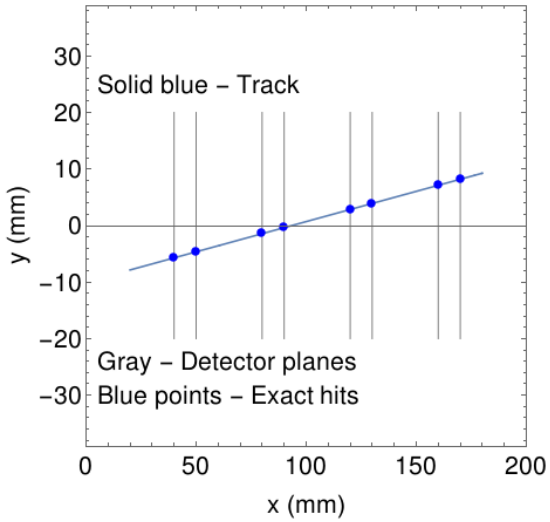
CLAS12, Track-Based, SVT Alignment with Millepede

G.P. Gilfoyle

Outline:
- The problem
- A toy model
- Basic idea
- Status

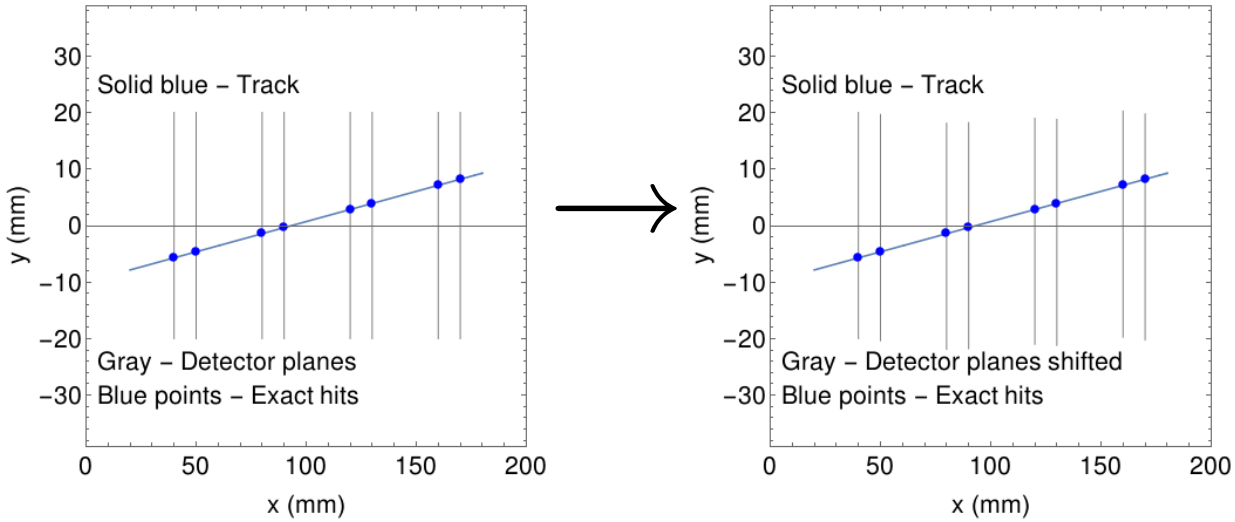
The Problem



Toy model:

- Straight tracks.
- Planar detectors.
- Shift detectors only in y.

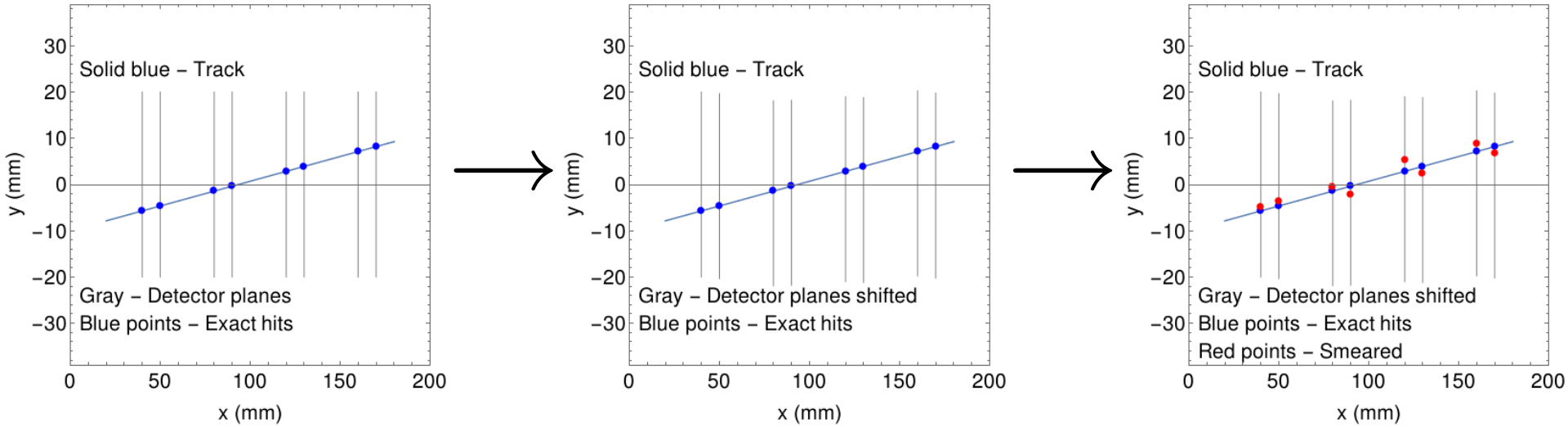
The Problem



Toy model:

- Straight tracks.
- Planar detectors.
- Shift detectors only in y.

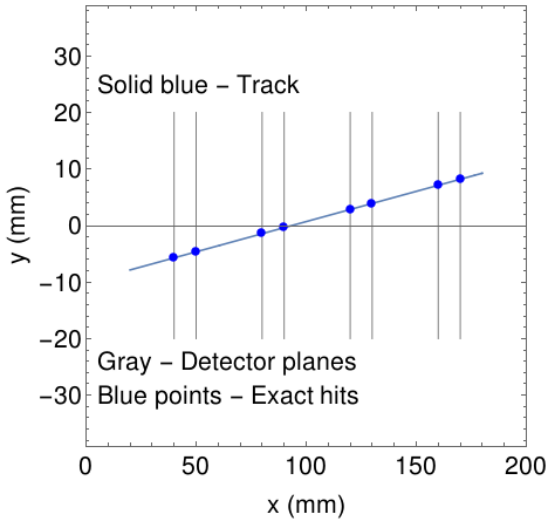
The Problem

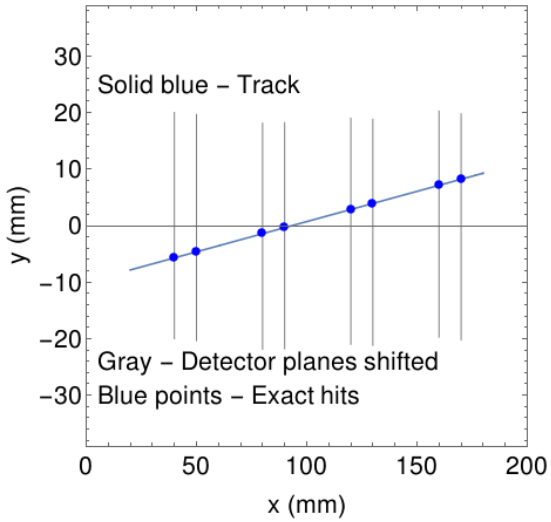


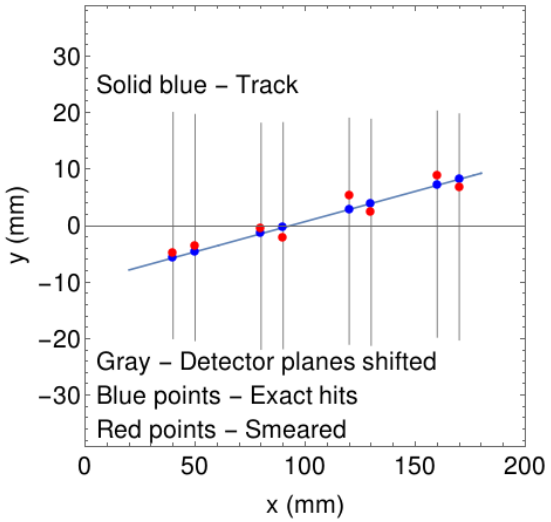
Toy model:

- Straight tracks.
- Planar detectors.
- Shift detectors only in y.

The Problem

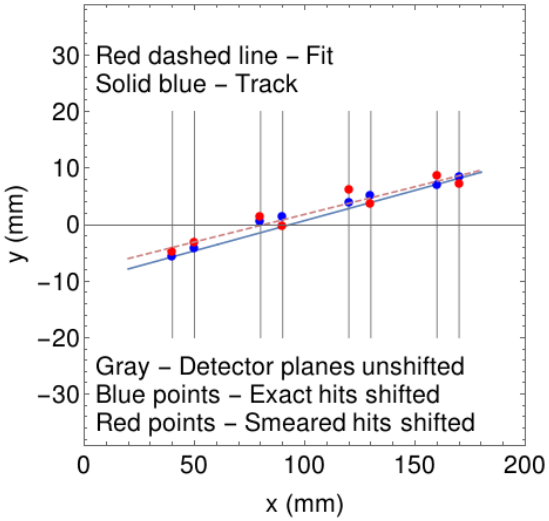






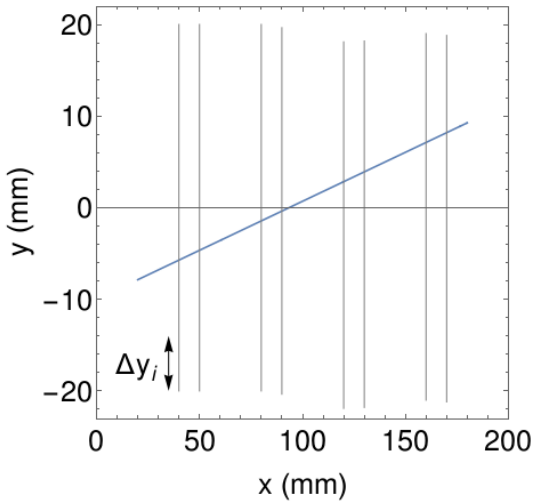
Toy model:

- Straight tracks.
- Planar detectors.
- Shift detectors only in y.



millepede: Linear Least Squares with Many Parameters

- In some least squares fit problems with many parameters those parameters can be divided into two classes.
  - Global - *i.e.* geometry.
  - Local - only present in subsets of the data, *i.e.* slope of a track.
- The code uses methods to solve the linear least square problem, irrespective of the number of local parameters.
- Up to ten thousand global parameters can be fitted.
- A simple test case:



Typically we fit the track with $y(x) = a + bx$.

In `millepede` use

$$y_{fit} = f(x, \vec{q}, \vec{p}) = \underbrace{\Delta y_1 + \Delta y_2 + ... + \Delta y_8}_{\text{global, } \vec{p}} + \underbrace{a + bx}_{\text{local, } \vec{q}} \quad .$$

Assume the initial fit with $\Delta y_i = 0$ is close to the final one so you can use the partial derivatives.

$$\frac{\partial z}{\partial \Delta y_i} = 1 \quad \frac{\partial z}{\partial a} = 1 \quad \frac{\partial z}{\partial b} = x$$

And use the residual $z = y_{meas} - f(x, \vec{q}, \vec{p})$.

CLAS12 millepede: Status

- Use the toy model described above as a tutorial.
- The code is running on the farm thanks to Mike Staib (CMU).
- Being used for HPS (Pelle Hansson and Alessandre Filippe) and GlueX (Mike Staib).
- A millepede event for the toy model.

Label	Measurement (mm)	Uncertainty (mm)	local derivatives		global derivatives
1	0.4378	1.4250	1.0	60.0	1.0
		...			
i	z_i	σ_i	1.0	x_i	1.0

- myMille
  - Running millepede requires two stages - (1) prepare a binary file with the data and (2) run the code that does the fitting called pede.
  - A C^{++} code to create the input binary called myMille has been written and tested with local tools.
- The code pede runs, reads the binary input file, but halts before fitting - "insufficient constraint equations!"?