSlot,1);

      for(Int_t iPad=0;iPad<64;iPad++)
        {
          offset_phillips0[iSlot][iPad]=GetEpsilonsByPosition(Position,iSlot,iPad,0);
          offset_phillips1[iSlot][iPad]=GetEpsilonsByPosition(Position,iSlot,iPad,1);
        }
    }
#elif __EPSILONS==20
  for(Int_t iSlot=0;iSlot<7;iSlot++)
    {
      offset_slot_phillips0[iSlot]=GetPureSlotEpsilonsByPosition(Position,iSlot,0);
      offset_slot_phillips1[iSlot]=GetPureSlotEpsilonsByPosition(Position,iSlot,1);
    }
#endif
```

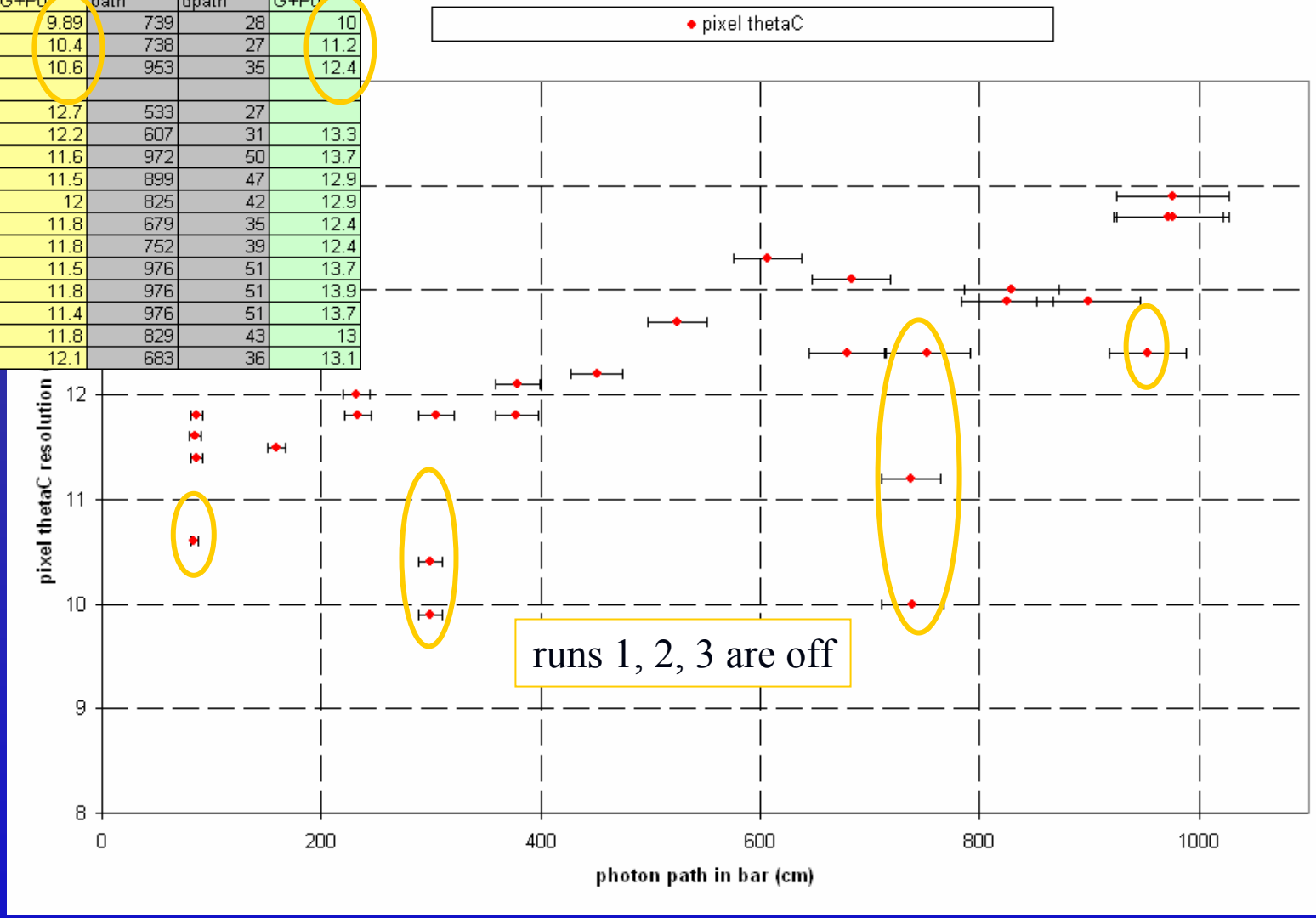
Position-dependent pad angles – example slot 4, pad 27, peak 1

	(thetaC in mrad)	(epsilon in ns)	
position 1:	836.75	0.003	
position 2:	837.88	-0.157	
position 3:	837.15	-0.072	
position 4:	837.93	-0.175	
position 5:	838.23	-0.151	
position 6:	836.89	-3.814	(difficult to fit because the peaks are almost overlapping)
position 7:	837.38	-0.264	(didn't attempt to separate peaks)

→ sizable variation of angles and times as a function of position

Use these angles to calculate thetaC resolution from pixels and from TOP.

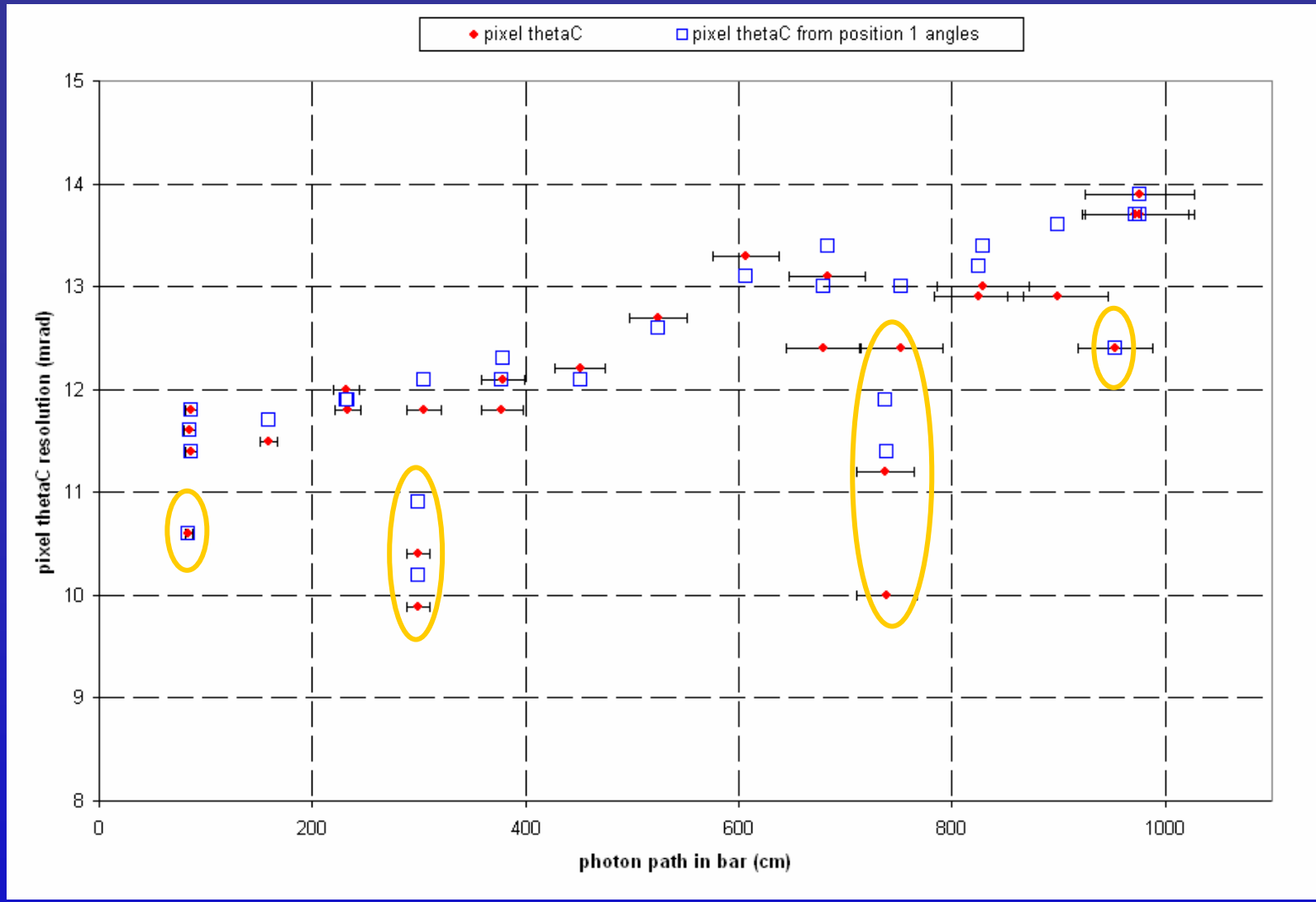
	direct		direct	reflected		reflected
run	path	dpath	G+P0	path	dpath	G+P0
1	299	11	9.89	739	28	10
2	299	11	10.4	738	27	11.2
3	84	3	10.6	953	35	12.4
4						
5	524	27	12.7	533	27	
6	451	24	12.2	607	31	13.3
7	85	5	11.6	972	50	13.7
8	159	8	11.5	899	47	12.9
9	232	12	12	825	42	12.9
10	378	19	11.8	679	35	12.4
11	305	16	11.8	752	39	12.4
12	86	5	11.5	976	51	13.7
12a	86	5	11.8	976	51	13.9
12b	86	5	11.4	976	51	13.7
13	233	12	11.8	829	43	13
14	379	20	12.1	683	36	13.1



Pixel thetaC vs. path *(plotted without errors – errors are probably ~0.5-1mrad)*

pixel resolutions from runs 1-3 are not consistent with runs 4-14

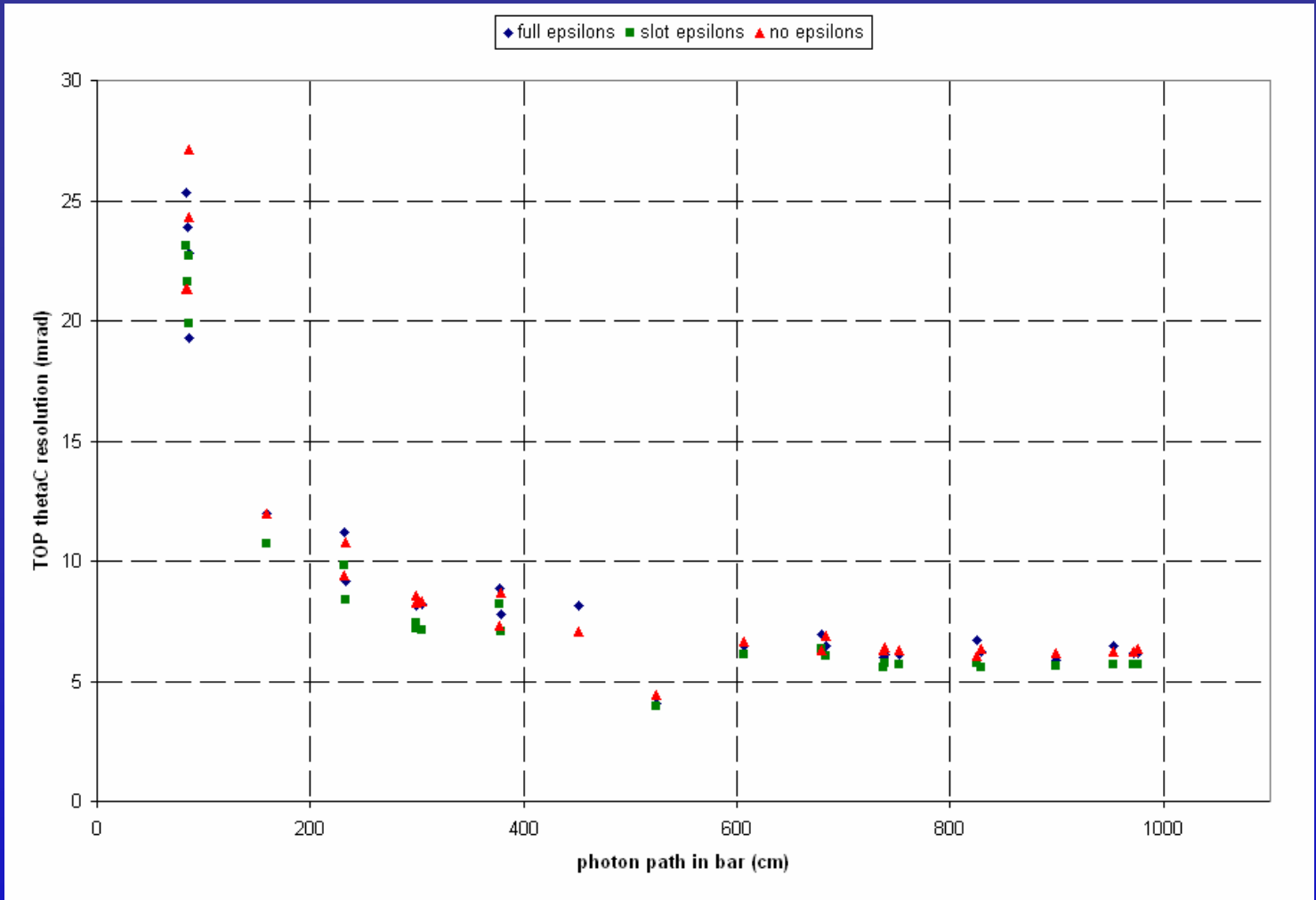
(runs 1-3 had fewer slots and pads instrumented – good res. maybe artifact of lack of coverage?)



Pixel thetaC vs. path *(plotted without errors – errors are probably ~0.5-1mrad)*

small resolution improvement from proper posn-dept angles (0.2-0.5mrad).

does not help with differentiating slope vs. step models (preference for slope)

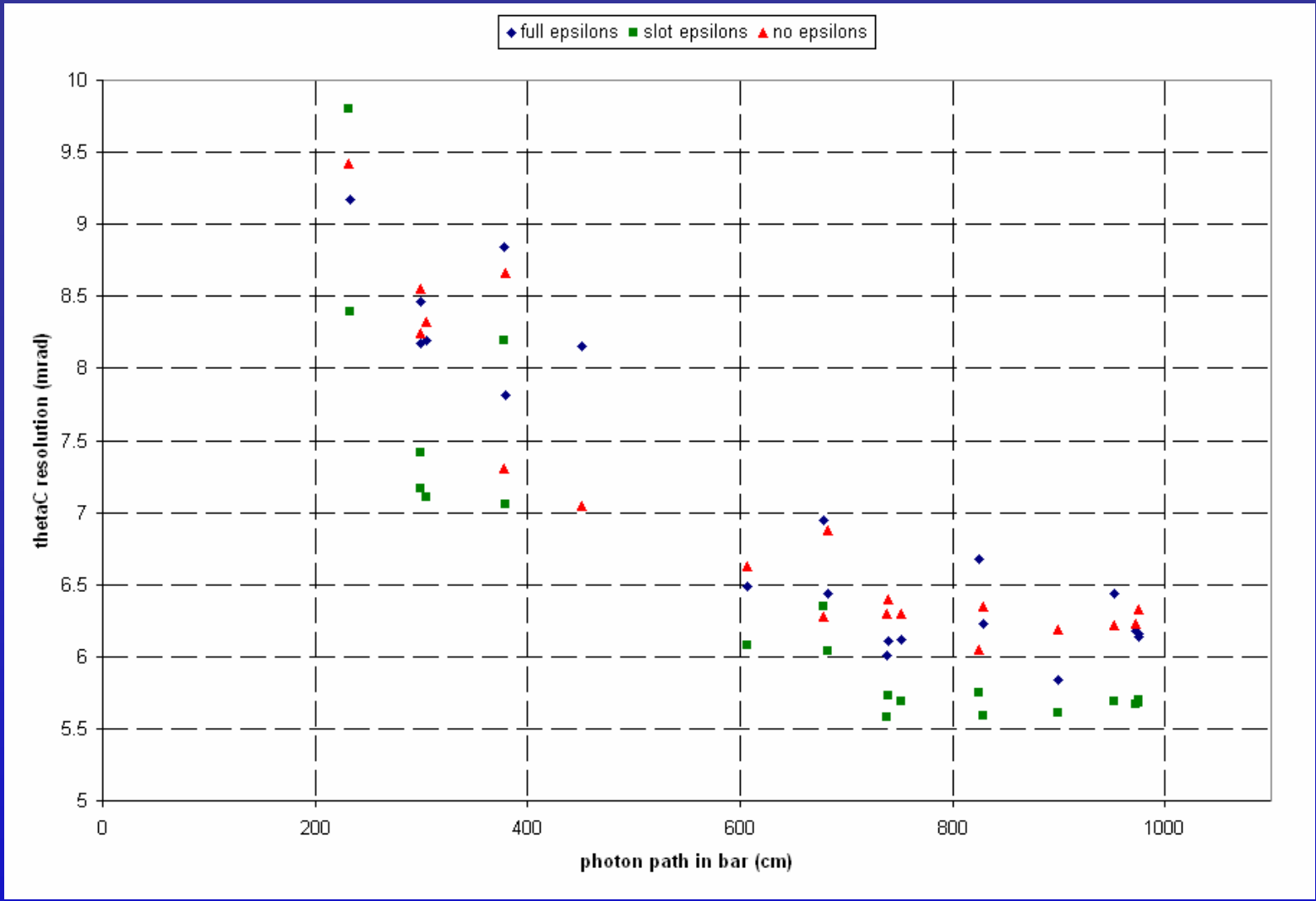


thetaC from TOP method vs. path *(assumes beta==1)*

plotting result of single Gaussian fit to peak (810-830mrad)

best results for "slot epsilons" = single epsilon per slot per peak per position (70 numbers)

position dependent angles improve resolution slightly (~0.1mrad level)



thetaC from TOP method (assumes beta==1)

zoom in thetaC resolution shows that slot epsilons are typically ~05mrad better than full epsilons (epsilons for each pad plus epsilon' per slot ~1000 numbers)

Influence of position-dependent slot epsilons on thetaC from TOP not clear

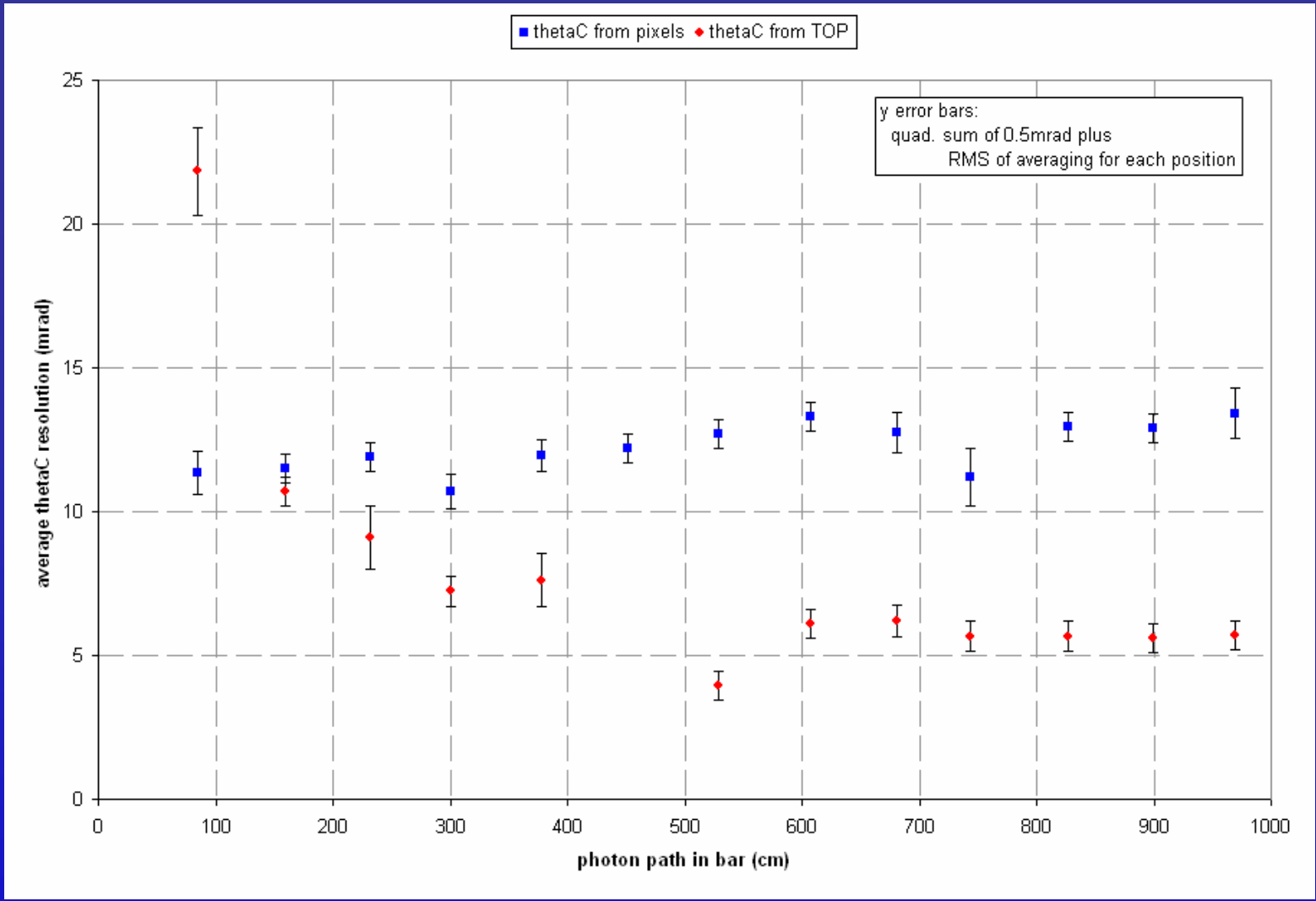
most runs significantly better with run-dept epsilon, some worse (run 10, double-checked)
not sure what that tells us – statistics of epsilon determination?

(checked full epsilons for anomalous run 10: resolution for pos 1 better than for pos 5 epsilons. Weird.)

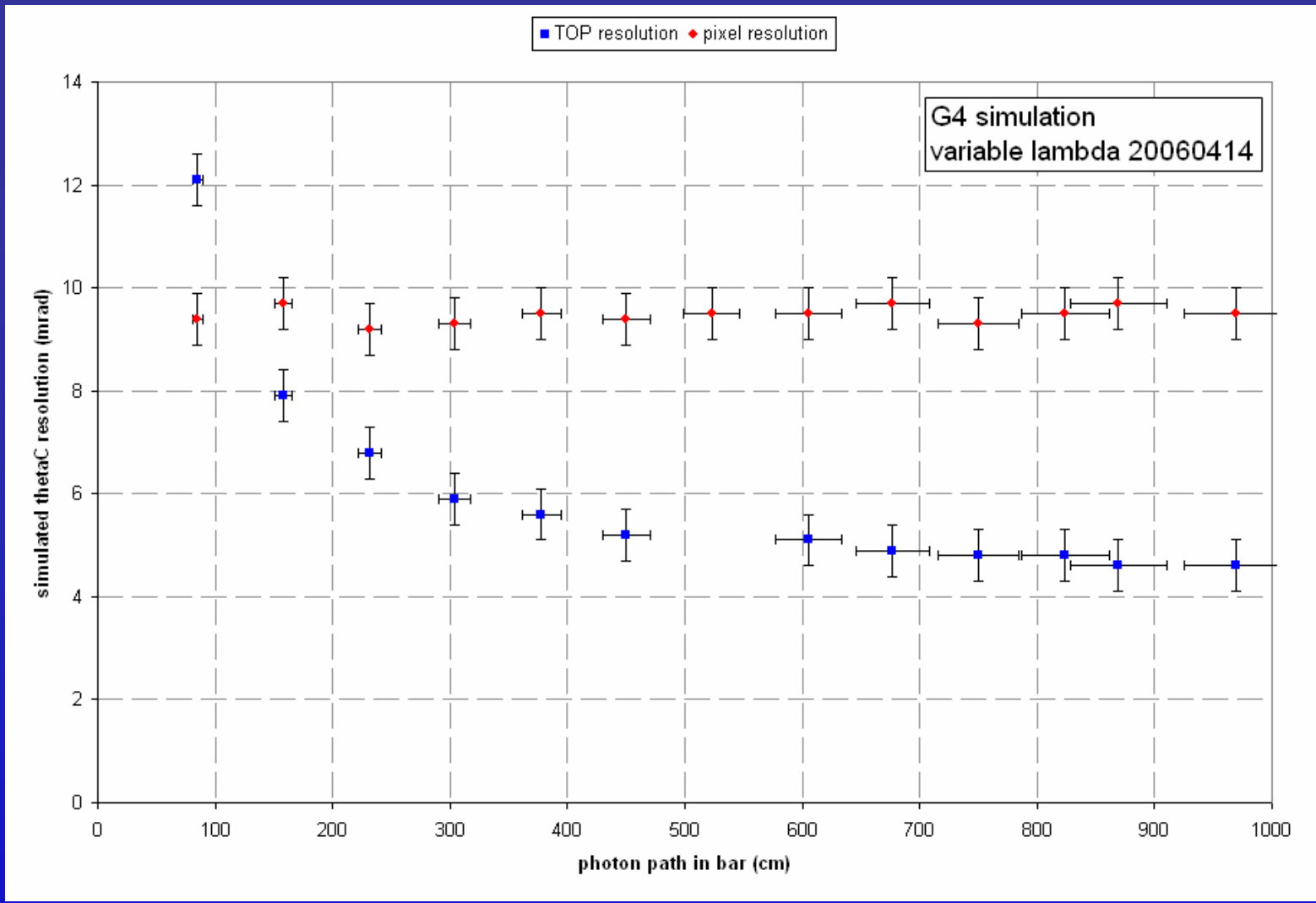
have not tried this for all run-dept pad/slot epsilon/epsilon' (big effort, low stat for some positions)

	pos-dept	pos1	
run	G+~~D0~~	G+~~D0~~	difference
1	5.73	6.07	-0.34
2	5.58	5.82	-0.24
3	5.69	5.69	0
4			
5			
6	6.08	6.63	-0.55
7	5.67	5.67	0
8	5.61	5.67	-0.06
9	5.75	5.54	0.21
10	6.35	5.69	0.66
11	5.69	5.78	-0.09
12a	5.68	5.68	0
12b	5.7	5.7	0
13	5.59	5.52	0.07
14	6.04	5.8	0.24

thetaC from TOP (resolution in mrad)

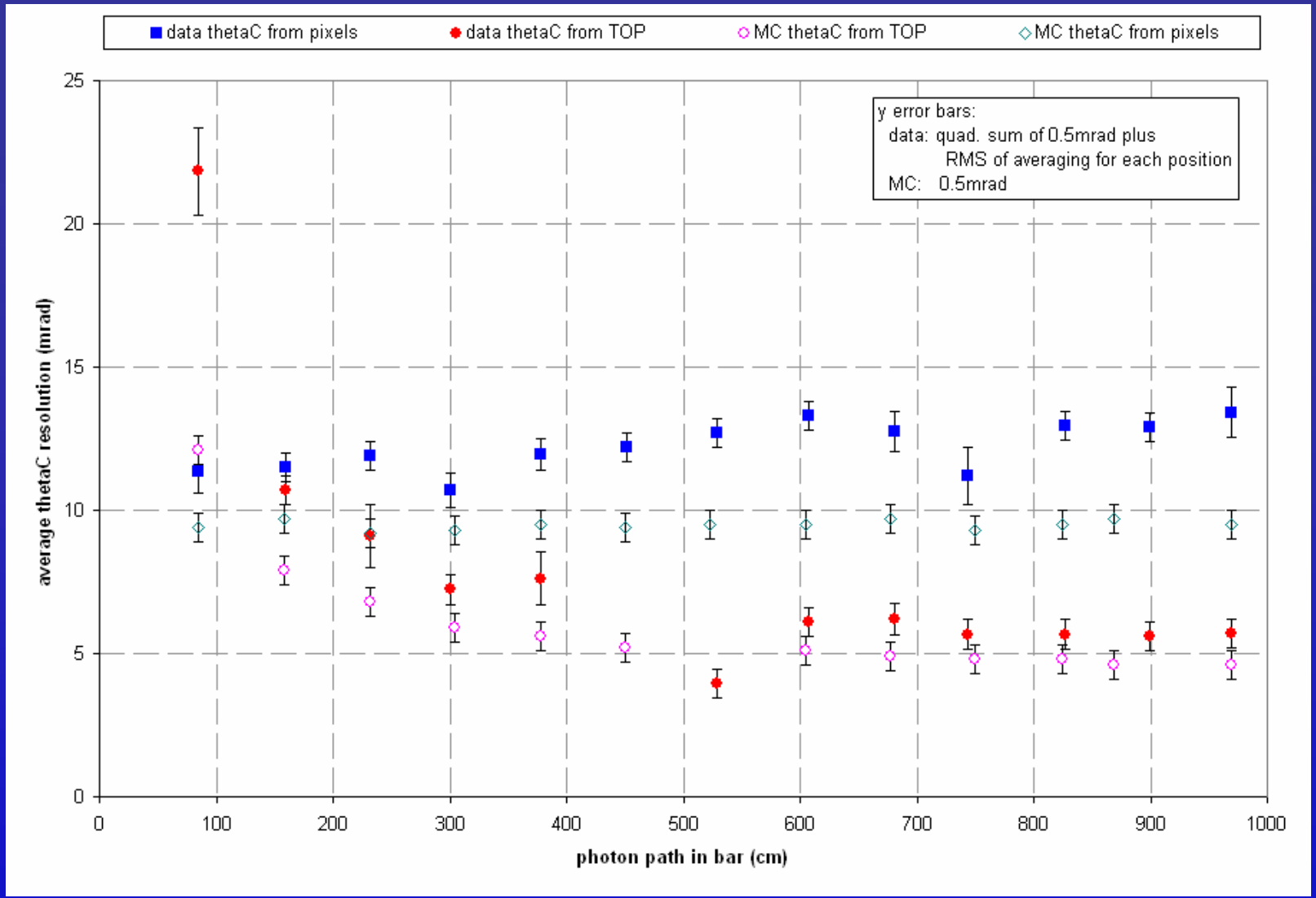


thetaC from TOP and methods averaged for each position (keeping all values)



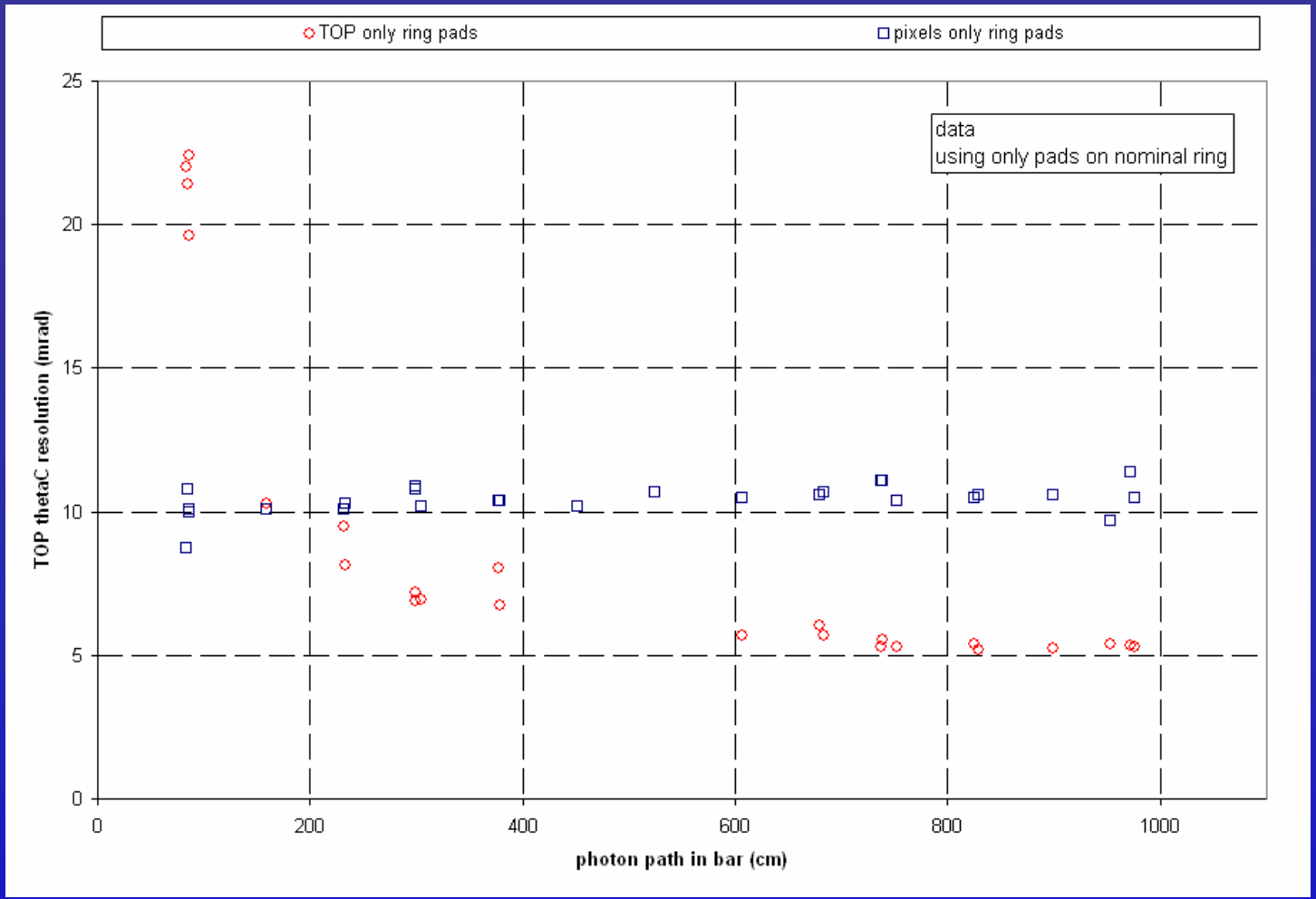
Analysis of Geant4 root files

variable lambda, default settings from April (Ivan's settings)



Comparison of data (position averaged) and G4 simulation

draw your own conclusions…

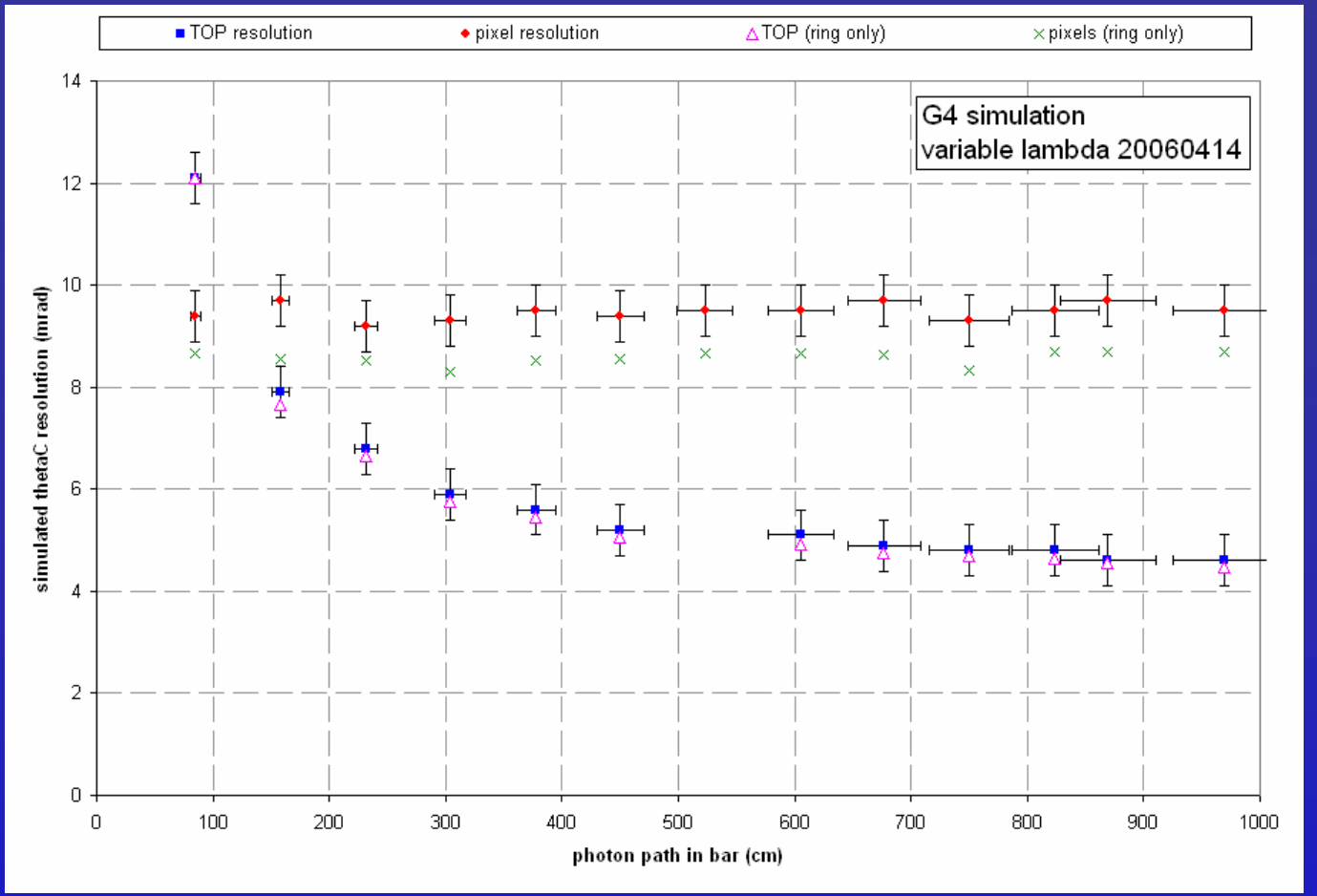


thetaC resolution if I use only pads along expected ring image

almost no grow of pixel thetaC resolution as function of path and better resolution

lower plateau resolution of thetaC from TOP (by ~0.3mrad)

resolution is clearly influenced by off-ring pads (1-3mrad)



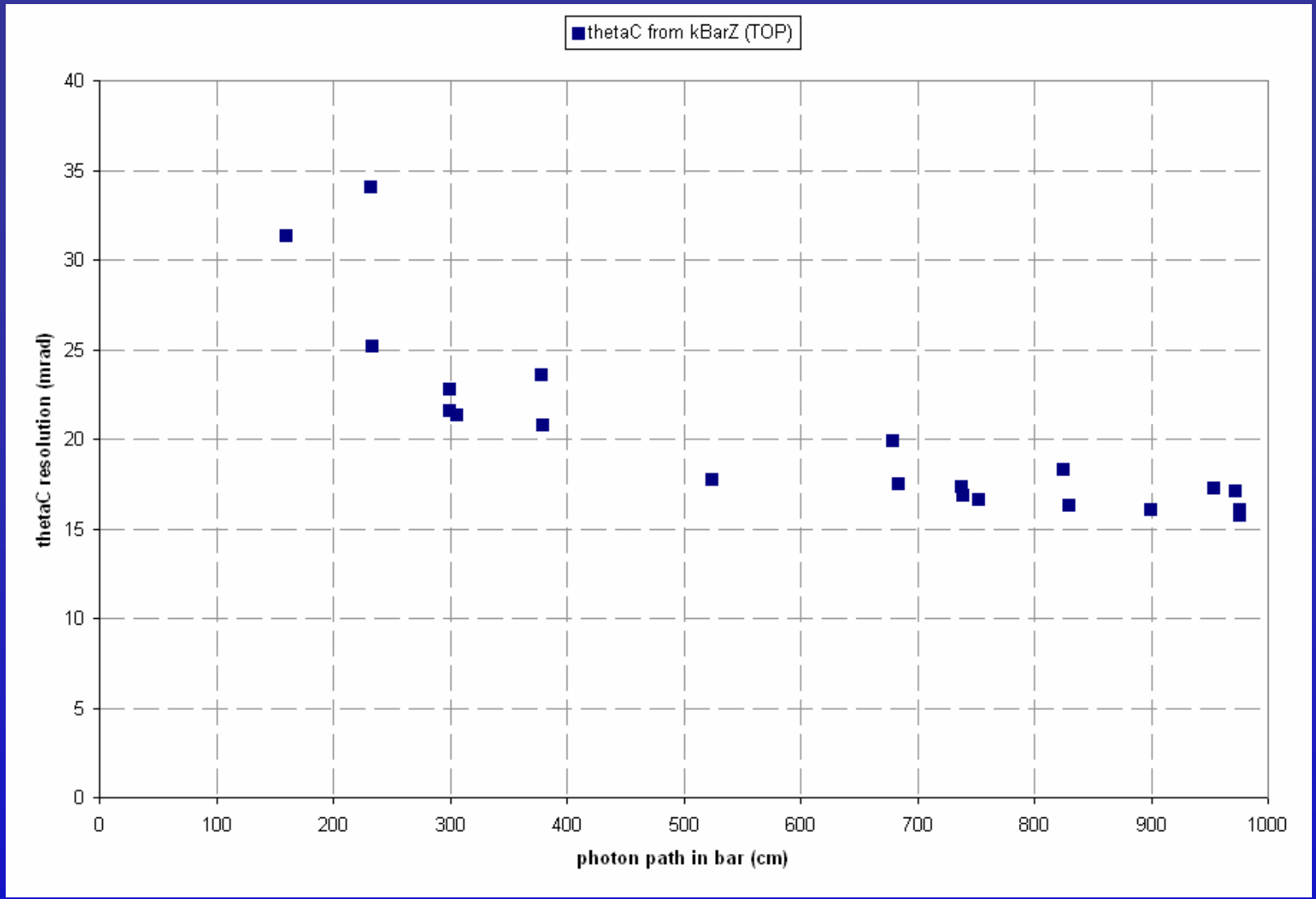
Geant4 thetaC resolution if I use only pads along expected ring image

shows that part (1mrad) of the resolution improvement

for the pixel thetaC is trivial (phase space?)

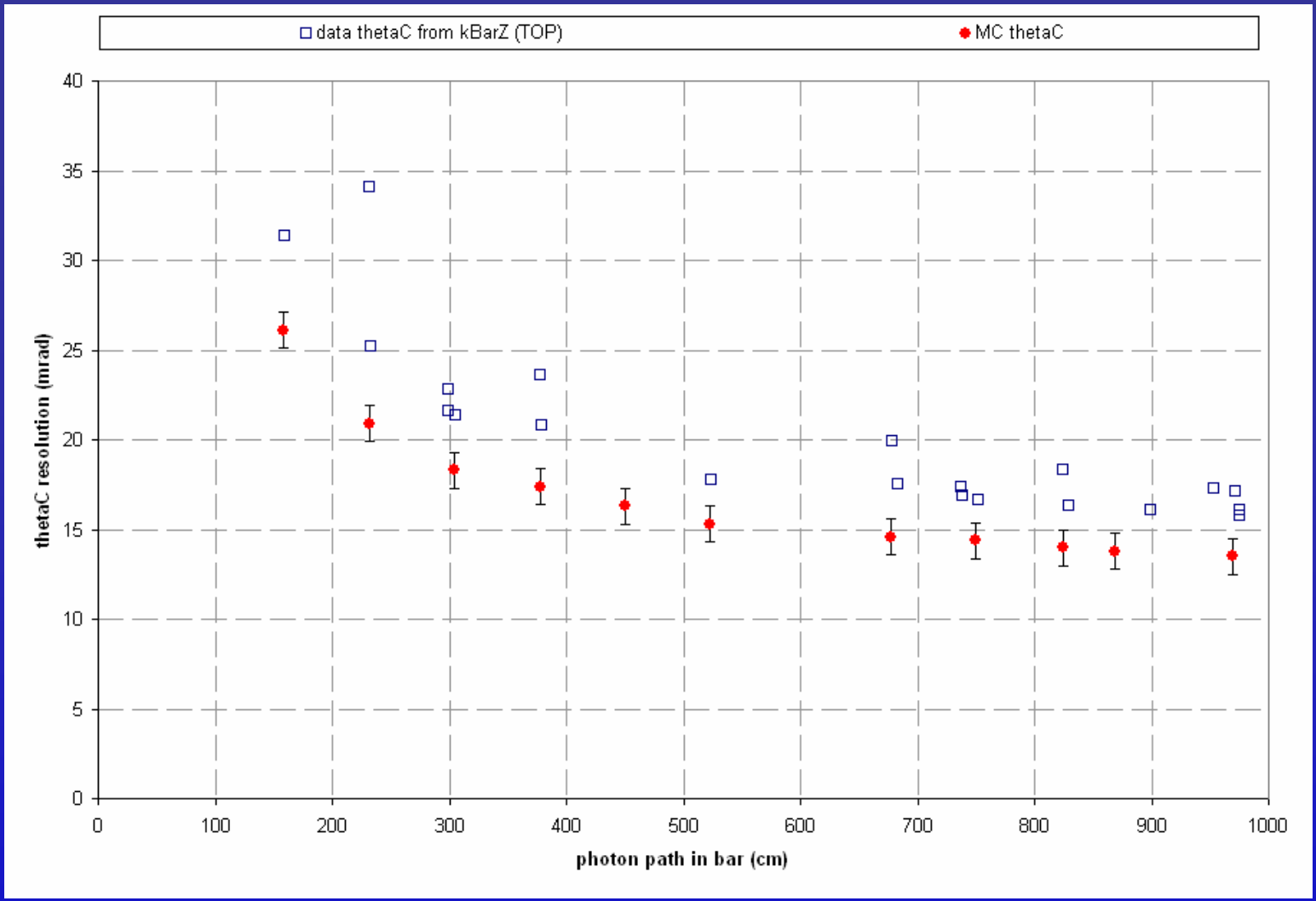
shows that MC does not see improvement for TOP thetaC

sort of interesting

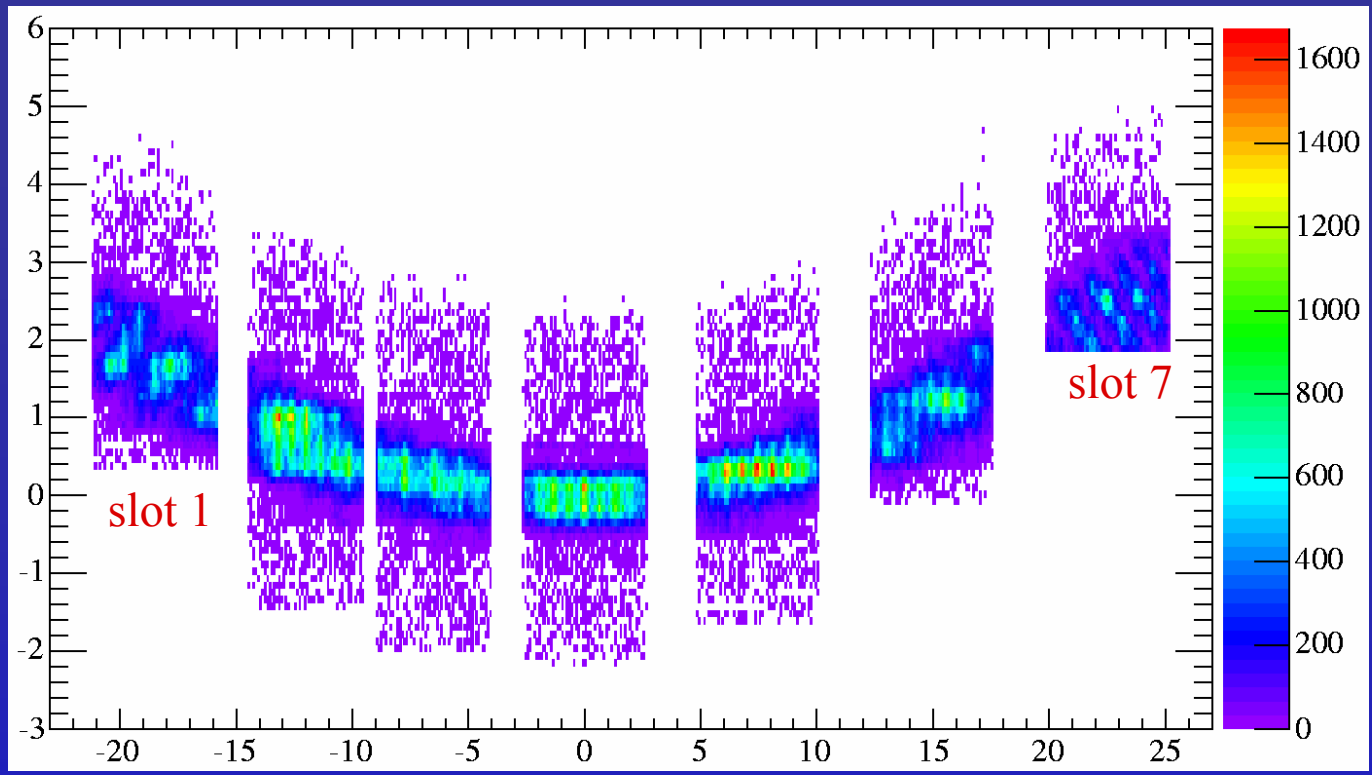


thetaC from kBar method (kBarZ from TOP assuming average v_g, kBarX from pixels, errors 1-2mrad)

bad resolution, drops as function of path



thetaC from kBar method for data and Geant4 (placed 1mrad error on MC points for comparison)
simulation predicts similar shape, ~3mrad better resolution than data



Geant4 with 7 PMTs in 7 slots

assumed: 64-ch Burle MCP in slots 1, 4, 5, 6, 7; 64-ch Hama MaPMT in slot 2;
256-ch new Hama MaPMT in slot 3

100k triggers in position 1

1.31M hits in slots 2-6; 221k hits in slot 1; 135k hits in slot 7 (partially cut off)

would gain 17% hits by instrumenting 32 pads of slot 1 (lower half)