Gauss-Newton Optimization and Rotations

Valentin Peretroukhin

October 3, 2019

Abstract

A brief summary of Non-Linear Least Squares, Gauss-Newton optimization, Maximum Likelihood and optimization over SO(3).

1 Linear Least Squares

We wish to solve the following problem:

$$E(\mathbf{x}) = \frac{1}{2}\mathbf{e}(\mathbf{x})^T\mathbf{e}(\mathbf{x}) \tag{1}$$

We notice that if it happens to be the case that we can express $\mathbf{e}$ as a linear function of $\mathbf{x}$, we get a quadratic with an analytic minimum. In other words, if

$$\mathbf{e}(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}, \tag{2}$$

then

$$\begin{align} E(\mathbf{x}) &= \frac{1}{2}(\mathbf{A}\mathbf{x} - \mathbf{b})^T(\mathbf{A}\mathbf{x} - \mathbf{b}) \tag{3} \\ &= \frac{1}{2}\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{b} \tag{4} \end{align}$$

and taking the gradient $\frac{\mathrm{d}E}{\mathrm{d}\mathbf{x}}$ and setting to 0, we arrive at

$$\begin{align} \frac{\mathrm{d}E}{\mathrm{d}\mathbf{x}} &= \frac{\mathrm{d}}{\mathrm{d}\mathbf{x}}\left(\frac{1}{2}\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{b}\right) \tag{5} \\ &= \mathbf{A}^T\mathbf{A}\mathbf{x} - \mathbf{A}^T\mathbf{b} = \mathbf{0}. \tag{6} \end{align}$$

Solving this, we arrive at the famous normal equations:

$$\begin{align} \mathbf{A}^T\mathbf{A}\mathbf{x}^* &= \mathbf{A}^T\mathbf{b} \tag{7} \\ \mathbf{x}^* &= (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}. \tag{8} \end{align}$$

2 Nonlinear Least Squares

$$E(\mathbf{x}) = \frac{1}{2}\mathbf{e}(\mathbf{x})^T\mathbf{e}(\mathbf{x}) \tag{9}$$

If $\mathbf{e}(\mathbf{x})$ is not linear, we can use Gauss-Newton optimization to iteratively find an optimal $\mathbf{x}$. We first must be given an initial *operating point*[1], $\mathbf{x}_{op}$ and linearize $\mathbf{e}(\mathbf{x})$ about that point. In other words, determine:

$$\begin{align}
\mathbf{e}(\mathbf{x}) &= \mathbf{e}(\mathbf{x}_{op} + \delta\mathbf{x}) \tag{10}\\
&\approx \mathbf{e}(\mathbf{x}_{op}) + \underbrace{\left.\frac{\partial \mathbf{e}}{\partial \mathbf{x}}\right|_{\mathbf{x}_{op}}}_{\mathbf{J}_e} \delta\mathbf{x} \tag{11}\\
&= \mathbf{e}(\mathbf{x}_{op}) + \mathbf{J}_e\delta\mathbf{x} \tag{12}
\end{align}$$

Now we can write our objective function as a function of $\delta\mathbf{x}$ instead of $\mathbf{x}$:

$$E(\delta\mathbf{x}) = \frac{1}{2}(\mathbf{e}(\mathbf{x}_{op}) + \mathbf{J}_e\delta\mathbf{x})^T(\mathbf{e}(\mathbf{x}_{op}) + \mathbf{J}_e\delta\mathbf{x}), \tag{13}$$

This is a parabola fit at $\mathbf{x}_{op}$. Comparing to our linear least squares problem, we can immediately write down the optimal $\delta\mathbf{x}^*$ as:

$$\begin{align}
\mathbf{J}_e^T\mathbf{J}_e\delta\mathbf{x}^* &= -\mathbf{J}_e^T\mathbf{e}(\mathbf{x}_{op}) \tag{14}\\
\delta\mathbf{x}^* &= -(\mathbf{J}_e^T\mathbf{J}_e)^{-1}\mathbf{J}_e^T\mathbf{e}(\mathbf{x}_{op}) \tag{15}
\end{align}$$

Given an $\delta\mathbf{x}^*$, we update $\mathbf{x}_{op}$,

$$\mathbf{x}_{op} \leftarrow \mathbf{x}_{op} + \delta\mathbf{x}^* \tag{16}$$

and iterate until convergence[2].

2.1 Multiple error terms

If our criterion looks like this

$$E(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{N}\mathbf{e}_i(\mathbf{x})^T\mathbf{e}_i(\mathbf{x}), \tag{17}$$

we can use the linearity of derivatives to re-write Equation (14) as:

$$\begin{align}
\left(\sum_{i=1}^{N}\mathbf{J}_{e_i}^T\mathbf{J}_{e_i}\right)\delta\mathbf{x}^* &= -\sum_{i=1}^{N}\mathbf{J}_{e_i}^T\mathbf{e}_i(\mathbf{x}_{op}) \tag{18}\\
\delta\mathbf{x}^* &= -\left(\sum_{i=1}^{N}\mathbf{J}_{e_i}^T\mathbf{J}_{e_i}\right)^{-1}\left(\sum_{i=1}^{N}\mathbf{J}_{e_i}^T\mathbf{e}_i(\mathbf{x}_{op})\right) \tag{19}
\end{align}$$

[1]This is sometimes a non-trivial problem. Bad initial operating points can lead to poor performance.

[2]There are many ways to define convergence, but a straightforward way is to check when $\|\delta\mathbf{x}^*\|$ falls below some small threshold.

2.2 Weighted error terms

If we are given **symmetric** weight matrices, $\mathbf{W}_i$, and our criterion looks like this

$$E(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{N} \mathbf{e}_i(\mathbf{x})^T \mathbf{W}_i \mathbf{e}_i(\mathbf{x}), \tag{20}$$

we can adapt our normal equations as follows:

$$\left(\sum_{i=1}^{N} \mathbf{J}_{e_i}^T \mathbf{W}_i \mathbf{J}_{e_i}\right) \delta\mathbf{x}^* = -\sum_{i=1}^{N} \mathbf{J}_{e_i}^T \mathbf{W}_i \mathbf{e}_i(\mathbf{x}_{op}) \tag{21}$$

$$\delta\mathbf{x}^* = -\left(\sum_{i=1}^{N} \mathbf{J}_{e_i}^T \mathbf{W}_i \mathbf{J}_{e_i}\right)^{-1} \left(\sum_{i=1}^{N} \mathbf{J}_{e_i}^T \mathbf{W}_i \mathbf{e}_i(\mathbf{x}_{op})\right) \tag{22}$$

2.3 Relation to Maximum Likelihood

If we assume that each of our errors, $\mathbf{e}_i$, is distributed according to the Gaussian density $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_i)$:

$$\mathbf{e}_i(\mathbf{x}) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_i) \tag{23}$$

and each error is independent of the others, then the likelihood of observing a set of N errors becomes the product:

$$p(\mathbf{e}_1(\mathbf{x}), ..., \mathbf{e}_N(\mathbf{x})) = \prod_{i=1}^{N} p(\mathbf{e}_i(\mathbf{x}); \mathbf{0}, \boldsymbol{\Sigma}_i). \tag{24}$$

To maximize this likelihood, we can take the logarithm of both sides (since its a monotonically increasing function, it doesn't affect the optimization):

$$\log p(\mathbf{e}_i, ..., \mathbf{e}_N) = \sum_{i=1}^{N} \log p(\mathbf{e}_i(\mathbf{x}); \mathbf{0}, \boldsymbol{\Sigma}_i). \tag{25}$$

Recalling the definition of a multivariate Gaussian, we have

$$p(\mathbf{e}_i(\mathbf{x})) = \alpha \exp\left\{-\frac{1}{2}\mathbf{e}_i(\mathbf{x})^T \boldsymbol{\Sigma}_i^{-1} \mathbf{e}_i(\mathbf{x})\right\} \tag{26}$$

where α does not depend on $\mathbf{x}$. Now we can write the log likelihood as:

$$\log p(\mathbf{e}_i, ..., \mathbf{e}_N) = -\sum_{i=1}^{N} \frac{1}{2}\mathbf{e}(\mathbf{x})^T \boldsymbol{\Sigma}_i^{-1} \mathbf{e}(\mathbf{x}) + C \tag{27}$$

where the constant C does not depend on $\mathbf{x}$. Maximizing the log likelihood is the same as minimizing its negative, so the final maximum likelihood estimate is given by:

$$\mathbf{x}_{ML} = \underset{\mathbf{x}}{\operatorname{argmin}}\{-\log p(\mathbf{e}_i(\mathbf{x}), ..., \mathbf{e}_N(\mathbf{x}))\} = \underset{\mathbf{x}}{\operatorname{argmin}} \frac{1}{2}\sum_{i=1}^{N} \mathbf{e}_i(\mathbf{x})^T \boldsymbol{\Sigma}_i^{-1} \mathbf{e}_i(\mathbf{x}). \tag{28}$$

Looks oddly familiar! If we interpret each $\mathbf{W}_i$ as the information matrix, $\boldsymbol{\Sigma}_i^{-1}$, of a Gaussian multivariate density that describes the probability of drawing a given zero-mean error, $\mathbf{e}_i$, then we can use weighted non-linear least squares to arrive at the maximum likelihood estimate. Convenient!

2.4 Salient Gauss-Newton Points

1. This vanilla routine only works for unconstrained states and Gauss Newton is only applicable to squared error objectives.

2. We need a good initial guess otherwise we might be stuck in local minima.

3. Notice that Gauss-Newton does not directly fit a quadratic about an operating point (i.e., it does NOT approximate $E(\mathbf{x} + \delta\mathbf{x})$). Instead it linearly approximates $\mathbf{e}(\mathbf{x}_{op} + \delta\mathbf{x})$ which results in a quadratic cost in $\delta\mathbf{x}$. The full Newton's method fits a quadratic to $E(\mathbf{x})$ about an operating point.

4. Gauss-Newton approximates Newton's method by approximating the Hessian of $E(\mathbf{x})$ as $\mathbf{J}_e^T\mathbf{J}_e$ (the full Hessian requires second order derivatives of $\mathbf{e}(\mathbf{x})$).

3 Optimizing Rotations

Recall that rotations belong to a special matrix Lie group called the *Special Orthogonal Group*. We can define the group as follows:

$$\mathrm{SO}(3) = \{\mathbf{C} \in \mathbb{R}^{3\times 3} |\ \mathbf{C}^T\mathbf{C} = \mathbf{1}, \det \mathbf{C} = 1\}. \tag{29}$$

Let's say we have some quadratic objective that includes a rotation. For example, consider finding the rotation matrix that best aligns pairs of vectors, $\mathbf{u}_i$ and $\mathbf{v}_i$ both in $\mathbb{R}^3$, such that the following objective is minimized (this is called the *Wahba Problem*):

$$E(\mathbf{C}) = \frac{1}{2}\sum_{i=1}^{N} \left(\mathbf{C}\mathbf{u}_i - \mathbf{v}_i\right)^T \left(\mathbf{C}\mathbf{u}_i - \mathbf{v}_i\right) \tag{30}$$

$$= \frac{1}{2}\sum_{i=1}^{N} \mathbf{e}_i(\mathbf{C})^T\mathbf{e}_i(\mathbf{C}) \tag{31}$$

In order to use our non-linear least squares formulation, we'll need to find some way of updating an operating point $\mathbf{C}_{op}$ with some small update.

3.1 Useful notation

For our derivations, the following skew-symmetric cross-product operator will be useful:

$$\mathbf{a}^\times = [\mathbf{a}]_\times = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}^\times = \begin{bmatrix} 0 & -a_2 & a_1 \\ a_2 & 0 & -a_0 \\ -a_1 & a_0 & 0 \end{bmatrix}. \tag{32}$$

Note that $\mathbf{a} \times \mathbf{b} = \mathbf{a}^\times\mathbf{b} = -\mathbf{b}^\times\mathbf{a}$. Also note that you may see this defined as $[\mathbf{a}]_\times$ or $\mathbf{a}^\wedge$.

3.2 Euler Angles

The first way we might consider solving this problem is to use Euler angles. Let's parametrize our rotation matrix with a 1-2-3 Euler angle sequence as:

$$\mathbf{C}(\boldsymbol{\theta}) = \mathbf{C}_3(\theta_3)\mathbf{C}_2(\theta_2)\mathbf{C}_1(\theta_1). \tag{33}$$

Now, our goal will be to start with an operating point $\boldsymbol{\theta}_{op}$ and solve for $\delta\boldsymbol{\theta}$ so that we can use the update $\boldsymbol{\theta}_{op} \leftarrow \boldsymbol{\theta}_{op} + \delta\boldsymbol{\theta}$.

Beginning with the definition of our error, we derive the following:

$$\begin{align}
\mathbf{e}_i(\boldsymbol{\theta}_{op} + \delta\boldsymbol{\theta}) &= \mathbf{C}(\boldsymbol{\theta}_{op} + \delta\boldsymbol{\theta})\mathbf{u}_i - \mathbf{v}_i \tag{34}\\
&\approx \underbrace{\mathbf{C}(\boldsymbol{\theta}_{op})\mathbf{u}_i - \mathbf{v}_i}_{\mathbf{e}_i(\boldsymbol{\theta}_{op})} + \underbrace{\left.\frac{\partial \mathbf{C}\mathbf{u}_i}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}_{op}}}_{\mathbf{J}_{e_i}} \delta\boldsymbol{\theta} \tag{35}\\
&= \mathbf{e}_i(\boldsymbol{\theta}_{op}) + \mathbf{J}_{e_i}\delta\boldsymbol{\theta}. \tag{36}
\end{align}$$

Now, how do we compute $\frac{\partial \mathbf{C}\mathbf{u}_i}{\partial \boldsymbol{\theta}}$? For this, we can either use straight trigonometric differentiation or, we can use the handy relation (refer to (Barfoot, 2017) for more details) that:

$$\begin{align}
\frac{\partial \mathbf{C}\mathbf{u}_i}{\partial \theta_3} &= [\mathbf{1}_3]_\times \mathbf{C}_3(\theta_3)\mathbf{C}_2(\theta_2)\mathbf{C}_1(\theta_1)\mathbf{u}_i \tag{37}\\
\frac{\partial \mathbf{C}\mathbf{u}_i}{\partial \theta_2} &= \mathbf{C}_3(\theta_3)\,[\mathbf{1}_2]_\times \mathbf{C}_2(\theta_2)\mathbf{C}_1(\theta_1)\mathbf{u}_i \nonumber\\
\frac{\partial \mathbf{C}\mathbf{u}_i}{\partial \theta_1} &= \mathbf{C}_3(\theta_3)\mathbf{C}_2(\theta_2)\,[\mathbf{1}_1]_\times \mathbf{C}_1(\theta_1)\mathbf{u}_i \nonumber
\end{align}$$

where $\mathbf{1}_i$ is the column vector of zeros with 1 in the ith spot (e.g., $\mathbf{1}_3 = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$). The Jacobian can then be composed as the horizontally stacked columns:

$$\frac{\partial \mathbf{C}\mathbf{u}_i}{\partial \boldsymbol{\theta}} = \begin{bmatrix}\frac{\partial \mathbf{C}\mathbf{u}_i}{\partial \theta_1} & \frac{\partial \mathbf{C}\mathbf{u}_i}{\partial \theta_2} & \frac{\partial \mathbf{C}\mathbf{u}_i}{\partial \theta_3}\end{bmatrix}. \tag{38}$$

Note if we do the optimization this way, we need to watch out that our Euler angles don't approach Gimbal lock!

3.2.1 A Tidy Derivation of the Euler Angle Jacobian

We can derive the Euler angle derivatives (Equation (37)) starting from a more general formula for the derivative of an exponential map that is proved in Gallego and Yezzi (2015). Namely, if $\mathbf{C}(\boldsymbol{\phi}) = \mathrm{Exp}\,(\boldsymbol{\phi})$, with $\boldsymbol{\phi} = \begin{bmatrix}\phi_1 & \phi_2 & \phi_3\end{bmatrix}^T$, then

$$\frac{\partial \mathbf{C}}{\partial \phi_i} = \frac{\phi_i\,[\boldsymbol{\phi}]_\times + \left[[\boldsymbol{\phi}]_\times(\mathbf{1} - \mathbf{C})\mathbf{1}_i\right]_\times}{\|\boldsymbol{\phi}\|^2}\mathbf{C}, \tag{39}$$

where $\mathbf{1}_i$ has the same column-vector definition as above.

Now, in our 1-2-3 Euler sequence, we can re-write our principal rotation matrices using the exponential map:

$$\begin{align}
\mathbf{C}(\boldsymbol{\theta}) &= \mathbf{C}_3(\theta_3)\mathbf{C}_2(\theta_2)\mathbf{C}_1(\theta_1) \tag{40}\\
\mathbf{C}(\boldsymbol{\theta}) &= \mathrm{Exp}\left(\begin{bmatrix}0\\0\\\theta_3\end{bmatrix}\right)\mathrm{Exp}\left(\begin{bmatrix}0\\\theta_2\\0\end{bmatrix}\right)\mathrm{Exp}\left(\begin{bmatrix}\theta_1\\0\\0\end{bmatrix}\right). \tag{41}
\end{align}$$

Note that because the Lie bracket of SO(3) is not 0 (in other words, rotations do not commute), we **cannot** simplify the above to $\mathrm{Exp}\left(\begin{bmatrix}\theta_1 & \theta_2 & \theta_3\end{bmatrix}^T\right)$. We could however, attempt to use the Baker-Campbell-Hausdorff (BCH) formula to simplify the above to a single exponential vector, but I am not sure that would result in anything analytic.

Instead, we can take the following approach. Note that

$$\begin{aligned}\frac{\partial \mathbf{C}}{\partial \theta_3} &= \frac{\partial \mathbf{C}_3}{\partial \theta_3}\mathbf{C}_2(\theta_2)\mathbf{C}_1(\theta_1) \\ \frac{\partial \mathbf{C}}{\partial \theta_2} &= \mathbf{C}_3(\theta_3)\frac{\partial \mathbf{C}_2}{\partial \theta_2}\mathbf{C}_1(\theta_1) \\ \frac{\partial \mathbf{C}}{\partial \theta_1} &= \mathbf{C}_3(\theta_3)\mathbf{C}_2(\theta_2)\frac{\partial \mathbf{C}_1}{\partial \theta_1}.\end{aligned} \tag{42}$$

Using Equation (39), we can show that $\frac{\partial \mathbf{C}_i}{\partial \theta_i}$ has a very simple expression. Notice that if $\boldsymbol{\phi}$ has the structure $\boldsymbol{\phi} = \theta_i \mathbf{1}_i$, then

$$\left[[\boldsymbol{\phi}]_\times (\mathbf{1} - \mathbf{C})\mathbf{1}_i\right]_\times = \left[\theta_i [\mathbf{1}_i]_\times \mathbf{1}_i - \theta_i [\mathbf{1}_i]_\times \mathbf{C}_i \mathbf{1}_i\right]_\times = [\mathbf{0} - \mathbf{0}]_\times = \mathbf{0}, \tag{43}$$

where we have used the fact that $\mathbf{C}_i \mathbf{1}_i = \mathbf{1}_i$ since $\mathbf{C}_i$ rotates about $\mathbf{1}_i$. Thus, Equation (39) becomes

$$\frac{\partial \mathbf{C}_i}{\partial \theta_i} = \frac{\theta_i [\theta_i \mathbf{1}_i]_\times}{\|\boldsymbol{\theta}\|^2}\mathbf{C}_i = \frac{\theta_i^2 [\mathbf{1}_i]_\times}{\theta_i^2}\mathbf{C}_i = [\mathbf{1}_i]_\times \mathbf{C}_i, \tag{44}$$

which combined with Equation (42) results in the required simple expression for the Jacobian. This completes the derivation.

3.3 Small Angle-Axis Perturbations

An alternative formulation is to keep our representation of rotations in matrix form (this will avoid Gimbal lock) but to change our update rule. Instead, let's look for a three-parameter small 'angle-axis' vector $\delta\boldsymbol{\phi}$, such that we can update an operating point, $\mathbf{C}_{op}$ in the following way.

Recall that the Euler's theorem tell's us that any rotation can be represented as an axis of rotation, $\hat{\mathbf{n}}$ and a rotation about that axis, ϕ. Combining both into a single vector, $\boldsymbol{\phi} = \phi\hat{\mathbf{n}}$, we can convert this to a rotation matrix using the Euler-Rodriguez formula:

$$\mathbf{C}(\boldsymbol{\phi}) = \mathbf{1} + \sin\phi\,[\hat{\mathbf{n}}]_\times + (1 - \cos\phi)\,[\hat{\mathbf{n}}]_\times^2. \tag{45}$$

If $\phi = \delta\phi$ is small, we have

$$\mathbf{C}(\delta\boldsymbol{\phi}) \approx \mathbf{1} + \delta\phi\,[\hat{\mathbf{n}}]_\times = \mathbf{1} + [\delta\boldsymbol{\phi}]_\times. \tag{46}$$

Recall that to linearize a non linear function of vector quantities, we used the expression '$\mathbf{x} = \mathbf{x}_{op} + \delta\mathbf{x}$'. We can write down an analogous expression for rotations using left perturbations:

$$\mathbf{C} = \mathbf{C}(\delta\boldsymbol{\phi})\mathbf{C}_{op} \approx (\mathbf{1} + [\delta\boldsymbol{\phi}]_\times)\mathbf{C}_{op} \tag{47}$$

Now we can substitute the rotation for the perturbed rotation, and derive a Jacobian in the following way:

$$\begin{aligned}
\mathbf{e}_i(\delta\boldsymbol{\phi}) &\approx (\mathbf{1} + [\delta\boldsymbol{\phi}]_\times)\mathbf{C}_{op}\mathbf{u}_i - \mathbf{v}_i && (48)\\
&= \mathbf{C}_{op}\mathbf{u}_i - \mathbf{v}_i + [\delta\boldsymbol{\phi}]_\times \mathbf{C}_{op}\mathbf{u}_i && (49)\\
&= \mathbf{e}_i(\mathbf{C}_{op}) \underbrace{- [\mathbf{C}_{op}\mathbf{u}_i]_\times}_{\mathbf{J}_{e_i}} \delta\boldsymbol{\phi} && (50)\\
&= \mathbf{e}_i(\mathbf{C}_{op}) + \mathbf{J}_{e_i}\delta\boldsymbol{\phi}. && (51)
\end{aligned}$$

where we have used the fact that $[\mathbf{a}]_\times \mathbf{b} = -[\mathbf{b}]_\times \mathbf{a}$. Now we can update our operating point as $\mathbf{C}_{op} \leftarrow \mathbf{C}(\delta\boldsymbol{\phi})\mathbf{C}_{op}$ until convergence.

Another way to see this is to derive the Jacobian a bit more directly. Consider that for a 'small' rotation, we can use our small-angle approximation to derive that,

$$\frac{\partial \mathbf{C}(\delta\boldsymbol{\phi})\mathbf{u}}{\partial \delta\boldsymbol{\phi}} = \frac{\partial}{\partial \delta\boldsymbol{\phi}}(\mathbf{1} + [\delta\boldsymbol{\phi}]_\times)\mathbf{u} = \frac{\partial}{\partial \delta\boldsymbol{\phi}}(\mathbf{u} - [\mathbf{u}]_\times \delta\boldsymbol{\phi}) = -[\mathbf{u}]_\times, \tag{52}$$

This then allows us to write:

$$\begin{aligned}
\frac{\partial \mathbf{C}\mathbf{u}}{\partial \delta\boldsymbol{\phi}} &= \frac{\partial \mathbf{C}(\delta\boldsymbol{\phi})\mathbf{C}_{op}\mathbf{u}}{\partial \delta\boldsymbol{\phi}} && (53)\\
&= -[\mathbf{C}_{op}\mathbf{u})]_\times, && (54)
\end{aligned}$$

which we can use as our Jacobian to solve for $\delta\boldsymbol{\phi}$.

References

Barfoot, T. D. (2017). *State Estimation for Robotics*. Cambridge University Press.

Gallego, G. and Yezzi, A. (2015). A compact formula for the derivative of a 3-D rotation in exponential coordinates. *J. Math. Imaging Vis.*, 51(3):378–384.