Unconstrained Optimization and Least Squares

Kasra Khosoussi

MIT 16.485: VNAV

Visual Navigation for Autonomous Vehicles

Massachusetts Institute of Technology

Review/Motivation: Point Estimation

find the "best" estimate of $\mathbf{x}$ given noisy measurements $\mathbf{z}$

1. Maximum likelihood (ML) estimate

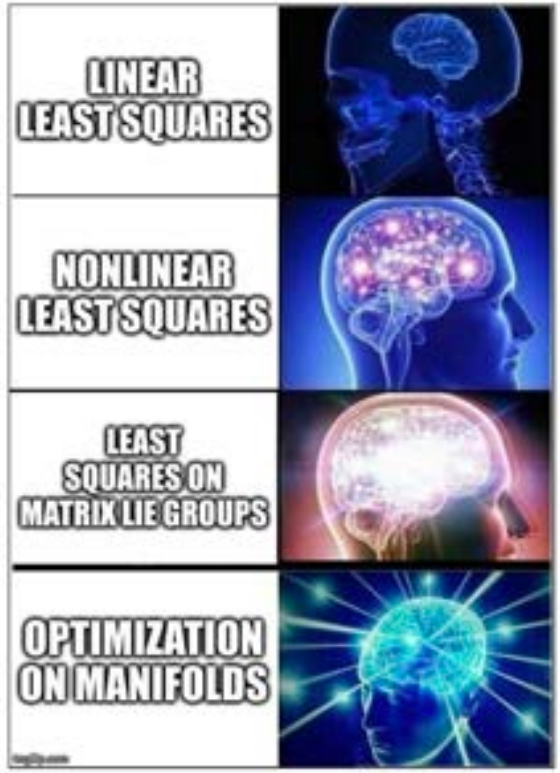
$$\hat{\mathbf{x}}_{\text{MLE}} \in \arg\max_{\mathbf{x}} \; p(\mathbf{z}|\mathbf{x})$$

2. Maximum-a-posteriori (MAP) estimate

$$\hat{\mathbf{x}}_{\text{MAP}} \in \arg\max_{\mathbf{x}} \; p(\mathbf{x}|\mathbf{z}) = \arg\max_{\mathbf{x}} \; p(\mathbf{z}|\mathbf{x})\, p(\mathbf{x})$$

- Under additive Gaussian noise and Gaussian priors ➔ **least squares**

Our Plan



- **Today's Lecture**:

  Unconstrained Optimization and Least Squares

- **Next Lectures**:

  Introduction to Optimization on Manifolds and Least Squares on Matrix Lie Groups

Basic Terminology

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

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

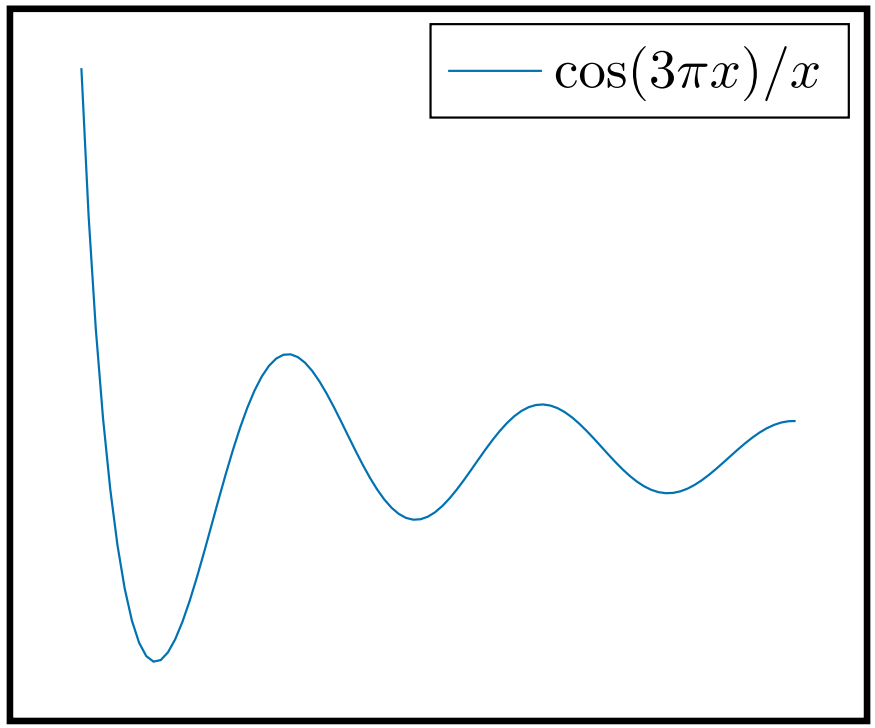
- $\mathbf{x}^\star$ is a *global* minimizer and $f(\mathbf{x}^\star)$ is a *global* minimum iff $f(\mathbf{x}^\star) \leq f(\mathbf{x})$

$$\text{for all } \mathbf{x} \in \mathbb{R}^n$$

- $\mathbf{x}^*$ is a *local* minimizer and $f(\mathbf{x}^*)$ is a *local* minimum iff $f(\mathbf{x}^*) \leq f(\mathbf{x})$

$$\text{for all } \mathbf{x} \in \mathcal{B}(\mathbf{x}^*, r) \text{ with positive radius } r$$

Example



Mixed blessing

- Many problems can be formulated as optimization problems
- Sometimes hard problems ➔ easy-looking optimization problems
- Deciding global (even local) optimality is NP-hard in general

Fun example: Fermat's Last *Theorem* (1637-1995)

Example 1. *Fermat's Last Theorem.* Some of the most difficult unsolved problems in mathematics can be posed as problems of finding a global minimum in a smooth nonconvex NLP. Consider Fermat's last theorem, unresolved since the year 1637. It states that there exists no positive integer solution (x, y, z) to the equation

$$x^n + y^n = z^n$$

when n is an integer ≥ 3 (here, $x, y, z \in \mathbb{R}^1$). Even though this conjecture has been shown to be true for several individual values of n, in general, it remains open. Obviously, Fermat's last theorem is not true iff the global minimum objective value in the following NLP is 0 and attained where α is a positive penalty parameter.

$$\begin{aligned} \text{minimize} \quad & (x^n + y^n - z^n)^2 \\ & + \alpha((1-\cos(2\pi x))^2 + (1-\cos(2\pi y))^2 + (1-\cos(2\pi z))^2 \\ & + (1-\cos(2\pi n))^2) \\ \text{subject to} \quad & x, y, z \geq 1, \quad n \geq 3. \end{aligned}$$

Murty and Kabadi (1987)

Structure

Structures: Smoothness

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

$$\begin{array}{c} \text{gradient} \in \mathbb{R}^n \\ \uparrow \\ \nabla f(\mathbf{x}) \end{array} \triangleq \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \vdots \\ \dfrac{\partial f}{\partial x_n} \end{bmatrix} \qquad \begin{array}{c} \text{Hessian} \in \text{Sym}(n) \\ \uparrow \\ \mathbf{H}(\mathbf{x}) \end{array} \triangleq \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \, \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \, \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2 \, \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2 \, \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n \, \partial x_1} & \dfrac{\partial^2 f}{\partial x_n \, \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

- f is (sufficiently) smooth and analytic ➔ Taylor expansion

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

Second-order Taylor approximation

- Local quadratic approximation

$$f(\mathbf{x}_0 + \mathbf{d}) \approx \hat{f}_{\mathbf{x}_0}(\mathbf{d}) \triangleq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top \mathbf{d} + \frac{1}{2}\,\mathbf{d}^\top \mathbf{H}(\mathbf{x}_0)\,\mathbf{d}$$

- Another interpretation after change of variables $\mathbf{x} \triangleq \mathbf{x}_0 + \mathbf{d}$

$$f(\mathbf{x}) \approx \hat{f}(\mathbf{x}) \triangleq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}\,(\mathbf{x} - \mathbf{x}_0)^\top \mathbf{H}(\mathbf{x}_0)\,(\mathbf{x} - \mathbf{x}_0)$$

Recognizing Local Minima

- First-order necessary condition for $f \in C^1(\mathbb{R}^n)$

$$\nabla f(\mathbf{x}) = \mathbf{0}$$

- Second-order necessary condition for $f \in C^2(\mathbb{R}^n)$

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

- Second-order sufficient condition for $f \in C^2(\mathbb{R}^n)$

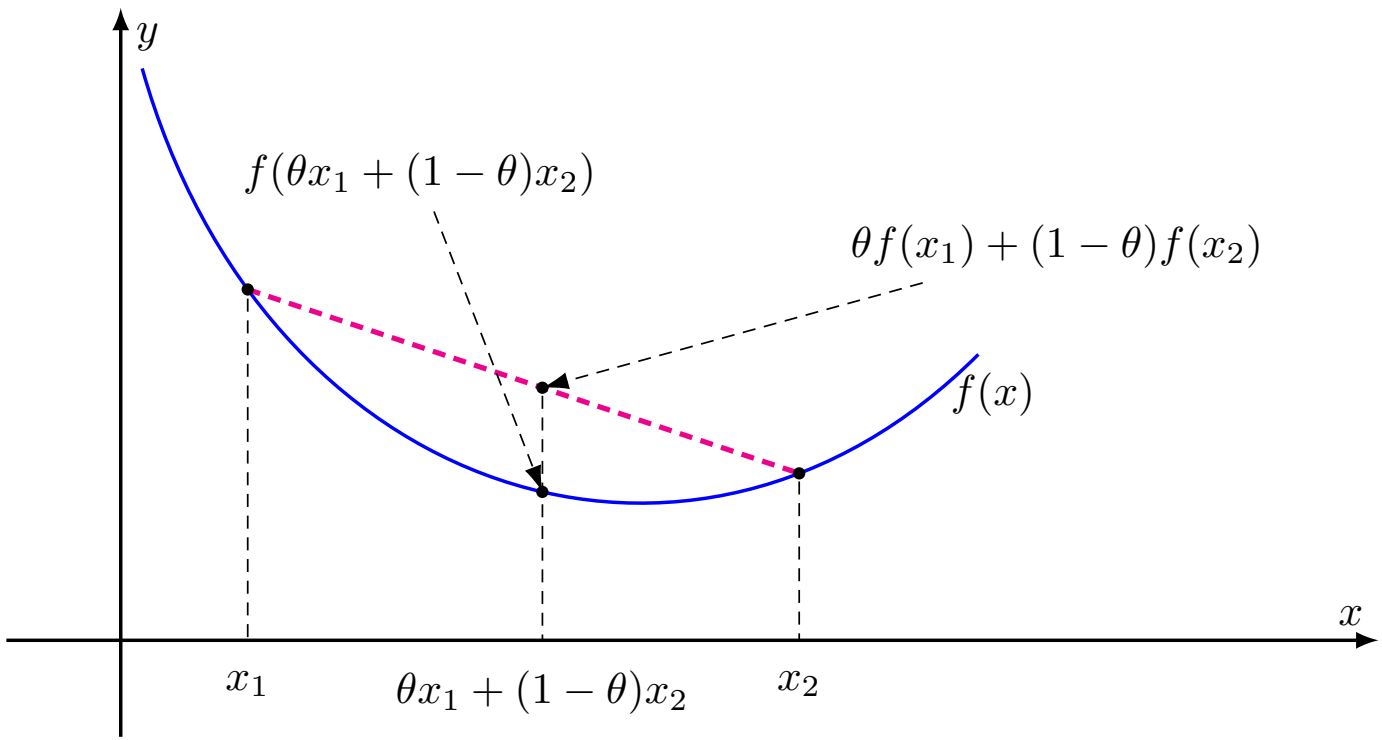
$$\nabla f(\mathbf{x}) = \mathbf{0} \quad \text{and} \quad \mathbf{H}(\mathbf{x}) \succ \mathbf{0}$$

Structure: Convexity

$f : \mathbb{R}^n \to \mathbb{R}$ ($\mathrm{dom} f = \mathbb{R}^n$) is convex iff:

1. For all $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$ and all $\theta \in [0,1]$:

$$f(\theta \mathbf{x}_1 + (1-\theta)\mathbf{x}_2) \leq \theta f(\mathbf{x}_1) + (1-\theta) f(\mathbf{x}_2)$$



Structure: Convexity

$f : \mathbb{R}^n \to \mathbb{R}$ ($\mathrm{dom} f = \mathbb{R}^n$) is convex iff:

1. For all $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$ and all $\theta \in [0,1]$:

$$f(\theta \mathbf{x}_1 + (1-\theta)\mathbf{x}_2) \leq \theta f(\mathbf{x}_1) + (1-\theta) f(\mathbf{x}_2)$$

2. First-order condition (differentiable f): For all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$:

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x})$$

   What happens when $\nabla f(\mathbf{x}) = \mathbf{0}$?

3. Second-order condition (twice differentiable f): For all $\mathbf{x} \in \mathbb{R}^n$:

$$\mathbf{H}(\mathbf{x}) \succeq \mathbf{0}$$

Problem 1: Linear Least-Squares

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

- $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$
- Gradient

$$\nabla f(\mathbf{x}) = \mathbf{A}^\top(\mathbf{A}\mathbf{x} - \mathbf{b})$$

- Hessian

$$\mathbf{H}(\mathbf{x}) = \mathbf{A}^\top \mathbf{A}$$

- Claim: f is convex (why?)
- Claim: $\nabla f(\mathbf{x}) = \mathbf{0}$ is necessary and sufficient for global optimality (why?)
- Claim: unique minimizer iff $\mathrm{rank}(\mathbf{A}) = n$ (why?)
- i.e., $\mathbf{A}$ is a tall matrix ($m \geq n$) with full column rank
- Just solve the normal equations:

$$(\mathbf{A}^\top \mathbf{A})\mathbf{x} = \mathbf{A}^\top \mathbf{b}$$

Problem 2: Nonlinear Least Squares (NLS)

$$f(\mathbf{x}) = \frac{1}{2}\|\mathbf{r}(\mathbf{x})\|^2 \qquad \mathbf{r} : \mathbb{R}^n \to \mathbb{R}^m \qquad \text{where } m \geq n$$

- $\mathbf{r}$ is smooth, but not necessarily affine anymore
- $\|\mathbf{r}(\mathbf{x})\|^2 = \sum_{i=1}^{m} r_i^2(\mathbf{x})$ where $r_i : \mathbb{R}^n \to \mathbb{R}$
- First-order Taylor:

$$r_i(\mathbf{x}) \approx r_i(\mathbf{x}_0) + \nabla r_i(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0)$$

- Stack r_i's:

$$\mathbf{r}(\mathbf{x}) \approx \mathbf{r}(\mathbf{x}_0) + \underbrace{\mathbf{J}(\mathbf{x}_0)}_{\text{Jacobian}} (\mathbf{x} - \mathbf{x}_0)$$

- Same story, different narrative (change of variable):

$$\mathbf{r}(\mathbf{x}_0 + \mathbf{d}) \approx \mathbf{r}(\mathbf{x}_0) + \mathbf{J}(\mathbf{x}_0)\mathbf{d}$$

Jacobian

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

Gauss-Newton

1. start from an initial guess $\mathbf{x}^0$

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

2. linearize the residual at the current guess $\mathbf{x}^k$

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

3. solve the resulting linear least squares to find the step $\mathbf{d}$

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

$$(\mathbf{J}_k^\top \mathbf{J}_k)\mathbf{d} = -\mathbf{J}_k^\top \mathbf{r}(\mathbf{x}^k)$$

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

Newton's Method

- Common Idea in Optimization: locally approximate the objective function around $\mathbf{x}^k$ by a simpler (often quadratic) model function
- "Optimal" Choice $\rightarrow$ Taylor (here $\mathbf{g}_k \triangleq \nabla f(\mathbf{x}^k)$ and $\mathbf{H}_k \triangleq \nabla^2 f(\mathbf{x}^k)$)

$$f(\mathbf{x}^k + \mathbf{d}) \approx m_k(\mathbf{d}) \triangleq \underbrace{f(\mathbf{x}^k)}_{\text{constant}} + \mathbf{g}_k^\top \mathbf{d} + \frac{1}{2}\mathbf{d}^\top \mathbf{H}_k \mathbf{d}$$

- $m_k(\mathbf{d})$ gives the local quadratic approximation
- Choose a $\mathbf{d}$ that is a stationary point (hopefully, minimizer) for $m_k(\mathbf{d})$:

$$\nabla m_k(\mathbf{d}) = \mathbf{0} \Rightarrow \mathbf{H}_k \mathbf{d} + \mathbf{g}_k = \mathbf{0}$$

- Well-defined (i.e., actually moves towards a local minimum) if $\mathbf{H}_k \succ \mathbf{0} \Rightarrow \boxed{\mathbf{d} = -\mathbf{H}_k^{-1}\mathbf{g}_k}$ and $\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{d}$

✗ In general, has no preference for local minima over other types of stationary points (local maxima or saddle points)

✓ **Very** fast ("quadratic") convergence near stationary points

Newton vs. Gauss-Newton

- Recall Nonlinear Least Squares (NLS) $f_{\text{NLS}}(\mathbf{x}) = \frac{1}{2}\|\mathbf{r}(\mathbf{x})\|^2$
- Verify that the gradient and Hessian of f_{NLS} are given by:

$$\nabla f_{\text{NLS}}(\mathbf{x}^k) =: \mathbf{g}_k = \mathbf{J}_k^\top \mathbf{r}(\mathbf{x}^k)$$

$$\nabla^2 f_{\text{NLS}}(\mathbf{x}^k) =: \mathbf{H}_k = \mathbf{J}_k^\top \mathbf{J}_k + \underbrace{\sum_{i=1}^{m} r_i(\mathbf{x}^k)\nabla^2 r_i(\mathbf{x}^k)}_{\mathbf{S}}$$

- Thus (pure) Newton step for NLS will be the solution of

$$(\mathbf{J}_k^\top \mathbf{J}_k + \mathbf{S})\mathbf{d} = -\mathbf{J}_k^\top \mathbf{r}(\mathbf{x}^k)$$

- Now compare this to Gauss-Newton step:

$$(\mathbf{J}_k^\top \mathbf{J}_k)\mathbf{d} = -\mathbf{J}_k^\top \mathbf{r}(\mathbf{x}^k)$$

Cont'd

$\Rightarrow$ Gauss-Newton (in NLS) approximates the Hessian matrix in Newton's method – less expensive than computing the Hessian

$\Rightarrow$ Gauss-Newton step will be "close" to Newton step (e.g., fast convergence close to a solution) if $\mathbf{S}$ is "small"

- e.g., when $\mathbf{r}$ is "close" to an affine function
- and/or when the residuals are "close" to zero at a local solution

$\Rightarrow$ $\mathbf{J}_k^\top \mathbf{J}_k$ in Gauss-Newton is a PSD approximation of Hessian in NLS — $\mathbf{S}$ can make Hessian non-PSD – (Thanks, Guass!)

Globalization Strategies

- Pure Newton's or Gauss-Newton iterations may fail to converge at all even to stationary points depending on the initial guess!
- Partly due to the fact that our model functions (and the linearization of residual in Gauss-Newton) are valid approximations of the original function **close** to $\mathbf{x}^k$, but these algorithms in pure form disregard this.
- $\mathbf{d}$ can be "too large" — we may end up increasing the objective!

$\Rightarrow$ Need safeguards ("globalization strategies") to converge (hopefully, to a **local** minimum) from any initial guess

- Note that "globalization" has nothing to do with "global" optimality here (that'd be way too ambitious for generic non-convex objectives)
- Two approaches: (i) Line Search and (ii) Trust-Region Methods

Globalization Strategies: Line Search

- **Idea:** $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha \mathbf{d}$ where α is the step size
- **Plan:** First find a "good" direction, then choose a "good" step size

"Good" Direction

$\mathbf{d}$ is a descent direction if $\exists \alpha_0 > 0$ such that $f(\mathbf{x}^{k+1}) < f(\mathbf{x}^k)$ for all $\alpha \in (0, \alpha_0)$

- Recall the definition of directional derivative at $\mathbf{x}^k$ along direction $\mathbf{d}$

$$Df(\mathbf{x}^k)[\mathbf{d}] \triangleq \lim_{\alpha \to 0} \frac{1}{\alpha} \Big(f(\mathbf{x}^k + \alpha \mathbf{d}) - f(\mathbf{x}^k) \Big) = \mathbf{g}_k^\top \mathbf{d}$$

Theorem

If the directional derivative along $\mathbf{d}$ is negative $\Rightarrow$ $\mathbf{d}$ is a descent direction

- What does this say about the angle between such $\mathbf{d}$'s and $\mathbf{g}_k$?

Cont'd

1. Pick a descent direction $\mathbf{d}$
   - Newton's direction is a descent direction if $\mathbf{H}_k \succ \mathbf{0}$ (why?)
   - Gauss-Newton direction is a descent direction if $\mathbf{J}_k$ is full column rank (why?)
   - More generally, $\mathbf{d} = -\mathbf{B}\mathbf{g}_k$ is a descent direction for any $\mathbf{B} \succ \mathbf{0}$ (why?)

2. Find the "best" step size α (exact line search) by solving

$$\underset{\alpha \in \mathbb{R}_{\geq 0}}{\text{minimize}} \;\; f(\mathbf{x}^k + \alpha \mathbf{d})$$

- In practice $\rightarrow$ inexact (backtracking) line search until achieve "sufficient" descent suffices: shrink an initial α until satisfy Armijo (or Wolfe) condition
- Resulting algorithms are sometimes called *damped* Newton/Gauss-Newton

Globalization Strategies: Trust-Region

- **Plan**: Pick max step size first, then choose the step $\mathbf{d}$
- How much do we **trust** our local approximate quadratic model away from $\mathbf{d} = \mathbf{0}$ (i.e., away from $\mathbf{x}^k$)?

1. Pick a maximum step size Δ_k
2. Pick $\mathbf{d}$ by solving the trust-region subproblem

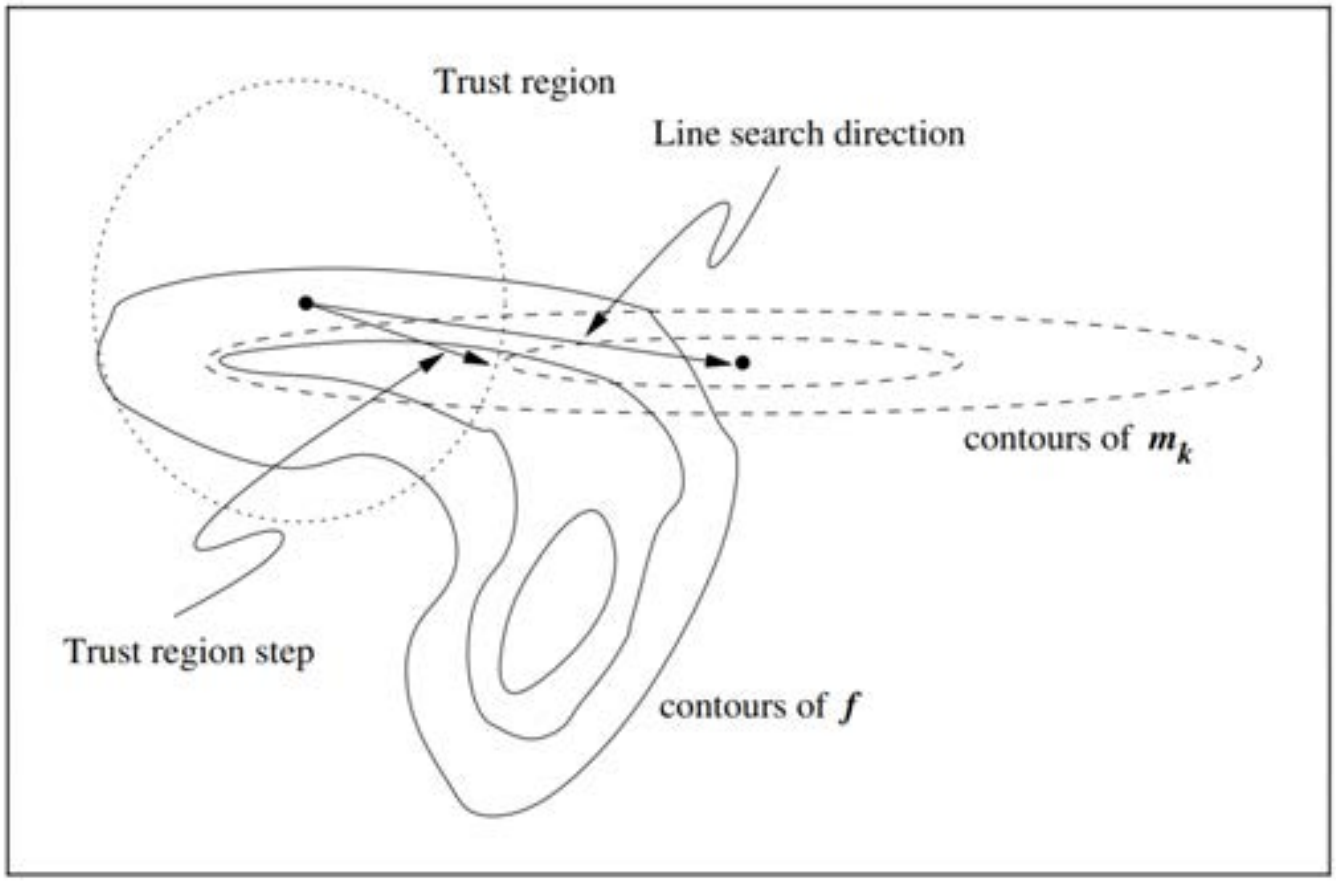
$$\underset{\mathbf{d}}{\text{minimize}} \;\; m_k(\mathbf{d}) \;\; \text{such that} \;\; \|\mathbf{d}\| \leq \Delta_k$$

3. Quantify and re-evaluate our trust on the model (i.e., Δ) based on

$$\frac{\text{actual reduction}}{\text{expected reduction}} = \frac{f(\mathbf{x}^k) - f(\mathbf{x}^k + \mathbf{d})}{m_k(\mathbf{0}) - m_k(\mathbf{d})}$$

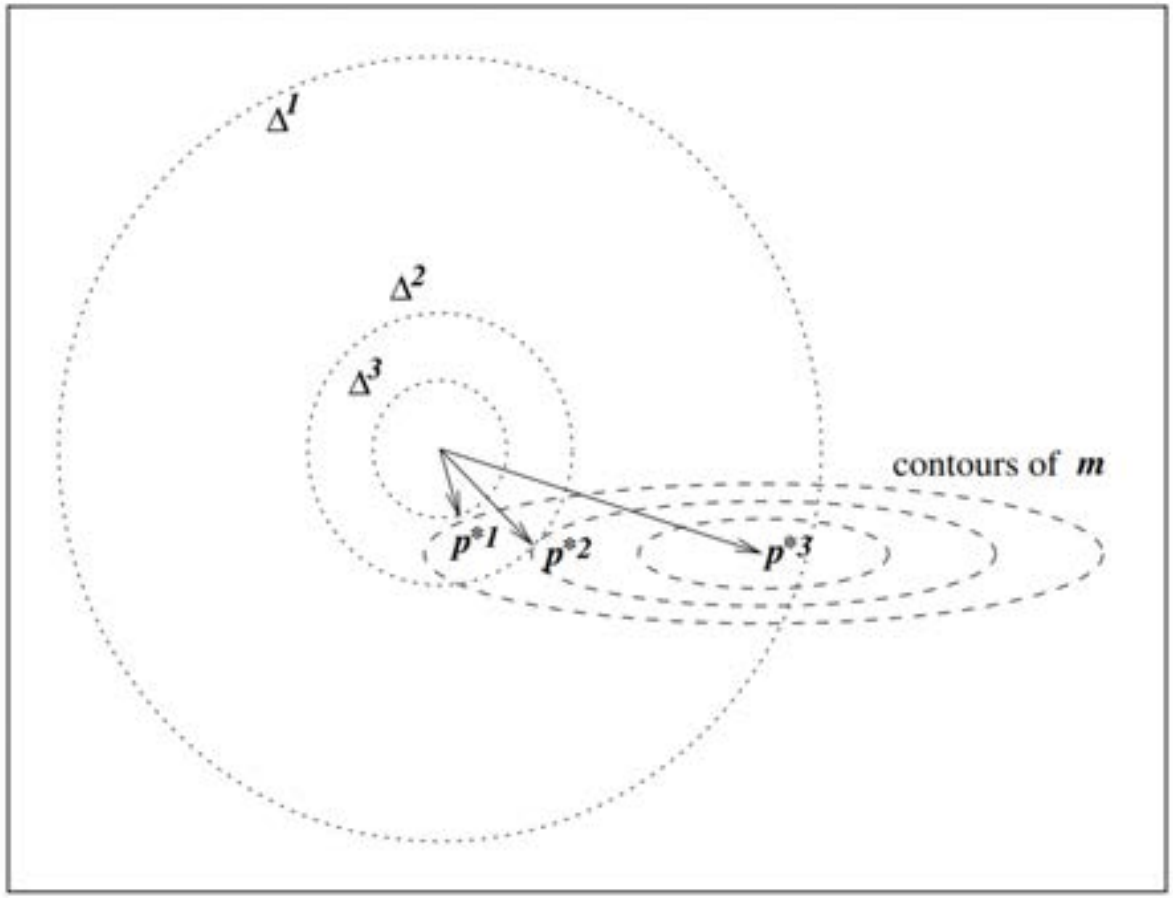
- If ratio is above a threshold, accept $\mathbf{d}$ (i.e., $\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{d}$) and scale Δ_k up by a factor (unless you get to a pre-defined global max value)
- If ratio is above a smaller threshold, accept $\mathbf{d}$ but don't change Δ_k
- Otherwise, reject $\mathbf{d}$ ($\mathbf{x}^{k+1} = \mathbf{x}^k$) and shrink Δ_k by a factor

Trust Region vs. Line Search



Figures from from Numerical Optimization by Nocedal and Wright

Trust Region



Figures from from Numerical Optimization by Nocedal and Wright

A Trust-Region Method: Levenberg-Marquardt

- Has a trust-region interpretation
- Instead of solving the trust-region subproblem, adds a penalty term $\lambda\|\mathbf{d}\|^2$ to $m_k(\mathbf{d})$ to penalize large $\mathbf{d}$

$$\frac{1}{2}\mathbf{d}^\top(\mathbf{H}_k)\mathbf{d}^\top + \mathbf{g}_k^\top\mathbf{d} + f(\mathbf{x}^k) + \lambda_k\|\mathbf{d}\|^2 = \frac{1}{2}\mathbf{d}^\top(\mathbf{H}_k + \lambda_k\mathbf{I})\mathbf{d}^\top + \mathbf{g}_k^\top\mathbf{d} + f(\mathbf{x}^k)$$

- Gives the solution of trust-region subproblem for a particular value of Δ_k
- Larger $\Delta_k \Leftrightarrow$ larger trust region $\Leftrightarrow$ smaller penalty factor λ_k
- Often implemented using λ_k (penalty) rather than Δ_k (explicit constraint)
- λ_k is updated similar to Δ_k
- Originally was purposed for nonlinear least squares:
  - Levenberg $(\mathbf{J}_k^\top\mathbf{J}_k + \lambda_k\mathbf{I})\mathbf{d} = -\mathbf{J}_k^\top\mathbf{r}(\mathbf{x}^k)$
  - Marquardt $(\mathbf{J}_k^\top\mathbf{J}_k + \lambda_k\mathrm{diag}(\mathbf{J}_k^\top\mathbf{J}_k))\mathbf{d} = -\mathbf{J}_k^\top\mathbf{r}(\mathbf{x}^k)$
- Interpolates between gradient descent (large λ) and (Gauss-)Newton (small λ) — (why?)

Our "Unconstrained Optimization" Trilogy: Big Reveal

- **Key idea**: Locally approximate the function with a quadratic model function and minimize the model

$$f(\mathbf{x}^k + \mathbf{d}) \approx f(\mathbf{x}^k) + \mathbf{g}_k^\top \mathbf{d} + \frac{1}{2}\mathbf{d}^\top \mathbf{B}\,\mathbf{d}$$

  (ideally, $\mathbf{B} \succ \mathbf{0}$)
- Setting the gradient to zero, we get:

$$\mathbf{B}\,\mathbf{d} = -\mathbf{g}_k$$

- If $\mathbf{B} = \mathbf{H}_k$ we get (pure) Newton (using actual second-order information!)
- If $\mathbf{B} = \mathbf{H}_k + \lambda \mathbf{I}$ we get (general) Levenberg-Marquardt
- In NLS problems, if $\mathbf{B} = \mathbf{J}_k^\top \mathbf{J}_k$ we get (pure) Gauss-Newton
- In NLS problems, if $\mathbf{B} = \mathbf{J}_k^\top \mathbf{J}_k + \lambda \mathbf{I}$ we get (NLS) Levenberg-Marquardt
- If $\mathbf{B} = \mathbf{I}$ we get gradient descent
- …

Direct Methods for Solving Linear Systems

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

$$\mathbf{A} = (\mathbf{J}_k^\top \mathbf{J}_k) \text{ and } \mathbf{b} = -\mathbf{J}_k^\top \mathbf{r}(\mathbf{x}^k)$$

  - e.g., in Levenberg-Marquardt

$$\mathbf{A} = (\mathbf{J}_k^\top \mathbf{J}_k + \lambda \mathbf{I}) \text{ and } \mathbf{b} = -\mathbf{J}_k^\top \mathbf{r}(\mathbf{x}^k)$$

- Do not invert $\mathbf{A}$!
  - Will lose structure (e.g., $\mathbf{A}$ may be sparse but $\mathbf{A}^{-1}$ will be generally dense)
  - Numerical stability
- We consider two direct methods based on Cholesky and QR factorizations

Cholesky solver

- Solving triangular systems (non-zero diagonal) is fast/easy (forward/backward substitution)

$$\begin{pmatrix} \ell_{11} & 0 & 0 \\ \ell_{21} & \ell_{22} & 0 \\ \ell_{31} & \ell_{32} & \ell_{33} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

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

$$\mathbf{L}\underbrace{\mathbf{L}^\top \mathbf{d}}_{\mathbf{y}} = \mathbf{b}$$

  ii. Solve $\mathbf{L}\mathbf{y} = \mathbf{b}$ via forward substitution
  iii. Solve $\mathbf{L}^\top \mathbf{d} = \mathbf{y}$ via backward substitution

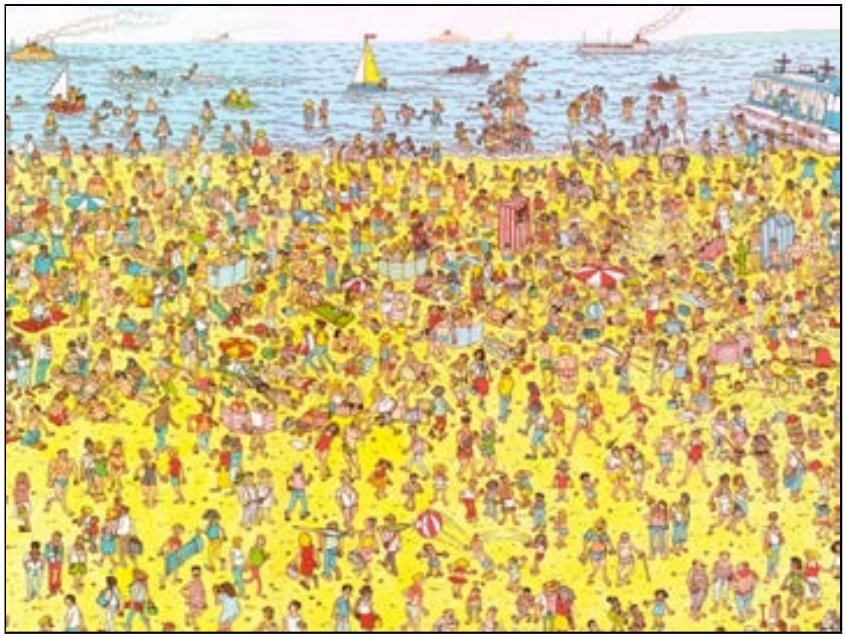
QR solver

- Note that $\mathbf{A} = \mathbf{M}^\top \mathbf{M}$ and $\mathbf{b} = \mathbf{M}^\top \mathbf{c}$ where $\mathbf{M} \in \mathbb{R}^{m \times n}$
  - e.g., in Gauss-Newton $\mathbf{M} = \mathbf{J}_k$ and $\mathbf{c} = -\mathbf{r}(\mathbf{x}^k)$
  - e.g., in Levenberg-Marquardt $\mathbf{M} = \begin{bmatrix} \mathbf{J}_k \\ \sqrt{\lambda}\mathbf{I}_n \end{bmatrix}$ and $\mathbf{c} = -\begin{bmatrix} \mathbf{r}(\mathbf{x}^k) \\ \mathbf{0} \end{bmatrix}$
- "Economic" QR factorization of $\mathbf{M} = \mathbf{QR}$
  - $\mathbf{Q} \in \mathbb{R}^{m \times n}$ and $\mathbf{Q}^\top \mathbf{Q} = \mathbf{I}_n$
  - $\mathbf{R} \in \mathbb{R}^{n \times n}$ is upper triangular
- Solve $\mathbf{R}\mathbf{d} = \mathbf{Q}^\top \mathbf{c}$ instead of $\mathbf{A}\mathbf{d} = \mathbf{b}$

$$
\begin{aligned}
\mathbf{A}\mathbf{d} = \mathbf{b} &\Rightarrow \mathbf{R}^\top \mathbf{Q}^\top \mathbf{Q}\mathbf{R}\mathbf{d} = \mathbf{R}^\top \mathbf{Q}\mathbf{c} && \mathbf{Q}^\top \mathbf{Q} = \mathbf{I}_n \\
&\Rightarrow \mathbf{R}^\top \mathbf{R}\mathbf{d} = \mathbf{R}^\top \mathbf{Q}^\top \mathbf{c} && \text{premultiply by } \mathbf{R}^{-\top} \\
&\Rightarrow \boxed{\mathbf{R}\mathbf{d} = \mathbf{Q}^\top \mathbf{c}} && \text{solve via backward substitution}
\end{aligned}
$$

- QR vs. Cholesky
  - ✓ QR does not need to form $\mathbf{A}$ - works with $\mathbf{J}_k$ or $\begin{bmatrix} \mathbf{J}_k \\ \sqrt{\lambda}\mathbf{I}_n \end{bmatrix}$
  - ✓ Better numerical stability than Cholesky
  - × Slower than Cholesky

To be continued …

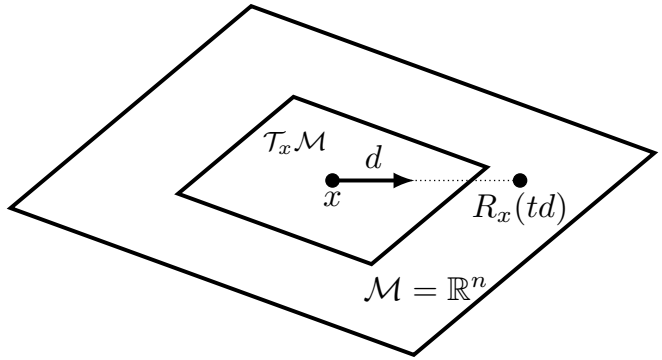


- We did not cover iterative (vs. direct) methods for solving (large) linear systems (see, e.g., conjugate gradient, Gauss-Seidel, etc)
- $\mathbf{A}\mathbf{d} = \mathbf{b}$ has a number of algorithmically exploitable structures in geometric estimation problems such as SLAM and bundle adjustment
- We will see how these structures can be exploited to speed up solvers

Today's Plan

- In robotics and computer vision, we often need to solve optimization problems involving rotations and poses – these variables do not live on (flat) vector spaces
- But in the previous lectures we ignored the constraints on such variables (just like flat-Earthers!)
- Can we use generic constrained optimization methods? Yeahnah …
- **Structure**: Our (constrained) decision variables (rotations and rigid-body transformations) are matrix Lie groups (smooth manifolds and groups)
- **Idea**: Exploit the smooth geometry of constraints and generalize Gauss-Newton (and other unconstrained algorithms) to do "unconstrained" optimization over our matrix Lie groups!
- Advantages:
  - ✓ Simpler, more natural, and faster methods
  - ✓ Iterations never leave the feasible set (manifold)
  - ✓ Can retain desirable traits of the algorithm in the unconstrained setting

Introduction to Optimization on Manifolds: Key Idea



Recall that in each iteration of unconstrained optimization over $\mathbb{R}^n$:

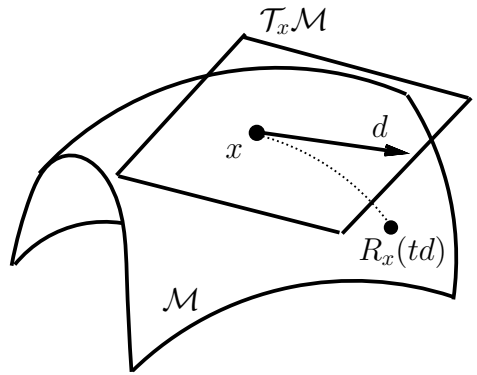
- Started at a point $x \in \mathbb{R}^n =: \mathcal{M}$
- Chose a "suitable" direction $d \in \mathbb{R}^n = \mathcal{T}_x\mathcal{M}$ ("tangent space" at x)
- Next iterate was found by moving along the line $x + td$ with step size t that gives us sufficient descent:

$$x \leftarrow x + td =: R_x(td) \in \mathcal{M}$$

- In other words, we walked on a (flat) curve $t \mapsto x + td$ that starts ($t = 0$) at x and has velocity d for t units of time
- This worked out in part because $x + td$ remained on our manifold $\mathcal{M} = \mathbb{R}^n$

Introduction to Optimization on Manifolds: Key Idea

Figure courtesy of Wen Huang



- This idea can be generalized to useful classes of manifolds beyond $\mathbb{R}^n$ (e.g., spheres, orthogonal matrices, rotations, rigid-body transformations)
- You can think of these manifolds (i.e., feasible set of our optimization problem) as "smooth surfaces" embedded in higher dimensional (Euclidean) ambient spaces (e.g., $\mathbb{R}^n$ or $\mathbb{R}^{n\times n}$)
- The idea is to treat constrained optimization problems with such constraints as "unconstrained" problems over the corresponding manifold
- But these manifolds are not "flat" anymore (i.e., not vector spaces)
- If we move on a line, we'll leave the manifold, resulting in infeasible points

Introduction to Optimization on Manifolds: Key Idea

- A natural idea is to move on smooth curves that live on the manifold $\gamma : (-\epsilon,\epsilon) \to \mathcal{M} : t \mapsto \gamma(t)$ and pass through x at $t = 0$; i.e., $\gamma(0) = x$
- Velocities of all such curves live on the tangent space to $\mathcal{M}$ at x, i.e., $\mathcal{T}_x\mathcal{M}$
- Fortunately, $\mathcal{T}_x\mathcal{M}$ is a vector space (i.e., $\mathcal{T}_x\mathcal{M} \cong \mathbb{R}^m$ for an m-dimensional manifold $\mathcal{M}$)! Therefore, (with the help of a Riemannian metric) we can use the same ideas that underpin unconstrained optimization methods over Euclidean spaces to choose a velocity (search direction) $d \in \mathcal{T}_x\mathcal{M}$
- After choosing a velocity $d \in \mathcal{T}_x\mathcal{M}$, we move on a curve that passes through x at $t = 0$ with initial velocity $\dot{\gamma}(0) = d$ for t units of time (e.g., selected via "line" search)
- Geodesics (generalization of straight lines in $\mathbb{R}^n$) are the most natural choices for γ – but in practice, we may prefer computationally cheaper and simpler alternatives (approximations) called retractions $R_x : \mathcal{T}_x\mathcal{M} \to \mathcal{M}$

Introduction to Optimization on Manifolds: Key Idea

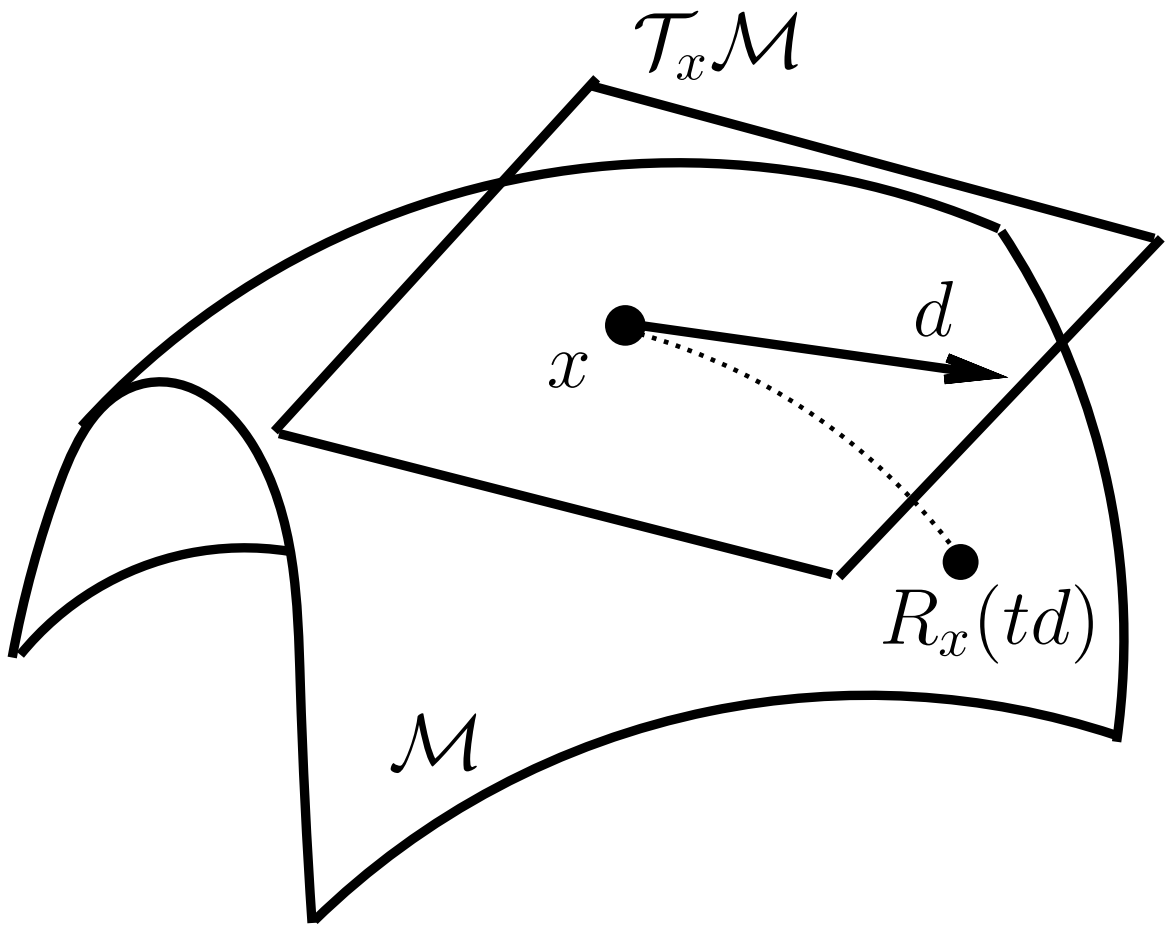


Figure courtesy of Wen Huang

Introduction to Optimization on Manifolds: Key Idea

In Riemannian optimization methods, until convergence we:

- **Lift**: "Lift" (pullback) the objective function to the tangent space $\mathcal{T}_x\mathcal{M}$ using a retraction
- **Solve**: Use ideas from unconstrained optimization methods to choose a "direction" (velocity) d on the tangent space $\mathcal{T}_x\mathcal{M}$
- **Retract**: Choose t (e.g., in line search methods) and move from x to $R_x(td) \in \mathcal{M}$ where $R_x : \mathcal{T}_x\mathcal{M} \to \mathcal{M}$ is a retraction and $R_x(td) = \gamma(t)$ for a curve $\gamma : \mathbb{R} \to \mathcal{M} : t \mapsto \gamma(t)$ such that $\gamma(0) = x$ and $\dot{\gamma}(0) = d$;

$$x \leftarrow R_x(td)$$

  - Note that this generalizes the Euclidean iteration $R_x^{\text{Euc}}(td) = x + td$

http://tiny.cc/flat-earth-society

[activate layers (colored circles on the left) one by one]

Optimization over Matrix Lie Groups

- The procedure that was just presented is quite general and can be easily implemented on any Riemannian manifold – we only need to be familiar with the geometry of our manifolds, choose a retraction, and use an optimization method on tangent spaces — in most cases, all of these are already well understood and readily available (see, e.g., Manopt)
- We are particularly interested in (nonlinear) least squares problems that involve elements of $\mathrm{SO}(p)$ and $\mathrm{SE}(p)$ where $p \in \{2,3\}$ (i.e., 2/3D rotations and poses)
- As we saw before, these manifolds are in fact matrix Lie groups and thus enjoy additional structures. This makes it even simpler to develop methods based on the lift-solve-retract framework
- Specifically, it turns out that instead of (explicitly) operating on different tangent spaces $\mathcal{T}_x\mathcal{M}$ as x evolves, we can (equivalently) always pullback to the tangent space at the identity element $\mathcal{T}_{\mathrm{Id}}\mathcal{M}$ (i.e., Lie algebra) and use matrix exponential to define retractions (in case of $\mathrm{SO}(p)$, this even gives us geodesics).

Review: Special Orthogonal Group $\mathrm{SO}(3)$

- We learned about $\mathrm{SO}(3)$ (rotations) and $\mathrm{SE}(3)$ (poses) in Lecture 3
- In matrix Lie groups, matrix exponential $\exp$ maps elements in the Lie algebra (i.e., tangent space at the identity element) to the Lie group

$$\exp(\mathbf{A}) \triangleq \mathbf{I} + \sum_{k=1}^{\infty} \frac{\mathbf{A}^k}{k!}$$

- Lie algebra (e.g., $\mathfrak{se}(3)$ and $\mathfrak{so}(3)$) has vector-space structure
- Basis "vectors" (generators)

$$\widehat{\boldsymbol{\phi}} \in \mathfrak{so}(3) \Leftrightarrow \widehat{\boldsymbol{\phi}} = \phi_1 \mathbf{G}_1 + \phi_2 \mathbf{G}_2 + \phi_3 \mathbf{G}_3$$

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

$$\mathbf{G}_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \quad \mathbf{G}_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \quad \mathbf{G}_3 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

- $\widehat{\boldsymbol{\phi}} = [\boldsymbol{\phi}]_\times \Rightarrow \widehat{\boldsymbol{\phi}}\,\mathbf{a} = \boldsymbol{\phi} \times \mathbf{a}$

Review: Special Euclidean Group $\mathrm{SE}(3)$

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

$$\widehat{\begin{bmatrix}\boldsymbol{\phi}\\ \boldsymbol{\rho}\end{bmatrix}} \in \mathfrak{se}(3) \Leftrightarrow \widehat{\begin{bmatrix}\boldsymbol{\phi}\\ \boldsymbol{\rho}\end{bmatrix}} = \phi_1\mathbf{G}_1 + \phi_2\mathbf{G}_2 + \phi_3\mathbf{G}_3 + \rho_1\mathbf{G}_4 + \rho_2\mathbf{G}_5 + \rho_3\mathbf{G}_6$$

where

$$\mathbf{G}_1 = \begin{bmatrix}0&0&0&0\\0&0&-1&0\\0&1&0&0\\0&0&0&0\end{bmatrix} \quad \mathbf{G}_2 = \begin{bmatrix}0&0&1&0\\0&0&0&0\\-1&0&0&0\\0&0&0&0\end{bmatrix} \quad \mathbf{G}_3 = \begin{bmatrix}0&-1&0&0\\1&0&0&0\\0&0&0&0\\0&0&0&0\end{bmatrix}$$

$$\mathbf{G}_4 = \begin{bmatrix}0&0&0&1\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{bmatrix} \quad \mathbf{G}_5 = \begin{bmatrix}0&0&0&0\\0&0&0&1\\0&0&0&0\\0&0&0&0\end{bmatrix} \quad \mathbf{G}_6 = \begin{bmatrix}0&0&0&0\\0&0&0&0\\0&0&0&1\\0&0&0&0\end{bmatrix}$$

Nonlinear Least Squares over Matrix Lie Groups

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

Example:

- $\mathbf{x}_1 \in \mathcal{M}_1 = \mathrm{SE}(3) \subset \mathbb{R}^{4\times 4}$ (3D pose)
- $\mathbf{x}_2 \in \mathcal{M}_2 = \mathrm{SO}(3) \subset \mathbb{R}^{3\times 3}$ (3D rotation)

- As we saw before, $\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{d}$ is not valid anymore (e.g., adding $\mathbf{d}$ to a rotation matrix results in an infeasible point)
- Choose a search direction $\widehat{\mathbf{d}}$ on the Lie algebra of $\mathcal{M}$[1]
- Use $\mathbf{x}^{k+1} = \mathbf{x}^k \exp(\widehat{\mathbf{d}})$ to move "along"[2] $\widehat{\mathbf{d}}$ from $\mathbf{x}^k$ to $\mathbf{x}^{k+1}$
- $\mathbf{x}^{k+1}$ is a feasible point (why?)

[1] Review the definition of hat operator

[2] Modulo some technical details

Linearizing Residual

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

$$\mathbf{r}(\mathbf{x}^k+\mathbf{d}) \approx \mathbf{r}(\mathbf{x}^k)+\mathbf{J}_k\mathbf{d} \qquad \text{where} \qquad \mathbf{J}_k = \left.\frac{\partial \mathbf{r}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^k} = \left.\frac{\partial \mathbf{r}(\mathbf{x}^k+\mathbf{d})}{\partial \mathbf{d}}\right|_{\mathbf{d}=\mathbf{0}}$$

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

$$\mathbf{r}(\mathbf{x}^k \exp(\widehat{\mathbf{d}})) \approx \mathbf{r}(\mathbf{x}^k) + \mathfrak{J}_k\mathbf{d} \qquad \text{where} \qquad \mathfrak{J}_k \triangleq \left.\frac{\partial \mathbf{r}(\mathbf{x}^k \exp(\widehat{\mathbf{d}}))}{\partial \mathbf{d}}\right|_{\mathbf{d}=\mathbf{0}}$$

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

$$\mathbf{r}(\mathbf{x}^k \exp(\widehat{\mathbf{d}})) \approx \mathbf{r}(\mathbf{x}^k) + \mathfrak{J}_k\mathbf{d} \qquad \text{where} \qquad \mathfrak{J}_k \triangleq \left.\frac{\partial \mathbf{r}(\mathbf{x}^k \exp(\widehat{\mathbf{d}}))}{\partial \mathbf{d}}\right|_{\mathbf{d}=\mathbf{0}}$$

Lift-Solve-Retract for NLS over Matrix Lie Groups

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

1. Lift (pullback) to the tangent space at the indenity element (Lie algebra):

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

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

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

$$\mathbf{r}(\mathbf{x}^k \exp(\widehat{\mathbf{d}})) \approx \mathbf{r}(\mathbf{x}^k) + \mathfrak{J}_k \mathbf{d} \qquad \text{(definition of } g)$$

2. Solve for $\mathbf{d}$ by solving a (flat) linear least squares

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

Several Tips for Computing $\mathfrak{J}_k$

- Note that $\mathfrak{J}_k$ is evaluated at $\mathbf{d} = \mathbf{0}$
- A first-order approximation of $\exp(\widehat{\mathbf{d}})$ at $\mathbf{d} = \mathbf{0}$ (why?)

$$\exp(\widehat{\mathbf{d}}) \approx \mathbf{I} + \widehat{\mathbf{d}}$$

- Use the chain rule and vectorization of matrices (for convenience):

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

- Usual Jacobian - compute partial derivatives wrt elements of $\mathbf{d}$
- You can also use $\widehat{\mathbf{d}} = \sum_i d_i \mathbf{G}_i$ and take derivatives w.r.t. each d_i

  (i.e., columns of $\mathfrak{J}_k$)
- Useful identity: $\text{vec}(\mathbf{AB}) = (\mathbf{I} \otimes \mathbf{A})\text{vec}(\mathbf{B})$ where $\otimes$ denotes Kronecker product

Example with Multiple Variables

- Consider $\|\mathbf{r}(\mathbf{x}_1,\mathbf{x}_2)\|^2$ where $\mathbf{x}_1 \in \mathbb{R}^3$ (e.g., 3D point) and $\mathbf{x}_2 \in \mathrm{SO}(3)$

$$\|\mathbf{r}(\mathbf{x}_1^k + \mathbf{d}_1, \mathbf{x}_2^k \exp(\widehat{\mathbf{d}_2}))\|^2 \approx \|\mathbf{r}(\mathbf{x}_1^k,\mathbf{x}_2^k) + \mathbf{J}_{1,k}\mathbf{d}_1 + \mathfrak{J}_{2,k}\mathbf{d}_2\|^2$$

$$\mathbf{J}_{1,k} \triangleq \left.\frac{\partial \mathbf{r}(\mathbf{x})}{\partial \mathbf{x}_1}\right|_{\mathbf{x}=(\mathbf{x}_1^k,\mathbf{x}_2^k)} = \left.\frac{\partial \mathbf{r}(\mathbf{x}_1^k + \mathbf{d}_1, \mathbf{x}_2^k)}{\partial \mathbf{d}_1}\right|_{\mathbf{d}_1=\mathbf{0}}$$

$$\mathfrak{J}_{2,k} \triangleq \left.\frac{\partial \mathbf{r}(\mathbf{x}_1^k, \mathbf{x}_2^k \exp(\widehat{\mathbf{d}_2}))}{\partial \mathbf{d}_2}\right|_{\mathbf{d}_2=\mathbf{0}}$$

- Solve the resulting linear least squares
- Retract: $\mathbf{x}_1^{k+1} = \mathbf{x}_1^k + \mathbf{d}_1$ and $\mathbf{x}_2^{k+1} = \mathbf{x}_2^k \exp(\widehat{\mathbf{d}_2})$

16.485 Visual Navigation for Autonomous Vehicles (VNAV)
Fall 2020