NA 568 - Winter 2024

Optimization and Smoothing

Maani Ghaffari

March 11, 2024

Notation

- $A \succeq B \quad \Leftrightarrow \quad A - B$ is positive semidefinite
- $A \succ B \quad \Leftrightarrow \quad A - B$ is positive definite
- $\|x\| := \|x\|_2 := \sqrt{x^\mathsf{T} x}$
- $\|x\|_1 := |x_1| + \cdots + |x_n| = \sum_{i=1}^n |x_i|$
- $\|x\|_p := (|x_1|^p + \cdots + |x_n|^p)^{1/p} = (\sum_{i=1}^n |x_i|^p)^{1/p}$
- $x \cdot y := \langle x,y \rangle := x^\mathsf{T} y$
- Norm ball $\mathcal{B}(x_c,r) := \{x \in \mathbb{R}^n : \|x - x_c\| \leq r\}$

- Objective function $f : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$ and decision variable $x \in \mathbb{R}^n$

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x)$$

- Global minimum

$$f(x^\star) \leq f(x) \qquad \underbrace{\forall x \in \mathbb{R}^n}_{\text{global}}$$

- Local minimum

$$f(x^*) \leq f(x) \qquad \underbrace{\forall x \in \mathcal{B}_{r>0}(x^*)}_{\text{local}}$$

Structures: Smoothness

- $f : \mathbb{R}^n \to \mathbb{R}$

$$\begin{array}{c} \text{gradient} \in \mathbb{R}^n \\ \uparrow \\ \nabla f(x) \end{array} := \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \qquad \begin{array}{c} \text{Hessian} \in \mathbb{S}^n \\ \uparrow \\ H(x) \end{array} := \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \, \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \, \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \, \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \, \partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 f}{\partial x_n \, \partial x_1} & \frac{\partial^2 f}{\partial x_n \, \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

- f is continuously differentiable (and analytic) $\to$ Taylor's Theorem

$$f(x+d) = f(x) + \nabla f(x)^\mathsf{T} d + \frac{1}{2} \, d^\mathsf{T} H(x) \, d + o(\|d\|^2)$$

Recognizing Local Minima

- First-order *necessary* condition

$$\nabla f(x) = 0$$

- Second-order *necessary* condition

$$\nabla f(x) = 0 \quad \text{and} \quad H(x) \succeq 0$$

- Second-order *sufficient* condition

$$\nabla f(x) = 0 \quad \text{and} \quad H(x) \succ 0$$

Linear Least Squares (LS)

$f(x) = \frac{1}{2}\|Ax - b\|^2$

- Gradient: $\nabla f(x) = A^\mathsf{T} Ax - A^\mathsf{T} b$
- Hessian: $H(x) = A^\mathsf{T} A$

Assumption

- $A \in \mathbb{R}^{m \times n}$ *and* $b \in \mathbb{R}^m$
- $m \geq n \Leftrightarrow A$ *is a tall matrix*
- $\mathrm{rank}(A) = n$ *(i.e., columns of* A *are linearly independent)*

$\nabla f(x) = 0$ is necessary and sufficient for global optimality.
Unique minimizer iff $\mathrm{rank}(A) = n$.

$A \in \mathbb{R}^{m \times n}$ has linearly independent columns $\Leftrightarrow A^\mathsf{T} A \succ 0$.

$$\nabla f(x^\star) = 0 \Rightarrow x^\star = (A^\mathsf{T} A)^{-1} A^\mathsf{T} b$$

- A is full (column) rank $\Rightarrow A^\mathsf{T} A \succ 0$ is invertible
- Solve a linear system — "Normal Equations"

$$(A^\mathsf{T} A) x^\star = A^\mathsf{T} b$$

- Cholesky ($A^\mathsf{T} A = L L^\mathsf{T}$) or QR ($A = QR$) factorization

Example: LS Target Tracking (Smoothing)

A target is moving in a 2D plane. The ownship position is known and fixed at the origin. We have access to noisy measurements that directly observe the target 2D coordinates at any time step. There is no knowledge of the target motion, but we assume target is close to its previous location to constrain the state.

$$x_k = x_{k-1} + w_k,$$

$$z_k = H_k x_k + v_k,$$

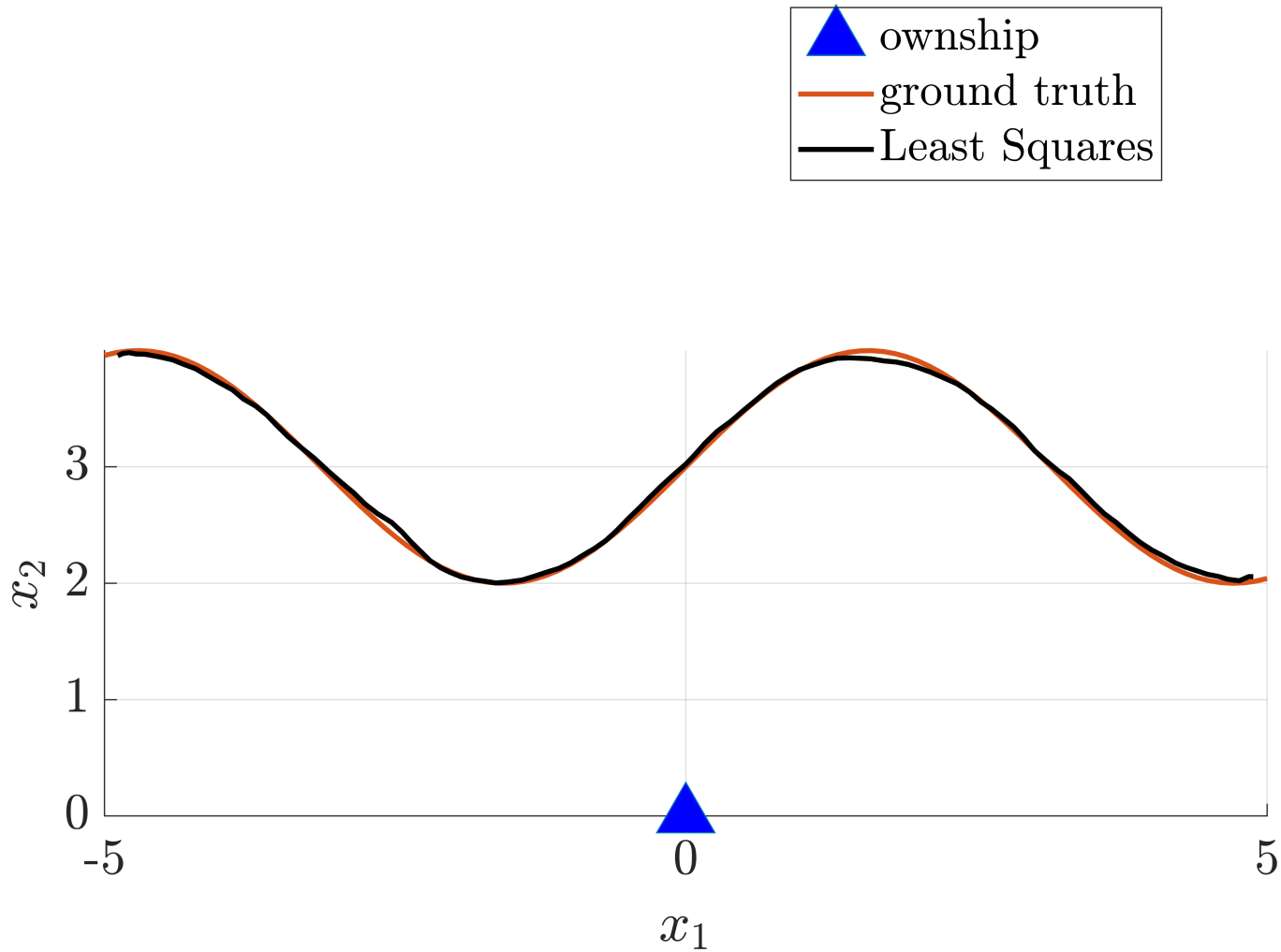
$$H_k = I,$$

$$Q_k = \mathrm{Cov}[w_k] = 0.03^2 I,$$

$$R_k = \mathrm{Cov}[v_k] = 0.05^2 I.$$

Example: LS Target Tracking (Smoothing)

See `ls_single_target.m` for code.



Example: Linear Regression with ℓ_2-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: Linear Regression with ℓ_2-Regularizer

To find $w \in \mathbb{R}^{N+1}$, we solve the following regularized least squares problem.

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

or

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

where $t = \operatorname{vec}(t_1, \ldots, t_N)$ 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: Linear Regression with ℓ_2-Regularizer

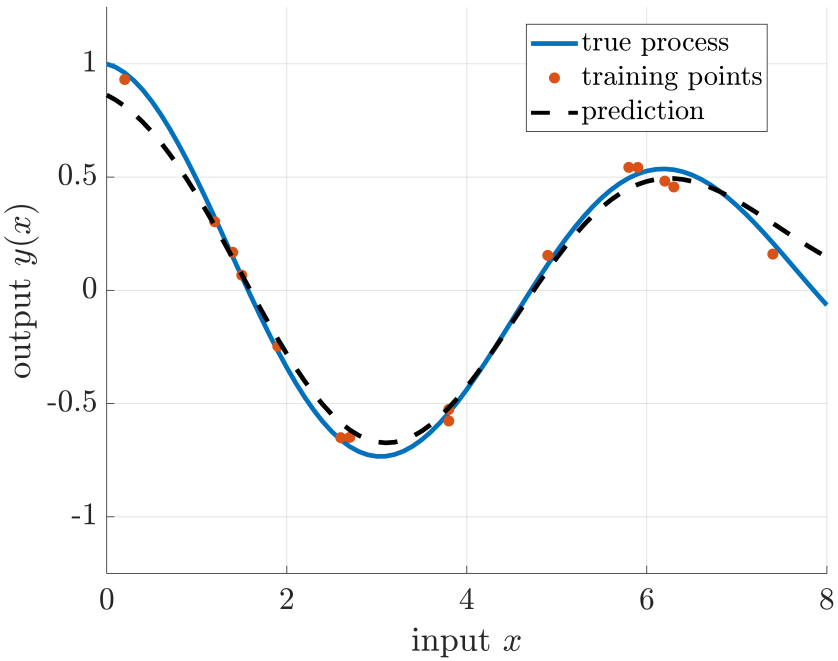
$$f(w) = \frac{1}{2}\|t - \Phi w\|^2 + \frac{\lambda}{2}\|w\|^2$$

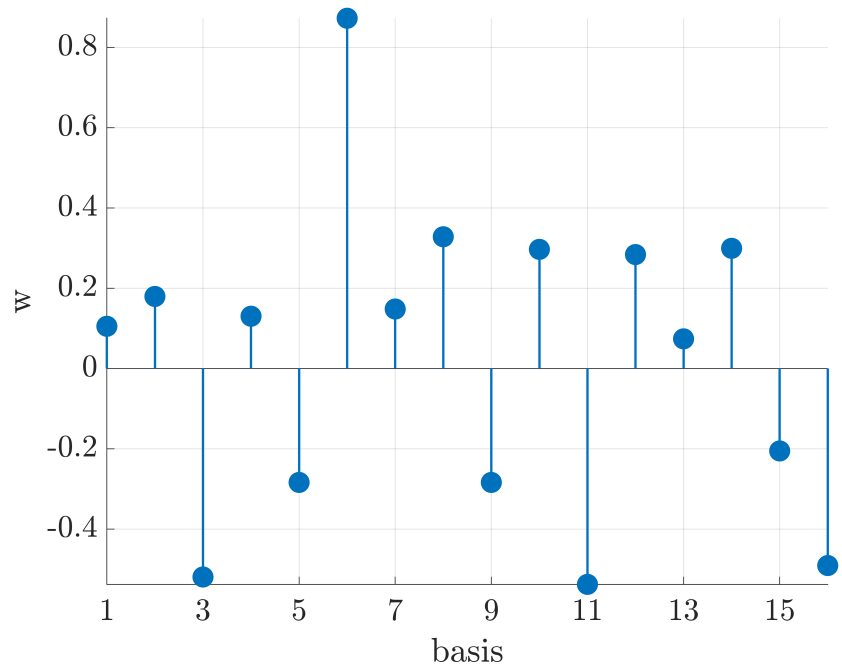
$$\nabla f(w) = \Phi^\mathsf{T}\Phi w - \Phi^\mathsf{T} t + \lambda w$$

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

Example: Linear Regression with ℓ_2-Regularizer

See `lin_reg.m` for code.





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:

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

Jacobian

$$J(x) := \frac{\partial r(x)}{\partial x} = \begin{bmatrix} \frac{\partial r_1}{\partial x_1} & \frac{\partial r_1}{\partial x_2} & \dots & \frac{\partial r_1}{\partial x_n} \\ \frac{\partial r_2}{\partial x_1} & \frac{\partial r_2}{\partial x_2} & \dots & \frac{\partial r_2}{\partial x_n} \\ \vdots & \vdots & \dots & \vdots \\ \frac{\partial r_m}{\partial x_1} & \frac{\partial r_m}{\partial x_2} & \dots & \frac{\partial r_m}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

Gauss-Newton (GN)

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

Levenberg-Marquardt

- adds a penalty term $\lambda\|d\|^2$ to penalize a large d

$$\frac{1}{2}d^{\mathsf{T}}(H_k + \lambda I)d^{\mathsf{T}} + g_k^{\mathsf{T}}d + f(x^k)$$

- nonlinear least squares:
  - Levenberg $(J_k^{\mathsf{T}}J_k + \lambda I)d = -J_k^{\mathsf{T}}r(x^k)$
  - Marquardt $(J_k^{\mathsf{T}}J_k + \lambda\,\mathrm{diag}(J_k^{\mathsf{T}}J_k))d = -J_k^{\mathsf{T}}r(x^k)$
- interpolation between gradient descent (large λ) and Gauss-Newton (small λ)

Direct methods for solving linear systems

- Ultimately need to solve $Ad = b$ where $A \in \text{Sym}(n)$ and $b \in \mathbb{R}^n$
  - e.g., in Gauss-Newton

    $$A = (J_k^\mathsf{T} J_k) \text{ and } b = -J_k^\mathsf{T} r(x^k)$$

  - e.g., in Levenberg-Marquardt

    $$A = (J_k^\mathsf{T} J_k + \lambda I) \text{ and } b = -J_k^\mathsf{T} r(x^k)$$

- Do not invert A! (and do not associate with people who invert A)
  - will lose structure (e.g., A may be sparse but A^{-1} will be generally dense)
  - numerical stability

- We consider two direct methods based on Cholesky and QR factorizations.

Cholesky solver

- Solving triangular systems is fast/easy (forward/backward substitution):

$$\begin{bmatrix} \ell_{11} & 0 & 0 \\ \ell_{12} & \ell_{22} & 0 \\ \ell_{13} & \ell_{23} & \ell_{33} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

- Cholesky decomposition (assuming $A \succ 0$)
  i. $A = LL^\mathsf{T}$ where L is lower triangular and thus L^T is upper triangular

$$L \underbrace{L^\mathsf{T} d}_{y} = b$$

  ii. solve $Ly = b$ via forward substitution
  iii. solve $L^\mathsf{T} d = y$ via backward substitution

QR Solver

- QR factorization of $A = QR$
  - $Q \in \mathbb{R}^{m \times n}$ and $Q^\mathsf{T} Q = I_n$
  - $R \in \mathbb{R}^{n \times n}$ is upper triangular

- Solve $Rd = Q^\mathsf{T} c$ instead of $Ad = b$

$$Ad = b \Rightarrow Q^\mathsf{T} QRd = Q^\mathsf{T} c \quad Q^\mathsf{T} Q = I_n$$

$$\Rightarrow \boxed{Rd = Q^\mathsf{T} c}$$ solve via backward substitution

QR vs. Cholesky

- ✓ QR does not need to form A - works with J_k or $\begin{bmatrix} J_k \\ \sqrt{\lambda} I_n \end{bmatrix}$
- ✓ Better numerical stability than Cholesky
- × Slower than Cholesky

Iteratively 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.

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

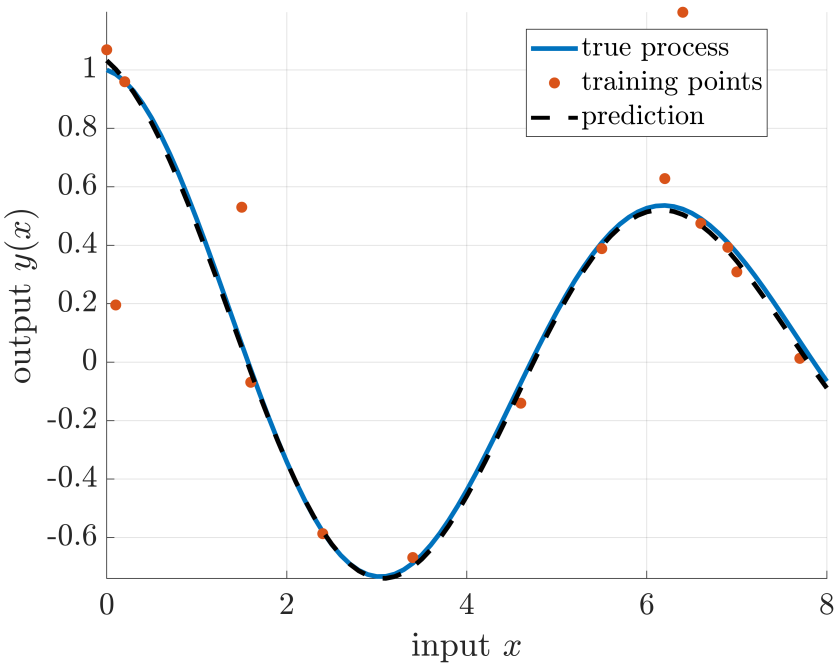
$$\rho(r) = \frac{\alpha^2}{2} \log(1 + \frac{r^2}{\alpha^2}) \quad \text{and} \quad \frac{\partial \rho}{\partial r} = \frac{r}{1 + \frac{r^2}{\alpha^2}}$$

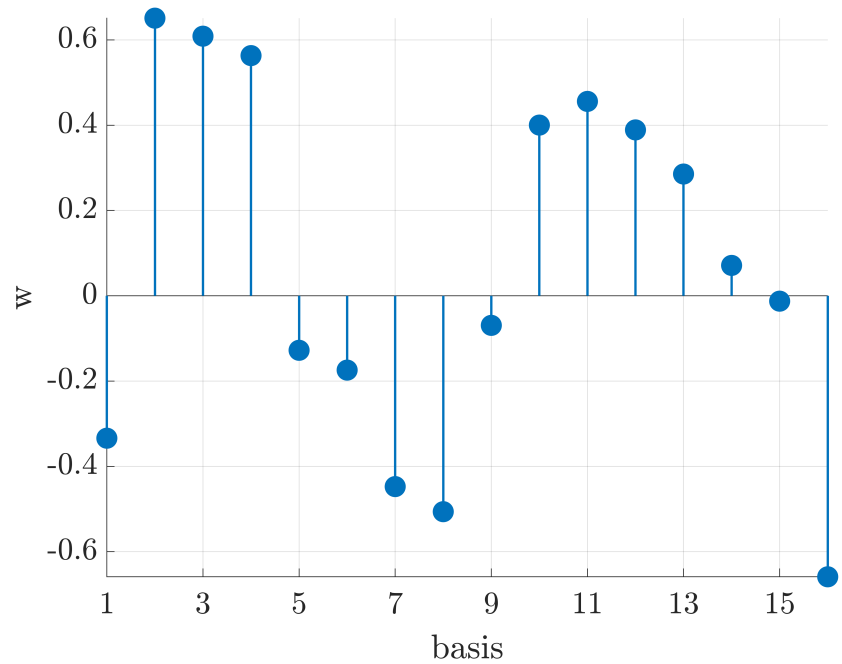
$$w(r) = \frac{\partial \rho\,(r)}{\partial r} \cdot \frac{1}{r} = \frac{1}{1 + \frac{r^2}{\alpha^2}}.$$

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.





Recall: Matrix Lie Groups

- $\exp(A) := \sum_{k=0}^{\infty} \frac{A^k}{k!}$, 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 (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

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

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

Least Squares over Matrix Lie Groups

$f(x) = \frac{1}{2}\|r(x_1, \cdots, x_n)\|^2$ 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.

Lift-Solve-Retract

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

1. Lift:

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

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

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

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

2. Solve:

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

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:

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

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.

$$
\begin{aligned}
& X \exp(\xi) X^{-1} = \exp(\mathrm{Ad}_X \xi) \\
\implies & X \exp(\xi) = \exp(\mathrm{Ad}_X \xi) X \\
\implies & \exp(\xi) X = X \exp(\mathrm{Ad}_X \xi)
\end{aligned}
$$

- 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:

$$\mathrm{BCH}(\xi_1^\wedge, \xi_2^\wedge) = \xi_1^\wedge + \xi_2^\wedge + \mathrm{HOT},$$

$$\exp(\xi_1)\exp(\xi_2) \approx \exp(\xi_1 + \xi_2).$$

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:

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

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

  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,

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

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

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

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

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

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

Example: Pose Synchronization using GN

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

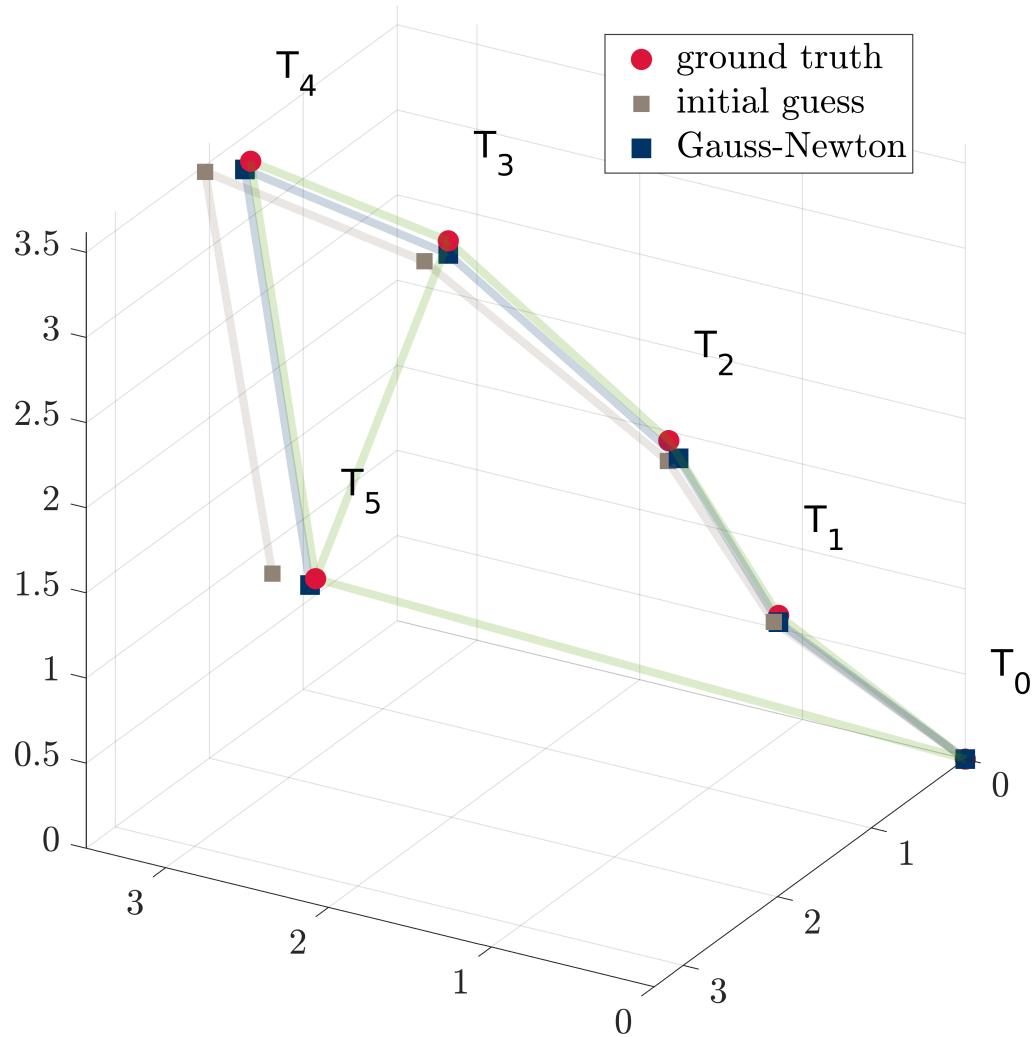
$$
\begin{aligned}
r_{ij}(X_i \exp(\xi_i)) &= \log(U_i^{-1}(X_i \exp(\xi_i)^{-1} X_j) \\
&= \log(U_i^{-1} \exp(-\xi_i) X_i^{-1} X_j) \\
&= \log(U_i^{-1} X_i^{-1} X_j \exp(-\mathrm{Ad}_{X_j^{-1} X_i} \xi_i)) \\
&\approx \log(U_i^{-1} X_i^{-1} X_j) - J_r^{-1}(\log(U_i^{-1} X_i^{-1} X_j)) \mathrm{Ad}_{X_j^{-1}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

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

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

Jacobians

Suppose we can parameterize a transformation $g \in \mathcal{G}$ using a vector of local coordinates $q \in \mathbb{R}^n$ such that $g(q) : \mathbb{R}^n \to \mathcal{G}$.

The Jacobian J is a matrix that relates the rate of change $\frac{d}{dt}q = \dot{q}$ to $\xi^\wedge \in \mathfrak{g}$, i.e.,

$$\xi = J\dot{q}.$$

From the reconstruction equations, we can consider the velocity ξ in the body frame or the spatial frame, which are related via the adjoint map $\xi_s = \mathrm{Ad}_g \xi_b$.

Then we obtain the following definitions for the left and right Jacobians.

$$\left(g^{-1}\dot{g}\right)^{\vee} = \xi_b = J_r(q)\dot{q},$$
$$\left(\dot{g}g^{-1}\right)^{\vee} = \xi_s = J_l(q)\dot{q}.$$

Remark

The particular choice of the group exponential coordinates leads to the group Jacobians presented earlier.