CLAS12, Track-Based, SVT Alignment with Millepede

G.P. Gilfoyle
The Problem

Toy model:
- Straight tracks.
- Planar detectors.
- Shift detectors only in $y$. 
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

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

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 $z = y_{\text{meas}} - f(x, \vec{q}, \vec{p})$. 
Test 1

- Set $\Delta y_i = 0.0$ for all detector planes, simulate 26,000 straight tracks.
- Fix $\Delta y_i = 0$ for planes 1-2 and 5-8 and let $\Delta y_i$ vary for planes 3-4.

Simulation Input:  

- $\Delta y_3 = 0.0\ mm$
- $\Delta y_4 = 0.0\ mm$

millepede Output:  

- $\Delta y_3 = -0.0041 \pm 0.0067\ mm$
- $\Delta y_4 = -0.0018 \pm 0.0066\ mm$
Test 1
- Set $\Delta y_i = 0.0$ for all detector planes, simulate 26,000 straight tracks.
- Fix $\Delta y_i = 0$ for planes 1-2 and 5-8 and let $\Delta y_i$ vary for planes 3-4.

Simulation Input: millepede Output:

$\Delta y_3 = 0.0 \, mm$  $\Rightarrow$  $\Delta y_3 = -0.0041 \pm 0.0067 \, mm$

$\Delta y_4 = 0.0 \, mm$  $\Rightarrow$  $\Delta y_4 = -0.0018 \pm 0.0066 \, mm$

Test 2
- Set $\Delta y_i = 0.0$ for planes 1-2, 5-8, $\Delta y_i = -2.0 \, mm$ for planes 3-4, and simulate 26,000 straight tracks.
- Fix $\Delta y_i = 0$ for planes 1-2 and 5-8 and let $\Delta y_i$ vary for planes 3-4.

Simulation Input: millepede Output:

$\Delta y_3 = -2.0 \, mm$  $\Rightarrow$  $\Delta y_3 = -1.999 \pm 0.007 \, mm$

$\Delta y_4 = -2.0 \, mm$  $\Rightarrow$  $\Delta y_4 = -2.012 \pm 0.007 \, mm$
Randomly select misalignments $\Delta y_i$ uniformly over the range $\Delta y_i = -1 \text{ mm} \rightarrow 1 \text{ mm}$ and get the following.

$$\Delta y_i = \{0.0, -0.05, 0.75294, 0.90532, 0.83337, -0.571876, 0.934063, 0.710517\}$$

Two constraints are needed so set the constraints on planes 1-2 to the input values $\Delta y_1 = 0 \text{ mm}$ and $\Delta y_2 = -0.05 \text{ mm}$.

Excellent agreement with the inputs.
• How sensitive is the millepede fit to the accuracy of the constraints?
• Use same set of misalignments as before.
• Two constraints are still needed. Set the constraint on plane 1 as before to \( \Delta y_1 = 0 \, \text{mm} \), but use the ‘wrong’ value for plane 2 \( \Delta y_2 = 0 \, \text{mm} \).

![Graph showing residual vs. z for different misalignments.](image)

• Significant disagreement with the inputs especially as large z.
How sensitive is the millepede fit to a shift in plane 1?

Use same set of misalignments as before.

Two constraints are still needed. Set the constraint on plane 1 to $\Delta y_1 = 0.20 \ mm$, and for plane 2 $\Delta y_2 = \Delta y_1 - 0.05 \ mm = 0.15 \ mm$.

Relative fit results are the same with overall shift added.
Simulated a single track in an idealized planar detector.

Working on approach to fitting the track, getting derivatives, ...
CLAS12 millepede: 3D Distance equation

\[ d^2 = \left( -\frac{(x_2 - x_1)(x_1 - x_0)(x_2 - x_1) + (y_1 - y_0)(y_2 - y_1) + (z_1 - z_0)(z_2 - z_1)}{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} - x_0 + x_1 \right)^2 + \\
\left( -\frac{(y_2 - y_1)(x_1 - x_0)(x_2 - x_1) + (y_1 - y_0)(y_2 - y_1) + (z_1 - z_0)(z_2 - z_1)}{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} - y_0 + y_1 \right)^2 + \\
\left( -\frac{(z_2 - z_1)(x_1 - x_0)(x_2 - x_1) + (y_1 - y_0)(y_2 - y_1) + (z_1 - z_0)(z_2 - z_1)}{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} - z_0 + z_1 \right)^2 \]  

(1)
\[ \frac{\partial d^2}{\partial x_1} = \]
\[
\frac{\partial d^2}{\partial x_1} = 2 \left( 1 - 2 (-x_1 + x_2)^2 (-x_0 + x_1) (-x_1 + x_2) + (-y_0 + y_1) (-y_1 + y_2) + (-z_0 + z_1) (-z_1 + z_2) \right) / \\
\left( (-x_1 + x_2)^2 + (-y_1 + y_2)^2 + (-z_1 + z_2)^2 \right)^2 - \frac{(x_0 - 2 x_1 + x_2) (-x_1 + x_2)}{(-x_1 + x_2)^2 + (-y_1 + y_2)^2 + (-z_1 + z_2)^2} + \\
\frac{(-x_0 + x_1) (-x_1 + x_2) + (-y_0 + y_1) (-y_1 + y_2) + (-z_0 + z_1) (-z_1 + z_2)}{(-x_1 + x_2)^2 + (-y_1 + y_2)^2 + (-z_1 + z_2)^2} \\
\right) / \\
\left( (-x_0 + x_1) (-x_1 + x_2) + (-y_0 + y_1) (-y_1 + y_2) + (-z_0 + z_1) (-z_1 + z_2) \right) / \\
\left( (-x_1 + x_2)^2 + (-y_1 + y_2)^2 + (-z_1 + z_2)^2 \right)^2 - \frac{(x_0 - 2 x_1 + x_2) (-y_1 + y_2)}{(-x_1 + x_2)^2 + (-y_1 + y_2)^2 + (-z_1 + z_2)^2} \\
\left( (-y_0 + y_1) (-y_1 + y_2) + (-y_0 + y_1) (-y_1 + y_2) + (-z_0 + z_1) (-z_1 + z_2) \right) / \\
\left( (-x_1 + x_2)^2 + (-y_1 + y_2)^2 + (-z_1 + z_2)^2 \right)^2 + 2 \left( (-x_1 + x_2) (-y_1 + y_2) \\
\left( (-x_0 + x_1) (-y_0 + y_1) (-y_1 + y_2) + (-z_0 + z_1) (-z_1 + z_2) \right) / \\
\left( (-x_1 + x_2)^2 + (-y_1 + y_2)^2 + (-z_1 + z_2)^2 \right)^2 \right) - \frac{(x_0 - 2 x_1 + x_2) (-z_0 + z_1)}{(-x_1 + x_2)^2 + (-y_1 + y_2)^2 + (-z_1 + z_2)^2} \\
\left( (-z_0 + z_1) (-x_0 + x_1) (-x_1 + x_2) + (-y_0 + y_1) (-y_1 + y_2) + (-z_0 + z_1) (-z_1 + z_2) \right) / \\
\left( (-x_1 + x_2)^2 + (-y_1 + y_2)^2 + (-z_1 + z_2)^2 \right)^2 
\]
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.

<table>
<thead>
<tr>
<th>Label</th>
<th>Measurement $(mm)$</th>
<th>Uncertainty $(mm)$</th>
<th>local derivatives</th>
<th>global derivatives</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.4378</td>
<td>1.4250</td>
<td>1.0</td>
<td>60.0</td>
</tr>
<tr>
<td>$i$</td>
<td>$z_i$</td>
<td>$\sigma_i$</td>
<td>1.0</td>
<td>$x_i$</td>
</tr>
</tbody>
</table>

- 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, and with the ‘proper’ constraints appears to work - thanks to Mike Staib, Alessandre Filippe, and Pele Hansson.
Generate a sample of straight tracks in our 2D, toy detector model.
Limit the sample to events with hits in all eight detector planes.
Generated over 26,000 events that satisfy this criteria.