NA 568 - Winter 2022

Optimization and Smoothing I

Maani Ghaffari

March 10, 2022

UNIVERSITY OF MICHIGAN

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

Notation

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

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

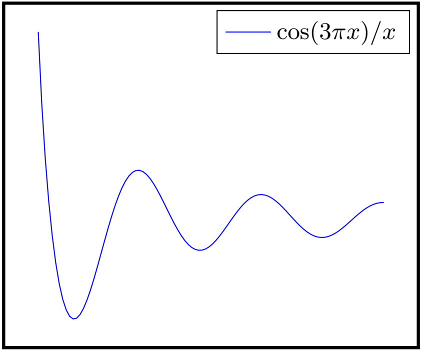
- Global minimum

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

- Local minimum

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

Example



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

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

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

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

Second-order Taylor Approximation

- Local quadratic approximation

$$f(x_0 + d) \approx f(x_0) + \nabla f(x_0)^{\mathsf{T}} d + \frac{1}{2}\, d^{\mathsf{T}} H(x_0)\, d$$

- Change of variables $x := x_0 + d$

$$f(x) \approx f(x_0) + \nabla f(x_0)^{\mathsf{T}} (x - x_0) + \frac{1}{2}\, (x - x_0)^{\mathsf{T}} H(x_0)\, (x - x_0)$$

Recognizing Local Minima

- First-order *necessary* condition

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

- Second-order *necessary* condition

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

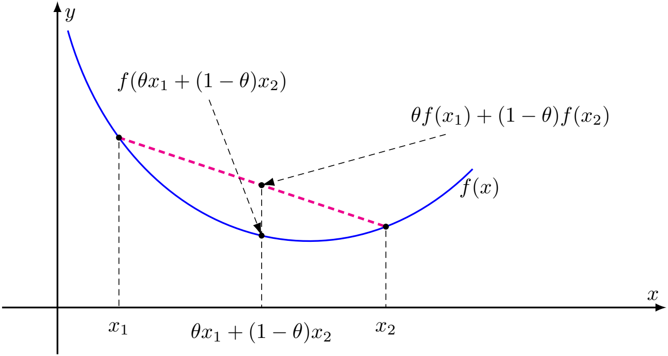
- Second-order *sufficient* condition

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

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

1. For all $x_1, x_2 \in \mathbb{R}^n$ and all $\theta \in [0,1]$:

$$f(\theta x_1 + (1-\theta)x_2) \leq \theta f(x_1) + (1-\theta) f(x_2)$$



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

1. For all $x_1, x_2 \in \mathbb{R}^n$ and all $\theta \in [0,1]$:

   $$f(\theta x_1 + (1-\theta)x_2) \leq \theta f(x_1) + (1-\theta) f(x_2)$$

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

   $$f(y) \geq f(x) + \nabla f(x)^{\mathsf{T}}(y - x)$$

3. Second-order condition: For all $x \in \mathbb{R}^n$:

   $$H(x) \succeq 0$$

Problem 1: 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)*

Claim

$\nabla f(x) = 0$ is necessary and sufficient for global optimality.

Claim

Unique minimizer iff $\mathrm{rank}(A) = n$.

Lemma

$A \in \mathbb{R}^{m\times n}$ *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 ($A^\mathsf{T} A = LL^\mathsf{T}$) 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.

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

$$z_k = H_k x_k + v_k,$$

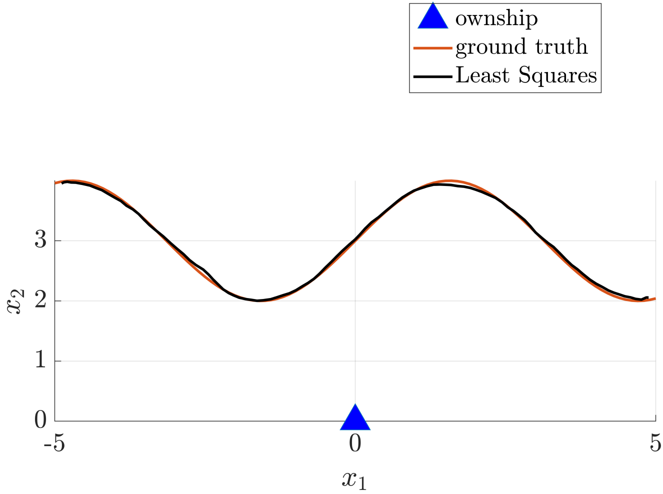
$$H_k = I,$$

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

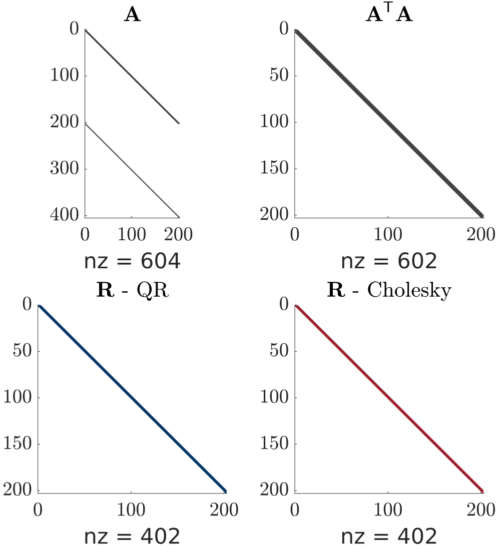
$$R_k = \mathrm{Cov}[v_k] = 0.05^2 I.$$

Example: LS Target Tracking (Smoothing)

See `ls_single_target.m` for code.



Example: LS Target Tracking (Smoothing)



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.

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

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

$\phi_0 = 1$ 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.

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

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

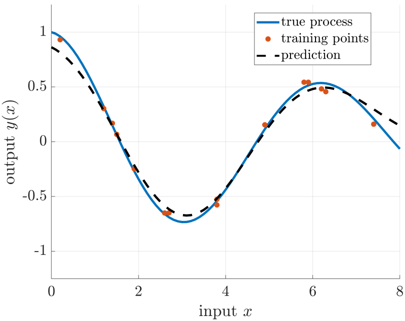
$$f(w) = \frac{1}{2}\|t - \Phi w\|^2 + \frac{\lambda}{2}\|w\|^2$$

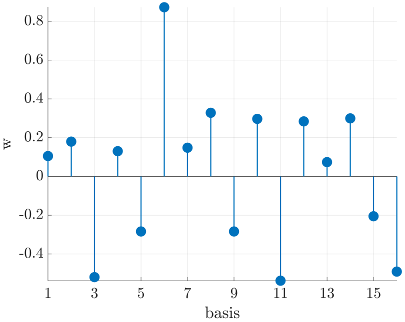
$$\nabla f(w) = \Phi^\mathsf{T}\Phi w - \Phi^\mathsf{T} t + \lambda w$$

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

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

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

Gauss-Newton (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 $x^{k+1} = x^k + d$

- Gradient $g_k := \nabla f(x^k)$
- Hessian $H_k := \nabla^2 f(x^k)$
- Second-order Taylor:

$$f(x^k + d) \approx m_k(d) := \frac{1}{2} d^\top H_k d + g_k^\top d + \underbrace{f(x^k)}_{\text{constant}}$$

- $m_k(d)$ gives the local quadratic approximation
- Find d by minimizing $m_k(d)$:

$$\nabla m_k(d) = 0 \Rightarrow H_k d + g_k = 0$$

- Well-defined if $H_k \succ 0 \Rightarrow \boxed{d = -H_k^{-1} g_k}$ and $x^{k+1} = x^k + d$
- In general has no preference for local minima over local maxima (i.e., stationary points)
- Very fast (quadratic) convergence near solutions

- Nonlinear least squares

$$f_{\mathsf{NLS}}(x) = \frac{1}{2}\|r(x)\|^2$$

- Gradient and Hessian of f_{NLS}

$$\nabla f_{\mathsf{NLS}}(x^k) =: g_k = J_k^{\mathsf{T}} r(x^k)$$

$$\nabla^2 f_{\mathsf{NLS}}(x^k) =: H_k = J_k^{\mathsf{T}} J_k + \underbrace{\sum_{i=1}^{m} r_i(x^k)\nabla^2 r_i(x^k)}_{S}$$

Newton vs. Gauss-Newton

- Newton iteration

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

- Gauss-Newton iteration

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

- Gauss-Newton is expected to behave like Newton (fast convergence close to a solution) if S is "small" (e.g., small-residual regime $r_i(x^\star) \approx 0$)

- $J_k^\mathsf{T} J_k$ is a PSD approximation of Hessian — S can make Hessian non-PSD!

Example: NLS Target Tracking (Smoothing) using GN

A target is moving in a 2D plane. The ownship position is known and fixed at the origin. We have access to relative noisy range and bearing measurements of the target position at any time step.

$$x_k = f(u_k, x_{k-1}) + w_k = x_{k-1} + w_k$$

$$z_k = h(x_k) + v_k = \begin{bmatrix} \sqrt{x_{1,k}^2 + x_{2,k}^2} \\ \operatorname{atan2}(x_{1,k}, x_{2,k}) \end{bmatrix} + v_k$$

$$Q_k = \operatorname{Cov}[w_k] = 0.05^2 I,\ R_k = \operatorname{Cov}[v_k] = \operatorname{diag}(0.1^2, 0.05^2)$$

$$H_k = \begin{bmatrix} \frac{x_{1,k}}{\sqrt{x_{1,k}^2+x_{2,k}^2}} & \frac{x_{2,k}}{\sqrt{x_{1,k}^2+x_{2,k}^2}} \\ \frac{x_{2,k}}{x_{1,k}^2+x_{2,k}^2} & \frac{-x_{1,k}}{x_{1,k}^2+x_{2,k}^2} \end{bmatrix}$$

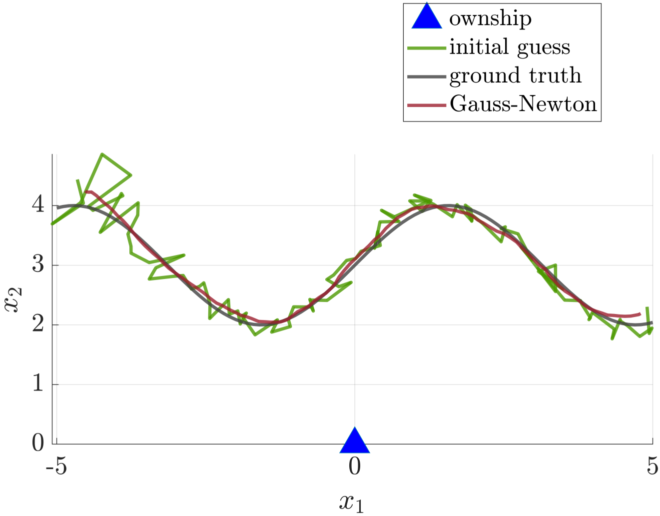
Example: NLS Target Tracking (Smoothing) using GN

There is no knowledge of the target motion, but we assume target is close to its previous location to constrain the state.

$$
\begin{aligned}
r_1(x_{k-1,k}) &:= x_k - f(u_k, x_{k-1}) = x_k - x_{k-1}, \\
r_2(x_k) &:= z_k - h(x_k), \\
r(x_{k-1,k}) &= \mathrm{vec}(r_1(x_{k-1,k}), r_2(x_k)).
\end{aligned}
$$

Example: NLS Target Tracking (Smoothing) using GN

See `nls_single_target.m` for code.



Globalization Strategies: Line Search

- $x^{k+1} = x^k + \alpha d$ where α is the step size
- d is a descent direction if $f(x^{k+1}) < f(x^k)$ for a sufficiently small step size

directional derivative $$\lim_{\alpha \to 0} \frac{f(x^k + \alpha d) - f(x^k)}{\alpha} = g_k^\mathsf{T} d$$

$g_k^\mathsf{T} d < 0 \Rightarrow d$ is a descent direction

1. Pick a *descent* direction d
   - Newton direction is a descent direction if $H_k \succ 0$
   - Gauss-Newton direction is a descent direction if J_k is full column rank
2. Find the best step size α (exact line search)

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

- In practice $\rightarrow$ *inexact* line search (backtracking) + armijo rule
- Leads to *damped* Newton/Gauss-Newton

Globalization Strategies: Trust-Region Methods

- **Q.** How much do we trust our local approximate quadratic model $m_k(d)$ away from $d = 0$?

1. Pick a maximum step size Δ
2. Pick d by solving the trust-region subproblem

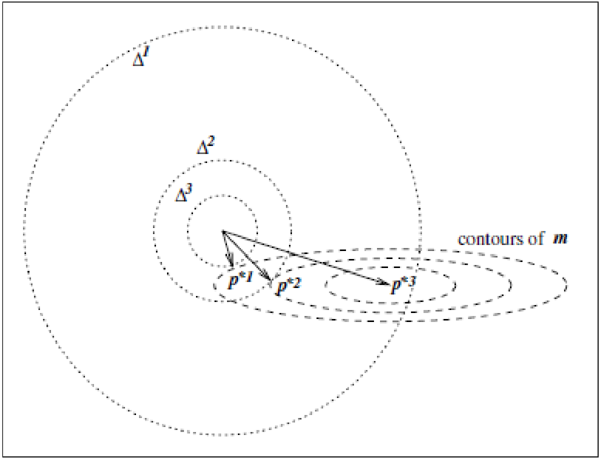
$$\underset{d}{\text{minimize}} \;\; m_k(d) \;\; \text{s.t.} \; \|d\| \leq \Delta$$

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

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

4. If ratio is below a threshold, reject d and shrink Δ by a factor; otherwise accept d and increase Δ by a factor

Trust region



- has a trust-region interpretation
- instead of solving the trust-region subproblem, adds a penalty term $\lambda\|d\|^2$ to $m_k(d)$ to penalize a large d

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

- larger $\Delta \Leftrightarrow$ larger trust region $\Leftrightarrow$ smaller penalty factor λ
- λ is updated similar to Δ
- 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 $A \in \mathsf{Sym}(n)$ 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.

- 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

- “Economic” 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 $Ad = b$

$$Ad = b \Rightarrow Q^\mathsf{T} Q R d = Q^\mathsf{T} c \qquad Q^\mathsf{T} Q = I_n$$

$$\Rightarrow \boxed{Rd = Q^\mathsf{T} c}$$

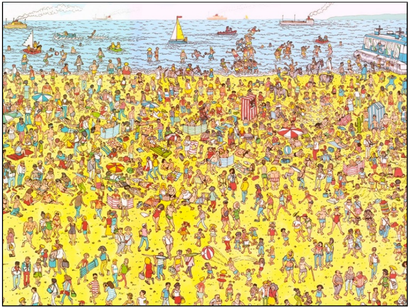
solve via backward substitution

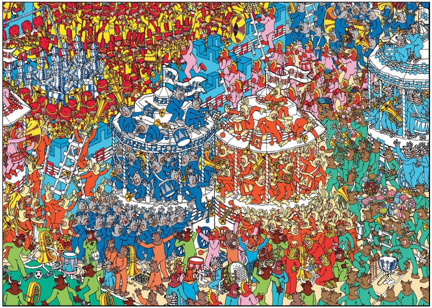
QR vs. Cholesky

- ✓ 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

Structure





- Plenty of structures in $Ad = b$ in SLAM, bundle adjustment, etc.
- We will see how these structures can be exploited to speed up solvers.