Chordal Based Error Function for 3-D Pose-Graph Optimization

Supplementary Material

This paper has been published on the IEEE Robotics and Automation Letters (RA-L).
Please cite this work as:

```
@article{aloise2019chordal,
title={Chordal Based Error Function for 3-D Pose-Graph Optimization},
author={Aloise, Irvin and Grisetti, Giorgio},
journal={IEEE Robotics and Automation Letters},
volume={5},
number={1},
pages={274--281},
year={2019},
publisher={IEEE}
}
```

Chordal Based Error Function for 3D Pose-Graph Optimization

Supplementary Material

Irvin Aloise and Giorgio Grisetti
Department of Computer, Control and Management Engineering
Sapienza University of Rome
Email: {ialoise,grisetti}@diag.uniroma1.it

I. SE(3) Mappings

In the remaining of this section, we will assume that a generic $\mathrm{SE}(3)$ object $\mathbf{A}$ is composed as follows:

$$\mathbf{A} \in \mathrm{SE}(3) = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}. \tag{1}$$

A possible minimal representation for this object could be using 3 Cartesian coordinates and the 3 Euler angles for the orientation, namely

$$\mathbf{a} = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^\top. \tag{2}$$

In order to pass from one representation to the other, we have to define suitable mapping functions. In this case, we use the following notation:

$$\mathbf{A} = \mathrm{ev2t}(\mathbf{a}) \tag{3}$$

$$\mathbf{a} = \mathrm{et2v}(\mathbf{A}). \tag{4}$$

Starting from Eq. (3), while the translational component $\mathbf{t}$ is easy to recover from $\mathbf{a}$, the rotational part requires to compose the rotation matrix as follows:

$$\mathbf{R} = \mathbf{R}(\phi, \theta, \psi) = \mathbf{R}_x(\phi)\, \mathbf{R}_y(\theta)\, \mathbf{R}_z(\psi). \tag{5}$$

As a result, indicating with c the $\cos$ and with s the $\sin$ of an angle, matrix $\mathbf{R}$ is built as:

$$\mathbf{R} = \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix} = \begin{bmatrix} c\theta\, c\psi & -c\theta\, s\psi & s\theta \\ m & n & -c\theta\, s\phi \\ p & q & c\theta\, c\phi \end{bmatrix} \tag{6}$$

where

$$\begin{aligned} m &= c\phi\, s\psi + s\phi\, c\psi\, s\theta \\ n &= c\phi\, c\psi - s\phi\, s\theta\, s\psi \\ p &= s\phi\, s\psi - c\phi\, c\psi\, s\theta \\ q &= s\phi\, c\psi + c\phi\, s\theta\, s\psi \end{aligned}$$

On the contrary, through Eq. (4) we compute the minimal parametrization $\mathbf{a}$ starting from $\mathbf{A}$. Again, the rotational part we have to compute ϕ, θ and ψ from Eq. (6).

Another possible minimal representation for a $\mathrm{SE}(3)$ object could use the unit-quaternion instead of the Euler angles. In this case, given the unit-quaternion $\mathbf{q} = [q_w\ q_x\ q_y\ q_z]^\top$ normalized by q_w, the 6D minimal representation of $\mathbf{A}$ is

$$\mathbf{a} = \begin{bmatrix} x & y & z & q_x & q_y & q_z \end{bmatrix}^\top. \tag{7}$$

Again, we have to define proper mapping functions to pass from one parametrization to the other, namely:

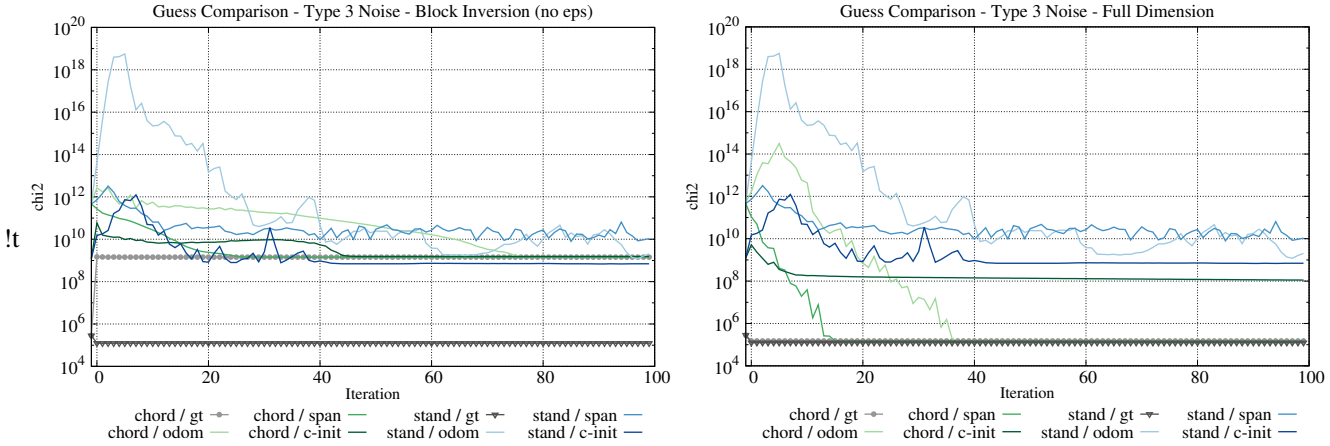
$$\mathbf{A} = \mathrm{v2t}(\mathbf{a}) \tag{8}$$

$$\mathbf{a} = \mathrm{t2v}(\mathbf{A}). \tag{9}$$

The function $\mathrm{v2t}$ computes the rotational component of $\mathbf{A}$ from the unit quaternion as follows:

$$\mathbf{R} = \mathbf{R}(\mathbf{q}) = \begin{bmatrix} 1-2q_y^2-2q_z^2 & 2q_xq_y-2q_zq_w & 2q_xq_z+2q_yq_w \\ 2q_xq_y+2q_zq_w & 1-2q_x^2-2q_z^2 & 2q_yq_z-2q_xq_w \\ 2q_xq_z-2q_yq_w & 2q_yq_z+2q_xq_w & 1-2q_x^2-2q_y^2 \end{bmatrix}. \tag{10}$$

On the contrary, to pass from the extended parametrization to the minimal one, we have to compute the quaternion components from Eq. (10). Also in this case, the task is not straightforward and involves several non-linear functions.



(a) No ϵ parameter.

(b) $\epsilon = 10^{-6}$.

Fig. 1: Comparison between the computation of $\bar{\mathbf{\Omega}}_{ij}$ with and without the ϵ parameter. Dataset is `sphere-b` with noise of type `3B`. We performed 100 iteration of the g^2o Gauss-Newton (GN) algorithm. These results are obtained using only one noise realization.

II. Information Matrix Projection onto the Chordal Error Space

In this section we provide more details on how we project the original 6D measurement information matrix $\mathbf{\Omega}_{ij}$ onto the chordal error space, resulting in $\bar{\mathbf{\Omega}}_{ij}$. To do such task, one can use either the first order error propagation or the Unscented Transform. In our case, using the former method does not capture entirely the non-linearity of the object (due to the first-order approximation). For this reason, we used the latter method to perform the projection. In the remaining of this section, we will make use of the notation introduced in Sec. I of this document. Given the original measurement $\langle \mathbf{Z}_{ij}, \mathbf{\Omega}_{ij} \rangle$, we know that $\mathbf{\Omega}_{ij}$ is expressed in the unit-quaternion based error space. Thus to compute the sigma points as follows:

- we compute the mean and covariance of the original measurement distribution as $\mathbf{z}_{ij} = \text{t2v}(\mathbf{Z}_{ij})$ and $\mathbf{\Sigma}_{ij} = \mathbf{\Omega}_{ij}^{-1}$ respectively
- from the distribution $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{ij})$ we sample the 6D sigma points $X^{(6)} = \langle \mathbf{x}^{(6)}, w_m^{(6)}, w_c^{(6)} \rangle$
- we compute the 12D sigma points $X^{(12)}$ from the original 6D ones $X^{(6)}$ as follows

$$
\begin{aligned}
\mathbf{x}_0^{(12)} &= \text{flatten}(\text{v2t}(\mathbf{z}_{ij})) && w_{m_0}^{(12)} = w_{m_0}^{(6)} && w_{c_0}^{(12)} = w_{c_0}^{(6)} && \\
\mathbf{x}_i^{(12)} &= \text{flatten}(\text{v2t}(\mathbf{z}_{ij})\ \text{v2t}(+\mathbf{x}_i^{(6)})) && w_{m_i}^{(12)} = w_{m_i}^{(6)} && w_{c_i}^{(12)} = w_{c_i}^{(6)} && [i = 1 \dots n] \\
\mathbf{x}_i^{(12)} &= \text{flatten}(\text{v2t}(\mathbf{z}_{ij})\ \text{v2t}(-\mathbf{x}_i^{(6)})) && w_{m_i}^{(12)} = w_{m_i}^{(6)} && w_{c_i}^{(12)} = w_{c_i}^{(6)} && [i = n \dots 2n]
\end{aligned}
$$

- reconstruct the distribution $\mathcal{N}(\mu^{(12)}, \bar{\mathbf{\Sigma}}_{ij})$ from 12D sigma points $X^{(12)}$.

If $\bar{\mathbf{\Sigma}}_{ij}$ would have been full rank, we could compute $\bar{\mathbf{\Omega}}_{ij}$ simply as $\bar{\mathbf{\Omega}}_{ij} = \bar{\mathbf{\Sigma}}_{ij}^{-1}$. Still, since we are projecting elements of a 6D space onto a 12D one, $\bar{\mathbf{\Sigma}}_{ij}$ is rank deficient and we cannot invert it. To overcome this issue, we can condition the eigenvalues that are collapsed in $\bar{\mathbf{\Sigma}}_{ij}$. More in detail, given a threshold $\epsilon > 0$ and the eigendecomposition of the 12D covariance matrix $\bar{\mathbf{\Sigma}}_{ij} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1}$, we can compute the conditioned eigenvalues as follow:

$$
\lambda = \begin{cases} \lambda & \text{if} \quad \lambda \geq \epsilon \\ \lambda + \epsilon & \text{if} \quad \lambda < \epsilon \end{cases}
$$

We can now build the conditioned covariance matrix as

$$
\bar{\mathbf{\Sigma}}_{ij}^{cond} = \mathbf{Q}\ \mathbf{\Lambda}^{cond}\ \mathbf{Q}^{-1} \tag{11}
$$

and compute the 12D information matrix as $\bar{\mathbf{\Omega}}_{ij} = (\bar{\mathbf{\Sigma}}_{ij}^{cond})^{-1}$. Note that, if we only keep up to the 6 most informative eigenvalues that are not null and we build $\bar{\mathbf{\Omega}}_{ij}^{cond}$ as $\bar{\mathbf{\Omega}}_{ij}^{cond} = (\mathbf{Q}\mathbf{\Lambda}^{(6)}\mathbf{Q}^{-1})^{-1}$ we do not need the ϵ parameter. In this case we noticed that the χ^2 evaluated with our error function is similar to the one computed with the standard error function. Still, since we are not using the entire information encoded in the error space, the convergence properties will be worsened - as shown in Fig. 1. For this reason, in our experiments we preferred to use all the eigenvalues to reconstruct $\bar{\mathbf{\Sigma}}_{ij}^{cond}$. However, in this last case, ϵ becomes an parameter of the optimization process, acting as a regularization term.

III. JACOBIANS' COMPUTATION

In this section, we provide the derivation of the Jacobians when using our error function, using the notation introduced in Sec. I of this document. In our case, the increments are parameterized as Cartesian coordinates and 3 Euler angles, namely:

$$\boldsymbol{\Delta}\mathbf{x} = [\Delta x_x \ \Delta x_y \ \Delta x_z \ \Delta x_\phi \ \Delta x_\theta \ \Delta x_\psi]^\top.$$

Given a edge $\mathbf{Z}_{ij} \in \mathrm{SE}(3)$ connecting node $\mathbf{X}_i$ to $\mathbf{X}_j$ and its prediction $\hat{\mathbf{Z}}_{ij} = \mathbf{h}(\mathbf{X}_i, \mathbf{X}_j) \in \mathrm{SE}(3)$, the error vector is computed as:

$$\mathbf{e}_{ij}(\mathbf{X}_i, \mathbf{X}_j) = \hat{\mathbf{Z}}_{ij} \boxminus \mathbf{Z}_{ij} = \mathrm{flatten}(\mathbf{X}_i^{-1}\mathbf{X}_j) - \mathrm{flatten}(\mathbf{Z}_{ij}). \tag{12}$$

Applying a small state perturbation $\boldsymbol{\Delta}\mathbf{x}$ to Eq. (12), the perturbed error becomes:

$$\mathbf{e}_{ij}(\mathbf{X}_i \boxplus \boldsymbol{\Delta}\mathbf{x}_i, \mathbf{X}_j \boxplus \boldsymbol{\Delta}\mathbf{x}_j) = \mathrm{flatten}\left((\mathrm{ev2t}(\boldsymbol{\Delta}\mathbf{x}_i)\mathbf{X}_i)^{-1} \ (\mathrm{ev2t}(\boldsymbol{\Delta}\mathbf{x}_j)\mathbf{X}_j)\right) - \mathrm{flatten}(\mathbf{Z}_{ij}) \tag{13}$$

The Jacobian $\mathbf{J}_j$ is computed performing the partial derivative of Eq. (13) with respect to $\boldsymbol{\Delta}\mathbf{x}_j$:

$$\mathbf{J}_j = \left.\frac{\partial\, \mathbf{e}_{ij}(\mathbf{X}_i, \mathbf{X}_j \boxplus \boldsymbol{\Delta}\mathbf{x}_j)}{\partial \boldsymbol{\Delta}\mathbf{x}_j}\right|_{\substack{\boldsymbol{\Delta}\mathbf{x}_i = 0 \\ \boldsymbol{\Delta}\mathbf{x}_j = 0}}. \tag{14}$$

We indicate with $\mathbf{R}_x(\alpha)$, $\mathbf{R}_y(\alpha)$ and $\mathbf{R}_z(\alpha)$ the matrices describing a rotation of α around axis x, y and z respectively. Additionally, we indicate with $\mathbf{R}'_x$, $\mathbf{R}'_y$ and $\mathbf{R}'_z$ the derivatives of the previously defined rotation matrices:

$$\mathbf{R}'_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\sin(\alpha) & -\cos(\alpha) \\ 0 & \cos(\alpha) & -\sin(\alpha) \end{bmatrix} \quad \mathbf{R}'_y = \begin{bmatrix} -\sin(\alpha) & 0 & \cos(\alpha) \\ 0 & 0 & 0 \\ -\cos(\alpha) & 0 & -\sin(\alpha) \end{bmatrix} \quad \mathbf{R}'_z = \begin{bmatrix} -\sin(\alpha) & -\cos(\alpha) & 0 \\ \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{15}$$

We name $\mathbf{R}'_{x0}$, $\mathbf{R}'_{y0}$ and $\mathbf{R}'_{z0}$ the matrices resulting posing $\alpha = 0$ in Eq. (15):

$$\mathbf{R}'_{x0} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \quad \mathbf{R}'_{y0} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \quad \mathbf{R}'_{y0} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{16}$$

Finally, we refer to the column of each matrix in Eq. (16) as $\mathbf{r}'_{k0}$ with $k = \{x, y, z\}$. Returning to Eq. (14), thanks to the fact that we are using the Euclidean minus in Eq. (13), $\mathbf{J}_j$ is composed as follows:

$$\mathbf{J}_j = \begin{bmatrix} \mathrm{flatten}(\frac{\partial \mathbf{h}}{\partial \Delta x_x}) & \mathrm{flatten}(\frac{\partial \mathbf{h}}{\partial \Delta x_y}) & \mathrm{flatten}(\frac{\partial \mathbf{h}}{\partial \Delta x_z}) & \mathrm{flatten}(\frac{\partial \mathbf{h}}{\partial \Delta x_\phi}) & \mathrm{flatten}(\frac{\partial \mathbf{h}}{\partial \Delta x_\theta}) & \mathrm{flatten}(\frac{\partial \mathbf{h}}{\partial \Delta x_\psi}) \end{bmatrix} \tag{17}$$

where $\mathbf{h} = \mathbf{h}(\mathbf{X}_i, \mathbf{X}_j)$ represents the predicted measurement. The quantities in Eq. (17) are easily computed in close form as follows:

$$\frac{\partial \mathbf{h}}{\partial \Delta x_x} = \begin{bmatrix} \mathbf{R}_i^\top \mathbf{R}_j & \mathbf{R}_i^\top [1\ 0\ 0]^\top \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \quad \frac{\partial \mathbf{h}}{\partial \Delta x_x} = \begin{bmatrix} \mathbf{R}_i^\top \mathbf{R}_j & \mathbf{R}_i^\top [0\ 1\ 0]^\top \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \quad \frac{\partial \mathbf{h}}{\partial \Delta x_x} = \begin{bmatrix} \mathbf{R}_i^\top \mathbf{R}_j & \mathbf{R}_i^\top [0\ 0\ 1]^\top \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{18}$$

$$\frac{\partial \mathbf{h}}{\partial \Delta x_\phi} = \begin{bmatrix} \mathbf{R}_i^\top \mathbf{R}'_{x0} \mathbf{R}_j & \mathbf{R}_i^\top \mathbf{R}'_{x0} \mathbf{t}_j \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \quad \frac{\partial \mathbf{h}}{\partial \Delta x_\theta} = \begin{bmatrix} \mathbf{R}_i^\top \mathbf{R}'_{y0} \mathbf{R}_j & \mathbf{R}_i^\top \mathbf{R}'_{y0} \mathbf{t}_j \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \quad \frac{\partial \mathbf{h}}{\partial \Delta x_\psi} = \begin{bmatrix} \mathbf{R}_i^\top \mathbf{R}'_{z0} \mathbf{R}_j & \mathbf{R}_i^\top \mathbf{R}'_{z0} \mathbf{t}_j \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{19}$$

Finally, $\tilde{\mathbf{J}}_i$ can be computed straightforwardly from Eq. (17), leading to the relation

$$\mathbf{J}_i = -\mathbf{J}_j \tag{20}$$

The numerical property expressed in Eq. (20) can potentially speedup the computation of matrix $\mathbf{H}$, since for each edge we only have to compute a single Jacobian - *i.e.* less FLOPS are required for the linearization of each $\mathbf{Z}_{ij}$. We recall that to build the $\mathbf{H}$ matrix, for each edge $\mathbf{Z}_{ij}$ we have to compute the following $\mathbf{H}$ components:

$$\begin{aligned} \mathbf{H}_{ii} &= \mathbf{J}_i^\top \boldsymbol{\Omega}_{ij} \mathbf{J}_i \qquad & \mathbf{H}_{jj} &= \mathbf{J}_j^\top \boldsymbol{\Omega}_{ij} \mathbf{J}_j \\ \mathbf{H}_{ij} &= \mathbf{J}_i^\top \boldsymbol{\Omega}_{ij} \mathbf{J}_j \qquad & \mathbf{H}_{ji} &= \mathbf{J}_j^\top \boldsymbol{\Omega}_{ij} \mathbf{J}_i. \end{aligned}$$

Moreover, we have to calculate also the $\mathbf{b}$ components relative to $\mathbf{Z}_{ij}$, namely:

$$\mathbf{b}_i = \mathbf{J}_i^\top \boldsymbol{\Omega}_{ij} \mathbf{e}_{ij} \qquad \mathbf{b}_j = \mathbf{J}_j^\top \boldsymbol{\Omega}_{ij} \mathbf{e}_{ij}.$$

Note that, the matrix involved have dimension 12×6 and 12×12, respectively for $\mathbf{J}_i/\mathbf{J}_j$ and $\mathbf{\Omega}_{ij}$. Therefore, computing the products $\mathbf{J}^\top \mathbf{\Omega}_{ij} \mathbf{J}$ and $\mathbf{J}^\top \mathbf{\Omega}_{ij} \mathbf{e}_{ij}$ result to be very time consuming. However, given that $\mathbf{J}_i = -\mathbf{J}_j$, the following relations hold:

$$\mathbf{H}_{jj} = \mathbf{H}_{ii} \qquad \mathbf{H}_{ij} = -\mathbf{H}_{ii} \qquad \mathbf{H}_{ji} = -\mathbf{H}_{ii}$$
$$\mathbf{b}_j = -\mathbf{b}_i.$$

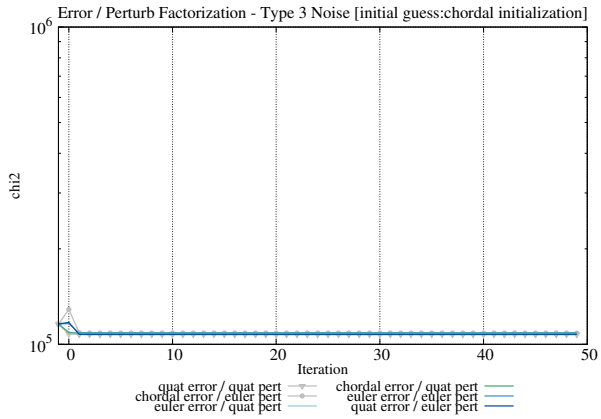
Thanks to the property in Eq. (20), we can compute the two time consuming products only once. If one embeds this property in a custom implementation of a generic Iterative Least-Squares (ILS) algorithm - *e.g.* GN or Levemberg-Marquardt (LM) - it leads to a sensible increase in terms of speed. Moreover, the $\mathbf{H}$ matrix results to be Laplacian. This characteristic might be exploited to increase the speed of the linear solver, however, this topic is not analyzed in this work.

IV. Comparing Error and Perturbation Parametrizations

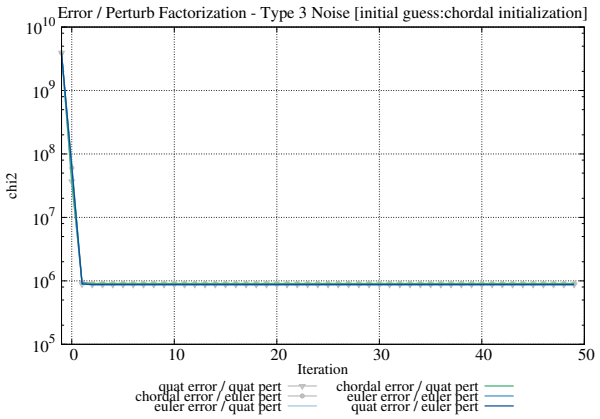
As reported in Sec. V-D, different combinations of error and increment parametrization can be used in Pose-Graph Optimization (PGO). In this section, we compare the following combinations of parametrizations:

- $\{$quaternion, euclidean$\}$ based standard error fuction + $\{$quaternion, euclidean$\}$ based increments
- chordal based error fuction + $\{$quaternion, euclidean$\}$ based increments.

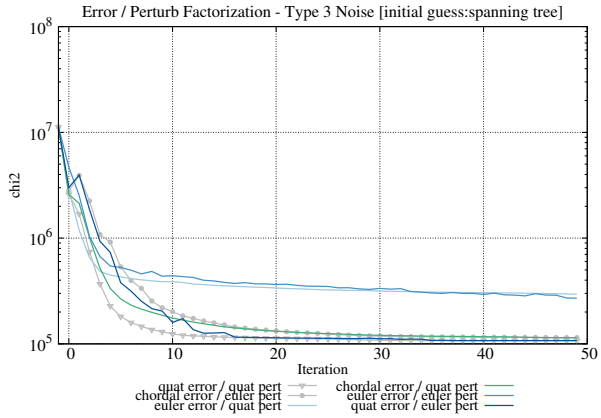
We compared the χ^2 evolution of all those approaches, recomputing it at each iteration with the standard standard function - *i.e.*the one that uses the unit quaternion parametrization - to avoid any bias in the evaluation. The optimization algorithm used is GN in all cases. In left column of Fig. 2 we reported the χ^2 of the `grid` dataset, using type `3A` noise to perturb the edges. In this case, both from good - Fig. 2a - or poor - Fig. 2c and Fig. 2e - our approach delivers similar performances compared to all the standard error functions. Still, when the covariances results to be highly non spherical - as they are using type `3B` noise - all the standard approaches exhibit convergence issue. On the contrary, when starting from a reasonable initial guess - *e.g.* the spanning tree - our approach always succeeds in finding the optimal configuration. From extremely poor initial configurations - *e.g.* the odometry - our approach result in sub-optimal configurations, but closer to the optimum compared to standard approaches. These evaluations are reported in the right column of Fig. 2. All the experiments refer to only one realization of noise.



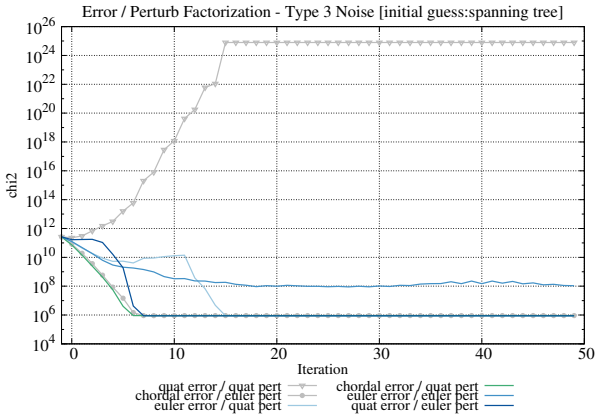
(a) Initialization: chordal; noise: 3A.



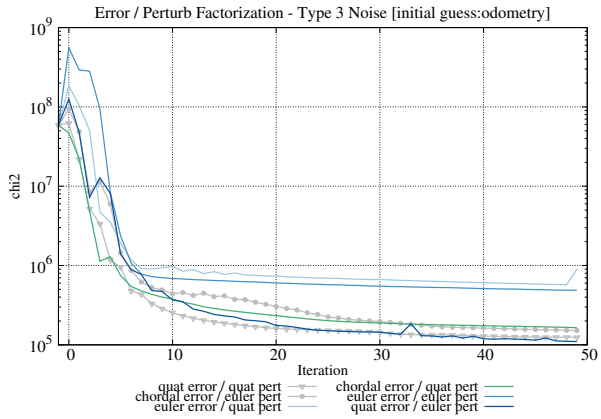
(b) Initialization: chordal; noise: 3B.



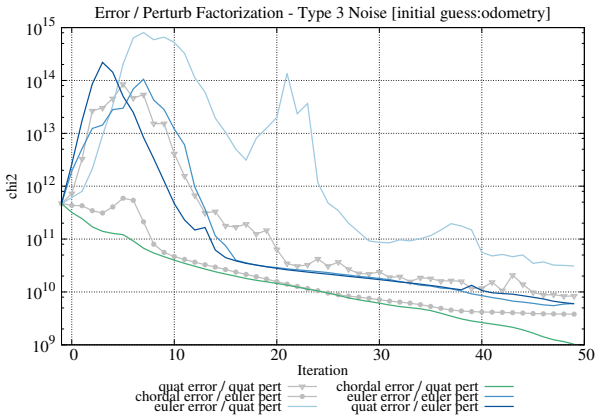
(c) Initialization: spanning tree; noise: 3A.



(d) Initialization: spanning tree; noise: 3B.



(e) Initialization: odometry; noise: 3A.



(f) Initialization: odometry; noise: 3B.

Fig. 2: Evolution of the χ^2 on the grid dataset using different parametrizations for the error and the perturbation vectors. The left column reports experiments using type 3A noise. In the right column are reported the comparisons using type 3B noise. In all cases we performed 50 iterations of GN optimization in the g^2o framework.