ILC – Main Linac Simulation February 2006 Report.

Paul Lebrun

ILC-LHC R&D in AMR

Outline

- Lucretia/Merlin/CHEF code evaluation and studies.
- Benchmarking Lucretia/Merlin on misaligned 23.4 MeV/m old Tesla lattice.
- Status report on steering algorithm implementation in Merlin
- Outlook.

Computer Code Studies, Goals

- Understand and evaluate the computing methodologies
- If appropriate, select an existing package to pursue more complicate, exhaustive (“start to end”) simulation where, ultimately, all relevant effects, control imperfections and failures are implemented.
- Necessary step for benchmarking and understanding code differences.
- Perhaps, moving towards a better tool-kit (long term goal).

Code Studies: Status

- 4 codes (Lucretia, Merlin, CHEF, Placet/Tao
- Lucretia (version from Sept 05): Operational on ilcsim.fnal.gov (just for me!, needs more MatLab license if others want to use it). Latest activities:
  - Implemented very simple interface to read the "nick23p4_misxy_1.txt" misalignment file received from Jeff Smith.
  - More checks on how emittances are calculated.

Code Studies: Status for Merlin

- Implemented same lattice and misalignment files.
- Mostly worked on the re-implementation of 1-to- 1, DFS and “brute force emittance minimization steering algorithms. (first implementation of DFS were never released as part of the standard version.)
- Minor effort on cavity wakefields, various plottings (Root-based).

Code Studies: Status for CHEF

- Received from Leo Michelotti 4 short examples codes based on the same lattice.
- Works one a lattice cell, verified for proton only.
- Awaiting for verification of the electron/positron code upgrade
- Plan to implement vertical emittance growth simulation to benchmark same problem as for Lucretia and Merlin.
- Although the current capabilities of CHEF with respect to other e+e- colliders are somewhat lacking, CHEF is interfaced with Synergia, which supports collective effects (space charge, electron clouds,..) which are absent in other codes.

Comments:

- Towards a Toolkit
  - As opposed to a fixed rigid executable (e.g., MAD, for instance) + private scripting codes to simulate controls algorithms and analysis tricks.
  - Allow better integration of advanced steering algorithms, for instance.
  - And implement more physics processes, well integrated with the core tracking code.
  - *But, it require the user to "code a bit", which can discourage many physicists..*

Utilities: graphics + math

- MATLAB offers a very powerful collection of mathematical function, matrix manipulations, and publication quality graphics. Extensively used in LIAR and Lucretia, and, to a lesser extend, in LLRF controls and simulation.
- Merlin and CHEF completely separate the simulation bit from these analysis tools. Merlin spits returns data in internal memory, not as graph or ASCII files.
- Octave and Root have been used by Synergia people to do this work. For my Merlin work, used almost exclusively Root.
- Mathematical utilities: MATLAB very well documented with respect to GNU, BLASS, etc.. “free-ware libraries” . But do we need extensive libraries?
- Running MATLAB on computer farms for more extensive simulation might become expensive and difficult to manage.

Package Architectures

- Lucretia -and to some extend LIAR(?)- architecture takes advantage from MATLAB high level constructs, which are mapped to the core tracking, written in C. Note that the replacement of MATLAB with Octave is not immediate, because, although the “m” code can be kept as is, the C-API has to be replaced by a corresponding C++, which matches the internal of Octave. Crude estimate based on a cursory look of the complexity and # of line of C code: ~ 3 man-weeks, at least. But do-able!..
- Merlin v3.0: Stand-alone package! Necessary math has been coded or re-coded from scratch. Just use the Unix Standard C math library!..
  - Of course, users are free to use other C++ packages.. Did so, in the context of steering..
- CHEF uses BOOST, Qt to build it's GUI,.. and many others.. A bit more complex to re-build, but a nice “rpm” have been build by the Synergia team=> easy to install/re-install
- All these packages are assembled around the concept of “Physics Modules”, e.g., “Beam Model”, “Lattices” or “Accelerator Model”, “Tracking”, “TuningAndAnalysis”. But details implementation can be quite different. Adding the fact that programming languages are different make it nearly impossible to exchange capabilities without re-coding... ==> I wil have to re-write code moving the “Steering” code written in the context of Merlin, to Matlab (completely!) and to CHEF (only interfaces to accelerator model, beam structures.).

Language Issues: “m”

- The “m” programming language can be considered a “dynamically typed language”: the type of a variable is never declared, and can eventually change in the course of execution. Great for prototyping, as it is also an interpreted language, in many instances (computing intensive Matlab function are pre-compiled). Trouble is, I am not a good typist, not a careful programmer: inherently more error prone: example:

```
S_yvsp = 0.;
for nn = 1 : 1 : numPartOut
      S_yvsp = S_yvsp + beam.Bunch(1).x(3) * beam.Bunch(1).x(6,nn);
end
```

compiles, runs and produces wrong answer!!, due to a missing index!..

The corresponding C++ code would not compile...

Language Issues: C++ ?

- But C++ is often too difficult to learn.. it seems...
- Personal view: For large scale, complex simulation problems
  - it is worth learning !
  - Well supported at Fermilab.
  - Now used consistently in HEP
  - ...

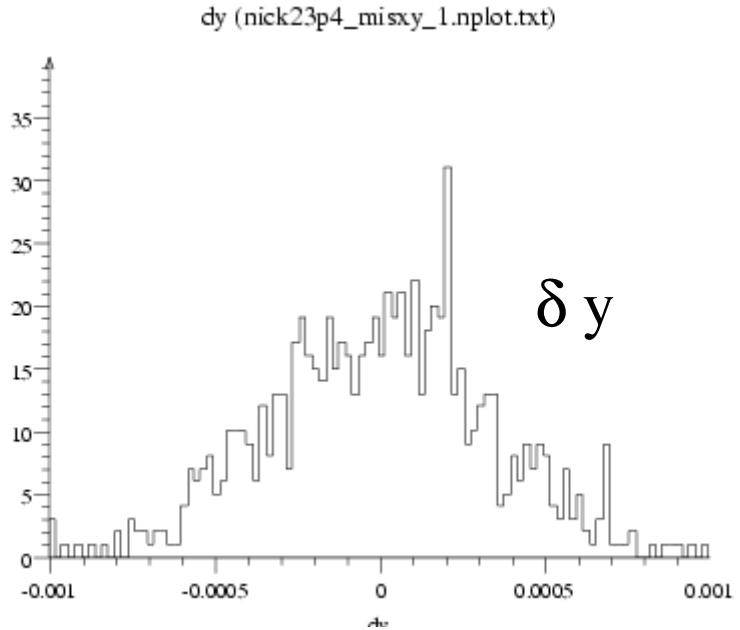
BenchMark, Lucretia/Merlin

- Description of the chosen problems.
  - final goal: Vertical Emittance preservation via adhoc steering in the Main Linac or Bunch compressor.
  - Need for an intermediate problem, not too trivial, and not overly complex such that differences can not be systematically tracked. ==> Must produce lots of intermediate results.
- Timing result.
- Physics results on the misaligned Tesla Main Linac in presence of transverse wake fields.

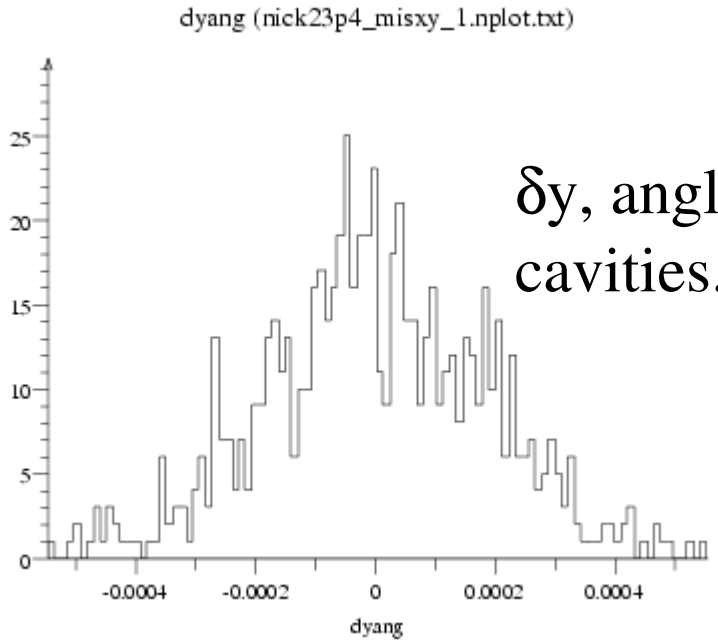
Benchmark Problem.

- Using the “23.4 Mv/m”, Tesla Main Linac design. The corresponding “xsif” file has been studied by Pt, Kirti, Nikolai in the context of LIAR, and by Jeff Smith in Bmad. Implemented this lattice in both Lucretia and Merlin.
- “Misaligned” the lattice, same displacement and angle in both codes (i.e., not random!) (see plot next slide)
- Same 6D Gaussian beam, matched to the lattice. But different macro-particles.. Beam Parameters:
  - Initial.Momentum = 5.0; Initial.SigPUncorrel = 0.140; Initial.sigz = 0.3e-3;
  - Initial.x.NEmit = 8.0e-6; Initial.y.NEmit = 2.0e-8;
  - Initial.x.Twiss.beta = 89.309; Initial.y.Twiss.beta = 50.681;
  - Initial.x.Twiss.alpha = -1.451; Initial.y.Twiss.alpha = 0.873;
  - Initial.Q = 2.0e10 * 1.6e-19;

Mis-alignment data



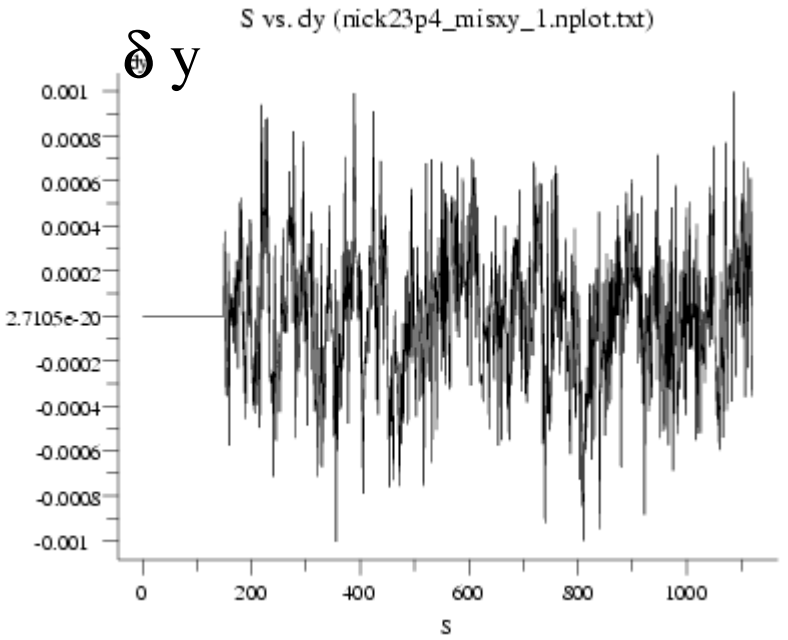
δ y



δy, angle, cavities.

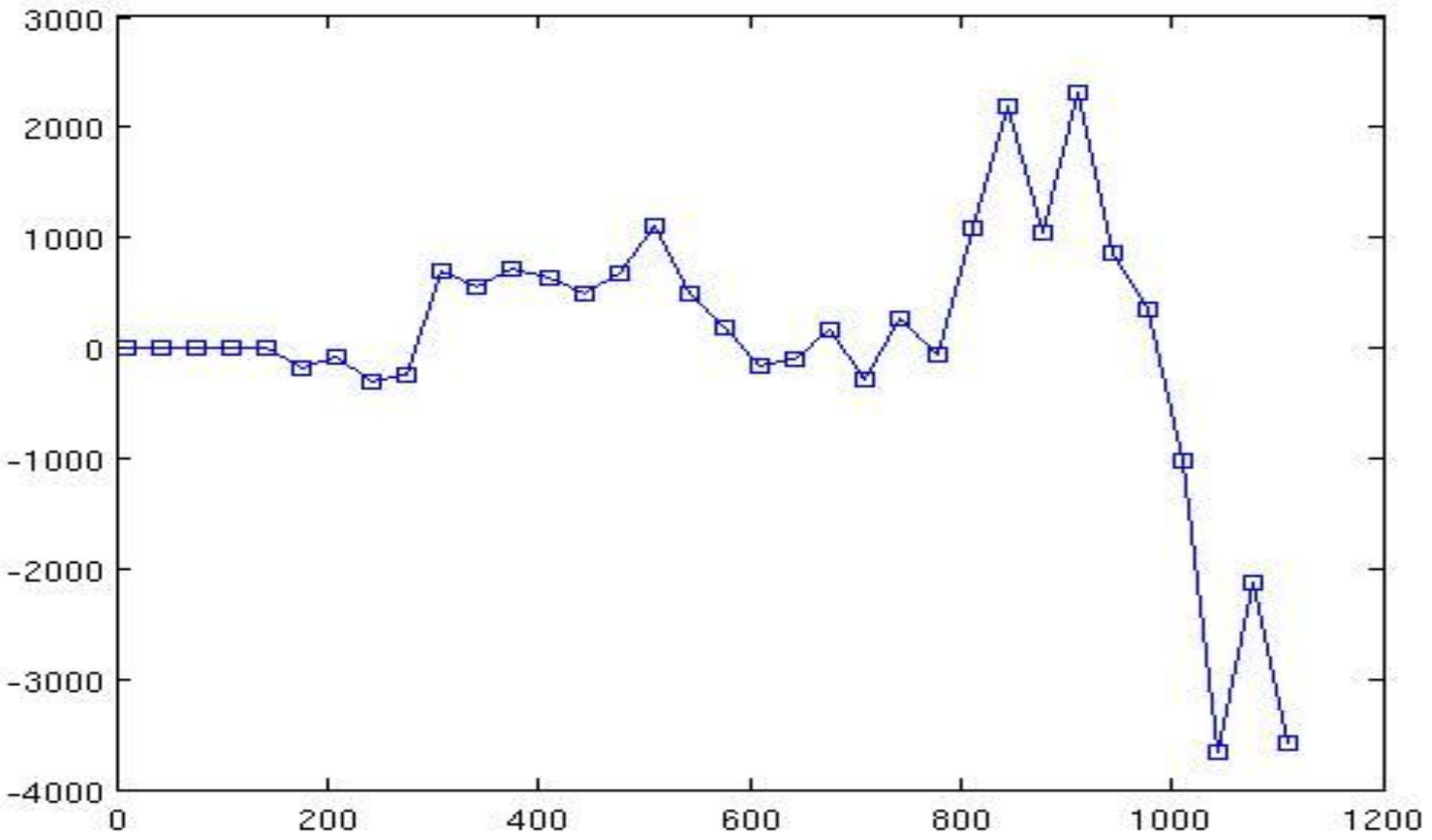
- Lattice is perfect until S = 151 m
- Only vertical displacement, sigma ~ 337 microns.
- No Quad tilts (rolls)

δ y



Resulting Orbit, S ~< 1.05 km, Lucretia

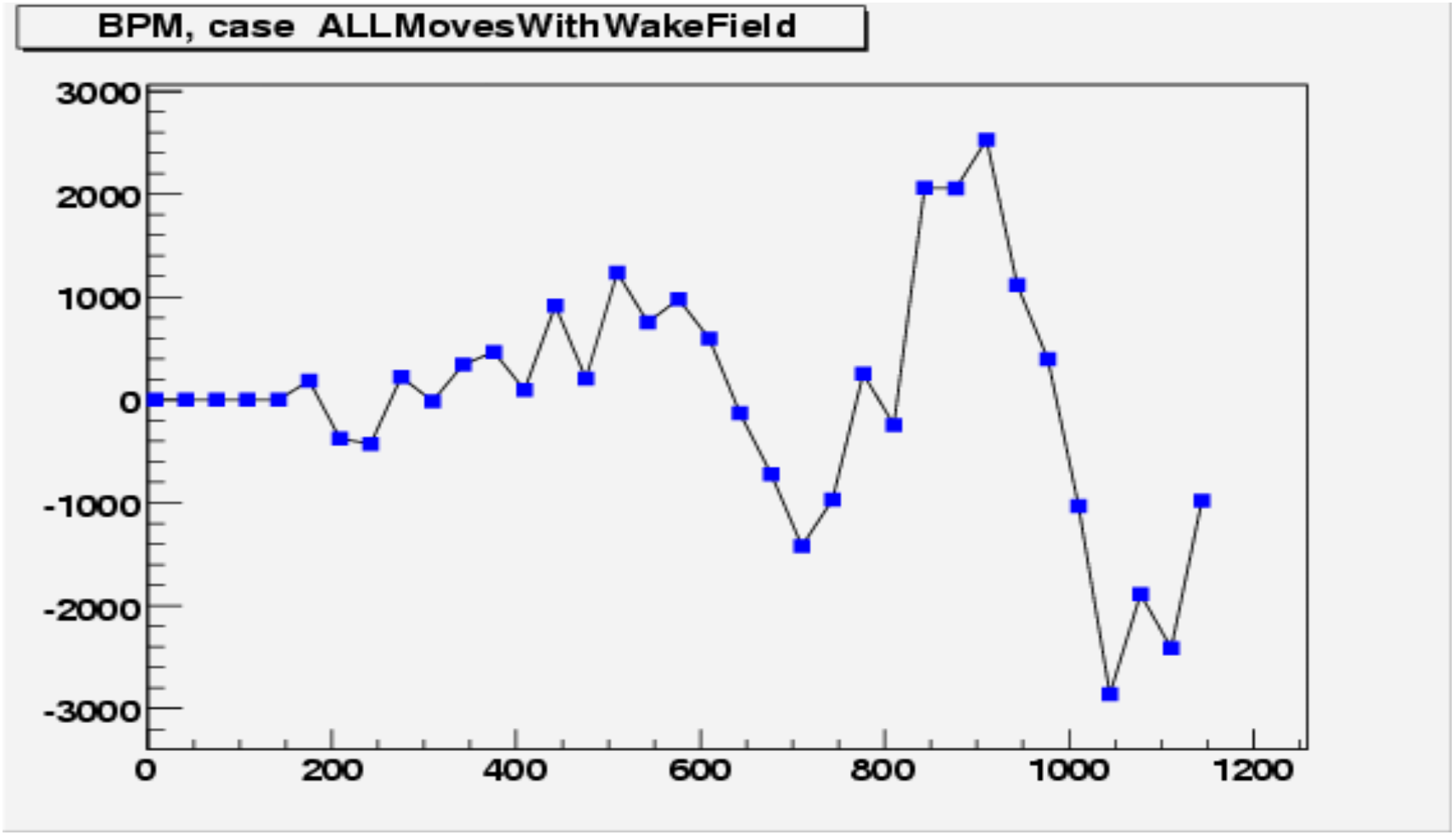
δ y, BPM Measurements (microns)



S (m.)

Resulting Orbit, S ~< 1.05 km, Merlin

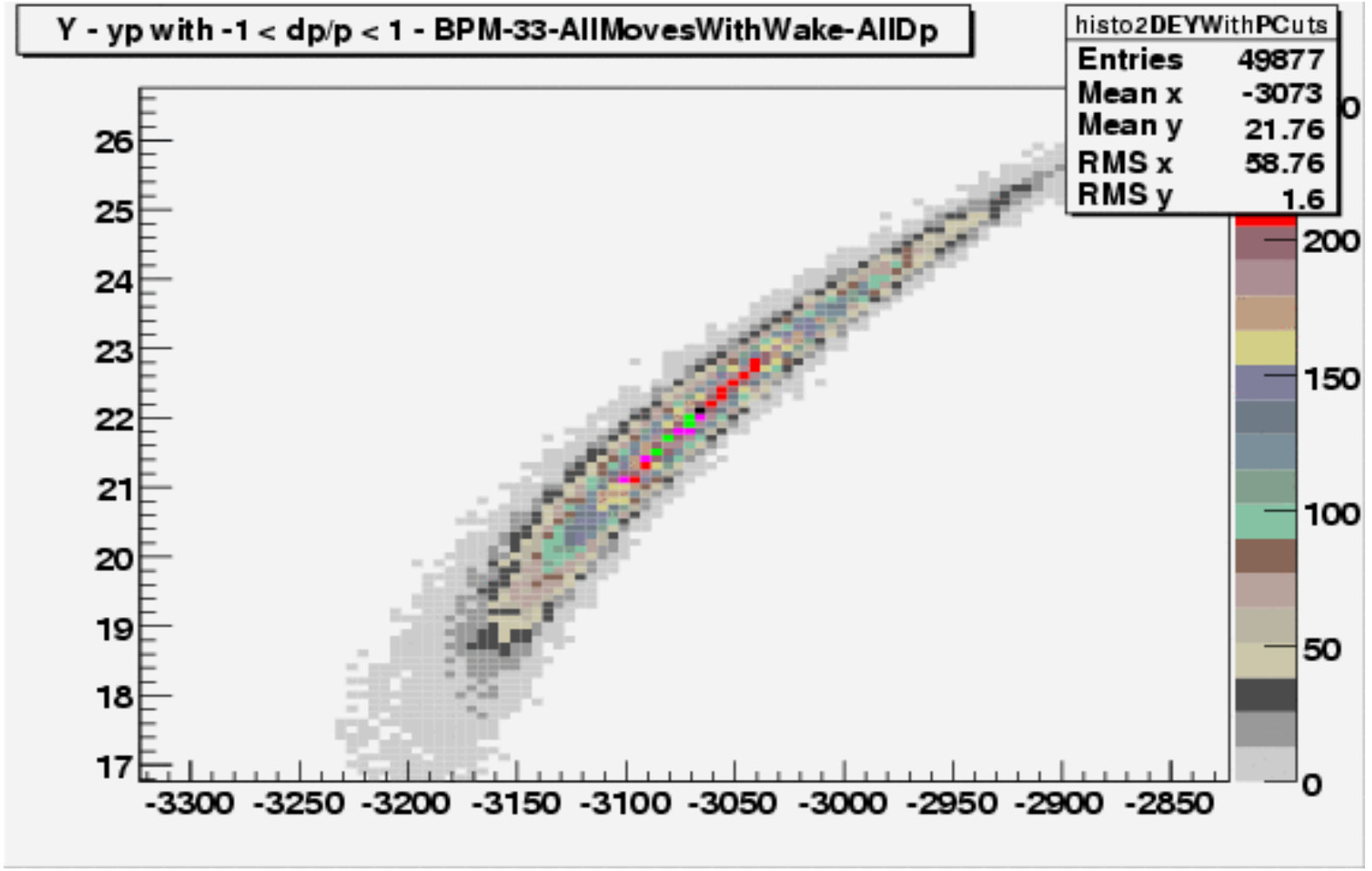
δ y, BPM Measurements (microns) .



S (m.)

Vertical Phase Space at S ~< 1.05 km, Merlin

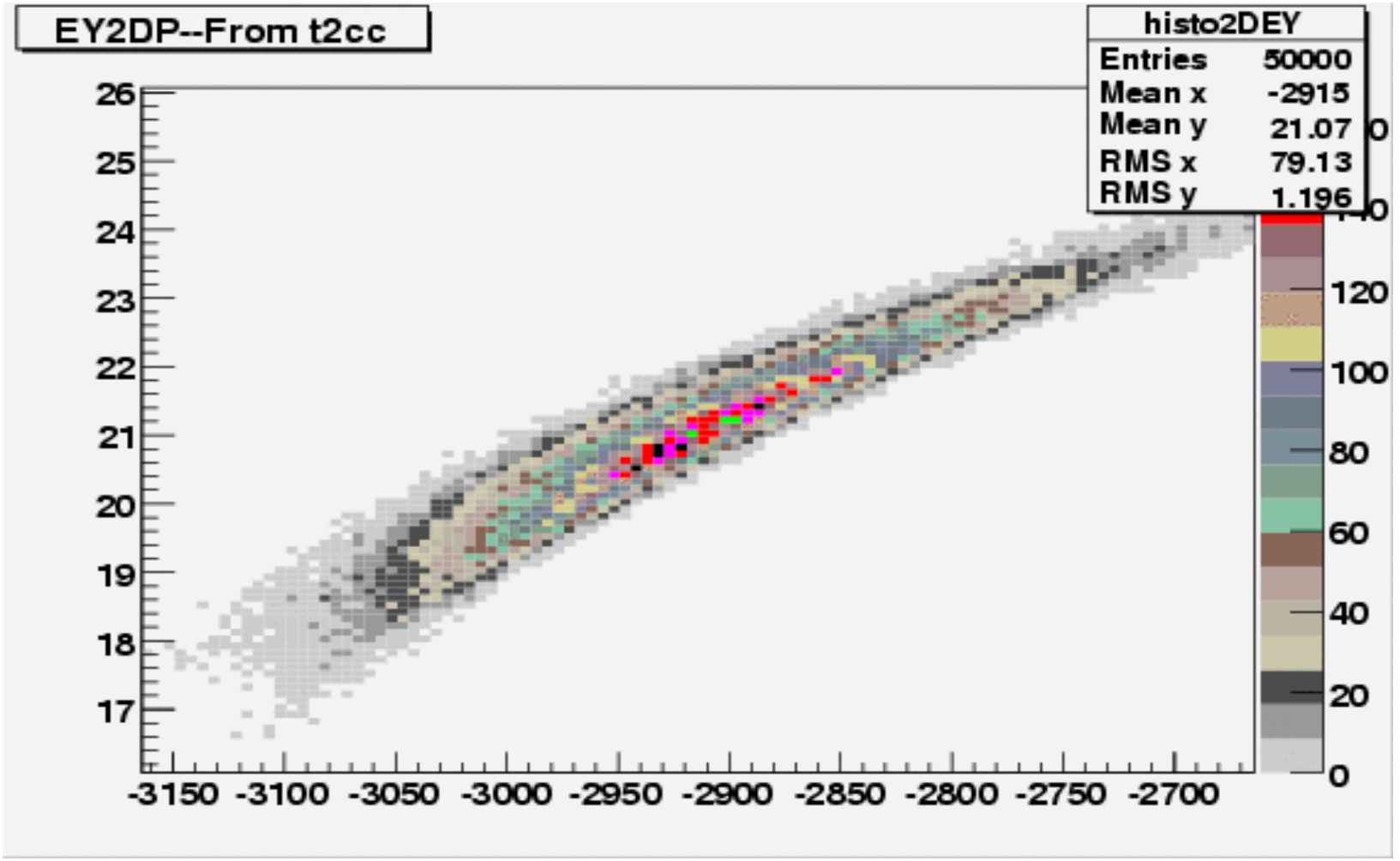
y'(micro-radiants)



y (microns)

Vert. Phase Space at S ~< 1.05 km, Lucretia

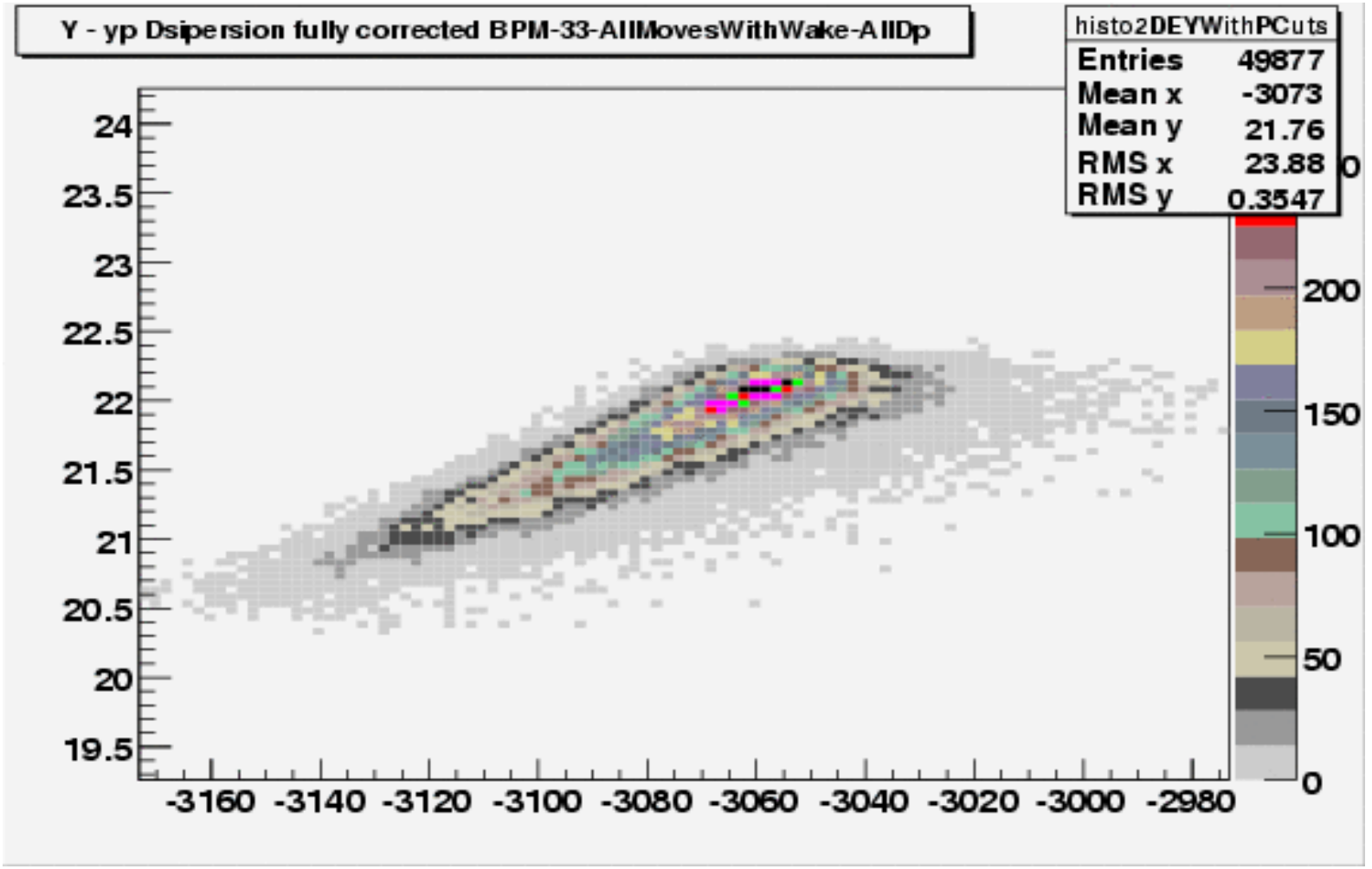
y'(micro-radiants)



y (microns)

Vert. Phase Space at S ~< 1.05 km, Merlin Momentum-y & Momentum yp de-correlated. (i.e., full dispersion corrected.)

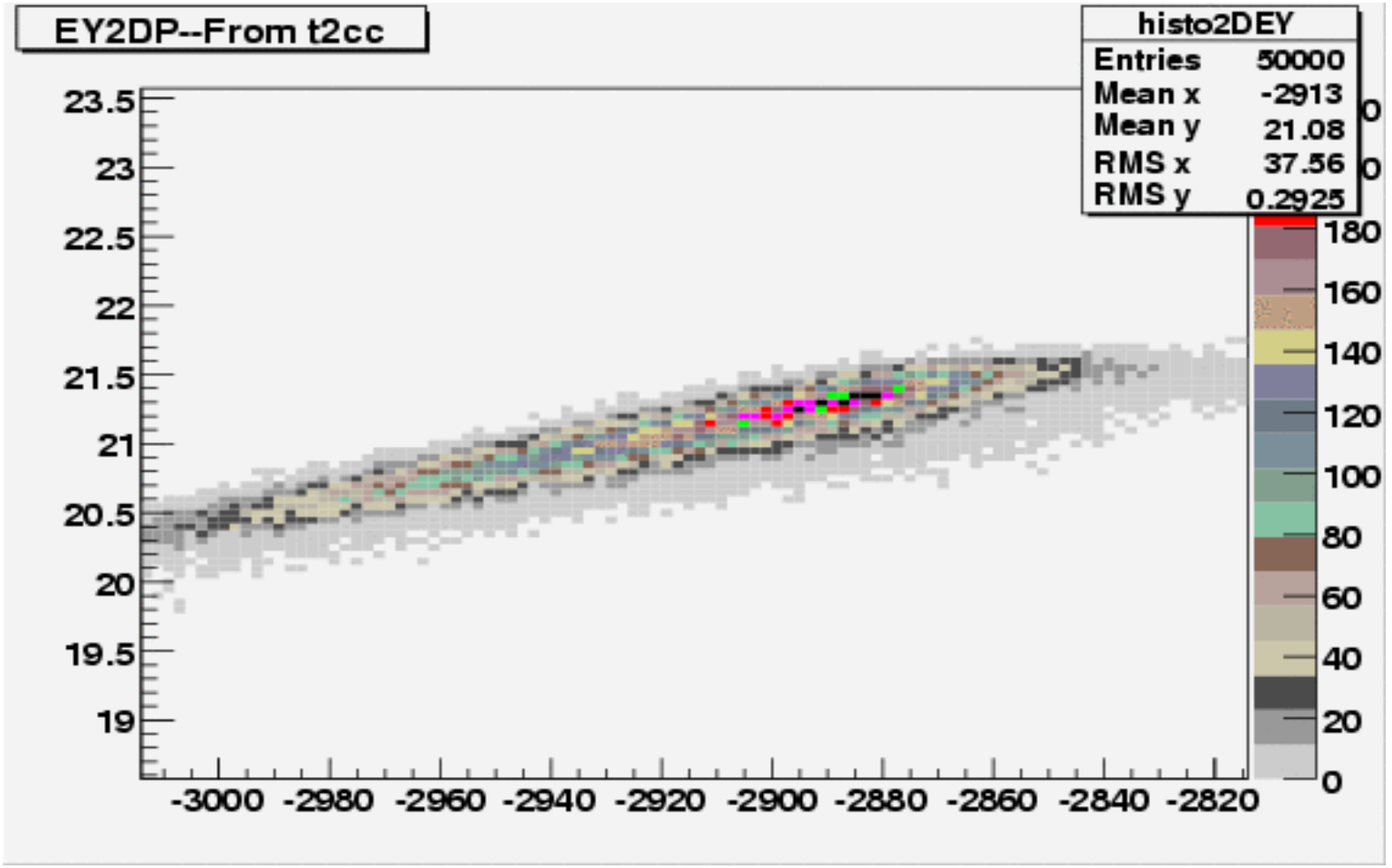
y'(micron-radiants)



y (microns)

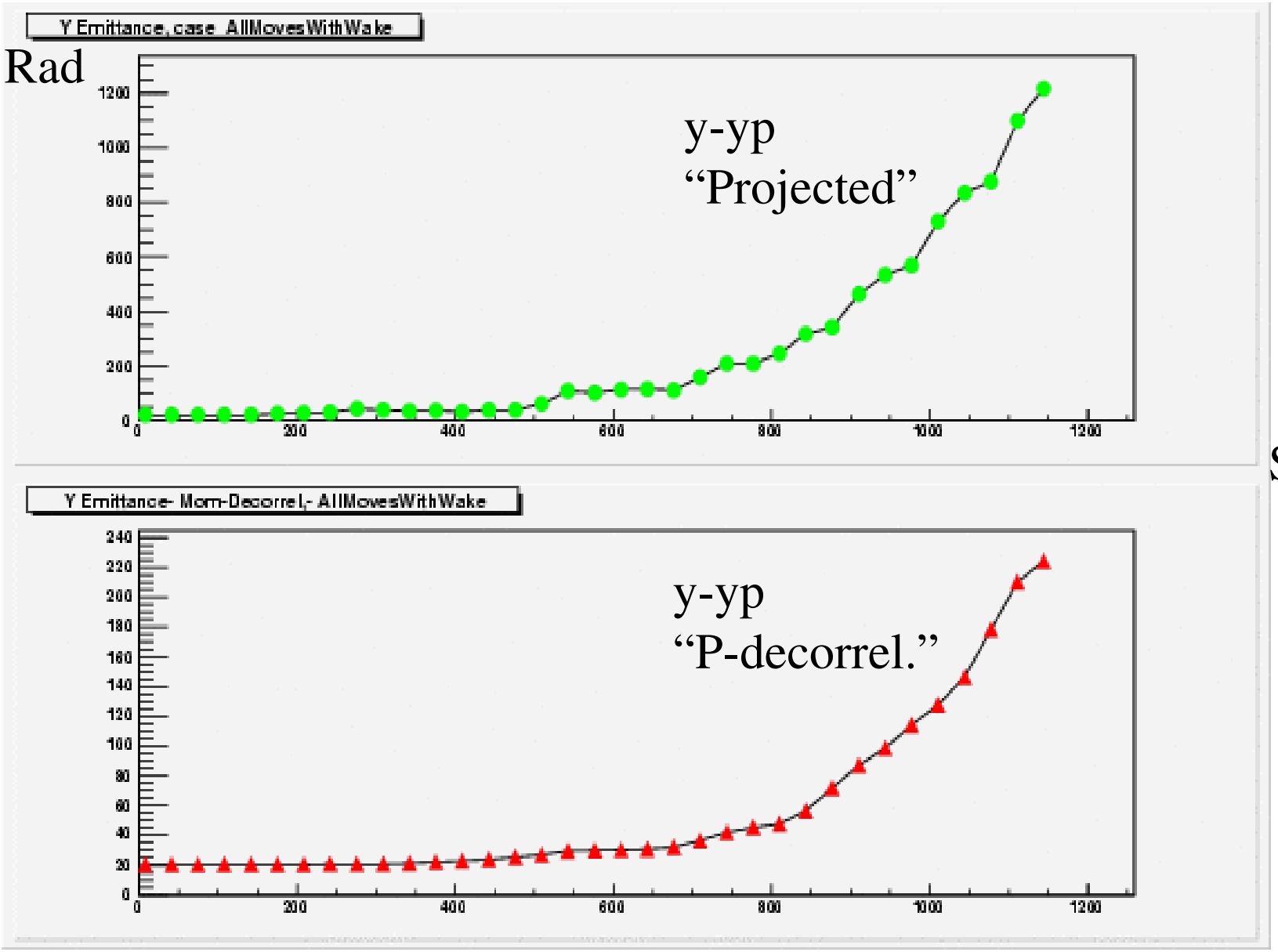
Vert. Phase Space at S ~< 1.05 km, Lucretia Momentum-y & Momentum yp decorrelated. (i.e., entirely dispersion corrected.)

y'(micro-radiants)



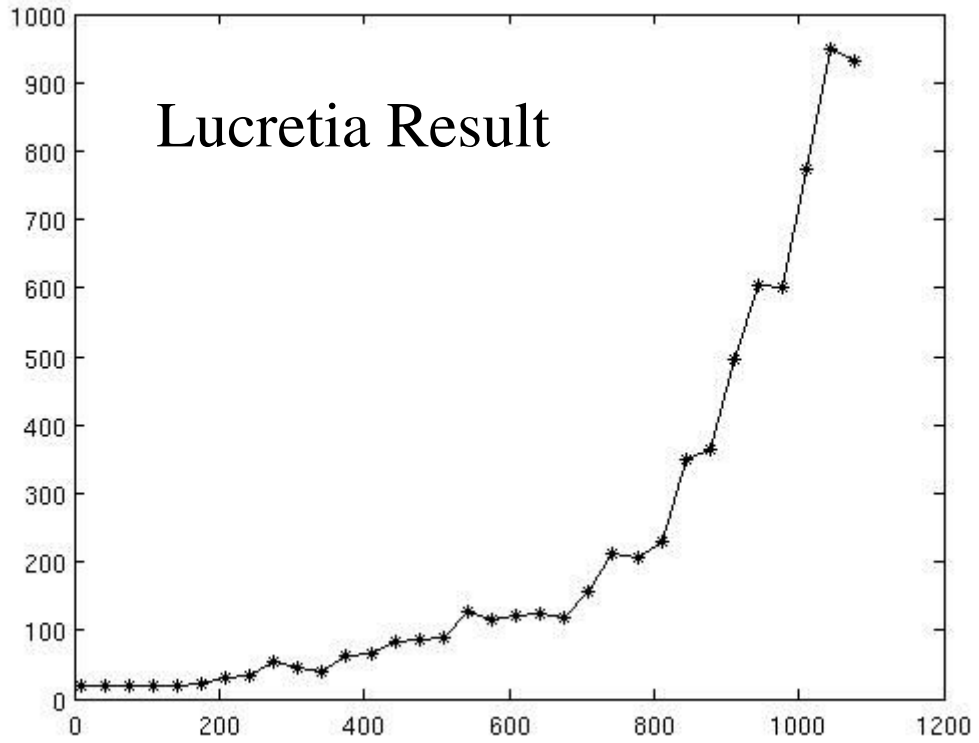
y (microns)

Vertical Emittance vs S, Merlin



Vertical Emittance vs S, cont'd

Nm Rad



Projected emitance!. "normal mode" emittance ~50 nm

At BPM 34 (S ~ 1.05 km.)
Merlin = 1,097 nm.
Lucretia = 949. nm

=> Agreeing with 10 to 20%

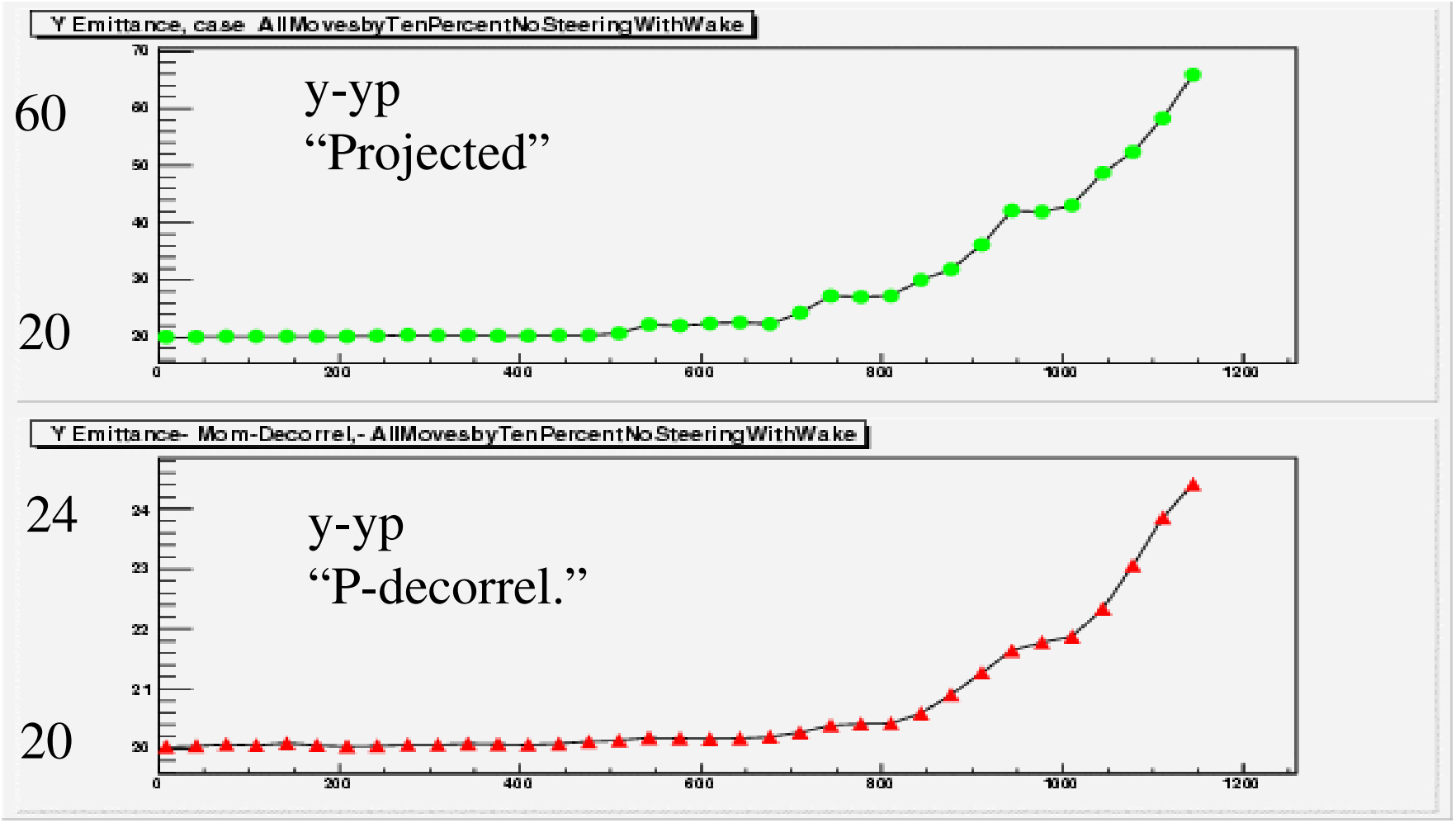
Rapidly varying with S, and phase advance (due to the strong correlation with Dispersion!) => sensitive test.

But is this good enough?

Vertical Emittance vs S, smaller disturbances
Misalignement 1/10 of previous plots

Merlin Result

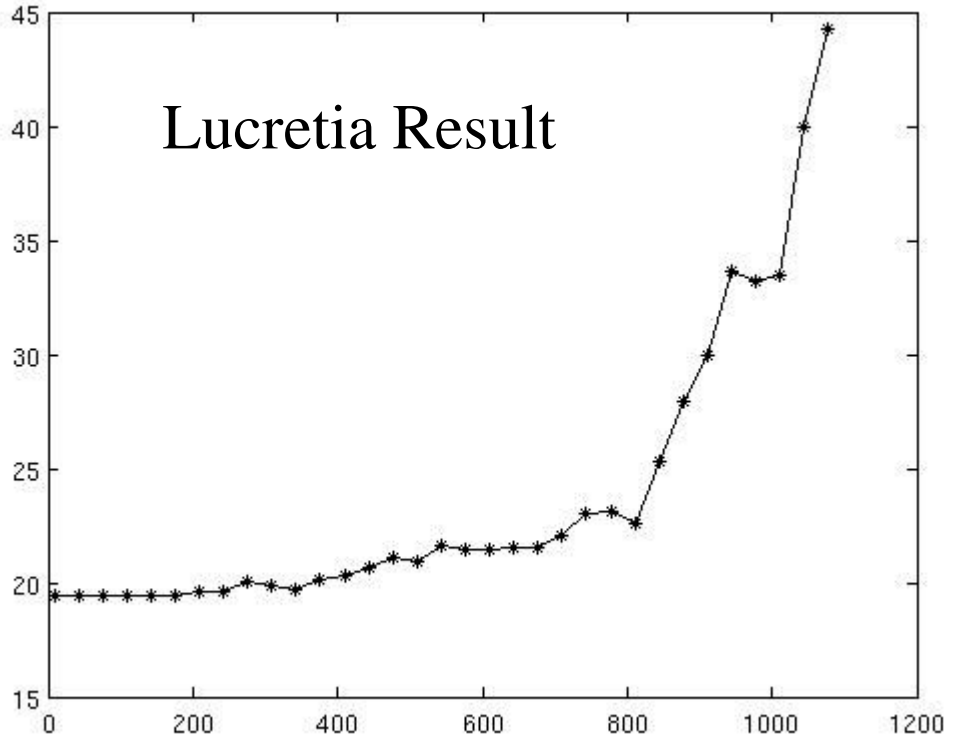
Nm Rad



S(m)

Vertical Emittance vs S, smaller disturbances Misalignement 1/10 of previous plots

Nm Rad



Projected emitance!. "normal mode" emittance ~50 nm

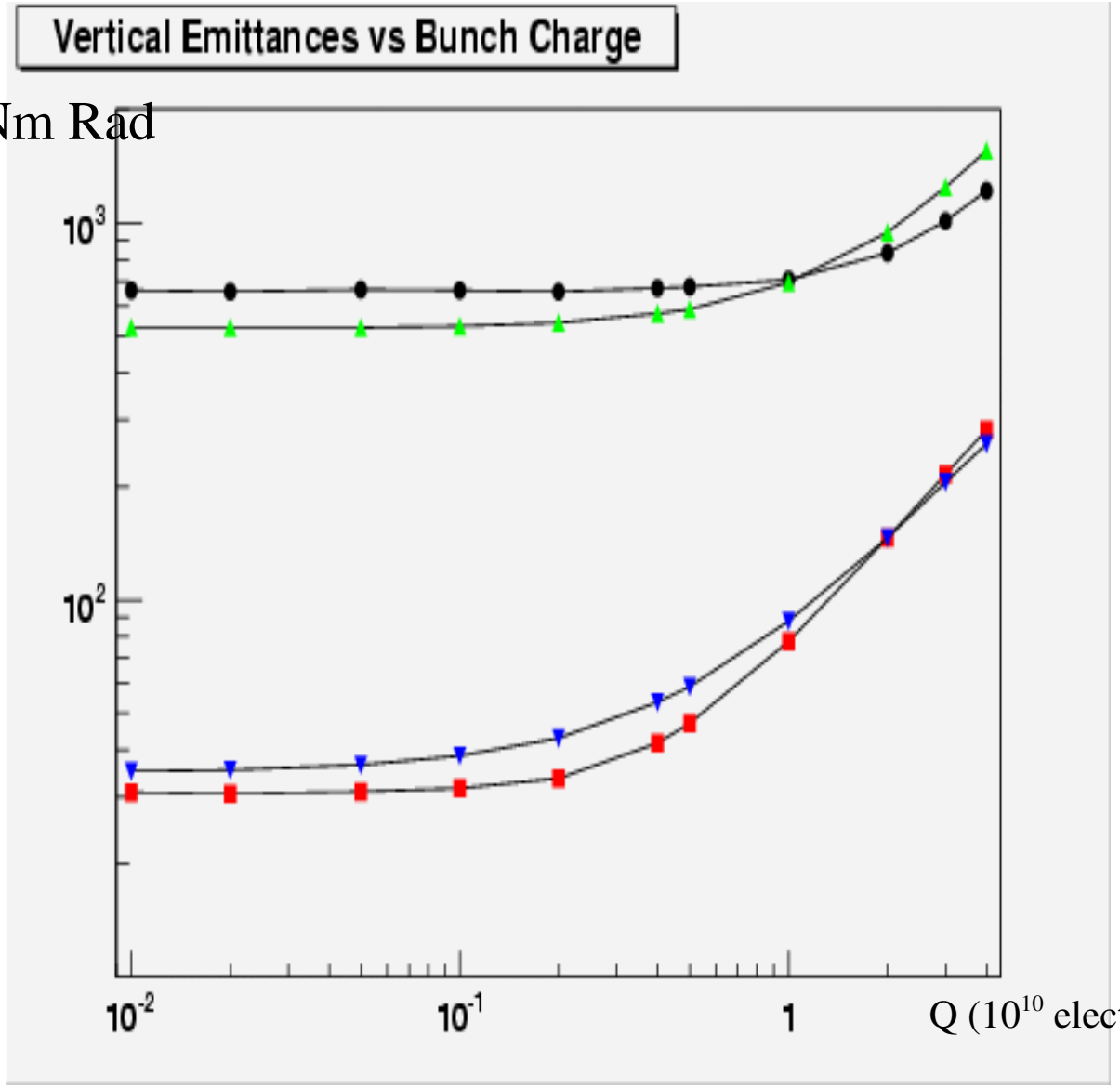
At BPM 34 (S ~ 1.05 km.)
Merlin = ~60 nm.
Lucretia = 35 to 40 nm

=> worse discrepency!!

Non-linearities!..
Both vs S
vs displacements

A one percent of nominal misalignment give only a ~ 1 nM emittance grotwth at 1 km.

Vertical Emittance growth vs Bunch Charge.



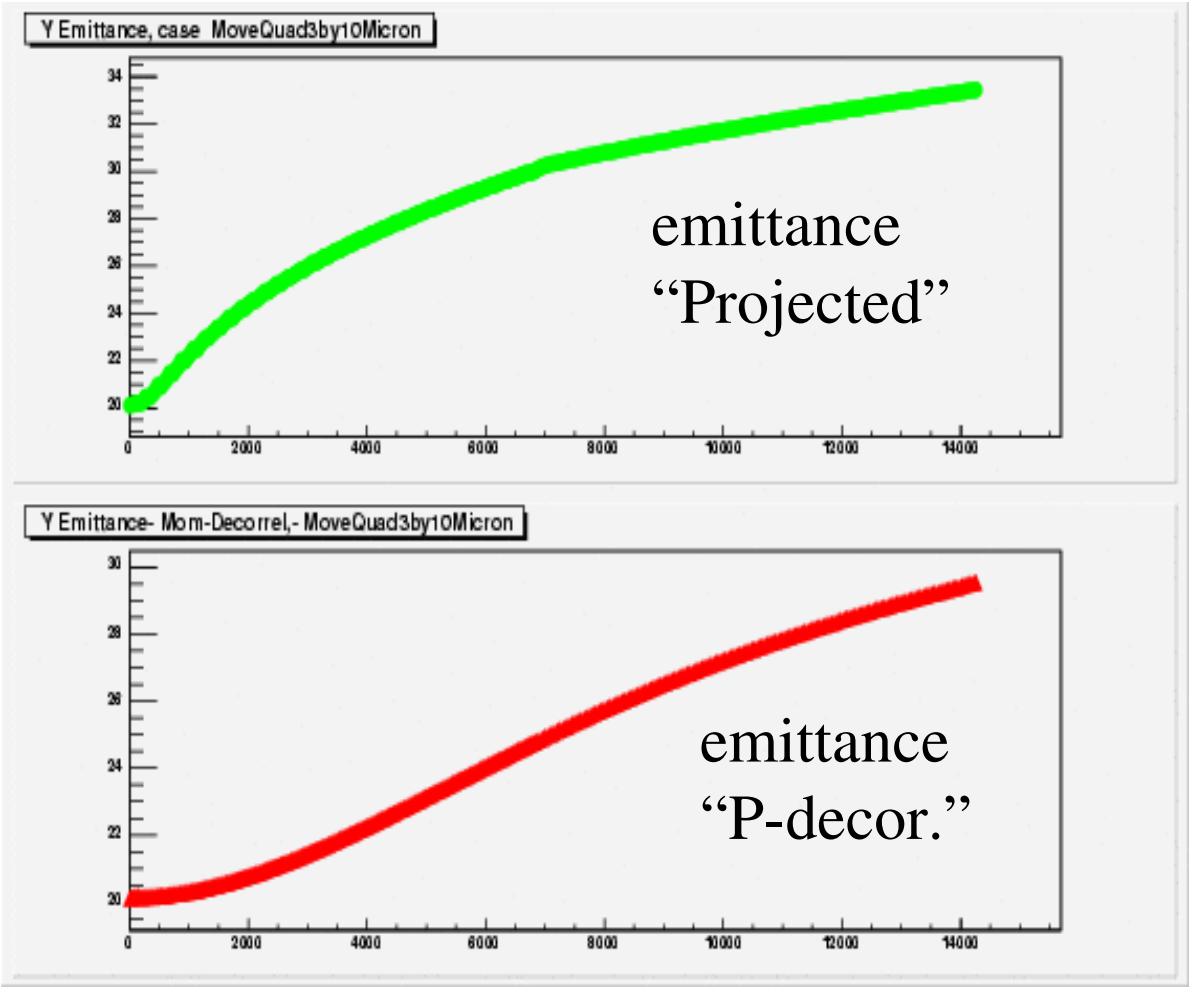
At S = 1. km.

Black dots:
Projected emittances, Merlin
Red Square:
Mom de-correlated,
Merlin
Green up triangles:
Projected emittances, Lucretia
Blue Square:
Mom de-correlated,
Lucretia.

Note: Transverse Wake field do matter if beam is not steered!.

Simpler tests, suggested by N. Solyak
One Quadrupole Move (#4), by ~ 1 to 10 μ m

Nm Rad



At S = 14 km.

LIAR, move by ~1 micron
-> emittance 39 nm
Merlin ~ 20.3 ! nm

Merlin, 10 microns move,
(either quad 3,4,5)
worse is only 34 nM (proj.)
and 30 (P-de-correlated)

???? To be pursued.

Timing Results: Lucretia/Merlin

- Done on the same system, same CPU:, Intel(R) Xeon(TM) MP CPU 3.00GHz
- Test Job:
  - Loading the 23.4 lattice, + misalignment file
  - Generate the input beam, 100,000 particles.
  - Tracking over 1.1 km, recording BPM, computing emittances at each BPM.
  - Producing ~ 3 plots.
- Result: Lucretia ~ 3 minutes, Merlin 2 minutes.
  - Timing depends on how much analysis is done!

Steering Algorithms: status

- In Lucretia: only 1-1 algorithm
- In Merlin,
  - 1-1 .... implemented (or re-implemented)
  - Dispersion Free Steering, DFS' (with varying the weight assigned to Dispersion, orbit deviation and dipole kicks)
    - w/o pre-steering into a region..
  - Brute force minimization of vertical emittance (just coded!)
  - Missing : Balistic alignment
  - Easy to add other other algorithms at this point, as most of the code consist of controls utilities, simulated DAQ from BPMs, emittance calculations, interfaces to non-linear optimizer.

Steering Algorithms, in general

- Happy to have the opportunity to re-implement such algorithms, a great way to learn how this machine will be controlled! Nice Beam dynamic problems..
- Learned why DFS works better than the simple 1-1 :
  - first cancellation of unknown beam offsets
  - Since excessive, uncorrected Dispersion is to first order the source of emittance growth, minimizing the dispersion everywhere is the way to go..

DFS' implementation, some details.

- Should have tried to follow as much as possible Jeff S. algorithm in BMAD, but failed to appreciate many subtleties in his F95 code. Also, wanted to understand a bit what I was doing... But concept of overlapping region where tuning is performed, sequentially, is preserved.
- Non-linear Optimizer is different.. Could translated from f95, but preferred to use the new C++ Minimization package written by M Fischler. and D. Sachs, from CD, based on their deep understanding of Minuit. This package has growth potential, more algorithms could be added...
- Based on a genuine Dispersion calculation at the BPM locations, not a mere difference off/on full energy.
- The set of cavities for which the voltage is turned down does not currently map to a unique klystron sets => a bit unrealistic.

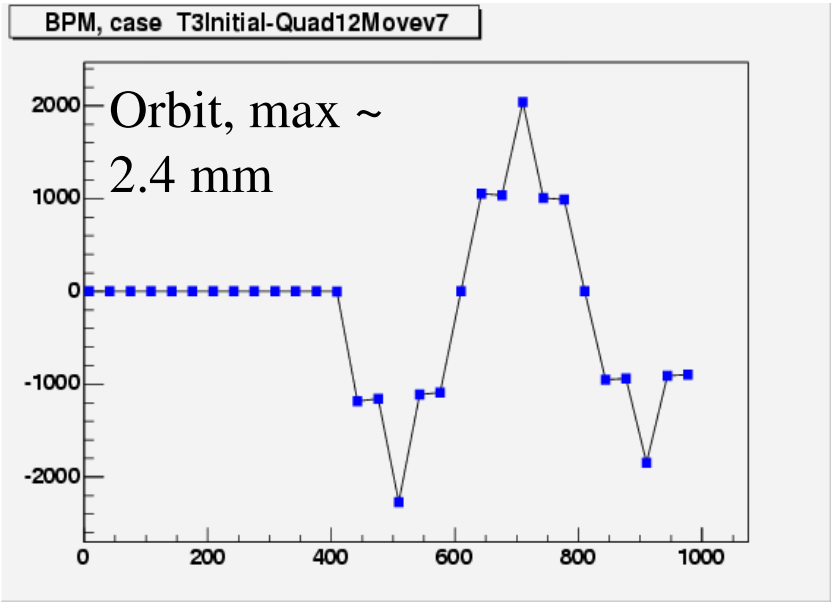
DFS', What are we minimizing?

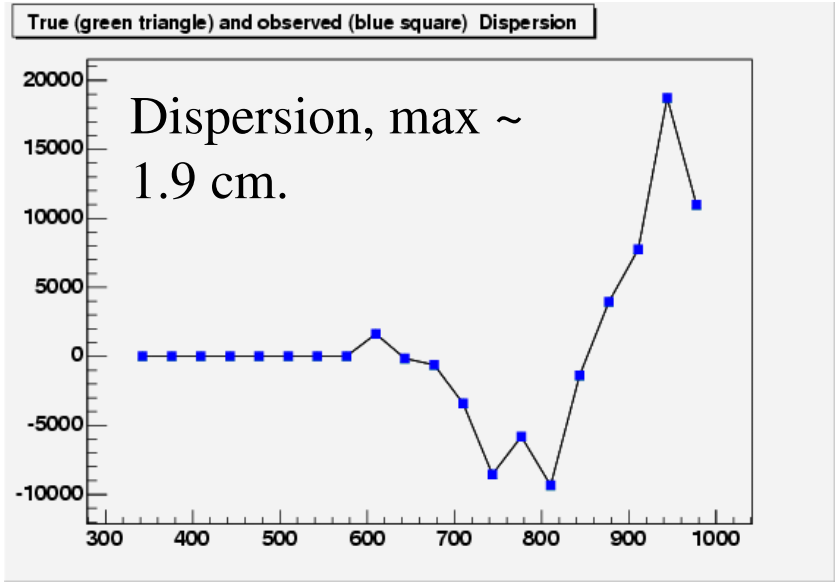
- For a region (typically, 20 dipoles, ~ 3.3 betatron wave length), minimize the following sum :
  - $\Sigma_{bpm}\ (W_d * D^2 + W_o * bpm^2 + W_{dip} * Kick^2)$
  - where the weight ("W") factors are to be tweaked, the BPM values are left uncorrected (Offsets are not known..). Pure and simple DFS is W_o = W_dip = 0

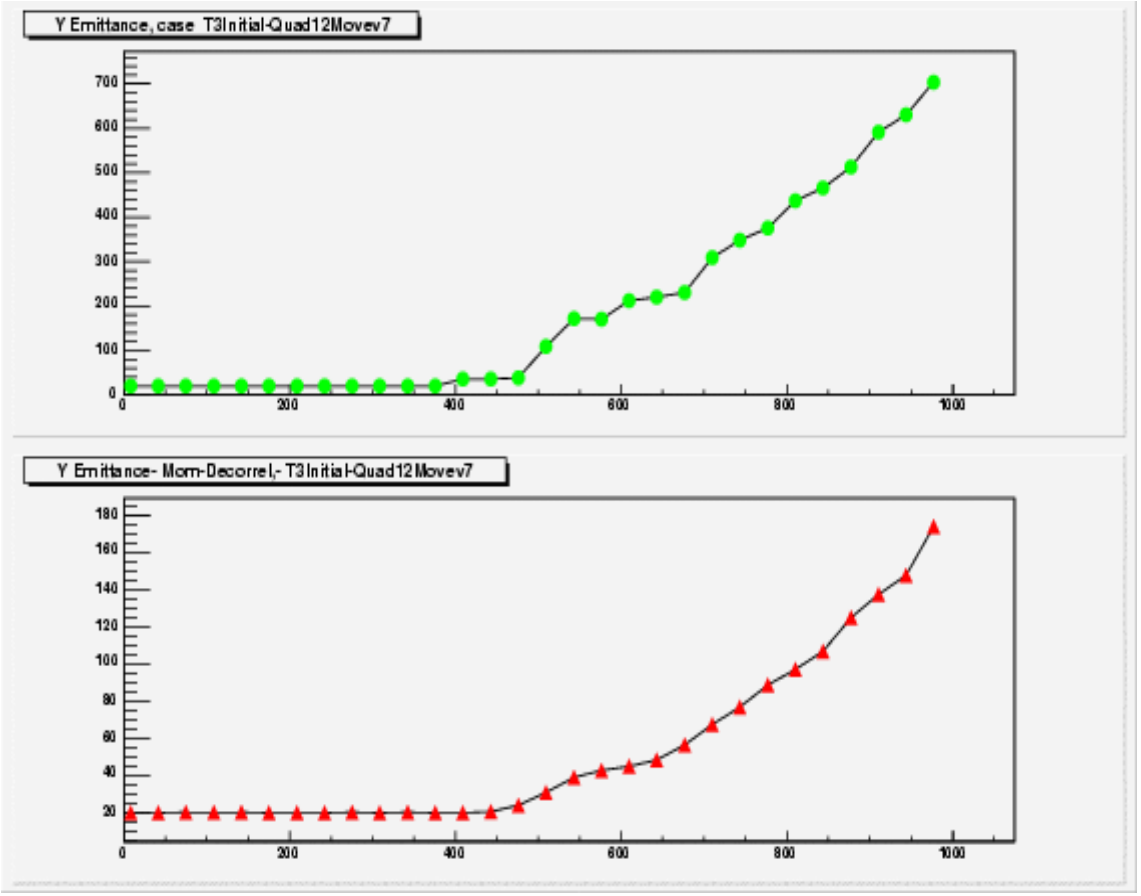
DFS: more details...

- “Migrad” minimization algorithm got seriously confused when attempting to tune ~ 20 dipoles “at once”, in a single minimization sequence. It simply did not converged ran for ever..
- Revert to a simpler algorithm “Simplex” based, with only 4 dipole being allowed to be changed at a time, sweeping a region ~ 5 to 10 times, i.e., going back and forth along the beam line, and adjusting only 4 dipoles at a time.
- Minimize done “step by step”, and if the estimated distance to the minimum is less than a fixed, small fraction (< 1%) of the function to be minimized, then, move on to the next set of 4 dipoles. This minimal fraction decreases as the sweep number increases, that is, in early sweeps, alignment are crude and gets refined as an acceptable solution is found.
- Always incremental changes to a given dipole setting. Always start previously found solution.
- Tedious and unrealistic, perhaps. but it seems to work.
- Many variant are easy to code at this point.

DFS, single quadrupole motion, at BPM 13, displaced by 200 microns: Initial Condition

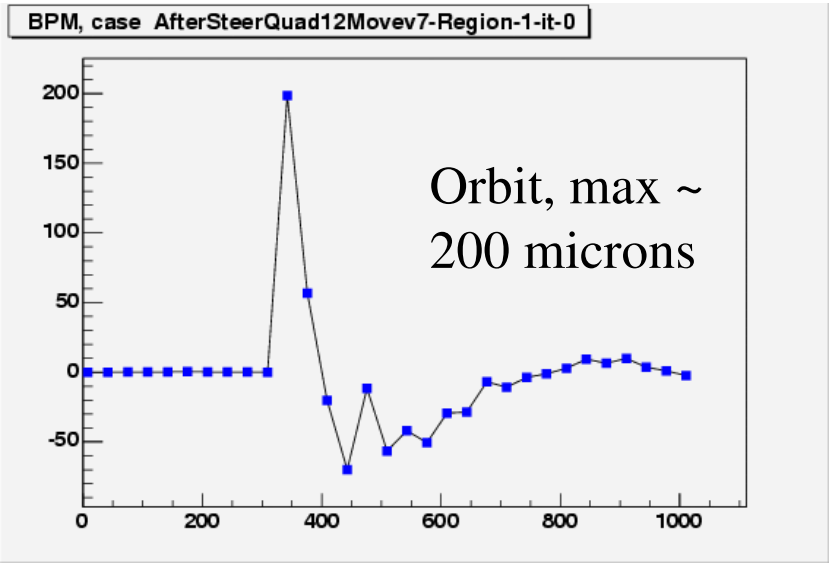


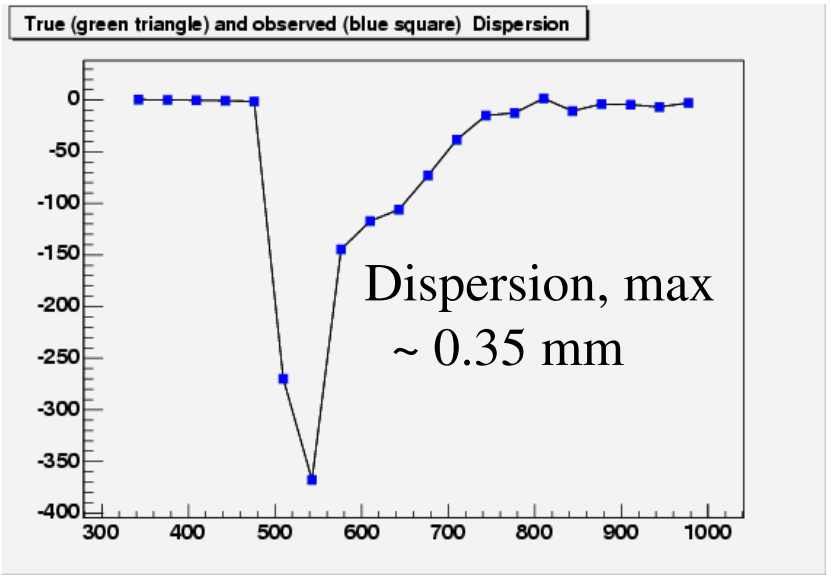


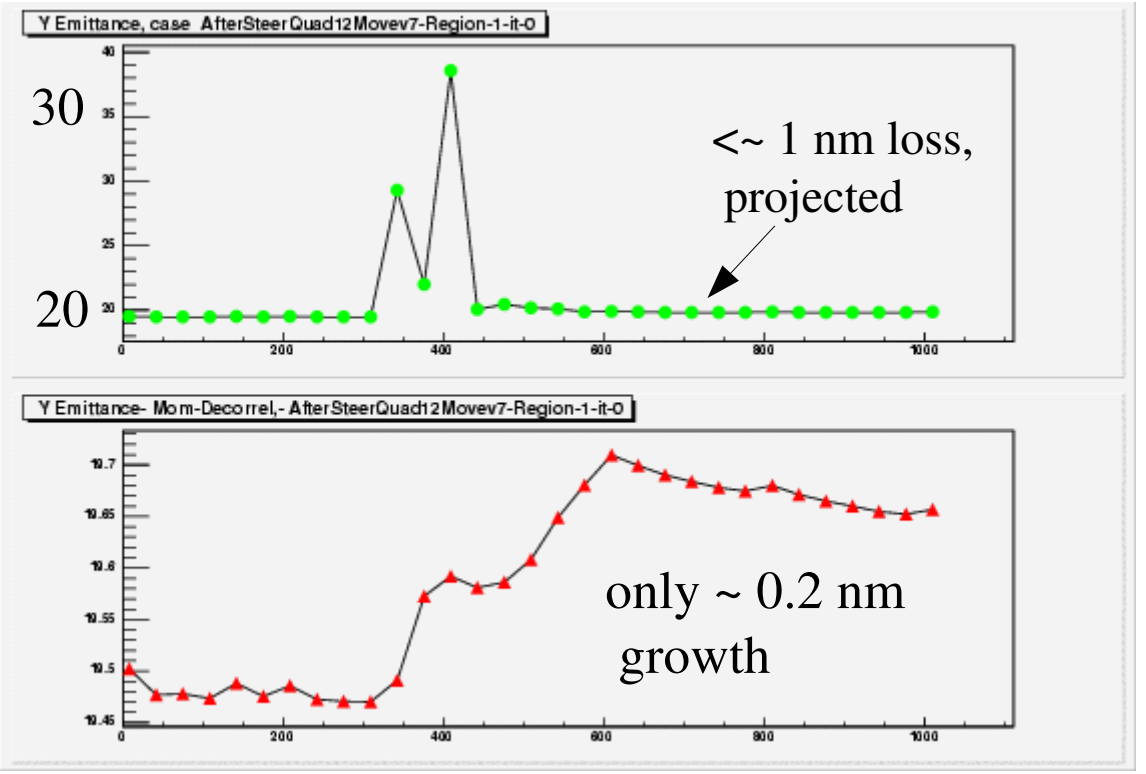


Projected emittance grows to 700 nm over a distance of 1 km.
Dispersion corrected emitt. grows to 170 nm.

DFS, single quadrupole motion, at BPM 13, displaced by 200 microns: Final Conditions



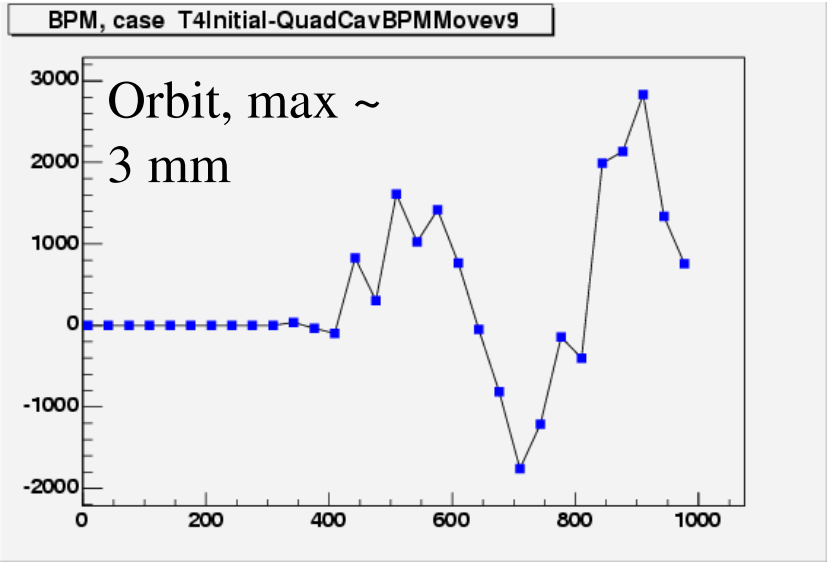


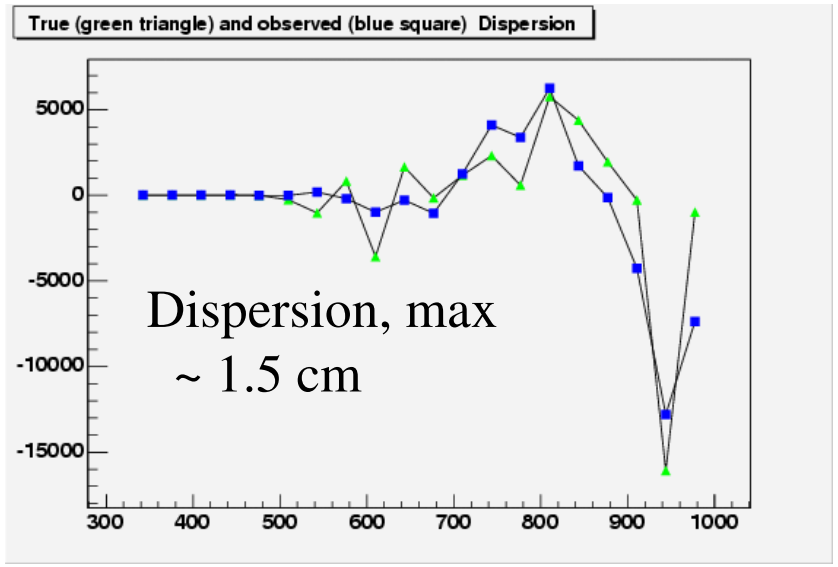


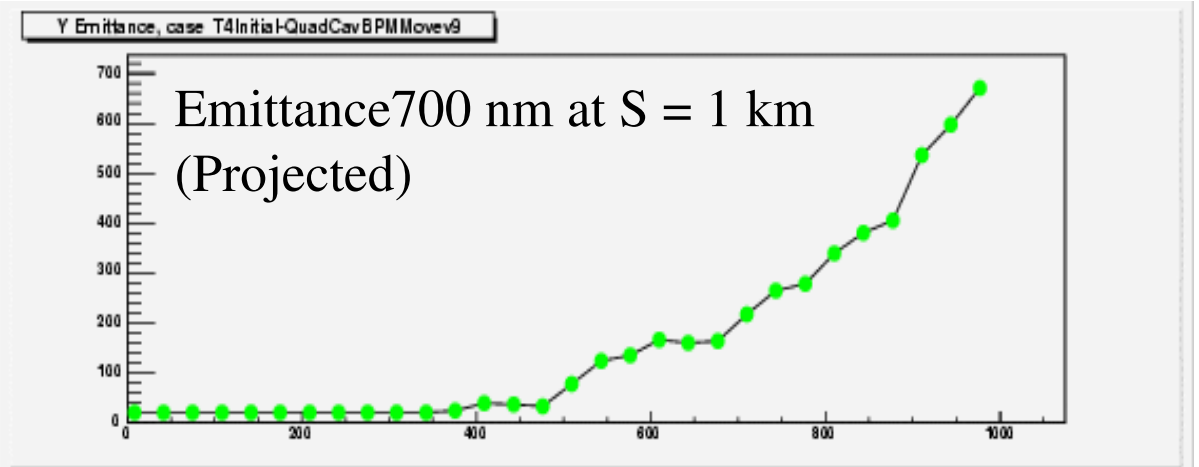
Looks like the algorithm decided to make a small "dispersion bump" ~ 100 m upstream of the displaced quadrupole. The dispersion is nearly fully corrected at 800 mm.

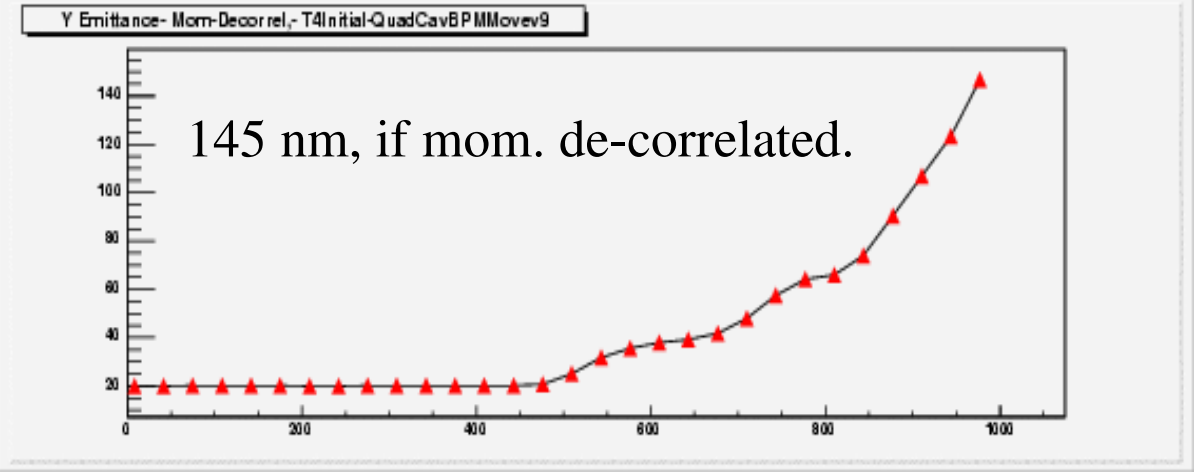
Note: BPM resolution is 5 Angstrom !

Starting at dipole 10, moving all Quads, BPM, Cavities, following Jeff. S. mis-alignment file.



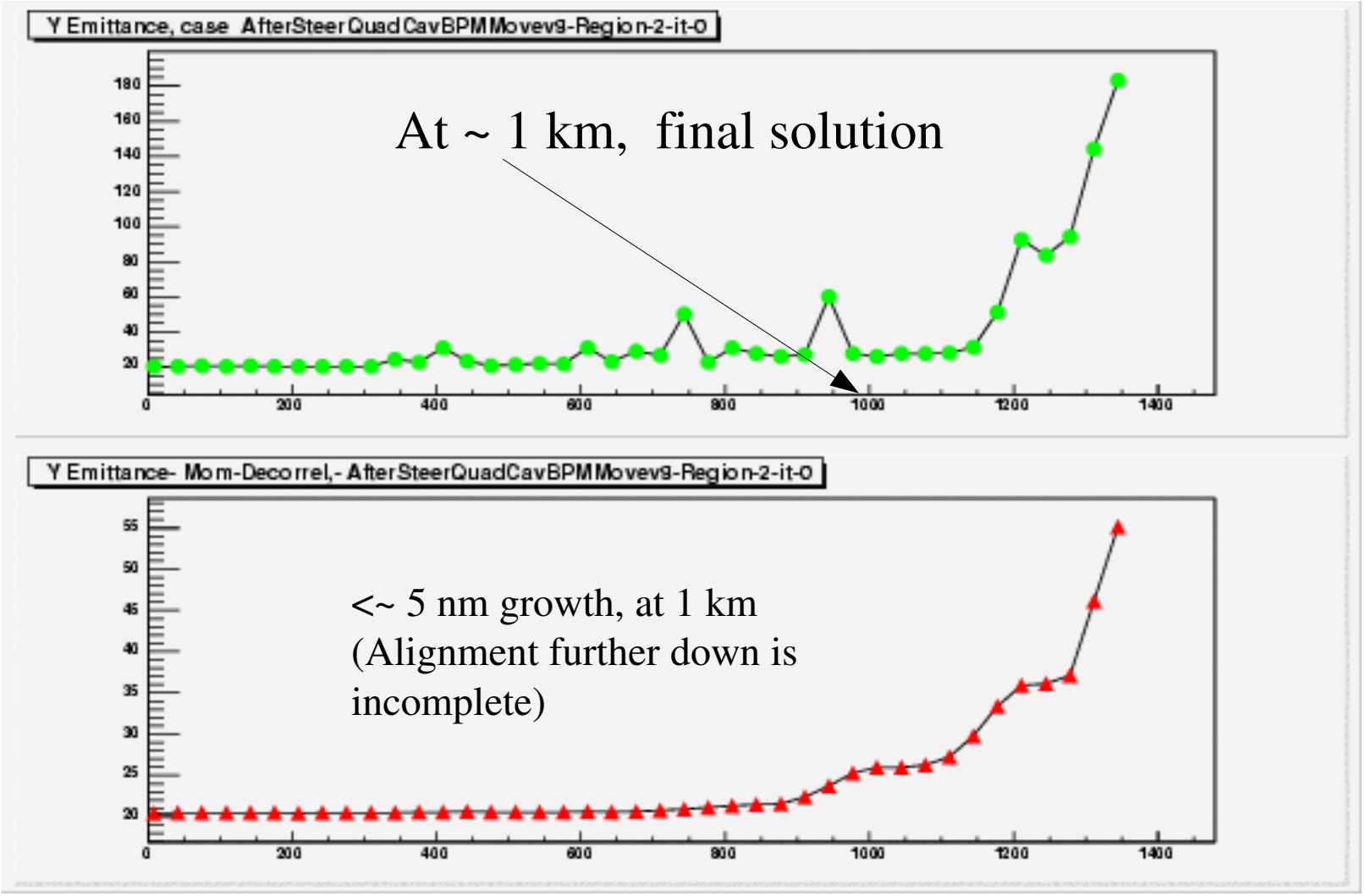






Note: BPM resolution is 5 Angstrom !
BPM have unknown offsets, i.e., misaligned.

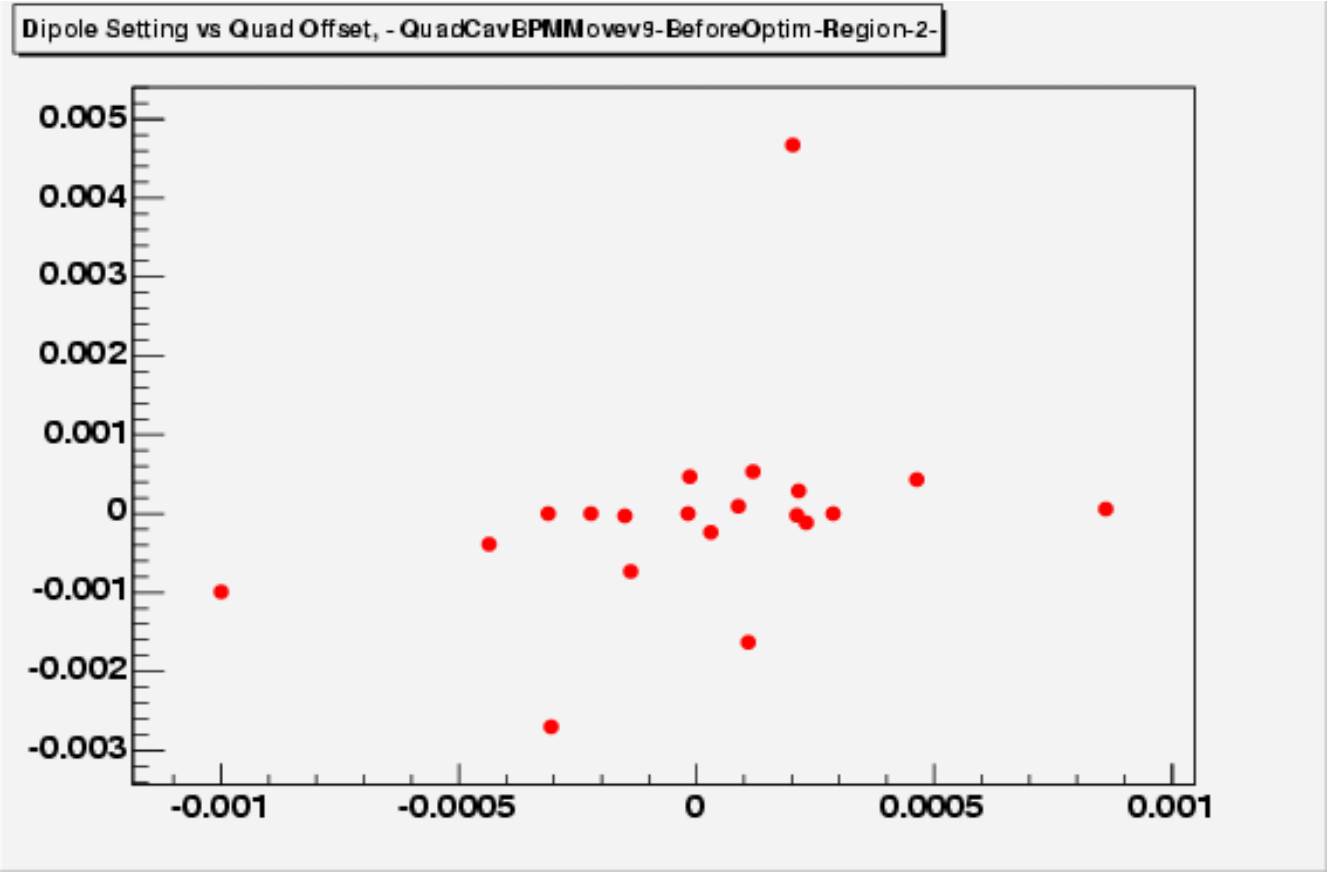
DFS steer this beam line, between dipole 10 and dipole ~ 33



Effective Dispersion bumps don't quite compensate correctly....

The “end game” is a problem, (did only 2 region, end of 2nd region has no overlap..)

Analysis of this solution, at the end of the 2nd region



Plotted here are the dipole setting (kick, mRad) vs Quadrupole mis-alignment, in m. One dipole is nearly pegged at its (arbitrarily chosen) maximum current, a kick of 5 mRad. We see no correlation between quadrupole displacement and corresponding corrector setting. Dispersion is corrected downstream of where it is created, a feature of this algorithm...

DFS in Merlin, status..

- Currently, quite slow and tedious: ~ 3,600 sequences of Off/On Energy setting and orbit measurement, per regions! That's 12 minutes, real time... What if the machine moves during this time?
- This simple algorithm fails completely if the BPM resolution is 5 microns==> no convergence, takes a lot more iterations.
- No beam jitter!
- No Quad. rolls !
- No ground motion!

Outlook on Steering Algorithm Studies

- Need to be a bit more clever at non-linear optimization techniques.
- Use HOM info? (proposed, need to do simulation work..)
- More realistic mis-alignments, resolutions, jitters,.....
- Klystrons constraints: common set of cavities, how fast can we control them? (at 5 Hz ?)
- If quad (or Cavities ?) are on movers, implement steering by both dipole correctors setting and position adjustments.
- Dynamical steering: as the ground moves, can we keep up?