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 $y$. 

\[ 	ext{Diagram showing straight tracks and shifted detector planes.} \]
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 $y$. 
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 \( z = y_{\text{meas}} - f(x, \vec{q}, \vec{p}) \).
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>σ_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.
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
CLAS12 millepede: Results

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 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:**

- $\Delta y_3 = -2.0 \ mm$
- $\Delta y_4 = -2.0 \ mm$

**millepede Output:**

- $\Delta y_3 = -1.999 \pm 0.007 \ mm$
- $\Delta y_4 = -2.012 \pm 0.007 \ mm$
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:

\[
\begin{align*}
\Delta y_3 &= 0.0 \text{ mm} \\
\Delta y_4 &= 0.0 \text{ mm}
\end{align*}
\]

\[
\begin{align*}
\Delta y_3 &= -0.0041 \pm 0.0067 \text{ mm} \\
\Delta y_4 &= -0.0018 \pm 0.0066 \text{ mm}
\end{align*}
\]

Test 2
- Set $\Delta y_i = 0.0$ for planes 1-2, 5-8, $\Delta y_i = -2.0 \text{ mm}$ 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:

\[
\begin{align*}
\Delta y_3 &= -2.0 \text{ mm} \\
\Delta y_4 &= -2.0 \text{ mm}
\end{align*}
\]

\[
\begin{align*}
\Delta y_3 &= -1.999 \pm 0.007 \text{ mm} \\
\Delta y_4 &= -2.012 \pm 0.007 \text{ mm}
\end{align*}
\]

Issues: Effect of number of constraints, rank defect, deciphering $millepede$ output...