Alignment in 

Xiaocong Ai

ACTS Developers Workshop, Sept 27, 2022

Alignment parameters

- Detector element placement description:
  - Translation (3 parameters) + Rotation (3x3 rotation matrix), i.e. 12 parameters
- Alignment parameters:
  - Translation + rotation about three axes using Euler angles, i.e. 6 parameters
  - Caveat: a rotation can be expressed in 24 equivalent sequence of Euler angles

```
enum AlignmentIndices : unsigned int {
  // Center of geometry object in global 3D cartesian coordinates
  eAlignmentCenter0 = 0u,
  eAlignmentCenter1 = eAlignmentCenter0 + 1u,
  eAlignmentCenter2 = eAlignmentCenter0 + 2u,
  // Rotation angle around global x/y/z axis of geometry object
  eAlignmentRotation0 = 3u,
  eAlignmentRotation1 = eAlignmentRotation0 + 1u,
  eAlignmentRotation2 = eAlignmentRotation0 + 2u,
  // Last uninitialized value contains the total number of components
  eAlignmentSize,
};
```

Alignment context

- Acts::GeometryContext is passed to all geometry-dependent algorithms in ACTS, i.e. :
  - algorithms can access the right version of geometry
  - alignment algorithm can create or update a version of geometry

```
Acts::Result<void> updateAlignmentParameters(
    const Acts::GeometryContext& gctx,
    const std::vector<Acts::DetectorElementBase*>& alignedDetElements,
    const AlignedTransformUpdater& alignedTransformUpdater,
    AlignmentResult& alignResult,
    Acts::LoggerWrapper logger = Acts::getDummyLogger()) const;
```

Track-based alignment

- Idea: tracks share the same detector geometry (a.k.a. global track parameters α) though they have their own track parameters (a.k.a. local track parameters x_i)
- The global track parameters can be estimated by minimizing the chi2 sum of a set of tracks:

$$\chi^2 = \sum_i {\chi_i}^2 = \sum_i [\vec{m}_i - \vec{h}_i(\vec{x}_i(\vec{\alpha}), \vec{\alpha})]^T V^{-1} [\vec{m}_i - \vec{h}_i(\vec{x}_i(\vec{\alpha}), \vec{\alpha})]$$

- This involves solving the non-linear equation iteratively, i.e. α is updated iteratively to approach its optimal value:

$$\frac{d^2\chi^2}{d^2\vec{\alpha}} \Big|_{\vec{\alpha_0}} \Delta\vec{\alpha} = -\frac{d\chi^2}{d\vec{\alpha}} \Big|_{\vec{\alpha_0}}$$

Available ingredients

$r = m - h(x, \alpha)$ The track residual

V The measurement covariance

$H = \left.\frac{\partial h(x)}{\partial x}\right|_{x_0}$ The projection matrix from (bound) track parameters to measurement

C The covariance of track parameters at different measurements:
Straightforward with global chi2 fitter. But can be provided by Kalman Filter as well (Calculated by Acts::detail::globalTrackParametersCovariance)

$A_{k\ell} \equiv \frac{\partial r_k}{\partial \alpha_\ell}$ The derivative of residual w.r.t. alignment parameters
Calculated by Surface::alignmentToBoundDerivative

The first and second derivatives for a single track is calculated in Acts::detail::trackAlignmentState

$$\frac{d\chi^2}{d\alpha} = 2A^T V^{-1}\left(V - HCH^T\right)V^{-1}r,$$

$$\frac{d^2\chi^2}{d\alpha^2} = 2A^T V^{-1}\left(V - HCH^T\right)V^{-1}A.$$

$$\left.\frac{d^2\chi^2}{d\alpha^2}\right|_{\alpha_0} \Delta\alpha = -\left.\frac{d\chi^2}{d\alpha}\right|_{\alpha_0}$$

Solved with Eigen LU decomposition (claimed stable and well tested with large matrices)

Derivatives w.r.t. alignment parameters

- Currently, the derivative of track residual is calculated w.r.t. the rotation of a surface about fixed global x/y/z axis, i.e. Tait–Bryan angles, extrinsic rotation
  - $R = R_z(\gamma)R_y(\beta)R_x(\alpha)$ (rotation of α about global x → β about global y → γ about global z)

$$= \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

$$= \begin{bmatrix} \cos\beta\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \cos\beta\sin\gamma & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma \\ -\sin\beta & \sin\alpha\cos\beta & \cos\alpha\cos\beta \end{bmatrix}$$

- Will change to provide the derivatives w.r.t. rotation around local axes of surface
- Also provide derivatives w.r.t. the rotation of layers/volumes, as more coarse alignment is often used

A naive alignment algorithm

```
template <typename fitter_t>
template <typename trajectory_container_t,
          typename start_parameters_container_t, typename fit_options_t>
Acts::Result<ActsAlignment::AlignmentResult>
ActsAlignment::Alignment<fitter_t>::align(
    const trajectory_container_t& trajectoryCollection,
    const start_parameters_container_t& startParametersCollection,
    const ActsAlignment::AlignmentOptions<fit_options_t>& alignOptions) const {
```

- It takes a set of tracks and figure out the sets of detector elements which can be aligned and are requested to be aligned => the initial value of α
- Perform the fit for each track, and estimate the chi2 derivatives w.r.t. α
- Update α using provided alignment parameter updater
- Stop iteration of the above two steps when provided converging criteria is met

Interface to Millepede II

- Interface to Millepede is not available yet
- Implementation of a Mille data writer should be straightforward

Steer.txt specifiying the location of the data file etc.

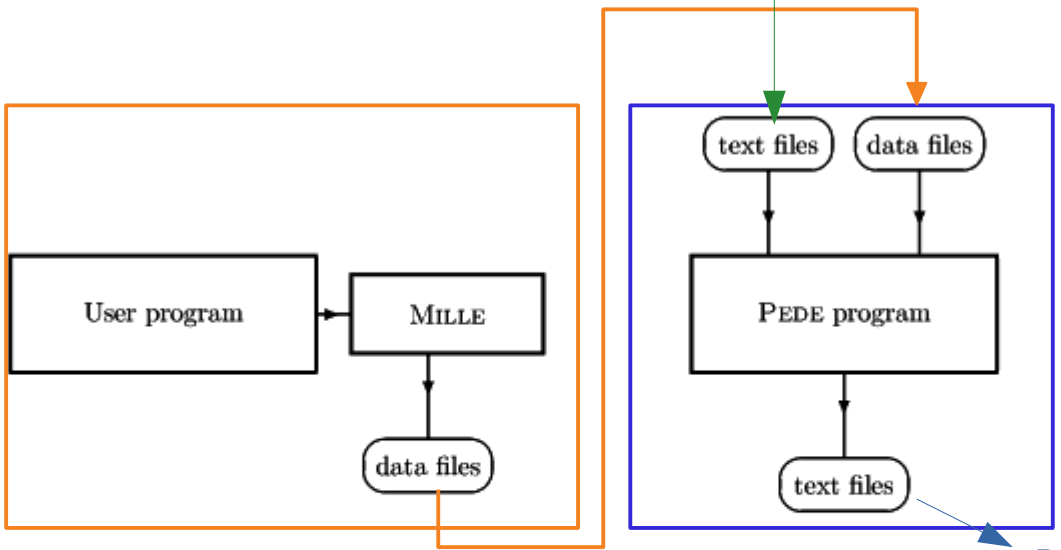

Figure 1: The subprogram MILLE (left), called inside the user program, and the stand-alone program PEDE (right), with the data flow from text and data files to text files.

Result file Millepede.res

Binary data files (residual, measurement error, derivatives w.r.t. track parameters and alignment parameters) need to be written out first

Standalone program executed via command lines (within user program): e.g. `pede -c steer.txt`

Summary

- Basic ingredients (residuals, alignment parameters derivatives) and geometry context for track-based alignment is available in ACTS
- Will provide derivatives with alignment parameters as rotation around local axes
- Consider to provide derivatives w.r.t. layer/volume geometry
- Consider to add interface to MillepedeII
- Needs dedicated alignment tests. Might start with small numbers of alignment parameters
- More development manpower is needed with increasing interest in ACTS alignment