ATLAS NOTE

ATL-SOFT-PUB-2014-003

June 13, 2014

Common Framework Implementation for the Track-Based Alignment of the ATLAS Detector

The ATLAS Collaboration

Abstract

The document presents ATLAS implementation of the generic track-based alignment framework. Being modular and highly configurable, it is applicable to all tracking systems in ATLAS; the silicon and drift tubes of the Inner Detector, as well as the gaseous Muon System. The implementation of the common parts and virtual interfaces is separated from the detector system specific parts. The formalism of the alignment based on the least squares principle and the general layout of the software implementation within the Athena framework are presented.

1 Introduction

While track reconstruction in modern high energy physics experiments is a very complex task, alignment of the detector elements to ever increasing precision is even more complex.

In ATLAS [1], alignment of two different tracking systems, the *Inner Detector* (ID) [2] and *Muon Spectrometer* (MS) [3], is required. The ID, consisting of the pixel detector at the innermost radii, the *Semiconductor Tracker* (SCT) using silicon strips and a *Transition Radiation Tracker* (TRT), are aligned using tracks. Until recently, two independent alignment algorithms were in place: one used for the Pixel and SCT alignment [4, 5]; and the other used for TRT alignment [6]. Both followed a global χ^2 minimization approach using reconstructed tracks. For the MS, the *Monitored Drift Tube* (MDT) chambers, *Cathode Strip Chambers* (CSC) and *Thin Gap Chambers* (TGC) all must all be aligned using tracks.

In this note we describe the common framework allowing the alignment of the ATLAS Muon Spectrometer and Inner Detector components. This framework is based on the alignment algorithms already in use for ID alignment.

We present the track-based alignment formalism, as well as its implementation in the common software framework, Athena [7]. The fundamental formalism is described in section 2 while section 3 describes the implementation of the core generic algorithms as well as the ID and MS specific parts.

2 Alignment Approaches

2.1 χ^2 Alignment

The chapter summarizes the formalism for alignment using reconstructed tracks. It starts with a description of track fitting using the Newton-Raphson method, and then extends the formalism of track fitting to include a fit for the alignment parameters.

2.1.1 Track fitting with the Newton-Raphson method

The Newton-Raphson uses an iterative approach to find the best fit of a track to a set of measurements. The quality of the fit is characterized by a track χ^2, determined from the distances between the track measurements and the fitted track. The track is parametrized by a set of five fit parameters, as well as additional parameters to allow for the effects of multiple Coulomb scattering (MCS). For a track fit without scattering, the track parameters $\boldsymbol{\tau} = (d_0, z_0, \phi_0, \theta_0, Q/p)$, all defined at a perigee either on the beam line or at the muon spectrometer entrance, are used to define the track [8].

For a track without scattering effects, the track χ^2 is calculated from the biased measurement residuals, $r_i = (e_i(\boldsymbol{\tau}) - m_i)$, where m_i is the local coordinate of the i_{th} measurement, and e_i is the local coordinate of the extrapolation of the fitted track to the surface on which the i_{th} measurement is recorded. The track χ^2 is then defined as the sum of the squares of the residuals divided by the measurement uncertainties, σ_i:

$$\chi^2 = \sum_i \left(\frac{r_i}{\sigma_i}\right)^2. \tag{1}$$

Using vector notation, where $\boldsymbol{r}$ is a vector of track residuals and Ω is the covariance matrix of the corresponding measurements ($\boldsymbol{m}$), the track χ^2 can be expressed as:

$$\chi^2 = \boldsymbol{r}^T \Omega^{-1} \boldsymbol{r}. \tag{2}$$

The track χ^2 minimization is done using the first and second derivatives of the χ^2 with respect to track

parameters. Defining the derivative $G = \mathrm{d}\boldsymbol{r}/\mathrm{d}\boldsymbol{\tau}$, the condition for the minimization of χ^2 is:

$$\frac{\mathrm{d}\chi^2}{\mathrm{d}\boldsymbol{\tau}} = 2G^T\Omega^{-1}\boldsymbol{r} = 0. \tag{3}$$

The value of $\boldsymbol{\tau}$ satisfying Eq. (3) is found using the Newton-Raphson method. This is done iteratively by evaluating the first and second derivatives of χ^2 with respect to track parameters evaluated using the track parameters of the current iteration, $\boldsymbol{\tau}_0$:

$$\left.\frac{\mathrm{d}\chi^2}{\mathrm{d}\boldsymbol{\tau}}\right|_{\tau_0} = 2G^T\Omega^{-1}\boldsymbol{r}_0 \tag{4}$$

and

$$\left.\frac{\mathrm{d}^2\chi^2}{\mathrm{d}\boldsymbol{\tau}^2}\right|_{\tau_0} = 2G^T\Omega^{-1}G, \tag{5}$$

giving the solution for $\boldsymbol{\tau}$:

$$\boldsymbol{\tau}_1 = \boldsymbol{\tau}_0 - \left(\frac{\mathrm{d}^2\chi^2}{\mathrm{d}\boldsymbol{\tau}^2}\right)^{-1}\frac{\mathrm{d}\chi^2}{\mathrm{d}\boldsymbol{\tau}}. \tag{6}$$

If the derivative G is constant, the problem is linear and the solution is exact. In general the derivative G depends on the track parameters, $\boldsymbol{\tau}$, and $\boldsymbol{\tau}_1$ is merely closer to the set of track parameters that minimizes χ^2. Recomputing the first and second derivatives at $\boldsymbol{\tau}_1$ can be used to get a new set of track parameters, $\boldsymbol{\tau}_2$, and the procedure can be repeated until a convergence criterion is met.

2.1.2 Track fitting with multiple scattering effects

The track fit can be improved by allowing for the track to scatter as it passes through material in the detector. To include the effects of multiple scattering, terms are added directly to the track χ^2:

$$\chi^2 = \sum_i \left(\frac{\rho_i(\boldsymbol{\tau},\boldsymbol{\theta})}{\sigma_i}\right)^2 + \sum_j \frac{(\hat{\theta}-\theta_j)^2}{\Theta_{jj}}. \tag{7}$$

Note that the residuals now also depend on the scattering angles, $\boldsymbol{\theta}$. The scattering expectation value, $\hat{\theta}$, is zero and its variance, Θ_{jj}, depends on the particle momentum and amount of material traversed [9]. The χ^2 has to be minimized for $\boldsymbol{\tau}$ and $\boldsymbol{\theta}$ simultaneously. Defining the derivative of residuals with respect to perigee and scattering parameters to be:

$$G \equiv \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\tau}} \qquad\qquad S \equiv \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\theta}} \tag{8}$$

the derivatives of χ^2 with respect to the perigee and scattering parameters are:

$$\frac{1}{2}\frac{\mathrm{d}\chi^2}{\mathrm{d}\boldsymbol{\tau}} = G^T\Omega^{-1}\boldsymbol{r}, \tag{9}$$

$$\frac{1}{2}\frac{\mathrm{d}\chi^2}{\mathrm{d}\boldsymbol{\theta}} = S^T\Omega^{-1}\boldsymbol{r} + \Theta^{-1}\boldsymbol{\theta}. \tag{10}$$

Neglecting second-order derivatives of residuals, the second derivatives of χ^2 with respect to perigee and scattering parameters are:

$$\frac{1}{2}\frac{\mathrm{d}^2\chi^2}{\mathrm{d}\boldsymbol{\tau}^2} = G^T\Omega^{-1}G, \tag{11}$$

$$\frac{1}{2}\frac{\mathrm{d}^2\chi^2}{\mathrm{d}\boldsymbol{\theta}^2} = S^T\Omega^{-1}S + \Theta^{-1}, \tag{12}$$

$$\frac{1}{2}\frac{\mathrm{d}^2\chi^2}{\mathrm{d}\boldsymbol{\theta}\mathrm{d}\boldsymbol{\tau}} = G^T\Omega^{-1}S. \tag{13}$$

The above can be written down in a compact form:

$$\frac{1}{2}\frac{\mathrm{d}\chi^2}{\mathrm{d}\boldsymbol{\pi}} = H^T V^{-1}\boldsymbol{\rho}, \tag{14}$$

$$\frac{1}{2}\frac{\mathrm{d}^2\chi^2}{\mathrm{d}\boldsymbol{\pi}^2} = H^T V^{-1} H, \tag{15}$$

where we additionally define:

$$\boldsymbol{\rho} \equiv \begin{pmatrix} \boldsymbol{r} \\ \boldsymbol{\theta} \end{pmatrix}, \tag{16}$$

$$V \equiv \begin{pmatrix} \Omega & 0 \\ 0 & \Theta \end{pmatrix}, \tag{17}$$

$$\boldsymbol{\pi} \equiv \begin{pmatrix} \boldsymbol{\tau} \\ \boldsymbol{\theta} \end{pmatrix}, \tag{18}$$

$$H \equiv \begin{pmatrix} G & S \\ \frac{\partial\boldsymbol{\theta}}{\partial\boldsymbol{\tau}} = 0 & \frac{\partial\boldsymbol{\theta}}{\partial\boldsymbol{\theta}} = \mathbf{1} \end{pmatrix}. \tag{19}$$

2.1.3 Track fitting with significant energy loss

Whenever energy loss due to material effects is large enough to produce significant fluctuations (e.g. combining the ID and MS measurements to fit a combined muon track), an extra term is added to the track χ^2, and an extra fit parameter (ΔE) accounts for the energy loss in the calorimeter. $\overline{\Delta E}$ is the expectation value for the energy loss for the known amount of traversed material and particle momentum, and $\sigma_{\Delta E}$ its statistical spread. The track χ^2 becomes:

$$\chi^2 = \sum_i \left(\frac{\rho_i(\boldsymbol{\tau}, \boldsymbol{\theta}, \Delta E)}{\sigma_i}\right)^2 + \sum_j \frac{(\hat{\theta} - \theta_j)^2}{\Theta_{jj}} + \frac{(\overline{\Delta E} - \Delta E)^2}{\sigma^2_{\Delta E}}. \tag{20}$$

The derivatives of residuals with respect to the additional fit parameter must be calculated, as well as the derivative of the additional term in the track χ^2.

When fitting hadronic tracks within the ID alone, the energy loss fluctuations may be neglected. Instead, the mean energy loss corresponding to the chosen particle hypothesis is attributed to every traversed material [10].

2.1.4 Global χ^2 method for alignment

The alignment parameters, $\boldsymbol{\alpha}$, are determined by minimizing the global χ^2. The global χ^2 is simply the sum of the χ^2 values for all tracks:

$$\chi^2_{global} = \sum_i \chi^2_i, \tag{21}$$

where χ^2_i is the χ^2 of the i_{th} track. The residuals, $r_i(\boldsymbol{\pi}, \boldsymbol{\alpha})$, now depend on the alignment parameters as well as the track fit parameters.

The total derivatives of the χ^2 with respect to the alignment parameters needs to be determined first. Defining:

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\alpha}} = \frac{\partial}{\partial\boldsymbol{\alpha}} + \frac{\mathrm{d}\boldsymbol{\pi}}{\mathrm{d}\boldsymbol{\alpha}}\frac{\partial}{\partial\boldsymbol{\pi}}, \tag{22}$$

the required derivative $\frac{\mathrm{d}\boldsymbol{\pi}}{\mathrm{d}\boldsymbol{\alpha}}$ is determined from the condition that, once at a minimum, the global χ^2 is at a minimum with respect to track parameters:

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\alpha}}\frac{\partial\chi^2}{\partial\boldsymbol{\pi}} = 0, \tag{23}$$

resulting in

$$\frac{\mathrm{d}\boldsymbol{\pi}}{\mathrm{d}\boldsymbol{\alpha}} = -\left(\frac{\partial^2\chi^2}{\partial\boldsymbol{\pi}^2}\right)^{-1}\frac{\partial^2\chi^2}{\partial\boldsymbol{\alpha}\partial\boldsymbol{\pi}}. \tag{24}$$

Defining:

$$A \equiv \frac{\partial\boldsymbol{\rho}}{\partial\boldsymbol{\alpha}}, \tag{25}$$

neglecting second derivatives and using the fact that the covariance matrix of the track parameters, C, is

$$C = \frac{1}{2}\left(\frac{\mathrm{d}^2\chi^2}{\mathrm{d}\boldsymbol{\pi}^2}\right)^{-1} = \left(H^T V^{-1} H\right)^{-1}, \tag{26}$$

the total derivative operator with respect to $\boldsymbol{\alpha}$ can be rewritten as:

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\alpha}} = \frac{\partial}{\partial\boldsymbol{\alpha}} - A^T V^{-1} H C \frac{\partial}{\partial\boldsymbol{\pi}}. \tag{27}$$

The first and second derivative of the global χ^2 with respect to $\boldsymbol{\alpha}$ are then:

$$\frac{\mathrm{d}\chi^2}{\mathrm{d}\boldsymbol{\alpha}} = 2\sum_{\text{tracks}} A^T V^{-1}(V - HCH^T)V^{-1}\boldsymbol{\rho} \tag{28}$$

$$\frac{\mathrm{d}^2\chi^2}{\mathrm{d}\boldsymbol{\alpha}^2} = 2\sum_{\text{tracks}} A^T V^{-1}(V - HCH^T)V^{-1}A. \tag{29}$$

Here the term HCH^T represents the covariance of the track parameters in measurement space. The matrix given by

$$R = V - HCH^T \tag{30}$$

is the covariance matrix of the residuals of the track fit.

2.1.5 Newton-Raphson method for Global χ^2 alignment

As with the method for track fitting described in Section 2.1.1, an iterative approach is used to solve for the alignment parameters. The first and second derivatives are obtained using Eqs. 28 and 29 evaluated for the initial set of alignment parameters, $\boldsymbol{\alpha}_0$, giving the solution for $\Delta\boldsymbol{\alpha}_0$:

$$\Delta\boldsymbol{\alpha}_0 = -\left(\left.\frac{\mathrm{d}^2\chi^2}{\mathrm{d}\boldsymbol{\alpha}^2}\right|_{\boldsymbol{\alpha}_0}\right)^{-1}\left.\frac{\mathrm{d}\chi^2}{\mathrm{d}\boldsymbol{\alpha}}\right|_{\boldsymbol{\alpha}_0}. \tag{31}$$

This is repeated for successive iterations until the $\Delta\boldsymbol{\alpha}$ is negligible.

2.1.6 Locality ansatz

If the initial track parameters, $\boldsymbol{\pi}_0$, are those that minimize the track χ^2 for the initial set of alignment parameters, then $H^T V^{-1}\boldsymbol{\rho}$ is zero, and the first term in Eq. (28) simplifies to:

$$\left.\frac{\mathrm{d}\chi^2}{\mathrm{d}\boldsymbol{\alpha}}\right|_{\boldsymbol{\pi}_o} = 2\sum_{\text{tracks}} A^T V^{-1}\boldsymbol{\rho}. \tag{32}$$

Thus only residuals r_i for which $A = \partial r_i/\partial \boldsymbol{\alpha}$ are non-zero contribute to the first derivative, and all other residuals and their derivatives do not. In particular, contributions to the χ^2 from chambers not being aligned, multiple scattering, and energy loss in the calorimeter can all be ignored. This useful property dubbed the *locality ansatz* was first defined in [6] and provides an important simplification for the software implementation.

2.1.7 Adding constraints on track parameters

It is of particular importance for alignment to be able to constrain track parameters in order to eliminate unwanted biases [11]. This can be included naturally in the global χ^2 method by adding extra terms to the expression for the χ^2 in Eq. 21. For an individual track one has:

$$\chi^2_{\text{cons}} = r^T V^{-1} \boldsymbol{\rho} + (\boldsymbol{\pi} - \boldsymbol{q})^T T^{-1} (\boldsymbol{\pi} - \boldsymbol{q}), \tag{33}$$

where $\boldsymbol{q}$ is a vector defining the constraint on the track parameters ($\boldsymbol{\pi}$) and T is its covariance matrix. The additional constraint leads to the modified 14 and 15 expressions:

$$\frac{1}{2}\frac{\mathrm{d}\chi^2}{\mathrm{d}\boldsymbol{\pi}}_{\text{cons}} = H^T V^{-1} \boldsymbol{\rho} + T^{-1}(\boldsymbol{\pi} - \boldsymbol{q}), \tag{34}$$

$$\frac{1}{2}\frac{\mathrm{d}^2\chi^2}{\mathrm{d}\boldsymbol{\pi}^2}_{\text{cons}} = H^T V^{-1} H + T^{-1}. \tag{35}$$

Within the ATLAS tracking model the above is realized by adding a pseudo-measurement on a track [8]. The solution for the alignment parameters ($\boldsymbol{\alpha}$) is given by Eq. 31 where for each constrained track:

$$C = \frac{1}{2}\left(\frac{\mathrm{d}^2\chi^2}{\mathrm{d}\boldsymbol{\pi}^2}_{\text{cons}}\right)^{-1} \tag{36}$$

is consequently used and the first derivative of the global χ^2 reads:

$$\left.\frac{\mathrm{d}\chi^2}{\mathrm{d}\boldsymbol{\alpha}}\right|_{\alpha_0} = 2\sum_{\text{tracks}} A^T V^{-1}(V - HCH^T)V^{-1}\boldsymbol{\rho}(\boldsymbol{\alpha}_0) - A^T V^{-1} H C T^{-1}(\boldsymbol{\pi}(\boldsymbol{\alpha}_0) - q). \tag{37}$$

It is worth noting at this point, that if tracks have been refitted with the imposed constraint the locality ansatz fantastically simplifies Eq. 37. Vanishing expression 34 leads to Eq. 37 simplifying to Eq. 32. This property is used in the ATLAS implementation.

2.1.8 Vertex fitting with the reduced track model

The χ^2 fit of the alignment parameters can be extended to require a common origin for a group of tracks stemming from a common interaction vertex. This could be done using conventional vertex fitting algorithms as proposed in [12] but it would require retaining the full correlation matrix between all tracks concerned. Such a functionality is not available from the standard vertex fitters implemented for ATLAS. This is why an alternative approach that is particularly suitable for that purpose has been developed. It consists of a common fit of all vertexed tracks under the explicit assumption they share a common origin. In order to achieve that a new reduced track model is needed. Each constituent track is thus defined by only three perigee parameters $\boldsymbol{\pi} = (\phi_0,\ \theta_0,\ Q/p)$. Impact parameters of contributing tracks are reduced to a set of common vertex parameters ($\boldsymbol{b} = (x_b, y_b, z_b)$). Residuals of the measurements on track explicitly depend on three parameters:

$$\boldsymbol{r} \equiv \boldsymbol{r}(\boldsymbol{\pi}, \boldsymbol{b}, \boldsymbol{\alpha}). \tag{38}$$

The generic solution from Eq. 31 still holds, however, the full derivative takes a more complicated form:

$$\begin{aligned}
\frac{d}{d\boldsymbol{\alpha}} &= \frac{\partial}{\partial\boldsymbol{\alpha}} + \frac{\partial}{\partial\boldsymbol{\pi}}\frac{\mathrm{d}\boldsymbol{\pi}}{\mathrm{d}\boldsymbol{\alpha}} + \frac{\partial}{\partial\boldsymbol{b}}\frac{\mathrm{d}\boldsymbol{b}}{\mathrm{d}\boldsymbol{\alpha}}, \\
\frac{\mathrm{d}\boldsymbol{\pi}}{\mathrm{d}\boldsymbol{\alpha}} &= -CH^T V^{-1}(A + F\frac{\mathrm{d}\boldsymbol{b}}{\mathrm{d}\boldsymbol{\alpha}}), \\
\frac{\mathrm{d}\boldsymbol{b}}{\mathrm{d}\boldsymbol{\alpha}} &= -\underbrace{\left(\sum_{\text{tracks}}^{\text{vtx}} F^T W F\right)}_{M_b}{}^{-1}\left(\sum_{\text{tracks}}^{\text{vtx}} F^T W A\right),
\end{aligned} \tag{39}$$

where $F \equiv \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{b}}$ were additionally defined. Despite the above complexity, the final solution can be written as:

$$\delta\boldsymbol{\alpha} = -\mathcal{M}^{-1}\mathcal{V} \tag{40}$$

where the matrix $\mathcal{M}$ and the vector $\mathcal{V}$ take the new form:

$$\mathcal{M} = 2\sum_{\text{tracks}}(A^T W A) - 2\sum_{\text{vtx}}\left(\sum_{\text{tracks}}^{\text{vtx}}(A^T W F)M_b^{-1}\sum_{\text{tracks}}^{\text{vtx}}(F^T W A)\right) \tag{41}$$

$$\mathcal{V} = 2\sum_{\text{tracks}}(A^T W \boldsymbol{\rho}_0) - 2\sum_{\text{tracks}}\left(\sum_{\text{tracks}}^{\text{vtx}}(A^T W F)M_b^{-1}F^T W\boldsymbol{\rho}_0\right) \tag{42}$$

and we used:

$$W \equiv V^{-1}\left(V - HCH^T\right)V^{-1}. \tag{43}$$

It is important to note that, before the expressions 41 and 42 can be used for the alignment fit, the tracks concerned for the vertex fit (together with their residuals) need to be redefined in such a way that they do originate from a common vertex seed and the remaining three perigee parameters are defined with respect to it. In particular the track angles (ϕ_0 & θ_0) denote rotations around the vertex seed position.

2.1.9 Vertex fitting with the locality ansatz

After the above redefinition the tracks will not, in general, remain in the χ^2 minimum of their individual fits. This will not affect the alignment fit as Eq. 42 makes no implicit assumptions about the χ^2 of the input tracks. However, in order to ensure the locality ansatz is valid (Section 2.1.6), one needs to refit tracks using the new, reduced parametrization. The standard track fitter used in ATLAS [13] does not offer such a functionality. However, the reference perigee point can be chosen in any arbitrary way and a pseudo-measurement can be added to the collection of real measurements associated with the track [8]. Consequently, the fitter can force the trajectory through a chosen space-point with an arbitrary accuracy. This is achieved by setting the new perigee in the vertex seed position and constraining the two impact parameters (d_0 & z_0) to be zero. This renders both the residuals and their derivatives with respect to track parameters (H) in agreement with the new formalism. The reduced covariance matrix C_{R} can be obtained from the standard one:

$$C_{\text{R}} = \begin{pmatrix} & & \\ & (C^{-1})_{n-2} & \\ & & \end{pmatrix}^{-1} \tag{44}$$

following the observation that the additional pseudo-measurement does not affect the sub-matrix $(C^{-1})_{n-2}$. With the above conditions satisfied Eq. 42 is simplified to:

$$\mathcal{V}_{\text{LTRK}} = 2\sum_{\text{tracks}}(A^T V^{-1}\boldsymbol{\rho}_0) - 2\sum_{\text{tracks}}\left(\sum_{\text{tracks}}^{\text{vtx}}(A^T W F)M_b^{-1}F^T V^{-1}\boldsymbol{\rho}_0\right) \tag{45}$$

and more importantly maintains the simplicity of local implementation.

It is worth noting that the locality ansatz can be promoted to the level of the vertex fit itself, thus reducing equation 42 to a very basic form:

$$\mathcal{V}_{\mathrm{LVTX}} = 2 \sum_{\mathrm{tracks}} (A^T V^{-1} \boldsymbol{\rho}_0). \tag{46}$$

Although tempting, this option does not bring in practice any simplification to the algorithm and additionally requires extra step in the loop, namely a vertex refit followed by one more iterations of the track fit in order to recover locality at the track level. Altogether, this presents no advantage from the algorithmic point of view. For that reason, the ATLAS implementation adopted the track locality paradigm alone and realizes algorithmically Eq. 45.

2.1.10 Adding a constraint on the vertex position

The fitted common vertex can be further constrained to an arbitrary position, e.g. to be compatible with the beam interaction region. This can be realised in a standard way, in analogy to adding constraints on track parameters described in Section 2.1.7. If the vertex position constraint is described by the three-component verctor $\mathbf{v}$ and its covaraince T, the first derivative vector and the second derivative matrix of Eqs. 41 and 42 are incremented by the extra terms:

$$\mathcal{M} \longrightarrow \mathcal{M} + \frac{\mathrm{d}\boldsymbol{b}}{\mathrm{d}\boldsymbol{\alpha}}^T T^{-1} \frac{\mathrm{d}\boldsymbol{b}}{\mathrm{d}\boldsymbol{\alpha}} \tag{47}$$

$$\mathcal{V} \longrightarrow \mathcal{V} + \frac{\mathrm{d}\boldsymbol{b}}{\mathrm{d}\boldsymbol{\alpha}}^T T^{-1} (\boldsymbol{b}_0 - \mathbf{v}), \tag{48}$$

where $\boldsymbol{b}_0$ is the initial estimate of the vertex position and the full derivative $\frac{\mathrm{d}\boldsymbol{b}}{\mathrm{d}\boldsymbol{\alpha}}$ is given in the Eq. 40.

2.1.11 Local χ^2 alignment

The power of the Global χ^2 method derivesis from its rigorous treatment of the correlations between alignable objects through the tracks connecting them. However, this approach becomes technically unfeasible for very large number of degrees of freedom (DoF). This is the case e.g. for the alignment of individual TRT straws, which involves $\approx 700,000$ parameters. To deal with such problems a simplified version of the χ^2 approach, dubbed the Local χ^2, has been put in place. It is based on minimization of the same χ^2 of Eq. 21, however, the implicit dependence on the fitted track parameters is dropped, which reduces eq. 22 to a simple form:

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\alpha}} = \frac{\partial}{\partial \boldsymbol{\alpha}}. \tag{49}$$

Consequently, Eq. 28 and Eq. 29 take a much simpler form:

$$\frac{\mathrm{d}\chi^2}{\mathrm{d}\boldsymbol{\alpha}} = 2 \sum_{\mathrm{tracks}} A^T \Omega^{-1} \boldsymbol{r} \tag{50}$$

$$\frac{\mathrm{d}^2\chi^2}{\mathrm{d}\boldsymbol{\alpha}^2} = 2 \sum_{\mathrm{tracks}} A^T \Omega^{-1} A. \tag{51}$$

and more importantly the problem breaks down to separate systems of equations describing individual alignable modules. The method loses part of its potential but at the same time eliminates all numerical challenges. Only many small systems of up to 6 parameters need to be solved. Moreover, besides pathologies, these systems are guaranteed to be positive definite and directly solvable.

3 Alignment Framework

The common alignment framework was developed in such a way that it could easily be used for various types of alignment. It was developed specifically for the global χ^2 and the local χ^2 alignment approaches. A single algorithm, `AlignAlg`, can be used for all types of alignment. `AlignAlg` is configured by means of jobOptions with tools to perform functions specific to various alignment approaches with jobOptions.

Several Event Data Model (EDM) objects containing information necessary for alignment are used in the common alignment framework. Whenever possible, they were derived from objects used in the standard ATLAS tracking EDM [8]. These are described here:

- `AlignModule`: a collection of detector elements, specifically `TrkDetElementBase` objects, grouped together to be aligned as a single body. Each `AlignModule` has its own global to alignment frame transform, as well as transforms to go from each detector element alignment frame to the `AlignModule` frame.
- `AlignTSOS`: an extension of `Trk::TrackStateOnSurface`, containing a pointer to the `AlignModule` to which it belongs, the type of measurement, and either the `RIO_OnTrack` or `CompetingRIOsOnTrack` belonging to the track.
- `AlignTrack`: an extension of `Trk::Track`, contains a vector of `AlignTSOS`, and the full covariance and derivative matrices if needed for alignment.
- `AlignPar`: a class containing the initial alignment parameter, the change in alignment parameter found by the alignment algorithm, and the final alignment parameter. It can be used to store the alignment information for either an `AlignModule` or a detector element, so it contains a pointer to either object.
- `AlignVertex`: a class containing all objects describing a common vertex including its position, covariance matrix, derivatives, pointers to all contributing tracks and optionally information on the constraint on its position. Appart from the default empty constructor, it contains a constructor taking a `VxCandidate` as an input. The main method of the `AlignVertex` class is `AlignVertex::fitVertex` which finds the vertex position and its covariance matrix from the contributing tracks. This method has to be executed before the vertex is used for constrained alignment.

As illustrated in Figure 1, alignment is done in the following steps:

1. Build Geometry. Here the `AlignModule` objects are defined, along with the alignment frames of reference, and transforms to go from the local and/or global frames to the alignment frames of reference. In addition, the degrees of freedom of the `AlignModule` objects to be aligned are defined. This is implemented as a tool inheriting from the `IGeometryManagerTool`.
2. Process Events.
   (a) Process Track Collection. Before processing individual tracks, an initial processing is done of the track collection. Any initial track or hit selection with track refitting is done in this stage. This is implemented as a tool inheriting from `IAlignTrackPreProcessor`.
   (b) Create `AlignTSOS` and store on `AlignTrack`. Loops over hits on track, creating `AlignTSOS` for each hit that is in an AlignModule, and stores the collection of `AlignTSOS` on `AlignTrack`. Even though the `AlignTrack` is created by `IAlignTrackPreProcessor`, the tool interface used for creating the `AlignTSOS` collection is (for historical reasons) named `IAlignTrackCreator`.

(c) Dress AlignTrack. Calculate additional information needed for alignment and store on `AlignTrack`. The tool interface is `IAlignTrackDresser`.

(d) Accumulate for the common vertex fit. In the `accumulateVTX` method, implemented by `IAlignTrackPreProcessor`, objects needed for the vertex fit are incremented by the contributing track information.

(e) Solve for the vertices accumulated in the previous step. This is done in the `solveVTX` method, implemented by `IAlignTrackPreProcessor`.

(f) Accumulate. More calculations are done if necessary and combined with information from earlier tracks. The tool interface used for accumulation is `IAlignTool`. This step is executed in a separate loop over `AlignTrack`s as the vertex fitting step has to be completed beforehand.

3. Solve. Once information from all tracks has been accumulated, the solving for alignment parameters is done by `IAlignTool`.

4. Process Alignment Constants. Alignment constants are written to databases by detector-specific tools inheriting from `ITrkAlignDBTool`.

More information about the concrete implementations of the interfaces described above and others are given in the following sections.

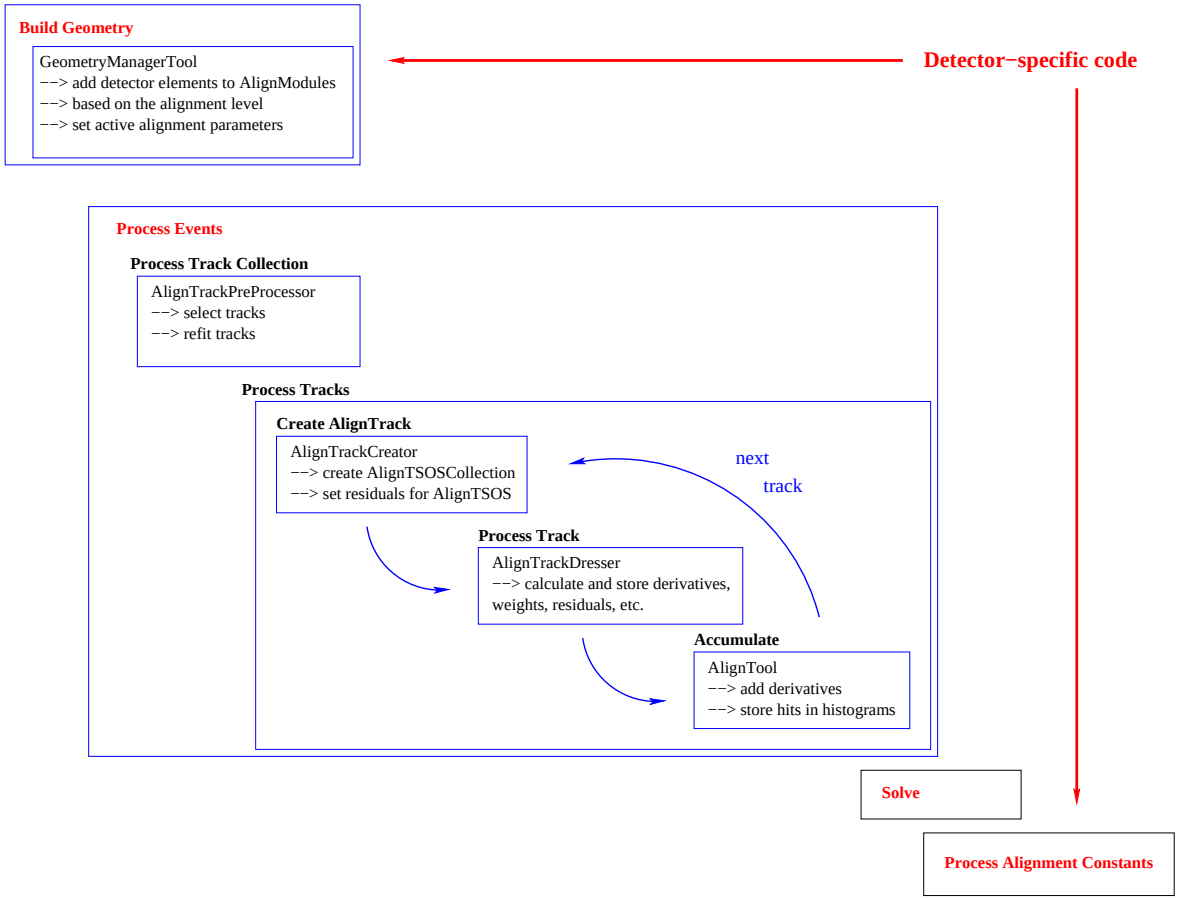

Figure 1: Overview of Common Alignment Framework

3.1 Building Geometry

Building the geometry is specific to the type of detector being aligned, so there is no implementation of `IGeometryManagerTool` in `TrkAlignGenTools`.

The tool inheriting from `IGeometryManagerTool` must implement the `ReadGeometry` method, the main method called by `AlignAlg`. It does the following:

- Defines the collection of detector elements in the `AlignModule` objects.
- Defines the parameters to be aligned.
- Defines the necessary transforms for the various frames of reference.

It returns the total number of degrees of freedom, with each degree of freedom being an alignment parameter for an `AlignModule`.

`IGeometryManagerTool` contains a pointer to an ntuple, which can be created by `AlignAlg` and set by `IGeometryManagerTool::setNtuple` method. This is for debugging and validation purposes only.

The geometry is created by a geometry manager tool, and is stored in a tool inheriting from `TrkAlignGenTools/AlignModuleTool`. The tool is also used to access the geometry information. `TrkAlignGenTools/AlignModuleTool` is a concrete implementation of `IAlignModuleTool` that is not detector-specific, and each implementation of `IGeometryManagerTool` should make use of one or more tools that inherit from `AlignModuleTool`.

The geometry manager tool creates the following and stores in its align module tool(s):

- A list of `AlignModule` objects, stored as a vector of vector of `AlignModule` objects for each detector type.
- A mapping for each type of detector from the detector element IdentifierHash (returned by `Trk::TrkDetElementBase::identifyHash()` method) to the `AlignModule`. This is used to make the identification of the `AlignModule` to which a hit on the track belongs much faster.
- A list of all the alignment parameters (`AlignPar` objects) for each `AlignModule`.
- A separate list of the alignment parameters for each `AlignModule` that are to be aligned.

The lists of alignment parameters are stored as 2-dimensional lists, with a vector of alignment parameters being stored for each `AlignModule` in an outer vector of `AlignModule` objects. A 1-dimensional vector is also used to simplify its use by the matrix tool. The list of `AlignModule` objects is also available as a 1-dimensional vector for the same reason.

3.2 Initial Track Processing (before Vertex Refit)

The tool inheriting from `IAlignTrackPreProcessor` can do any or all of the following:

1. Apply track selection to remove tracks.
2. Remove hits from tracks.
3. Refit tracks with or without a primary vertex constraint.
4. Create the `AlignTrack`, an extension of the `Trk::Track` EDM object.

5. If the track is refitted, store the full covariance and derivative matrices on the AlignTrack.

It is worthwhile to note that the ability to refit tracks with a primary vertex constraint is the main reason for the initial processing of the track collection before the processing done by IAlignTrackCreator.

TrkAlignGenTools/AlignTrackPreProcessor is a specific tool inheriting from IAlignTrackPreProcessor. It can be used for any type of alignment. It is configured by jobOptions with the following tools:

- TrackFitterTool: inherits from IGlobalTrackFitter, used to refit a track. The default fitter is Trk::GlobalChi2Fitter/MCTBFitter. IGlobalTrackFitter is used because it has the FullCovarianceMatrix and DerivMatrix methods.
- SLTrackFitterTool: inherits from IGlobalTrackFitter, optional fitter used to fit straight tracks. Needed for refitting straight muon tracks. If UseSingleFitter is set to True, the SLTrackFitterTool will not be used.
- TrackSelectorTool: inherits from ITrackSelectorTool, optional tool used to select tracks. Used if SelectTracks is True.

The method processTrackCollection is called in the execute method of the AlignAlg algorithm. The collection of Trk::Track objects is passed to AlignTrackPreProcessor in this method. For each track in the collection, the following is done:

1. The track is selected or rejected by TrackSelectorTool.
2. If the track passes selection, the track is refit by the appropriate track fitter.
3. If the track refit succeeds, an AlignTrack is created with no AlignTSOSCollection. The covariance and derivative matrices are set on the AlignTrack.
4. The AlignTrack is pushed back on the collection of AlignTrack objects.

The collection of AlignTrack objects is returned to AlignAlg.

3.3 Creation of AlignTSOSCollection

The tool inheriting from IAlignTrackCreator has the following functions and capabilities:

1. Create collection of AlignTSOS objects. The AlignTrack is created by the IAlignTrackPreProcessor in the previous step, but it contains only pointers to the fit matrices.
2. Select tracks passing through AlignModule objects.
3. Select hits to be used to create AlignTSOS objects. Note that hits are not removed from the track by AlignTrackCreator.
4. Store run and event numbers with selected tracks in a good event list, to be used for selection in later iterations.
5. Set residuals on AlignTSOS objects.

IAlignTrackCreator::processAlignTrack is called by AlignAlg for each track in the collection of AlignTrack objects returned by IAlignTrackPreProcessor::processTrackCollection. If the track passes the selection criteria, IAlignTrackCreator::processAlignTrack returns true.

TrkAlignGenTools/AlignTrackCreator is a specific tool inheriting from IAlignTrackCreator. It is configured by jobOptions with the following tools:

- ResidualCalculator: a tool used to calculate residuals for hits and scatterers on the track. ResidualCalculator is not used directly since it does not calculate residuals for scatterers.
- AlignModuleTool: described in Section 3.1. It is used to identify the AlignModule to which a hit on a track belongs.

3.4 AlignTrack Dressing

The tool inheriting from IAlignTrackDresser makes calculations for each track and stores the results of the calculations on the AlignTrack. The method dressAlignTrack is called by AlignAlg after processing by IAlignTrackCreator.

The calculations that need to be done depend on what type of alignment is being performed. For local and global χ^2 alignment, first and second derivatives of residuals with respect to alignment parameters are calculated. These are implemented with the TrkAlignGenTools/AlignTrackDresser tool, described below.

3.5 AlignTrackDresser with Derivatives

TrkAlignGenTools/AlignTrackDresser is the implementation of IAlignTrackDresser specific to local and global χ^2 alignment. It is configured by jobOptions to have an implementation of IDerivCalcTool specific to the type of derivatives being calculated. The currently used default derivative calculator tool, TrkAlignGenTools/AnalyticalDerivCalcTool is described in Appendix A.

In the method AlignTrackDresser::dressAlignTrack, the following is done:

- The residuals for all AlignTSOS on the AlignTrack are retrieved from the AlignTSOS (set by AlignTrackCreator) and stored in a vector on AlignTrack. This is done because the residuals are needed for calculations by GlobalChi2AlignTool (described in Section 3.8).
- The method IDerivCalcTool::setDerivatives is called, which calculates the first derivatives of hit residuals with respect to alignment parameters on the AlignTrack.
- The method IDerivCalcTool::setResidualCovMatrix is called, which calculates the residual covariance matrix (see Section 2.1).
- If the calculations are done successfully, then true is returned and the track and AlignAlg continues with processing. Otherwise, the track is skipped.

3.6 Accumulating for fitting of the vertices

TrkAlignGenTools/BeamSpotVertexPreProcessor is the current default implementation of IAlignTrackPreProcessor. Appart from the functionalities described earlier, it also implements methods needed for the accumulation and fitting of the common vertices implemented by the AlignVertex class. In the first loop over AlignTracks, after track dressing, the

IAlignTrackPreProcessor::accumulateVTX method is called by the IAlignAlg. The objects needed for the vertex fit as well as auxilliary objects needed for the vertex-constrained alignment are incremented by the information from the current track. In the current implementation vertex candidates are identified as primary vertices from the event record and selected in the BeamSpotVertexPreProcessor::processTrackCollection.

3.7 Fitting the vertices

The IAlignTrackPreProcessor::solveVTX method is called by the IAlignAlg just after the first loop over AlignTracks. It loops over all accumulated AlignVertex's and invokes the AlignVertex::fitVertex for all vertices containing more than one track.

3.8 Accumulation of Data

The tool inheriting from IAlignTool is responsible, in part, for accumulating the information calculated by IAlignTrackDresser for each track. The method accumulate is called by AlignAlg after the IAlignTrackDresser::dressAlignTrack has been called. In accumulate the final algebra objects which are used to solve for alignment corrections are incremented. In particular, for the Global χ^2 method, the contributions to the first and the second derivatives of the global χ^2, as defined in Eqs. 28 and 29 are calculated. This can be extended to the full vertex constraint described by Eqs. 41 and 45, if the option has been configured in IAlignTrackPreProcessor. Then, they are added to the vector of first derivatives and the matrix of second derivatives implemented by the TrkAlignMatrixTool.

3.9 Solving

When solving a linear system of equations the properties of the matrix determine which solution technique should be used. As described earlier, the matrix produced during alignment is symmetric and, without constraints, singular. The various constraint techniques will usually render the matrix positive definite. Practical experience shows that without any preconditioning the matrix will also be poorly conditioned. This is due to the existence of so-called "weak modes" of the solution, which correspond to geometry deformations leaving the global χ^2 of the system virtually unchanged. More specifically, weak modes transform helical trajectories of particles into other helical trajectories, possibly altering their reconstructed kinematic parameters.

Finding a solution to a system with a large number of DoF is both computationally intensive and memory intensive. The size and condition of the matrix may also require machine precision to be taken into account when choosing a solution technique.

The interface to the different solvers is realized by the TrkAlignMatrixTool and the TrkAlignAlgebraUtils.

3.9.1 Diagonalisation of the matrix

By far the best control of the solution can be obtained using diagonalisation, i.e. transforming the the whole system to its diagonal basis where all the parameters (directions) are linearly independent. Mathematically the transformation is given by the unitary rotation matrix $\mathcal{U}$:

$$\mathcal{M}X = Y \quad \Longrightarrow \quad \mathcal{U}\mathcal{M}\mathcal{U}^T\mathcal{U}X = \mathcal{U}Y \quad \Longrightarrow \quad \mathcal{D}X_D = Y_D, \tag{52}$$

where $\mathcal{D}$ is a diagonal matrix containing the eigenvalues of $\mathcal{M}$ (λ_i) on its diagonal. In this basis solution for each direction can be extracted independently. They are given by:

$$X_D^i = \frac{1}{\lambda_i} Y_D^i \quad \text{with} \quad \sigma(X_D^i) = \frac{1}{\sqrt{\lambda_i}} \tag{53}$$

Clearly, solution to the singular modes cannot be determined as their eigenvalues are zero, while the weak modes can have an arbitrarily large associated uncertainty, given by the square root of the reciprocal of their eigenvalue. Such modes must be excluded from the solution.

There are a number of software packages that are able to perform the diagonalisation of a large matrix. LAPACK's `DSPEV` [14] has been chosen as the baseline for the ATLAS implementation. Alternative options based on ROOT [15] linear algebra classes and CLHEP [16] are also available.

The computation time for diagonalisation in general scales as $O(DoF^3)$. Solving for very large systems soon becomes untenable on a single machine. Similarly, even with 64-bit words the numerical precision limits solutions for problems exceeding ≈10,000.

3.9.2 Direct solving

Even for very large problems, as long as the matrix remains sparse, direct solvers offer an accurate and cpu-efficient workaround. Also, as they do not invert or diagonalise the matrix they require much less memory than full diagonalisation.

The `MA27` [17] direct solver was implemented for the ATLAS alignment as it is freeware and found to be sufficiently performant at solving problems of the size ATLAS alignment. It has been shown that using such a procedure on a 35000×35000 matrix takes less than 10 minutes.

As direct solution does not offer the possibility of analysing and eliminating unwanted eigenmodes, other preconditioning techniques must be used in order to extract a meaningful solution.

3.9.3 The Soft Mode Cut

Weak modes must always be removed from the solution. The problem is aggravated when the system becomes too large to use diagonalisation in order to explicitly analyze the eigen-pulls. For such systems direct fast solvers are usually applied. That means that the system must be pre-conditioned prior to solving. A straightforward way dubbed the "soft-mode-cut" has been proposed for ATLAS [11]. The method benefits from the typically exponential nature of the eigen-spectrum and consists of incrementing diagonal elements of the matrix $\mathcal{M}$.

Cut-off in the diagonal basis

In the simplest case a common value is added and eigen-modes of the system remain unchanged. Only the eigen-values are incremented by the added quantity:

$$\mathcal{M}X = Y, \quad \mathcal{U}\mathcal{M}\mathcal{U}^T\mathcal{U}X = \mathcal{U}Y \;\; \Rightarrow \;\; \mathcal{D}X_D = Y_D \tag{54}$$

$$\mathcal{M} \to \mathcal{M} + \kappa\mathbf{1}, \quad \mathcal{D} \to \mathcal{D} + \kappa\mathbf{1}, \quad \lambda_i \to \lambda_i + \kappa. \tag{55}$$

As a result, all corrections get suppressed by the factor $\lambda_i/(\lambda_i + \kappa)$ that diverges significantly from one for $\lambda_i \ll \kappa$. As can be seen this simple operation provides an effective suppression of weak eigen-modes of the solution. Needless to say, the singularity of the matrix is removed by the preconditioning.

Cut-off in the natural basis

Turning again to the preconditioning described in Eq. 54, the addition of the constant term κ to the diagonal of $\mathcal{M}$ is equivalent to constraining the solution in the natural basis to its current value with $\sigma = 1/\sqrt{\kappa}$.

Recalling:

$$\mathcal{M}_{ij} = \frac{d\chi^2}{d\alpha_i d\alpha_j}, \qquad Y_i = \frac{d\chi^2}{d\alpha_i}, \tag{56}$$

the alignment may now be reparameterized by replacing the variables a by the ones normalized to their requested error:

$$\mathcal{M}_{ij} = \frac{d\chi^2}{d\frac{\alpha}{\sigma}_i d\frac{\alpha}{\sigma}_j}, \qquad \mathcal{M}_{ij} \longrightarrow \sigma(\alpha_i)\sigma(\alpha_j)\mathcal{M}_{ij} \tag{57}$$

Such a change of variables, of course, cannot change the solution to the χ^2 problem provided that the normalized variables get unfolded in the solution. However, if we add a unit matrix to the matrix $\mathcal{M}$ we effectively constrain each DoF to one sigma of its assumed uncertainty. Further, that is equivalent to accepting in the solution only those diagonal modes that lead to uncertainties on the derived alignment corrections not exceeding the assumed σ values.

At closer inspection, there is no need to actually change variables and redefine $\mathcal{M}$ and Y objects. The pre-conditioning gives:

$$(\sigma(\alpha_i)\sigma(\alpha_j)\mathcal{M}_{ij} + \mathbf{1})\frac{X_j}{\sigma(\alpha_j)} = \sigma(\alpha_i)Y_i \quad \Longrightarrow \quad (\mathcal{M}_{ij} + \mathcal{D}(\frac{1}{\sigma(\alpha_i)^2}))X_j = Y_i \tag{58}$$

Here, $\mathcal{D}(\frac{1}{\sigma(\alpha_i)^2})$ denotes a diagonal matrix with $\frac{1}{\sigma(\alpha_i)^2}$ elements on the diagonal. In summary, the scaled soft mode cut is realized by adding the diagonal matrix $\mathcal{D}$ to the original matrix $\mathcal{M}$.

This method proved very powerful in controlling solutions of large problems where the use of explicit diagonalisation is not feasible but one still needs to control the uncertainties on the extracted corrections. However, the error on the parameter corrections is, by construction, contained within the assumed σ values[1].

4 Implementation for Inner Detector Alignment

The Inner Detector (ID) [2] is the main tracking device of ATLAS. It is composed of two silicon subsystems: the Pixels and the SCT complemented by the gaseous drift straw tube system, the TRT. Other than the mechanical survey information from the assembly, the alignment of the ID relies entirely on the track-based alignment.

As a very complex system its alignment involves very large number of degrees of freedom additionally distributed over mechanically different devices offering fairly different measurement accuracy [18]. The ID adopted the global χ^2 alignment method as its main strategy. However to align $\approx 700,000$ degrees of freedom of the TRT straw tubes the local χ^2 method (see 2.1.11) is used.

In order to better reflect the mechanical structure of the ID and the expected magnitudes of misalignment, the following alignment levels and corresponding types of alignment modules have been defined:

- Level 0: each of the three ID subsystems (Pixels, SCT, TRT) is aligned as a rigid body.
- Level 1: the barrel and each endcap of the three ID subsystems, are aligned as separate rigid bodies.
- Level 2: the barrel cylinders and the endcap disks of the three silicon detectors as well as the barrel modules and the endcap wheels of the TRT are aligned as rigid bodies.
- Level 3: all of the silicon modules and TRT straws are aligned individually.

Importantly, alignment levels of the three subsystems can be combined in an arbitrary way.

[1] Note, this does not put any limit on the derived corrections themselves.

4.1 Software Implementation

The ID AlignModule's are created prior to data accumulation by the InDetGeometryManagerTool which implements the IGeometryManagerTool. The earlier benefits from the hierarchical underlying structure. Individual subsystems are managed by their respective tools: PixelGeometryManagerTool, SCTGeometryManagerTool and the TRTGeometryManagerTool. Additionally, the SiGeometryManagerTool deals with silicon-specific geometry setup. Each tool manages the different options for the alignment such as which degrees of freedoms are switched on or off, the softcuts for different degrees of freedom and the alignment level.

4.1.1 Geometry setup

The InDetAlignModuleTool is a tool derived from Trk::AlignModule that implements the method for returning a sub-TrkDetElementBase structure identifier Hash. This is only needed for TRT where the structure is a single straw and allows to set detector specific properties of AlignTSOS. Fan-out angle is set here for SCT measurements in the endcaps. The currently implemented geometry ManagerTools are:

- PixelGeometryManagerTool: the tool used to manage the modules of the pixel subdetector. This tool allows to select the alignment level for the Pixel:
  - Level 1: All pixel subdetector as one structure.
  - Level 1.5: Pixel barrel split in two parts, top and bottom. Endcaps are aligned as one structure each one.
  - Level 2: Pixel barrel layers and endcap disks.
  - Level 2.2: Each Pixel barrel layer is split into two half-shells.
  - Level 2.7: Considers the Pixel barrel staves as alignable structures.
  - Level 3: All Pixel modules are aligned.
- SCTGeometryManagerTool: the tool used to manage the modules of the SCT subdetector. This tool allows to select the alignment level for the SCT.
  - Level 1: Uses three structures, the SCT barrel and the two endcaps.
  - Level 2: SCT barrel layers and endcap disks.
  - Level 2.2: Each pixel barrel layer is splitted in two half-shells.
  - Level 2.5: The disks fo the endcaps are splitted in rings.
  - Level 2.7: Considered the SCT barrel staves as alignable structures.
  - Level 3: All SCT modules are aligned.
- TRTGeometryManagerTool: the tool used to manage the modules of the TRT subdetector. This tool allows to select the alignment level for the TRT.
  - Level 1: Uses 3 structures, the TRT barrel and the two endcaps.
  - Level 2: TRT barrel modules and endcap wheels.
  - Level 3: All TRT straws are aligned as independent structures.
- SiGeometryManagerTool: tool used to manage the silicon modules, it calls SCTGeometryManagerTool and PixelGeometryManagerTool.
- IndetGeometryManagerTool: tool used to manage the Inner Detector modules, it calls SiGeometryManagerTool and TRTGeometryManagerTool.

4.1.2 Database interface

There are three Database Tools; they are used to store the alignment corrections in the database. For the case of level 1, 2 and 3 the corrections are stored directly in the database. For the other intermediate levels (rings, staves, ...) the corrections need to be transformed to a level defined in the database and the tool transforms these corrections to level 3 corrections. There are three different tools:

- `SiTrkAlignDBTool`: manages the Pixel and SCT corrections.
- `TRTTrkAlignDBTool`: manages the TRT corrections.
- `InDetTrkAlignDBTool`: manages all the Inner Detector corrections, it calls the two previous tools.

5 Implementation for Muon Alignment

The muon spectrometer uses a number of methods to achieve the desired alignment. An optical alignment system is used to align most Monitored Drift Tube (MDT) chambers. While the optical alignment system is able to achieve the desired precision for most MDT chambers, there are a number of types of detectors that are not instrumented for optical alignment. For these chambers, muon tracks must be used for alignment. The chambers installed without optical alignment are the BEE chambers mounted on the endcap toroid housings, EEL chambers in sector 5, BIS8 chambers, and all small barrel chambers. Moreover, tracks can be used to align the MDT endcap relative to the MDT barrel, and to align the muon spectrometer relative to the inner detector. Lastly, track-based alignment provides an alternative in the event that optical alignment ceases to function for some MDT chambers, and also provides a valuable cross-check of the proper functioning of the optical alignment system.

To facilitate alignment of the muon spectrometer, the following types of alignment modules are defined:

- Level 0: the entire muon spectrometer is aligned as a rigid body.
- Level 1: the barrel and each endcap, including trigger chambers, are aligned as separate rigid bodies.
- Level 3: MDT, CSC, and/or TGC chambers are aligned individually.

The L0, L1, or L3 alignment modules are created

5.1 Software Implementation

EDM objects specific to muon alignment are defined in the `MuonAlignEvent` package. Tools specific to the implementation of the alignment framework for muon spectrometer alignment are contained in the `MuonAlignGenTools` package. These are described in the rest of this section.

5.1.1 MuonAlignEvent

The `CombinedMuonAlignModule` class is the implementation of `Trk::AlignModule` used to define groups of muon detectors for alignment. The implementation for muon chambers provides methods used to shift muon chambers by a small amount prior to refitting muon tracks. This is necessary for

determining numerical derivatives of χ^2 with respect to alignment parameters. The shifts are carried out by calling methods of the specific readout element classes of `MuonReadoutGeometry`.

`MdtAlignModule`, `CscAlignModule`, `TgcAlignModule`, and `RpcAlignModule` inherit from `CombinedMuonAlignModule` and exist to allow methods specific to muon subdetector types.

5.1.2 MuonAlignGenTools

The muon-specific implementation of `TrkAlignGenTools`, as well as other tools useful for muon alignment, are provided in the `MuonAlignGenTools` package. These are the following:

- `MuonGeometryManagerTool.` Implements `IGeometryManagerTool`.
- `MuonTrackPreProcessor`
- `MuonTrackCollectionProvider`
- `MuonAlignDBTool`

6 Summary

In the note, we presented the basics of the track-based software implementation in the ATLAS computing framework, Athena. The software is built in a modular and configurable manner allowing for efortless extensions and modifications. The alignment software is well integrated into the standard tracking realm of ATLAS reusing information, methods and data structures. The implementation presented is generic and has been used to align both the Inner Detector as well as the Muon System. Any combination of systems and subsystems may be configured for simultaneous alignment. The implementation clearly separates the core algorithm from the detector specific parts. The latter contain configuration of the specific geometry and definition of alignable parameters as well as storing of the derived alignment corrections in the database.

A Analytical Derivatives

The global χ^2 method for alignment described earlier in this document requires various partial derivatives being calculated. In its basic form the algorithm needs derivatives of track-hit residuals with respect to track parameters (H) and with respect to alignment parameters (A). Along the lines of the general philosophy of ATLAS the implementation of the alignment software attempts to maximally integrate with the general tracking realm in order to avoid duplications and incompatibilities. This is why it was chosen to use directly derivatives with respect to track parameters that are computed and used by the χ^2 based track fitter of ATLAS [10]. The derivative matrix H is imported directly from the fitter following the track refit. Derivatives with respect to alignment parameters have no precedence in the rest of ATLAS tracking software and hence must be calculated within the alignmet specific part, namely the `TrkAlignGenTools/AnalyticalDerivCalcTool`. The calculation is done purely analytically based on the known track direction at the intersection with the planar module (silicon wafers) or the closest approach to the cathode wire (TRT straw tubes) and the local degrees of freedom of the corresponding `AlignModule`. We define an auxiliary vector S:

$$\begin{aligned} S_x &= O^{\mathrm{LA}}_{xx} - Px^{\mathrm{loc}}/Pz^{\mathrm{loc}} * O^{\mathrm{LA}}_{xz} \\ S_y &= O^{\mathrm{LA}}_{yx} - Px^{\mathrm{loc}}/Pz^{\mathrm{loc}} * O^{\mathrm{LA}}_{yz} \\ S_z &= O^{\mathrm{LA}}_{zx} - Px^{\mathrm{loc}}/Pz^{\mathrm{loc}} * O^{\mathrm{LA}}_{zz} \end{aligned} \tag{59}$$

where Px^{loc} and Pz^{loc} are the track momentum components in the local frame of the measurement and O^{LA} is the rotation matrix from the local measurement frame to the reference frame of the alignable module. For example, the Inner Detector defines the local measurement frame in the following way:

1. Pixel Detector: x along the most sensitive measuring direction in the wafer plane (across the module); y along the orthogonal direction in the wafer plane (along the module); z perpendicular to the wafer plane.
2. SCT: x perpendicular to strips in the wafer plane; y along strips in the wafer plane; z perpendicular to the wafer plane.
3. TRT: x perpendicular to the track and the straw; y along the starw wire; z perpendicular to the wafer plane.

Additionally we define vector P_{ref} as the position of track-module itersection point in the reference frame of the alignable module. With the above definition and Eq. 59 the derivatives of the residual measured in the x direction w.r.t. alignment parameters read:

$$\begin{aligned} \partial r_x/\partial T &= S \\ \partial r_x/\partial R &= P_{\mathrm{ref}} \times S \end{aligned} \tag{60}$$

The global χ^2 vertex constrained fit requires additionally the partial derivatives of the residuals wih respect to the vertex position (F). These are also custom calculated by the `TrkAlignGenTools/AnalyticalDerivCalcTool` as a translation of the rigid track trajectory due to the shift of its origin. The vector of derivatives with respect to the three DoF's of the vertex position is given by:

$$\partial r_x/\partial b = -O^{\mathrm{AG}} S. \tag{61}$$

Here O^{AG} denotes the rotation matrix from the alignment to the global ATLAS frame.

References

[1] ATLAS Collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider*, JINST **3** (2008) S08003.

[2] ATLAS Collaboration, *ATLAS Inner Detector Technical Design Report*, vol. I. CERN-LHCC-1997-016, CERN, Geneva, 1997.

[3] ATLAS Collaboration, *ATLAS Muon Spectrometer Technical Design Report*, CERN-LHCC-1997-022, CERN, Geneva, 1997.

[4] P. Bruckman, S. Haywood, and A. Hicheur, *Global χ^2 approach to the alignment of the ATLAS silicon tracking detectors*, ATL-INDET-PUB-2005-002, 2005.

[5] S. Gonzalez-Sevilla and A. Hicheur, *Implementation of a global fit mehtod for the alignment of the silicon tracker in ATLAS ATHENA framework*, CHEP06 proceedings, Mumbai (India), 2006.

[6] A.Bocci and W.Hulsbergen, *TRT alignment for SR1 cosmics and beyond*, ATL-INDET-PUB-2007-009, 2007.

[7] ATLAS Collaboration, *ATLAS Computing Technical Design Report*, ATLAS-TDR-017, CERN-LHCC-2005-022, CERN, Geneva, 2005. See also `http://atlas-computing.web.cern.ch/atlas-computing/packages/athenaCore/athenaCore.php`.

[8] P. F. Akesson et al., *ATLAS Tracking Event Data Model*, ATL-SOFT-PUB-2006-004.

[9] K. Nakamura et al., *Review of Particle Physics*, Journal of Physics G **37** (2010) 1+.

[10] T. Cornelissen, *Track Fitting in the ATLAS Experiment*. PhD thesis, Amsterdam Univ., 2006. CERN-THESIS-2006-072.

[11] P. Bruckman, *Alignment of the ATLAS Inner Detector Tracking System - Solving the Problem*, Nuclear Physics B (Proceedings Supplements) **197** (2009) 158–162.

[12] W. Hulsbergen, *The Global covariance matrix of tracks fitted with a Kalman filter and an application in detector alignment.*, Nucl.Instrum.Meth. A **600** (2009) 471–477.

[13] T. Cornelissen et al., *The global χ^2 track fitter in ATLAS*, J. Phys.: Conf. Ser. **119** (2008) 032013.

[14] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third ed., 1999.

[15] Rene Brun and Fons Rademakers, *ROOT - An Object Oriented Data Analysis Framework*, Nucl. Inst. Meth. in Phys. Res. A **389** (1997) 81–86. See also `http://root.cern.ch/`.

[16] `http://proj-clhep.web.cern.ch/proj-clhep/`.

[17] I. Duff and J. Ried, *MA27–A set of Fortran subroutines for solving sparse symmetric sets of linear equations*, Her Majesty's Stationery Office, London (1982) .

[18] ATLAS Collaboration, *Alignment of the ATLAS Inner Detector Tracking System with 2010 LHC proton-proton collisions at $\sqrt{s}$ = 7 TeV*, Tech. Rep. ATLAS-CONF-2011-012, CERN, Geneva, Mar, 2011.