NOTES

Batch Estimation

Amro Al Baali

May 21, 2021

Contents

1 Why this document?

This document will be used as a guide to batch estimation on design variables that live in Lie groups. It'll be assumed that the random variables follow a Gaussian distribution. Therefore, the state estimation problem boils down to some least squares problem.

2 Linear least squares

Linear least squares is a special type of unconstrained optimization problem. It has the form

$$\min_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{2}\left(\mathbf{A}\mathbf{x}-\mathbf{b}\right)^{\mathsf{T}}\left(\mathbf{A}\mathbf{x}-\mathbf{b}\right). \tag{1}$$

The error function is really *affince*, but that's how it is.

The objective function can be expanded to a quadratic form

$$\begin{aligned} J(\mathbf{x}) &= \frac{1}{2}\left(\mathbf{A}\mathbf{x}-\mathbf{b}\right)^{\mathsf{T}}\left(\mathbf{A}\mathbf{x}-\mathbf{b}\right) && (2)\\ &= \frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{x}-\mathbf{b}^{\mathsf{T}}\mathbf{A}\mathbf{x}+\frac{1}{2}\mathbf{b}^{\mathsf{T}}\mathbf{b} && (3)\end{aligned}$$

which is a *convex* function in $\mathbf{x}$. Furthermore, the objective function is *strongly* quadratic function if $\mathbf{A}^{\mathsf{T}}\mathbf{A}$ is positive definite, which occurs if $\mathbf{A}$ has full *column rank*. [1]

[1] This assumption is usually valid since it is common to have more measurements than the number of design variables.

If $\mathbf{A}$ is full rank, then the optimization problem (1) has a unique minimizer and is given by solving the linear system

$$\mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{x}^{\star} = \mathbf{A}^{\mathsf{T}}\mathbf{b}. \tag{4}$$

This equation is referred to as the *normal equations* [1].

There are efficient ways to solve (4) than inverting $\mathbf{A}^{\mathsf{T}}\mathbf{A}$. These include Cholesky and QR factorizations [2, 1].

2.1 Weighted least squares

Most often, weighted least squares optimization is used where the objective function is of the form

$$J(\mathbf{x}) = \frac{1}{2}\left(\mathbf{A}\mathbf{x}-\mathbf{b}\right)^{\mathsf{T}}\mathbf{W}\left(\mathbf{A}\mathbf{x}-\mathbf{b}\right), \tag{5}$$

where $\mathbf{W}=\mathbf{W}^{\mathsf{T}}>0$ is a symmetric positive definite matrix. This form arises naturally in the maximum a posteriori (MAP) estimator over normally distributed measurements $\underline{\mathbf{b}} \sim \mathcal{N}\left(\mathbf{b},\mathbf{\Sigma}\right)$, where the weight matrix is assigned $\mathbf{W}=\mathbf{\Sigma}^{-1}$ [3].

However, since the weight matrix is positive definite, then the Cholesky decomposition can be used so that

$$\mathbf{L}^{\mathsf{T}}\mathbf{L} = \mathbf{W}, \tag{6}$$

where $\mathbf{L}$ is a lower triangular matrix. The $\mathbf{L}$ matrix can then be inserted into the objective function (5) which results in

$$\begin{aligned} J(\mathbf{x}) &= \frac{1}{2}\left(\mathbf{A}\mathbf{x}-\mathbf{b}\right)^\mathsf{T}\mathbf{W}\left(\mathbf{A}\mathbf{x}-\mathbf{b}\right) && (7)\\ &= \frac{1}{2}\left(\mathbf{A}\mathbf{x}-\mathbf{b}\right)^\mathsf{T}\mathbf{L}^\mathsf{T}\mathbf{L}\left(\mathbf{A}\mathbf{x}-\mathbf{b}\right) && (8)\\ &= \frac{1}{2}\left(\mathbf{L}\mathbf{A}\mathbf{x}-\mathbf{L}\mathbf{b}\right)^\mathsf{T}\left(\mathbf{L}\mathbf{A}\mathbf{x}-\mathbf{L}\mathbf{b}\right) && (9)\\ &= \frac{1}{2}\left(\tilde{\mathbf{A}}\mathbf{x}-\tilde{\mathbf{b}}\right)^\mathsf{T}\left(\tilde{\mathbf{A}}\mathbf{x}-\tilde{\mathbf{b}}\right), && (10) \end{aligned}$$

where

$$\begin{aligned} \tilde{\mathbf{A}} &= \mathbf{L}\mathbf{A}, && (11)\\ \tilde{\mathbf{b}} &= \mathbf{L}\mathbf{b}. && (12) \end{aligned}$$

The resulting form is a standard linear least squares problem. Therefore, without loss of generality, the least squares problem can be assumed to be of the form (1).

3 Euclidean nonlinear least squares

Nonlinear least squares optimization problem is given by

$$\min_{\mathbf{x}\in\mathbf{R}^n} \frac{1}{2}\mathbf{e}(\mathbf{x})^\mathsf{T}\mathbf{e}(\mathbf{x}), \tag{13}$$

where $\mathbf{e} : \mathbb{R}^n \to \mathbb{R}^m$ is some nonlinear *error function*.

One way to solve this optimization iteratively is by linearizing the error function $\mathbf{e}(\mathbf{x})$ at some operating point $\bar{\mathbf{x}}$. This results in the Gauss-Newton equation [3, 1].

First, define the error on the optimizer. That is, let

$$\mathbf{x} = \bar{\mathbf{x}} + \delta\mathbf{x}, \tag{14}$$

where $\bar{\mathbf{x}}$ is some operating point.[2] Plugging the error definition (14) into the error function gives

[2] The operating point will be updated at each iteration.

$$\begin{aligned} \mathbf{e}(\mathbf{x}) &= \mathbf{e}(\bar{\mathbf{x}} + \delta\mathbf{x}) && (15)\\ &\approx \mathbf{e}(\bar{\mathbf{x}}) + \mathbf{J}\delta\mathbf{x}, && (16) \end{aligned}$$

where $\mathbf{J} \in \mathbb{R}^{m\times n}$ is the Jacobian of the error function $\mathbf{e}$ with respect to its design variables $\mathbf{x}$.

Plugging the perturbed error function (16) into the objective function gives

$$
\begin{aligned}
\tilde{J}(\delta\mathbf{x}) &\coloneqq J(\bar{\mathbf{x}} + \delta\mathbf{x}) && (17)\\
&= \frac{1}{2}\mathbf{e}(\bar{\mathbf{x}} + \delta\mathbf{x})^{\mathsf{T}}\mathbf{e}(\bar{\mathbf{x}} + \delta\mathbf{x}) && (18)\\
&\approx \frac{1}{2}\left(\mathbf{e}(\bar{\mathbf{x}}) + \mathbf{J}\delta\mathbf{x}\right)^{\mathsf{T}}\left(\mathbf{e}(\bar{\mathbf{x}}) + \mathbf{J}\delta\mathbf{x}\right) && (19)\\
&= \frac{1}{2}\delta\mathbf{x}^{\mathsf{T}}\mathbf{J}^{\mathsf{T}}\mathbf{J}\delta\mathbf{x} + \mathbf{e}(\bar{\mathbf{x}})^{\mathsf{T}}\mathbf{J}\delta\mathbf{x} + \frac{1}{2}\mathbf{e}(\bar{\mathbf{x}})^{\mathsf{T}}\mathbf{e}(\bar{\mathbf{x}}) && (20)
\end{aligned}
$$

which is a quadratic[3] approximation of the objective function.

[3] Quadratic in $\delta\mathbf{x}$.

If $\mathbf{J}$ has full column rank, then $\tilde{J}(\delta\mathbf{x})$ is a strongly quadratic function. The minimizer of $\tilde{J}(\delta\mathbf{x})$ is then given by the solving the system of equations

$$\mathbf{J}^{\mathsf{T}}\mathbf{J}\delta\mathbf{x}^{\star} = -\mathbf{J}^{\mathsf{T}}\mathbf{e}(\bar{\mathbf{x}}). \tag{21}$$

Again, this system of equations can be solved efficiently using Cholesky and QR factorizations.

Finally, using the error definition (16), the operating point can be updated using

$$\bar{\mathbf{x}}^{(j+1)} = \bar{\mathbf{x}}^{(j)} + \delta\mathbf{x}^{\star}, \tag{22}$$

where the superscript (j) is added to denote the Gauss-Newton iteration.

Gauss-Newton may perform poorly if the residual is large [4, 5], thus it's advisable to use line search methods when updating the operating point. That is, use

$$\bar{\mathbf{x}}^{(j+1)} = \bar{\mathbf{x}}^{(j)} + \alpha^{(j)}\delta\mathbf{x}^{\star}, \tag{23}$$

where $\alpha^{(j)}$ is a step-length computed using some heuristics like backtracking [4].

Another popular method is Levenberg-Marquardt [1, 4] which can be thought of as a damped version of Gauss-Newton.

4 Non-Euclidean nonlinear least squares

If the design variables[4] $\mathbf{X}$ are elements of non-Euclidean space, say elements of a Lie group G,[5] then the Euclidean addition operator '+' wouldn't necessarily work. This complicates things a little bit, and a more generalized approach must be taken.

[4] Elements of Lie groups will be denoted by upper case boldface letters.
[5] For example, the Special Euclidean group of rotations $SO(3)$.

The optimization problem will first take the form

$$\min_{\mathbf{X}\in G} \frac{1}{2}\mathbf{e}(\mathbf{X})^{\mathsf{T}}\mathbf{e}(\mathbf{X}), \tag{24}$$

where $\mathbf{e} : G \to \mathbf{R}^m$.

The optimization problem can still be solved using Gauss-Newton method. As discussed in the previous section, the Jacobian of the error function with respect to the design variable is needed to compute the search direction. But what does that look like when the design variable is an element of a Lie group? A generalized 'addition' operator $\oplus$' is used.

For example, let $\mathbf{X} \in SO(3)$ be an element of the $SO(3)$ group. Then the element $\mathbf{X}$ can be perturbed by the column matrix $\boldsymbol{\xi} \in \mathbb{R}^3$ that lives in the Euclidean space isomorphic to the tangent space $\mathfrak{so}(3)$. The left-invariant increment operator '$\overset{\text{LI}}{\oplus}$' over $SO(3)$ is then defined as

$$\mathbf{X} \overset{\text{LI}}{\oplus} \boldsymbol{\xi} \triangleq \mathbf{X}\exp(-\boldsymbol{\xi}^{\times}), \tag{25}$$

$(\cdot)^{\times}$ is the skew-symmetric cross-operator defined as [3]

$$\begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{bmatrix}^{\times} \triangleq \begin{bmatrix} 0 & -\xi_3 & \xi_2 \\ \xi_3 & 0 & -\xi_1 \\ -\xi_2 & \xi_1 & 0 \end{bmatrix}. \tag{26}$$

Using such notation, the error function $\mathbf{e}$ can be perturbed with respect to the Lie *algebra* element (i.e., the $\boldsymbol{\xi} \in \mathbb{R}^n$ element). The error function can then be perturbed to be of the form

$$\mathbf{e}(\mathbf{X}) = \mathbf{e}(\bar{\mathbf{X}} \oplus \delta\boldsymbol{\xi}) \tag{27}$$

$$\approx \mathbf{e}(\bar{\mathbf{X}}) + \mathbf{J}\delta\boldsymbol{\xi}, \tag{28}$$

where $\mathbf{J} \in \mathbb{R}^{m\times n}$. Sola [6] provides a good explanation of Jacobians over Lie groups elements. An example will be provided in the following sections.

Just as in the Euclidean case, the Jacobian can then be used to compute the search direction

$$\mathbf{J}^{\mathsf{T}}\mathbf{J}\delta\boldsymbol{\xi}^{\star} = -\mathbf{J}^{\mathsf{T}}\mathbf{e}(\bar{\mathbf{X}}). \tag{29}$$

Finally, the design variable can be updated can be updated using

$$\mathbf{X}^{(j+1)} = \mathbf{X}^{(j)} \oplus \left(\alpha^{(j)}\delta\boldsymbol{\xi}^{\star}\right), \tag{30}$$

where $\alpha^{(j)}$ is the step-length computed and $\oplus$ is the defined increment operation.[6]

[6] The perturbations are left-, right-, left-invariant-, and right-invariant-perturbations [6, 7].

5 *Example on* $SE(2)$

Consider a robot moving on a plan like the one displayed in Fig. 1. Let z_k be a *point* on attached to the robot and w be a reference *point*. Furthermore, let $\mathcal{F}_{b_k}$ be the robot reference frame (at time k), and $\mathcal{F}_a$ be the reference frame (e.g., some inertial frame). The robot can then be described by the *pose*

$$\mathbf{T}_{ab_k}^{z_k w} = \begin{bmatrix} \mathbf{C}_{ab_k} & \mathbf{r}_a^{z_k w} \\ \mathbf{0} & 1 \end{bmatrix} \in SE(2), \tag{31}$$

where $\mathbf{C}_{ab_k} \in SO(2)$ and $\mathbf{r}_a^{z_k w} \in \mathbb{R}^2$. For simplicity and visual clarity, notation

$$\mathbf{X}_k \coloneqq \mathbf{T}_{ab_k}^{z_k w} \tag{32}$$

will be used in this document.[7]

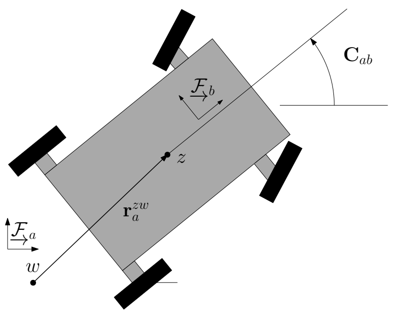

Figure 1: Robot with point z relative to point w. The body frame $\mathcal{F}_b$ is attached to the robot, and reference frame $\mathcal{F}_a$ is the reference frame.

[7] The $\mathbf{X}$ is usually reserved for 'design variable' so the Lie group should hopefully be inferred from the context.

5.1 *Process model*

The process model is given by [3]

$$\dot{\mathbf{X}}(t) = \mathbf{X}(t)\,\mathrm{Exp}(\mathbf{u}(t))\,\mathrm{Exp}(\underline{\boldsymbol{w}}(t)), \tag{33}$$

where

$$\mathbf{u}(t) = \begin{bmatrix} \omega_b^{ba}(t) \\ \mathbf{v}_a^{zw/a}(t) \end{bmatrix}, \tag{34}$$

are the *interoceptive* measurements, and

$$\underline{\boldsymbol{w}}(t) \sim \mathcal{GP}\left(\mathbf{0}, \boldsymbol{Q}\delta(t-t')\right), \tag{35}$$

is a zero-mean white-noise Gaussian Process (GP) with power spectral density $\boldsymbol{Q}$ [3].

The discrete-time kinematics can be described by [8]

$$\begin{aligned} \mathbf{X}_k &= \mathbf{X}_{k-1}\,\mathrm{Exp}(T_{k-1}\mathbf{u}_{k-1})\,\mathrm{Exp}(\underline{\mathbf{w}}_{k-1}) && (36) \\ &= \mathbf{X}_{k-1}\boldsymbol{\Xi}_{k-1}\underline{\mathbf{W}}_{k-1}, && (37) \end{aligned}$$

[8] Checkout [3] for a more complete derivation.

where T_{k-1} is the sampling period,

$$\begin{aligned} \boldsymbol{\Xi}_{k-1} &= \mathrm{Exp}(T_{k-1}\mathbf{u}_{k-1}) && (38) \\ &= \exp(T_{k-1}\mathbf{u}_{k-1}^{\wedge}), && (39) \end{aligned}$$

$$\mathbf{u}_{k-1}^{\wedge} = \begin{bmatrix} \left(\omega_{b_{k-1}}^{b_{k-1}a}\right)^{\times} & \mathbf{v}_a^{z_{k-1}w/a} \\ \mathbf{0} & 0 \end{bmatrix}, \tag{40}$$

and

$$\mathbf{w}_{k-1} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}_{k-1}\right), \tag{41}$$

is the process-noise and $\mathbf{Q}_{k-1}$ is the process-noise covariance matrix. [9]

[9] If Euler discretization is used to compute the process-noise covariance matrix $\mathbf{Q}_{k-1}$, then a correction must be made. Specifically, the covariance matrix would be given by

$$\begin{aligned} \mathbf{Q}_{k-1} &= \frac{T_{k-1}^2}{T_{k-1}}\boldsymbol{Q} && (42) \\ &= T_{k-1}\boldsymbol{Q}, && (43) \end{aligned}$$

where the $1/T_{k-1}$ scale factor corrects for intensity of the noise.

5.2 GPS measurement model

The GPS measurement model is given by

$$\mathbf{y}_k = \mathbf{r}_a^{z_k w} + \underline{\mathbf{n}}_k, \tag{44}$$

where

$$\underline{\mathbf{n}}_k \sim \mathcal{N}\left(\mathbf{0}, \mathbf{R}_k\right), \tag{45}$$

is measurement noise.

Another way to write the measurement function in terms of the Lie group element $\mathbf{X}_k \in SE(2)$ is

$$\begin{bmatrix} \mathbf{y}_k \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_a^{z_k w} + \underline{\mathbf{n}}_k \\ 1 \end{bmatrix} \tag{46}$$

$$= \begin{bmatrix} \mathbf{C}_{ab_k} & \mathbf{r}_a^{z_k w} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} + \begin{bmatrix} \underline{\mathbf{n}}_k \\ 0 \end{bmatrix} \tag{47}$$

$$= \mathbf{X}_k \mathbf{b} + \begin{bmatrix} \underline{\mathbf{n}}_k \\ 0 \end{bmatrix}. \tag{48}$$

References

[1] F. Dellaert and M. Kaess, “Factor Graphs for Robot Perception,” *Foundations and Trends in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017. [Online]. Available: http://www.nowpublishers.com/article/Details/ROB-043

[2] G. H. Golub and C. F. Van Loan, *Matrix computations*, fourth edition ed., ser. Johns Hopkins studies in the mathematical sciences. Baltimore: The Johns Hopkins University Press, 2013, oCLC: ocn824733531.

[3] T. D. Barfoot, *State Estimation for Robotics*. Cambridge: Cambridge University Press, 2017. [Online]. Available: http://ebooks.cambridge.org/ref/id/CBO9781316671528

[4] J. Nocedal and S. J. Wright, *Numerical optimization*, 2nd ed., ser. Springer series in operations research. New York: Springer, 2006, oCLC: ocm68629100.

[5] R. R. Fletcher, *Practical methods of optimization*, 2nd ed. Chichester ;: Wiley, 1987, section: xiv, 436 pages : illustrations ; 24 cm.

[6] J. Solà, J. Deray, and D. Atchuthan, “A micro Lie theory for state estimation in robotics,” *arXiv:1812.01537 [cs]*, Jun. 2019, arXiv: 1812.01537. [Online]. Available: http://arxiv.org/abs/1812.01537

[7] A. Barrau and S. Bonnabel, “Invariant Kalman Filtering,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 237–257, May 2018. [Online]. Available: https://www.annualreviews.org/doi/10.1146/annurev-control-060117-105010