NA 568 - Winter 2022

Optimization and Smoothing II

Maani Ghaffari

March 14, 2022

UNIVERSITY OF MICHIGAN

Thanks to Dr. Kasra Khosoussi (MIT) for sharing the notes.

Problem 2: Nonlinear Least Squares (NLS)

$f(x) = \frac{1}{2}\|r(x)\|^2$

- $r : \mathbb{R}^n \to \mathbb{R}^m$ $(m \geq n)$
- r is smooth, but not necessarily affine (i.e., $Ax + b$)
- $\|r(x)\|^2 = \sum_{i=1}^{m} r_i^2(x)$ where $r_i : \mathbb{R}^n \to \mathbb{R}$
- First-order Taylor expansion:

$$r_i(x) \approx r_i(x_0) + \nabla r_i(x_0)^{\mathsf{T}}(x - x_0)$$

- Stack r_i's:

$$r(x) \approx r(x_0) + \underbrace{J(x_0)}_{\text{Jacobian}}(x - x_0)$$

- Change of variable:

$$r(x_0 + d) \approx r(x_0) + J(x_0)d$$

1. Start from an initial guess x^0

for $k = 0,1,\cdots$ and until "convergence":

2. Linearize the residual at the current guess x^k

$$r(x^k + d) \approx r(x^k) + J(x^k)d$$

3. Solve the resulting linear least squares to find the step d

$$\underset{d}{\text{minimize}} \quad \|r(x^k) + J(x^k)d\|^2$$

$$(J_k^\mathsf{T} J_k)d = -J_k^\mathsf{T} r(x^k)$$

4. $x^{k+1} = x^k + d$

Iterratively Reweighted Least Squares (IRLS)

- We wish to minimize $f(x) = \frac{1}{2}\|r(x)\|_p^p$ (p-norm).

- This is no longer the least squares problem. So we can't use the Gauss-Newton algorithm.

- The trick is to convert it to a weighted least squares problem:

$$\|r(x)\|_p^p = r^\mathsf{T}(x) W r(x),$$

  and

$$W := \operatorname{diag}(|r_1(x)|^{p-2}, \ldots, |r_m(x)|^{p-2}).$$

- In practice, we start with $W = I$ and initialize x using the least squares solution. Then until convergence, we update W at each iteration and solve the least squares problem.

Example: Robust Linear Regression with ℓ_1-Regularizer

Given a dataset $\{(x_i,t_i)\}_{i=1}^N$, where x is the input and t is the target (output), we wish to find a linear model that explains data. The model is linear in weights with nonlinear basis functions.

$$y(x;w) = \sum_{j=0}^{N} w_j \phi_j(x) = w^\mathsf{T} \phi(x),$$

$$w = \mathrm{vec}(w_0,w_1,\ldots,w_N) \quad \text{and} \quad \phi = \mathrm{vec}(\phi_0,\phi_1,\ldots,\phi_N),$$

$\phi_0 = 1$ and w_0 is a bias parameter. A common basis function is the Gaussian (Squared Exponential) basis

$$\phi_j(x) = \exp\left(-\frac{(x-x_j)^2}{2s^2}\right),$$

The hyperparameter s is called the basis bandwidth or length-scale.

Example: Robust Linear Regression with ℓ_1-Regularizer

To find a robust (to outliers in data) and sparse estimate of $w \in \mathbb{R}^{N+1}$, we solve the following regularized problem.

$$\underset{w \in \mathbb{R}^{N+1}}{\text{minimize}} \quad \frac{1}{2} \sum_{i=1}^{N} |t_i - w^\mathsf{T} \phi(x_i)| + \frac{\lambda}{2} \|w\|_1,$$

or

$$\underset{w \in \mathbb{R}^{N+1}}{\text{minimize}} \quad f(w) := \frac{1}{2} \|t - \Phi w\|_1 + \frac{\lambda}{2} \|w\|_1,$$

where $t = \operatorname{vec}(t_1, \dots, t_N)$ and Φ is a $N \times N+1$ design matrix

$$\Phi = \begin{bmatrix} \phi^\mathsf{T}(x_1) \\ \vdots \\ \phi^\mathsf{T}(x_N) \end{bmatrix}.$$

Example: Robust Linear Regression with ℓ_1-Regularizer

$$f(w) = \frac{1}{2}\|t - \Phi w\|_1 + \frac{\lambda}{2}\|w\|_1$$

$$f(w) = \frac{1}{2}(t - \Phi w)^\mathsf{T} B(t - \Phi w) + \frac{\lambda}{2} w^\mathsf{T} G w$$

$$B := \operatorname{diag}(|t_1 - w^\mathsf{T}\phi(x_1)|^{-1}, \ldots, |t_N - w^\mathsf{T}\phi(x_N)|^{-1})$$

$$G := \operatorname{diag}(|w_0|^{-1}, \ldots, |w_N|^{-1})$$

$$\nabla f(w) = \Phi^\mathsf{T} B \Phi w - \Phi^\mathsf{T} B t + \lambda G w$$

$$\nabla f(w^\star) = 0 \Rightarrow \boxed{w^\star = (\Phi^\mathsf{T} B \Phi + \lambda G)^{-1} \Phi^\mathsf{T} B t}$$

Example: Robust Linear Regression with ℓ_1-Regularizer

Remark

To avoid division by zero, we use $\max(\delta, |t_i - w^\mathsf{T}\phi(x_i)|^{-1})$ *and* $\max(\delta, |w_j|^{-1})$. δ *is a small number, e.g.,* $1e-6$.

Remark

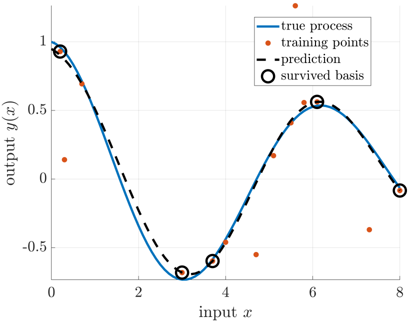
$\|\cdot\|_1$ *norm minimization is known as Least Absolute Deviation Regression and is robust to outliers in the data.The* ℓ_1*-regularizer results in a sparse weight vector.*

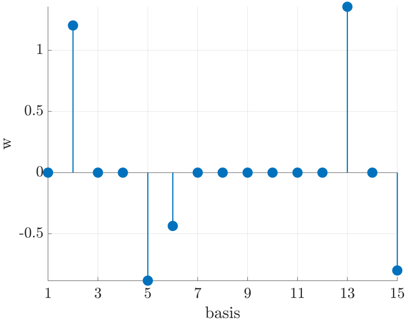
Remark

There is no guarantee that IRLS converges. If the solver hits the maximum number of iterations, do not trust the solution without an inspection!

Example: Robust Linear Regression with ℓ_1-Regularizer

See `lin_reg_ell1.m` for code.





Maximum likelihood Type Estimates (M-Estimates)

M-Estimation is a method for making an estimate robust to outliers. An M-Estimate of x is defined by

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \sum_{i=1}^{m} \rho\left(r_i(x)\right)$$

or by

$$\sum_{i=1}^{m} \frac{\partial \rho\left(r_i(x)\right)}{\partial x} = 0$$

Remark

A particular choice of $\rho(x) = -\log l(x)$ where $l(x)$ is the likelihood function leads to the ordinary maximum likelihood estimate.

Maximum likelihood Type Estimates (M-Estimates)

Using the chain rule we have

$$\sum_{i=1}^{m} \frac{\partial \rho\left(r_i(x)\right)}{\partial x} = \sum_{i=1}^{m} \frac{\partial \rho\left(r_i\right)}{\partial r_i} \cdot \frac{\partial r_i(x)}{\partial x} = 0$$

$$\sum_{i=1}^{m} \frac{\partial \rho\left(r_i\right)}{\partial r_i} \cdot \frac{r_i}{r_i} \cdot \frac{\partial r_i(x)}{\partial x} = \sum_{i=1}^{m} w(r_i) \frac{\partial r_i(x)}{\partial x} r_i = 0$$

where we defined $w(r_i) := \frac{\partial \rho(r_i)}{\partial r_i} \cdot \frac{1}{r_i}$.

Maximum likelihood Type Estimates (M-Estimates)

This allows us to redefine the problem using the following weighted least squares

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2}\sum_{i=1}^{m} w(r_i) r_i^2(x).$$

We can solve this problem by using IRLS.

Example: Robust Linear Regression via M-Estimation

To find an M-Estimate of $w \in \mathbb{R}^{N+1}$, we solve the following problem.

$$\underset{w \in \mathbb{R}^{N+1}}{\text{minimize}} \quad \sum_{i=1}^{N} \rho\left(t_i - w^{\mathsf{T}} \phi(x_i)\right),$$

We use the Cauchy loss function

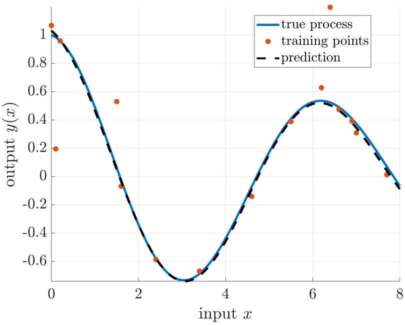
$$\rho(r) = \frac{\alpha^2}{2} \log(1 + \frac{r^2}{\alpha^2}) \quad \text{and} \quad \frac{\partial \rho}{\partial r} = \frac{r}{1 + \frac{r^2}{\alpha^2}}$$

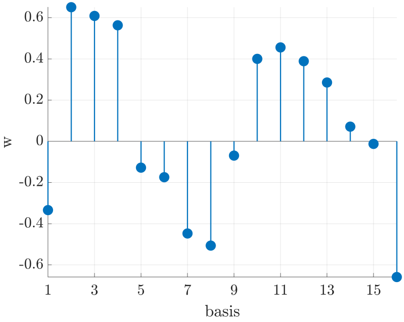
$$w(r) = \frac{\partial \rho\,(r)}{\partial r} \cdot \frac{1}{r} = \frac{1}{1 + \frac{r^2}{\alpha^2}}.$$

α is a parameter that controls where the loss begins to scale sublinearly.

Example: Robust Linear Regression via M-Estimation

See `lin_reg_m_estimation.m` for code.





- $\exp(A) := \sum_{k=0}^{\infty} \frac{A^k}{k!}$, and $\exp(0) = I$
- In matrix Lie groups, $\exp$ maps Lie algebra (i.e., $\mathfrak{se}(3)$ and $\mathfrak{so}(3)$) to Lie group (i.e., $\mathrm{SO}(3)$ and $\mathrm{SE}(3)$).
- $\mathfrak{se}(3)$ and $\mathfrak{so}(3)$ are vector spaces $\rightarrow$ basis "vectors" (a.k.a. generators)

$$\phi^\wedge \in \mathfrak{so}(3) \Leftrightarrow \phi^\wedge = \phi_1 G_1 + \phi_2 G_2 + \phi_3 G_3$$

where $\phi \in \mathbb{R}^3$ and

$$G_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, G_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, G_3 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

- $\phi^\wedge = (\phi)_\times \Rightarrow \phi^\wedge a = \phi \times a$

Similarly, for $\mathfrak{se}(3)$ consider $\phi \in \mathbb{R}^3$ and $\rho \in \mathbb{R}^3$ and the overloaded hat operator:

$$\begin{bmatrix} \phi \\ \rho \end{bmatrix}^\wedge \in \mathfrak{se}(3) \Leftrightarrow \begin{bmatrix} \phi \\ \rho \end{bmatrix}^\wedge = \phi_1 G_1 + \phi_2 G_2 + \phi_3 G_3 + \rho_1 G_4 + \rho_2 G_5 + \rho_3 G_6$$

where

$$G_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, G_2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, G_3 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$G_4 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, G_5 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, G_6 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$f(x) = \frac{1}{2}\|r(x_1, \cdots, x_n)\|^2$ where
$r : \mathcal{M}_1 \times \mathcal{M}_2 \times \cdots \times \mathcal{M}_n \to \mathbb{R}^m$

- e.g., $x_1 \in \mathcal{M}_2 = \mathbb{R}^3$ (3D point)
- e.g., $x_2 \in \mathcal{M}_1 = \mathrm{SE}(3)$ (pose)
- e.g., $x_3 \in \mathcal{M}_2 = \mathrm{SO}(3)$ (rotation)

- $x^{k+1} = x^k + d$ is not valid anymore (recall matrix groups are not closed under addition)

- Intuition: search along curves that live on the manifold.

- Gauss-Newton over $\mathbb{R}^n$

$$r(x^k + d) \approx r(x^k) + J_k d$$

$$J_k = \left.\frac{\partial r(x)}{\partial x}\right|_{x=x^k} = \left.\frac{\partial r(x^k + d)}{\partial d}\right|_{d=0}$$

- Gauss-Newton over $\mathrm{SO}(3)$ — $d \in \mathbb{R}^3$

$$r(x^k \exp(d^\wedge)) \approx r(x^k) + \mathfrak{J}_k d$$

$$\mathfrak{J}_k := \left.\frac{\partial r(x^k \exp(d^\wedge))}{\partial d}\right|_{d=0}$$

- Gauss-Newton over $\mathrm{SE}(3)$ — $d \in \mathbb{R}^6$

$$r(x^k \exp(d^\wedge)) \approx r(x^k) + \mathfrak{J}_k d$$

$$\mathfrak{J}_k := \left.\frac{\partial r(x^k \exp(d^\wedge))}{\partial d}\right|_{d=0}$$

Lift-Solve-Retract

Perturbation: $x^{k+1} = x^k \exp(d^\wedge)$

1 Lift:

$$g : \mathbb{R}^{n_d} \to \mathbb{R}^m : d \mapsto r(x^k \exp(d^\wedge))$$

e.g., $n_d = 3$ in $\mathrm{SO}(3)$ and $n_d = 6$ in $\mathrm{SE}(3)$.

$$g(d) \approx g(0) + \left.\frac{\partial g(d)}{\partial d}\right|_{d=0} d \qquad \text{Taylor at } d = 0$$

$$r(x^k \exp(d^\wedge)) \approx r(x^k) + \mathfrak{J}_k d$$

2 Solve:

$$\underset{d}{\text{minimize}} \qquad \frac{1}{2}\|r(x^k \exp(d^\wedge))\|^2 \approx \frac{1}{2}\|r(x^k) + \mathfrak{J}_k d\|^2$$

linear least squares $\Rightarrow$ normal equations

$$d = -(\mathfrak{J}_k^\mathsf{T}\mathfrak{J}_k)^{-1}\mathfrak{J}_k^\mathsf{T}(x^k)$$

3 Retract:

$$x^{k+1} = x^k \exp(d^\wedge)$$

Tips for computing $\mathfrak{J}_k$

- For $\|d\| \approx 0$:

$$\exp(d^\wedge) \approx I + d^\wedge$$

- Express $d^\wedge = \sum_i d_i G_i$ and take derivatives w.r.t. each d_i (i.e., columns of $\mathfrak{J}_k$).

- Chain rule and vectorization:

$$\mathfrak{J}_k = \left. \frac{\partial r(x^k \exp(d^\wedge))}{\partial d} \right|_{d=0} = \left. \frac{\partial r(s)}{\partial s} \right|_{s=\mathrm{vec}(x^k)} \left. \frac{\partial \mathrm{vec}(x^k \exp(d^\wedge))}{\partial d} \right|_{d=0}$$

Recall: Baker-Campbell-Hausdorff Series

For $X, Y, Z \in \mathfrak{g}$ with sufficiently small norm, the equation $\exp(X)\exp(Y) = \exp(Z)$ has a power series solution for Z in terms of repeated Lie bracket of X and Y. The beginning of the series is:

$$Z = X + Y + \frac{1}{2}[X,Y] + \frac{1}{12}[X,[X,Y]] + \frac{1}{12}[Y,[Y,X]] + \cdots .$$

First-Order Approximation using BCH

- The adjoint representation of a Lie group is a linear map that captures the non-commutative structure of the group.
- The following properties from adjoint representation and BCH formula for the first order approximation are useful.

$$X \exp(\xi^\wedge) X^{-1} = \exp((\mathrm{Ad}_X \xi)^\wedge)$$
$$\implies X \exp(\xi^\wedge) = \exp((\mathrm{Ad}_X \xi)^\wedge) X$$
$$\implies \exp(\xi^\wedge) X = X \exp((\mathrm{Ad}_{X^{-1}} \xi)^\wedge)$$

- In the above equations $X \in \mathcal{G}$ and $\xi^\wedge \in \mathfrak{g}$.

First-Order Approximation using BCH

- The BCH formula can be used to compound two matrix exponentials.
- If both terms are small, by keeping the first two terms ignoring the higher order terms, we have:

$$
\begin{aligned}
\mathrm{BCH}({\xi_1}^\wedge, {\xi_2}^\wedge) &= {\xi_1}^\wedge + {\xi_2}^\wedge + \mathrm{HOT}, \\
\exp({\xi_1}^\wedge)\exp({\xi_2}^\wedge) &\approx \exp({\xi_1}^\wedge + {\xi_2}^\wedge).
\end{aligned}
$$

First-Order Approximation using BCH

- When both terms are not small and assuming ξ is small, by keeping the linear terms in ξ, we have:

$$\log(\exp(r^\wedge)\exp(\xi^\wedge))^\vee \approx r + J_r^{-1}(r)\xi,$$

$$\log(\exp(\xi^\wedge)\exp(r^\wedge))^\vee \approx r + J_l^{-1}(r)\xi.$$

  where J_r and J_l are the right and left Jacobians of the Lie group $\mathcal{G}$, respectively.

- The left and right Jacobians are related through the adjoint map,

$$J_r(\xi) = \mathrm{Ad}_{\exp(-\xi^\wedge)} J_l(\xi).$$

Example with Multiple Variables

- Consider $\|r(x_1,x_2)\|^2$ where $x_1 \in \mathbb{R}^3$ and $x_2 \in \mathrm{SO}(3)$

$$\|r(x_1^k + d_1, x_2^k \exp(d_2{}^\wedge))\|^2 \approx \|r(x_1^k, x_2^k) + J_{1,k} d_1 + \mathfrak{J}_{2,k} d_2\|^2$$

$$J_{1,k} := \left.\frac{\partial r(x)}{\partial x_1}\right|_{x=(x_1^k, x_2^k)} = \left.\frac{\partial r(x_1^k + d_1, x_2^k)}{\partial d_1}\right|_{d_1=0}$$

$$\mathfrak{J}_{2,k} := \left.\frac{\partial r(x_1^k, x_2^k \exp(d_2{}^\wedge))}{\partial d_2}\right|_{d_2=0}$$

- Solve the resulting linear least squares;
- Retract: $x_1^{k+1} = x_1^k + d_1$ and $x_2^{k+1} = x_2^k \exp(d_2{}^\wedge)$.

Example: Pose Synchronization using GN

- Suppose a process model on matrix Lie group $\mathrm{SE}(3)$ where the state at any two successive keyframes at times-steps i and j is related using an input such as $U_i \in \mathrm{SE}(3)$. The deterministic process model is as follows.

$$X_j = f_{u_i}(X_i) := X_i U_i$$

- Substituting in the noisy process model we have

$$X_j = f_{u_i}(X_i) \exp(v_k{}^{\wedge})$$

  where $v_k \sim \mathcal{N}(0_{6,1}, \Sigma_v)$.

- Taking the $\log(\cdot)$ from both sides we arrive at the residual term.

$$r_{ij} := \log(f_{u_i}(X_i)^{-1} X_j)^{\vee} = \log(U_i^{-1} X_i^{-1} X_j)^{\vee}$$

Example: Pose Synchronization using GN

To compute the Jacobian, we perturb the residual using an incremental term, a retraction $X \leftarrow X \exp(\xi^\wedge)$, and apply a first order approximation as follow.

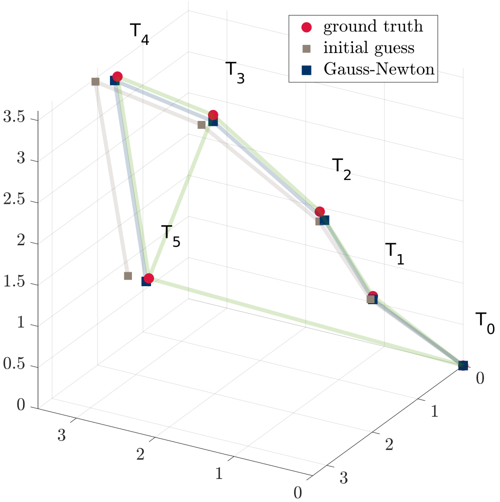
$$
\begin{aligned}
r_{ij}(X_i \exp({\xi_i}^\wedge)) &= \log(U_i^{-1}(X_i \exp({\xi_i}^\wedge))^{-1} X_j)^\vee \\
&= \log(U_i^{-1} \exp(-{\xi_i}^\wedge) X_i^{-1} X_j)^\vee \\
&= \log(U_i^{-1} X_i^{-1} X_j \exp((-\mathrm{Ad}_{X_j^{-1} X_i} \xi_i)^\wedge))^\vee \\
&\approx \log(U_i^{-1} X_i^{-1} X_j)^\vee - J_r^{-1}(\log(U_i^{-1} X_i^{-1} X_j)^\vee) \mathrm{Ad}_{X_j^{-1} X_i} \xi_i \\
&= r_{ij} - J_r^{-1}(r_{ij}) \mathrm{Ad}_{X_j^{-1} X_i} \xi_i
\end{aligned}
$$

$$
\begin{aligned}
r_{ij}(X_j \exp({\xi_j}^\wedge)) &= \log(U_i^{-1} X_i^{-1} X_j \exp({\xi_j}^\wedge))^\vee \\
&\approx \log(U_i^{-1} X_i^{-1} X_j)^\vee + J_r^{-1}(\log(U_i^{-1} X_i^{-1} X_j)^\vee) \xi_j \\
&= r_{ij} + J_r^{-1}(r_{ij}) \xi_j
\end{aligned}
$$

where $J_r(\cdot)$ is the right Jacobian of $\mathrm{SE}(3)$.

Example: Pose Synchronization using GN

See `pose_sync.m` for code.



Remark

This problem has further interesting structures and it is possible to develop global solvers. See:

Rosen, D. M., Carlone, L., Bandeira, A. S., & Leonard, J. J. (2019). SE-Sync: A certifiably correct algorithm for synchronization over the special Euclidean group. The International Journal of Robotics Research, 38(2–3), 95–125.

https://github.com/david-m-rosen/SE-Sync

- GICP cost function:

$$f_{\texttt{GICP}}(T) := \sum_{k}^{n} \|x_k - T \cdot y_k\|_{C_k}^2 := \sum_{k}^{n} r_k^{\mathsf{T}} r_k$$

- $r_k = L_k^{-1}(x_k - T \cdot y_k)$ and $C_k = L_k L_k^{\mathsf{T}}$

- Solve for the parameter $T \in \mathrm{SE}(3)$:

$$T^{\mathsf{OPT}} = \underset{T \in \mathrm{SE}(3)}{\arg\min}\, f_{\texttt{GICP}}(T)$$

To compute the Jacobian, we perturb the residual using $T \leftarrow \exp(\xi^\wedge)T$ and apply a first order approximation of the exponential map $\exp(\xi^\wedge) \approx I + \xi^\wedge$.

$$\begin{aligned} r_k\left(\exp(\xi^\wedge)T\right) &= L_k^{-1}(x_k - \exp(\xi^\wedge)T \cdot y_k) \\ &\approx L_k^{-1}(x_k - (I + \xi^\wedge)T \cdot y_k) \\ &= L_k^{-1}(x_k - T \cdot y_k) - L_k^{-1}(\xi^\wedge T \cdot y_k) \\ &= r_k(T) - L_k^{-1}(\xi^\wedge \cdot z_k), \end{aligned}$$

where we define $z_k := T \cdot y_k$ to be the source point after applying the transformation T.

$$r_k\left(\exp(\xi^\wedge)T\right) \approx r_k(T) - L_k^{-1}(\xi^\wedge \cdot z_k).$$

To find the Jacobian we need to solve $-L_k^{-1}(\xi^\wedge \cdot z_k) = J_k \xi$.

$$\begin{aligned} L_k^{-1}(-\xi^\wedge \cdot z_k) &= L_k^{-1}(-\phi^\wedge z_k - \rho) \\ &= L_k^{-1}(z_k^\wedge \phi - \rho) \\ &= L_k^{-1}\begin{bmatrix} z_k^\wedge & -I \end{bmatrix} \xi. \end{aligned}$$

We learn that $J_k = L_k^{-1}\begin{bmatrix} z_k^\wedge & -I \end{bmatrix}$ (a 3×6 matrix).

See `gicp_SE3.m` and `registration_example.m` for code.