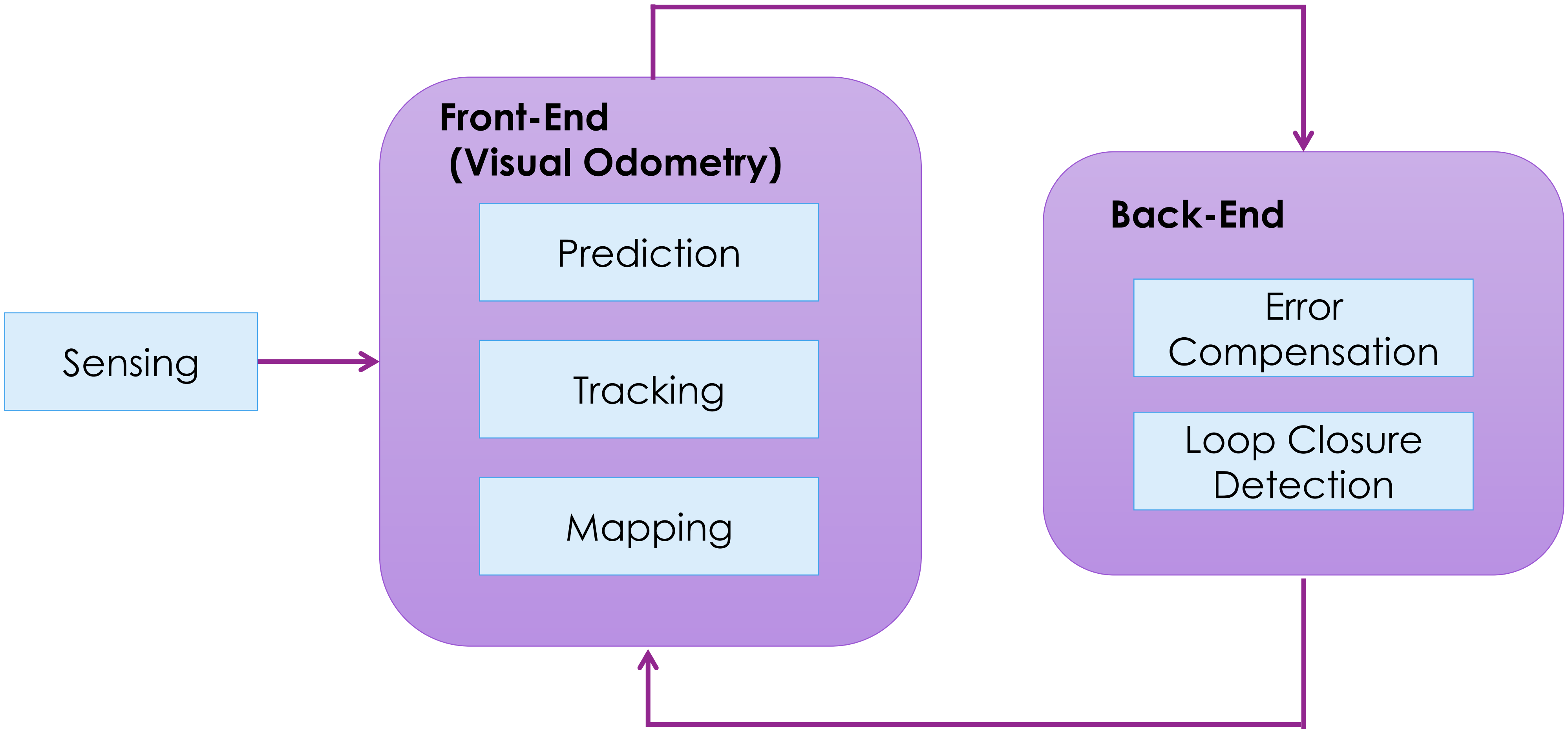
Robotic Navigation and Exploration

Unit 06: SLAM Back-end (II)

Min-Chun Hu anitahu@cs.nthu.edu.tw
CS, NTHU

SLAM Architecture



Error Compensation Methods

- Filter-based
  - Less computation
  - On-line optimization
  - Less accurate

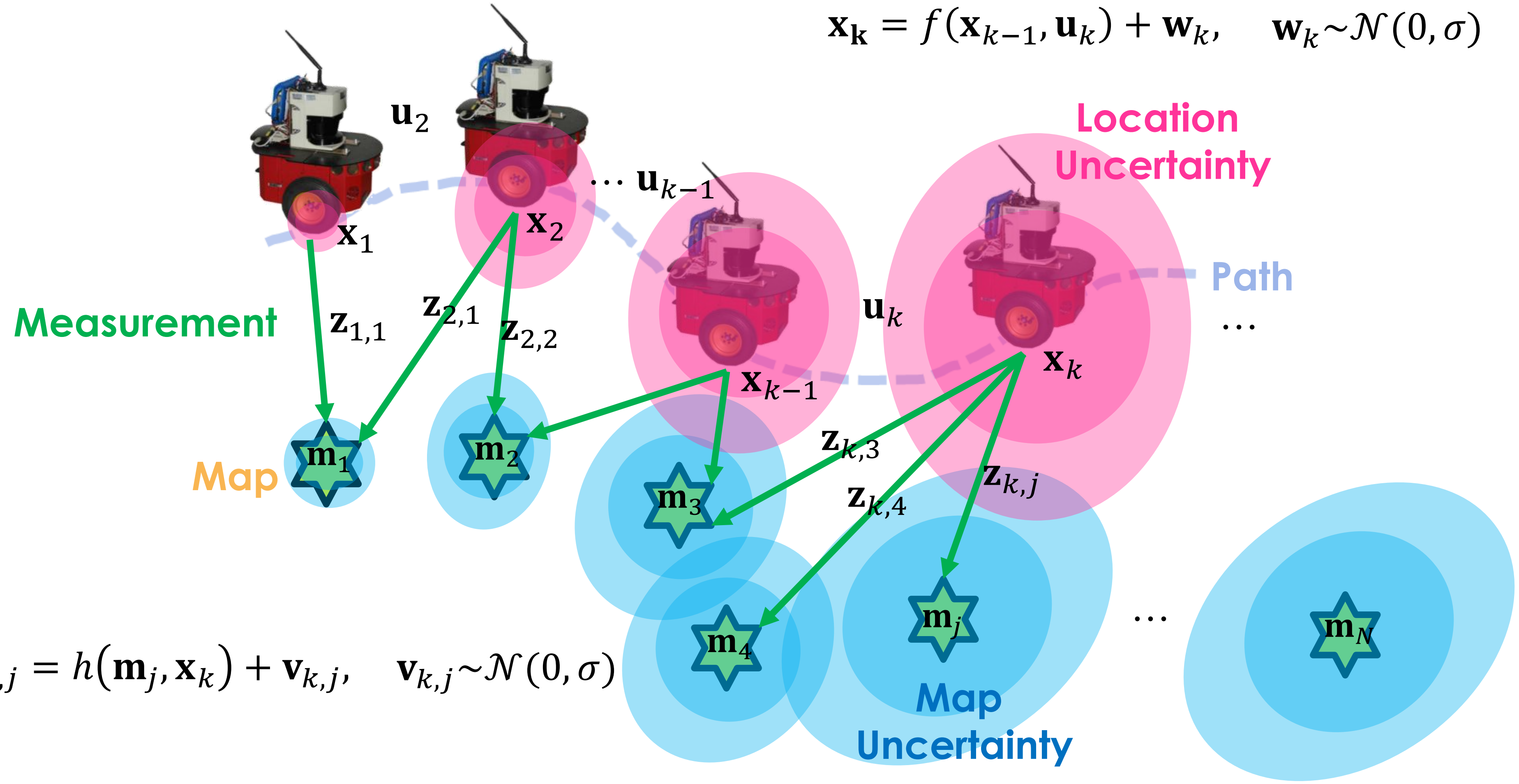
- Graph-based
  - Heavier computation
  - Off-line optimization
  - More accurate

Outline

- State Estimation and SLAM Problem
- SLAM Back-end (Error Compensation)
  - Filter-based Methods
    - Probability Theory and Bayes Filter
    - Kalman Filter (KF) / Extended Kalman Filter (EKF)
      - EKF-SLAM
    - Particle Filter
      - Fast-SLAM
  - Graph-based Methods
    - Pose Graph and Least-square Optimization
    - Gauss-Newton and Levenberg-Marquardt Algorithm
    - Sparse Matrix for Optimization

Recall the SLAM Problem

$$\mathbf{x_k} = f(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k, \quad \mathbf{w}_k \sim \mathcal{N}(0, \sigma)$$



$$\mathbf{z}_{k,j} = h(\mathbf{m}_j, \mathbf{x}_k) + \mathbf{v}_{k,j}, \quad \mathbf{v}_{k,j} \sim \mathcal{N}(0, \sigma)$$

State Estimation

$$\mathbf{x}\sim N(\boldsymbol{\mu},\boldsymbol{\Sigma})\quad P(\mathbf{x})=\frac{1}{\sqrt{(2\pi)^N\det(\Sigma)}}\exp(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}))$$

$$-\ln P(\mathbf{x})=\frac{1}{2}\ln((2\pi)^N\det(\boldsymbol{\Sigma}))+\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})$$

$$\mathbf{x}_k=f(\mathbf{x}_{k-1},\mathbf{u}_k)+\mathbf{w}_k,\quad \mathbf{w}_k\sim\mathcal{N}(0,\mathbf{R}_k)$$

$$\mathbf{z}_{k,j}=h(\mathbf{m}_j,\mathbf{x}_k)+\mathbf{v}_{k,j},\quad \mathbf{v}_{k,j}\sim\mathcal{N}(0,\mathbf{Q}_{k,j})$$

- Probability of $\{\mathbf{x}_1,\dots,\mathbf{x}_N,\mathbf{m}_1,\dots,\mathbf{m}_M\}$ given $\{\boldsymbol{u}_1,\dots,\boldsymbol{u}_N,\mathbf{z}_{1,1},\dots,\mathbf{z}_{N,M}\}$:

$$\underbrace{P(\mathbf{x},\mathbf{m}|\mathbf{z},\mathbf{u})}_{\text{posterior}}=\frac{P(\mathbf{z},\mathbf{u}|\mathbf{x},\mathbf{m})P(\mathbf{x},\mathbf{m})}{P(\mathbf{z},\mathbf{u})}\propto \underbrace{P(\mathbf{z},\mathbf{u}|\mathbf{x},\mathbf{m})}_{\text{likelihood}}\underbrace{P(\mathbf{x},\mathbf{m})}_{\text{prior}}$$

$$(\mathbf{x},\mathbf{m})^*_{MAP}=\operatorname{argmax} P(\mathbf{x},\mathbf{m}|\mathbf{z},\mathbf{u})=\operatorname{argmax} P(\mathbf{z},\mathbf{u}|\mathbf{x},\mathbf{m})P(\mathbf{x},\mathbf{m})$$

$$(\mathbf{x},\mathbf{m})^*_{MLE}=\operatorname{argmax} P(\mathbf{z},\mathbf{u}|\mathbf{x},\mathbf{m})$$

$$\boxed{\mathbf{z}_{k,j}=h(\mathbf{m}_j,\mathbf{x}_k)+\mathbf{v}_{k,j}}$$

$$P(\mathbf{z}_{k,j}\,|\mathbf{x}_\mathbf{k},\mathbf{m}_j)=\mathcal{N}(h(\mathbf{m}_j,\mathbf{x}_k),\mathbf{Q}_{k,j})$$

$$(\mathbf{x}_k,\mathbf{m}_j)^*_{MLE}=\operatorname{argmax}\mathcal{N}(h(\mathbf{m}_j,\mathbf{x}_k),\mathbf{Q}_{k,j})=\operatorname{argmin}\frac{1}{2}\left(\mathbf{z}_{k,j}-h(\mathbf{m}_j,\mathbf{x}_k)\right)^T\mathbf{Q}_{k,j}{}^{-1}\left(\mathbf{z}_{k,j}-h(\mathbf{m}_j,\mathbf{x}_k)\right)$$

State Estimation

$$(\mathbf{x}_k, \mathbf{m}_j)^*_{MLE} = \text{argmin} \frac{1}{2}\left(\mathbf{z}_{k,j} - h(\mathbf{m}_j, \mathbf{x}_k)\right)^T \mathbf{Q}_{k,j}{}^{-1}\left(\mathbf{z}_{k,j} - h(\mathbf{m}_j, \mathbf{x}_k)\right)$$

$$\mathrm{P}(\mathbf{z}, \mathbf{u} | \mathbf{x}, \mathbf{m}) = \prod_k P(\mathbf{u}_k | \mathbf{x}_{k-1}, \mathbf{x}_k) \prod_{k,j} P(\mathbf{z}_{k,j} | \mathbf{x}_k, \mathbf{m}_j)$$

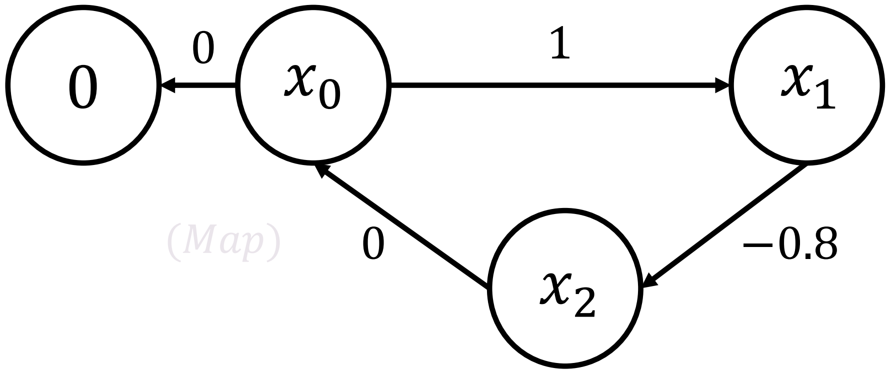
$$\mathbf{e}_{\mathbf{u},k} = \mathbf{x}_k - f(\mathbf{x}_{k-1}, \mathbf{u}_k)$$

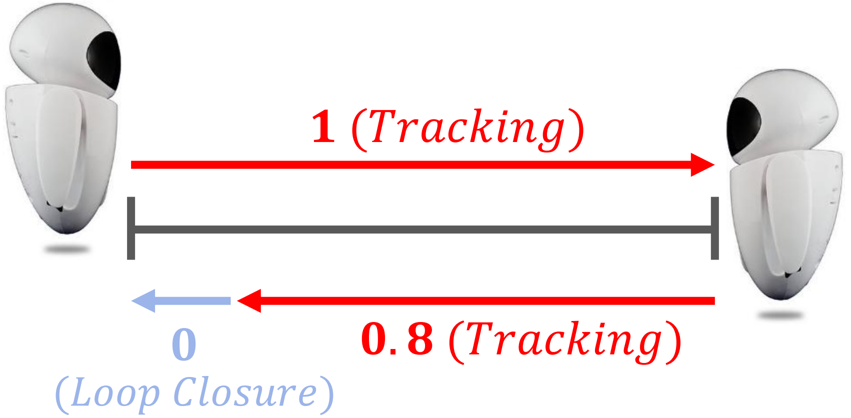
$$\mathbf{e}_{\mathbf{z},k,j} = \mathbf{z}_{k,j} - h(\mathbf{m}_j, \mathbf{x}_k)$$

$$\min F(\mathbf{x}, \mathbf{m}) = min \sum_k \mathbf{e}^T_{\mathbf{u},k} \mathbf{R}_K^{-1} \mathbf{e}_{\mathbf{u},k} + \sum_k \sum_j \mathbf{e}^T_{\mathbf{z},k,j} \mathbf{Q}_{K,j}^{-1} \mathbf{e}_{\mathbf{z},k,j}$$

Graph Optimization

Graph Optimization: 1D Example





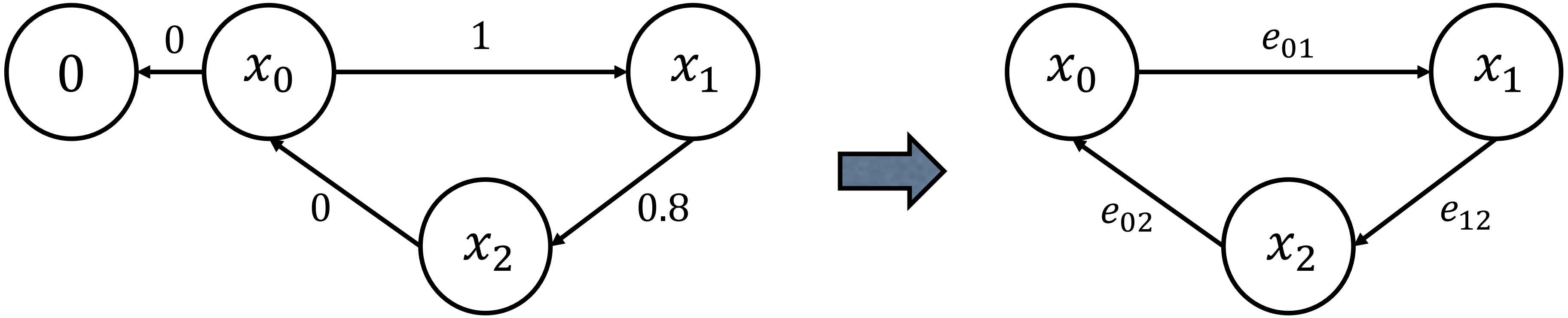
Error function

$$
\begin{aligned}
x_0 &= 0 \\
x_1 &= x_0 + 1 \\
x_2 &= x_1 - 0.8 \\
x_0 &= x_2 + 0
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
f_1 &= x_0 \\
f_2 &= x_1 - x_0 - 1 \\
f_3 &= x_2 - x_1 + 0.8 \\
f_4 &= x_0 - x_2
\end{aligned}
$$

$$\min_x \sum_i w_i f_i^2 = w_1 x_0^2 + w_2(x_1 - x_0 - 1)^2 + w_3(x_2 - x_1 + 0.8)^2 + w_4(x_0 - x_2)^2$$

$(Optimization)$

Graph Optimization: 1D Example



Error Function

$$e_{01} = x_1 - x_0 - 1$$
$$e_{12} = x_2 - x_1 - 0.8$$
$$e_{02} = x_0 - x_2$$

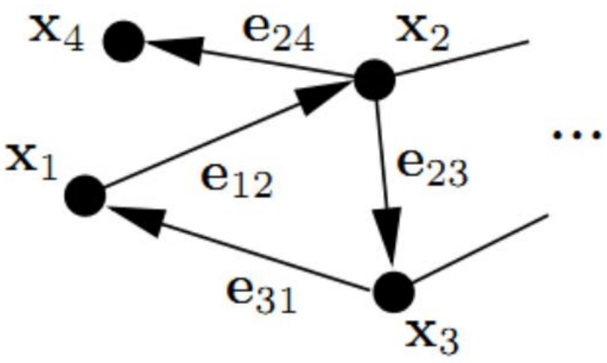
$$\min_x \sum_{i,j} w_{ij} e_{ij}^2 = w_{01}(x_1 - x_0 - 1)^2 + w_{12}(x_2 - x_1 + 0.8)^2 + w_{02}(x_0 - x_2)^2 \qquad (Optimization)$$

Graph Optimization: General Form

$$\min_x \sum_{i,j} w_{ij} e_{ij}^2 = w_{01}(x_1 - x_0 - 1)^2 + w_{12}(x_2 - x_1 + 0.8)^2 + w_{02}(x_0 - x_2)^2$$

$$\mathbf{F}(\mathbf{x}) = \sum_{\langle i,j \rangle \in \mathcal{C}} \underbrace{\mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})^\top \boldsymbol{\Omega}_{ij} \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})}_{\mathbf{F}_{ij}} \quad (1)$$

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}}\, \mathbf{F}(\mathbf{x}). \quad (2)$$



$$\begin{aligned} \mathbf{F}(\mathbf{x}) = & \ \mathbf{e}_{12}^\top \boldsymbol{\Omega}_{12} \mathbf{e}_{12} \\ & + \mathbf{e}_{23}^\top \boldsymbol{\Omega}_{23} \mathbf{e}_{23} \\ & + \mathbf{e}_{31}^\top \boldsymbol{\Omega}_{31} \mathbf{e}_{31} \\ & + \mathbf{e}_{24}^\top \boldsymbol{\Omega}_{24} \mathbf{e}_{24} \\ & + \ldots \end{aligned}$$

Graph Optimization Library

g2o - General Graph Optimization

Linux: build passing Windows: build passing

g2o is an open-source C++ framework for optimizing graph-based nonlinear error functions. g2o has been designed to be easily extensible to a wide range of problems and a new problem typically can be specified in a few lines of code. The current implementation provides solutions to several variants of SLAM and BA.

https://github.com/RainerKuemmerle/g2o

Ceres Solver

Ceres Solver is an open source C++ library for modeling and solving large, complicated optimization problems. It is a feature rich, mature and performant library which has been used in production at Google since 2010. Ceres Solver can solve two kinds of problems.

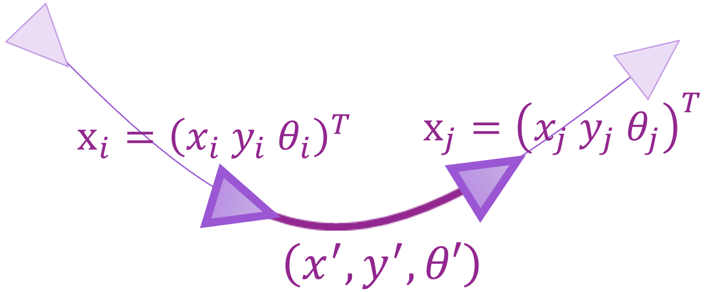
https://github.com/ceres-solver/ceres-solver

Graph Optimization for 2D Pose

- Consider the relation between two poses:

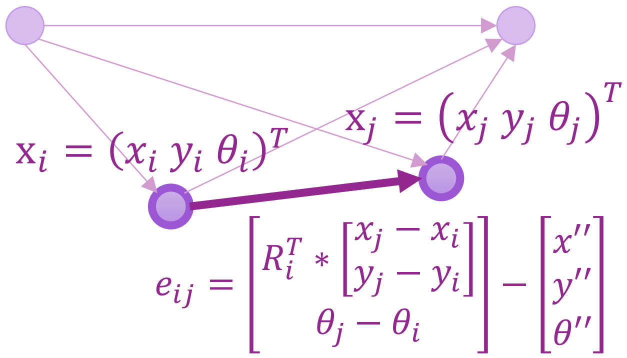
$$\begin{bmatrix} x_j \\ y_j \\ \theta_j \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix} + \begin{bmatrix} R_i * \begin{bmatrix} x' \\ y' \end{bmatrix} \\ \theta' \end{bmatrix}$$, in which $R_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix}$

And get $\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} R_i^T * \begin{bmatrix} x_j - x_i \\ y_j - y_i \end{bmatrix} \\ \theta_j - \theta_i \end{bmatrix}$



- After measuring the transform (x'', y'', θ'') between two nodes, we can write down the error term:

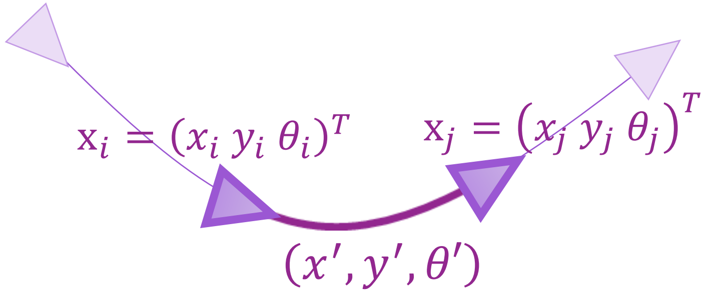
$$e_{ij} = \begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} - \begin{bmatrix} x'' \\ y'' \\ \theta'' \end{bmatrix} = \begin{bmatrix} R_i^T * \begin{bmatrix} x_j - x_i \\ y_j - y_i \end{bmatrix} \\ \theta_j - \theta_i \end{bmatrix} - \begin{bmatrix} x'' \\ y'' \\ \theta'' \end{bmatrix}$$



Graph Optimization for 2D Pose

- The goal is to find the optimal poses

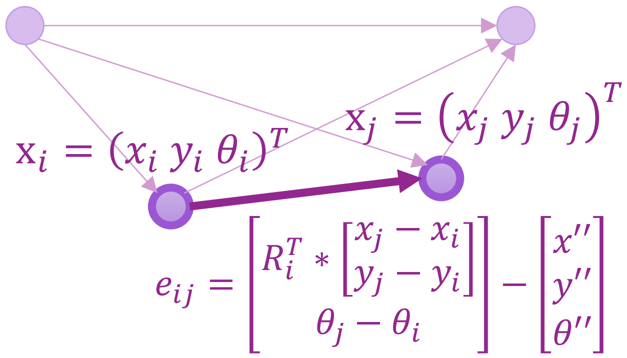
$$F = \sum_{i,j} e_{ij}^{\mathrm{T}} \Omega e_{ij} \qquad \begin{aligned} \mathrm{x} &= (x, y, \theta)^{\mathrm{T}} \\ \mathrm{x}^* &= \underset{\mathrm{x}}{\operatorname{argmax}} F(\mathrm{x}) \end{aligned}$$



- Approximate the object function by 1st order Taylor:

$$F \approx \sum_{i,j} e_{ij}(\mathrm{x}_i + \Delta\mathrm{x}_i, \mathrm{x}_j + \Delta\mathrm{x}_j)^T \Omega e_{ij}(\mathrm{x}_i + \Delta\mathrm{x}_i, \mathrm{x}_j + \Delta\mathrm{x}_j)$$

$$= \sum_{i,j} \left(e_{ij}(x_i, x_j) + A_{ij}\Delta\mathrm{x}_i + B_{ij}\Delta\mathrm{x}_j\right)^T \Omega \left(e_{ij}(x_i, x_j) + A_{ij}\Delta\mathrm{x}_i + B_{ij}\Delta\mathrm{x}_j\right) = \bar{\mathrm{F}}$$



, in which

$$A_{ij} = \frac{\partial e_{ij}}{\partial \mathrm{x}_i} = \begin{bmatrix} -R_i^T & \frac{\partial R_i^T}{\partial \theta_i} \begin{bmatrix} x_j - x_i \\ y_j - y_i \end{bmatrix} \\ 0 & -1 \end{bmatrix}_{3\times3}, B_{ij} = \frac{\partial e_{ij}}{\partial \mathrm{x}_j} = \begin{bmatrix} R_i^T & 0 \\ 0 & -1 \end{bmatrix}_{3\times3}$$

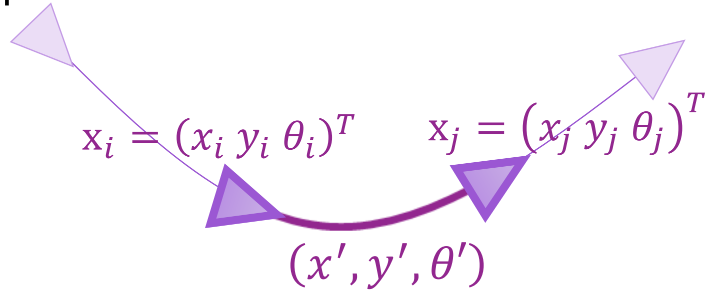
$$\bar{\mathrm{F}} = \sum_{i,j} \left(e_{ij}(x_i, x_j) + A_{ij}\Delta \mathrm{x}_i + B_{ij}\Delta \mathrm{x}_j\right)^T \Omega \left(e_{ij}(x_i, x_j) + A_{ij}\Delta \mathrm{x}_i + B_{ij}\Delta \mathrm{x}_j\right)$$

Graph Optimization for 2D Pose

- Apply Gauss-Newton method, we solve the 1st order approximation of object function:

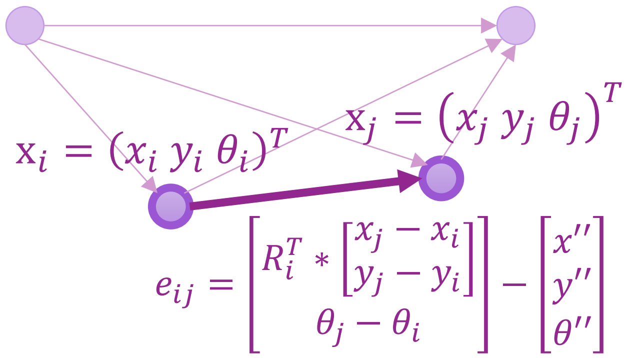
$$\frac{\partial \bar{\mathrm{F}}}{\partial \Delta \mathrm{x}_\mathrm{i}} = A_{ij}^T \Omega A_{ij} \Delta x_i + A_{ij}^T \Omega B_{ij} \Delta x_j + A_{ij}^T \Omega e_{ij} = 0,$$

$$\frac{\partial \bar{\mathrm{F}}}{\partial \Delta \mathrm{x}_\mathrm{j}} = B_{ij}^T \Omega A_{ij} \Delta x_i + B_{ij}^T \Omega B_{ij} \Delta x_j + B_{ij}^T \Omega e_{ij} = 0$$



- Transform the equation into matrix form:

$$\begin{bmatrix} A_{ij}^T \Omega A_{ij} & A_{ij}^T \Omega B_{ij} \\ B_{ij}^T \Omega A_{ij} & B_{ij}^T \Omega B_{ij} \end{bmatrix} * \begin{bmatrix} \Delta \mathrm{x}_\mathrm{i} \\ \Delta \mathrm{x}_\mathrm{j} \end{bmatrix} = \begin{bmatrix} -A_{ij}^T \Omega e_{ij} \\ -B_{ij}^T \Omega e_{ij} \end{bmatrix}$$



Solve the linear system by sparse Cholesky Factorization

$$H \Delta \mathrm{x} = -b \qquad\qquad (H + \lambda I)\Delta \mathrm{x} = -b$$

$\mathbf{H} \approx \mathbf{J}^\mathbf{T}\mathbf{J}$ (Gauss-Newton) (Levenberg-Marquardt)

Complete Algorithm

$$\mathbf{J}_{ij} = \left(\mathbf{0} \cdots \mathbf{0} \;\underbrace{\mathbf{A}_{ij}}_{\text{node } i}\; \mathbf{0} \cdots \mathbf{0} \;\underbrace{\mathbf{B}_{ij}}_{\text{node } j}\; \mathbf{0} \cdots \mathbf{0}\right).$$

$$\mathbf{H}_{ij} = \begin{pmatrix} \ddots & & & & \\ & \mathbf{A}_{ij}^T\boldsymbol{\Omega}_{ij}\mathbf{A}_{ij} & \cdots & \mathbf{A}_{ij}^T\boldsymbol{\Omega}_{ij}\mathbf{B}_{ij} & \\ & \vdots & \ddots & \vdots & \\ & \mathbf{B}_{ij}^T\boldsymbol{\Omega}_{ij}\mathbf{A}_{ij} & \cdots & \mathbf{B}_{ij}^T\boldsymbol{\Omega}_{ij}\mathbf{B}_{ij} & \\ & & & & \ddots \end{pmatrix}$$

$$\mathbf{b}_{ij} = \begin{pmatrix} \vdots \\ \mathbf{A}_{ij}^T\boldsymbol{\Omega}_{ij}\mathbf{e}_{ij} \\ \vdots \\ \mathbf{B}_{ij}^T\boldsymbol{\Omega}_{ij}\mathbf{e}_{ij} \\ \vdots \end{pmatrix}$$

Require: $\breve{\mathbf{x}} = \breve{\mathbf{x}}_{1:T}$: initial guess. $\mathcal{C} = \{\langle \mathbf{e}_{ij}(\cdot), \boldsymbol{\Omega}_{ij} \rangle\}$: constraints

Ensure: $\mathbf{x}^*$: new solution, $\mathbf{H}^*$ new information matrix

// find the maximum likelihood solution

while $\neg$converged **do**

$\mathbf{b} \leftarrow \mathbf{0} \quad \mathbf{H} \leftarrow \mathbf{0}$

for all $\langle \mathbf{e}_{ij}, \boldsymbol{\Omega}_{ij} \rangle \in \mathcal{C}$ **do**

// Compute the Jacobians $\mathbf{A}_{ij}$ and $\mathbf{B}_{ij}$ of the error function

$\mathbf{A}_{ij} \leftarrow \left.\frac{\partial \mathbf{e}_{ij}(\mathbf{x})}{\partial \mathbf{x}_i}\right|_{\mathbf{x}=\breve{\mathbf{x}}} \quad \mathbf{B}_{ij} \leftarrow \left.\frac{\partial \mathbf{e}_{ij}(\mathbf{x})}{\partial \mathbf{x}_j}\right|_{\mathbf{x}=\breve{\mathbf{x}}}$

// compute the contribution of this constraint to the linear system

$\mathbf{H}_{[ii]} \mathrel{+}= \mathbf{A}_{ij}^T\boldsymbol{\Omega}_{ij}\mathbf{A}_{ij} \quad \mathbf{H}_{[ij]} \mathrel{+}= \mathbf{A}_{ij}^T\boldsymbol{\Omega}_{ij}\mathbf{B}_{ij}$

$\mathbf{H}_{[ji]} \mathrel{+}= \mathbf{B}_{ij}^T\boldsymbol{\Omega}_{ij}\mathbf{A}_{ij} \quad \mathbf{H}_{[jj]} \mathrel{+}= \mathbf{B}_{ij}^T\boldsymbol{\Omega}_{ij}\mathbf{B}_{ij}$

// compute the coefficient vector

$\mathbf{b}_{[i]} \mathrel{+}= \mathbf{A}_{ij}^T\boldsymbol{\Omega}_{ij}\mathbf{e}_{ij} \quad \mathbf{b}_{[j]} \mathrel{+}= \mathbf{B}_{ij}^T\boldsymbol{\Omega}_{ij}\mathbf{e}_{ij}$

end for

// keep the first node fixed

$\mathbf{H}_{[11]} \mathrel{+}= \mathbf{I}$

// solve the linear system using sparse Cholesky factorization

$\boldsymbol{\Delta}\mathbf{x} \leftarrow \text{solve}(\mathbf{H}\,\boldsymbol{\Delta}\mathbf{x} = -\mathbf{b})$

// update the parameters

$\breve{\mathbf{x}} \mathrel{+}= \boldsymbol{\Delta}\mathbf{x}$

end while

$\mathbf{x}^* \leftarrow \breve{\mathbf{x}}$

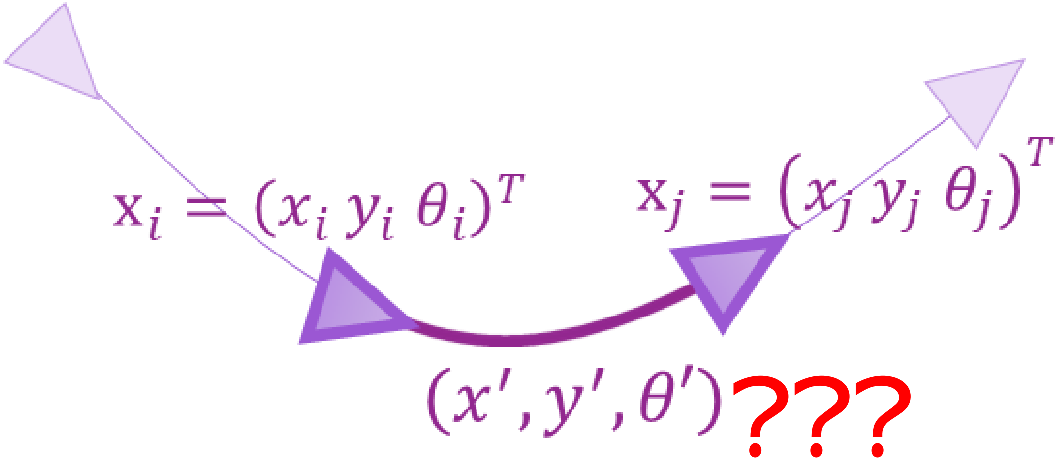
$\mathbf{H}^* \leftarrow \mathbf{H}$

// release the first node

$\mathbf{H}^*_{[11]} \mathrel{-}= \mathbf{I}$

return $\langle \mathbf{x}^*, \mathbf{H}^* \rangle$

How to get the transformation ?



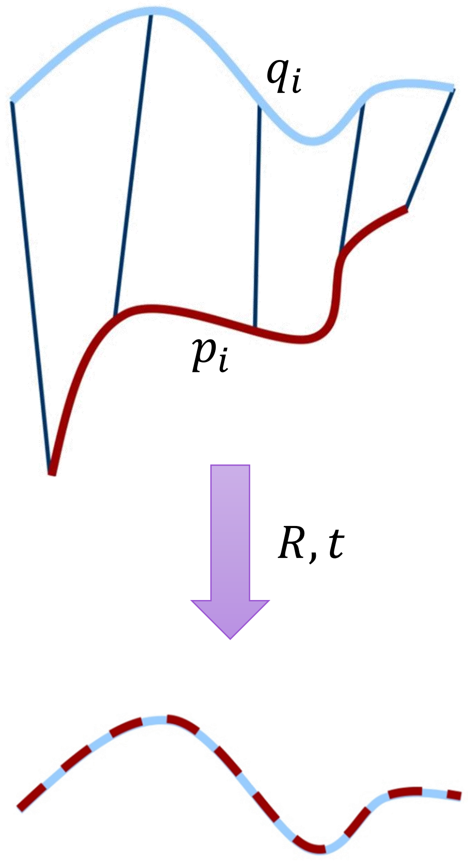
Scan-to-Scan Registration

- Given two matching points sets p_i and q_i, we aims to minimize the least square of registration error:

$$J = \frac{1}{2}\sum_{i=1}^{n}\|q_i - Rp_i - t\|^2$$

- Define the mean of points sets μ_p and μ_q, we can get

$$\begin{aligned}
\frac{1}{2}\sum_{i=1}^{n}\|q_i - Rp_i - t\|^2 &= \frac{1}{2}\sum_{i=1}^{n}\left\|q_i - Rp_i - t - (\mu_q - R\mu_p) + (\mu_q - R\mu_p)\right\|^2 \\
&= \frac{1}{2}\sum_{i=1}^{n}\left\|\left(q_i - \mu_q - R(p_i - \mu_p)\right) + (\mu_q - R\mu_p - t)\right\|^2 \\
&= \frac{1}{2}\sum_{i=1}^{n}\left\|\left(q_i - \mu_q - R(p_i - \mu_p)\right)\right\|^2 + \|\mu_q - R\mu_p - t\|^2 \cancel{+ 2\left(q_i - \mu_q - R(p_i - \mu_p)\right)^T(\mu_q - R\mu_p - t)}
\end{aligned}$$



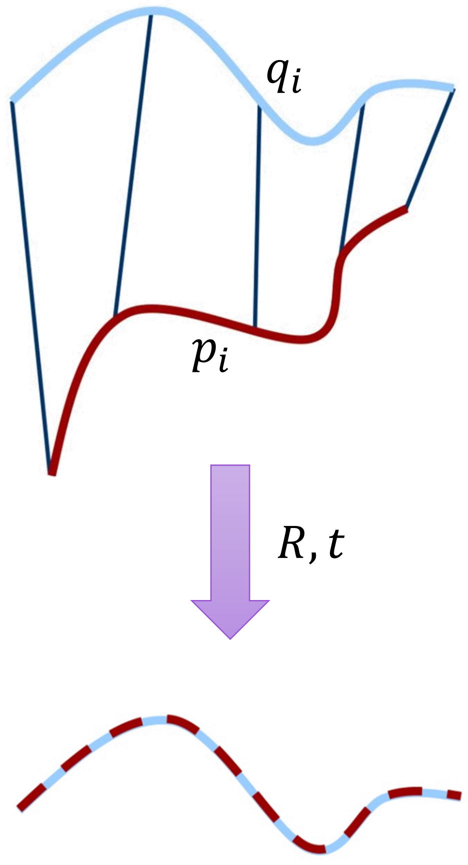
$$\boxed{\begin{aligned}
&\sum_{i=1}^{n}\left(q_i - \mu_q - R(p_i - \mu_p)\right)^T(\mu_q - R\mu_p - t) = (\mu_q - R\mu_p - t)^T\sum_{i=1}^{n}\left(q_i - \mu_q - R(p_i - \mu_p)\right) \\
&= (\mu_q - R\mu_p - t)^T\left(n\mu_q - n\mu_q - R(n\mu_p - n\mu_p)\right) = 0
\end{aligned}}$$

Scan-to-Scan Registration

- Define the relative location p_i' and q_i', the objective function becomes:

$$\frac{1}{2}\sum_{i=1}^{n}\left\|\left(q_i-\mu_q-R\left(p_i-\mu_p\right)\right)\right\|^2+\left\|\mu_q-R\mu_p-t\right\|^2$$
$$=\frac{1}{2}\sum_{i=1}^{n}\|(q_i{}'-Rp_i{}')\|^2+\left\|\mu_q-R\mu_p-t\right\|^2$$

$$\boxed{\begin{aligned}p_i'&=p_i-\mu_p,\\ q_i'&=q_i-\mu_q\end{aligned}}$$



- Divide the optimization process into two steps:

1. Rotation $R^*=\underset{\mathrm{R}}{\operatorname{argmin}}\frac{1}{2}\sum_{i=1}^{n}\|(q_i{}'-Rp_i{}')\|^2$

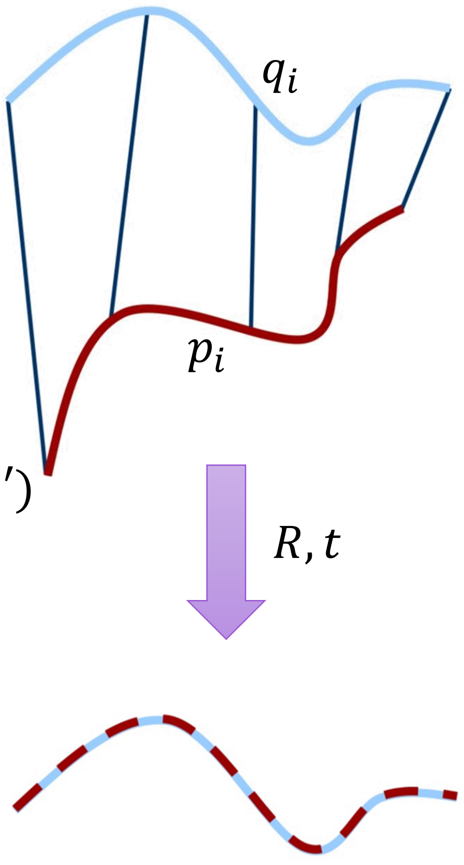
2. Translation $t^*=\mu_q-R^*\mu_p$

Scan-to-Scan Registration

- Solve the rotation term:

$$R^* = \underset{R}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{n} \|(q_i' - Rp_i')\|^2 = \underset{R}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{n} (q_i'^T q_i' + p_i'^T R^T R p_i' - 2 q_i'^T R p_i')$$

$$= \underset{R}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{n} (q_i'^T q_i' + p_i'^T p_i' - 2 q_i'^T R p_i') = \underset{R}{\operatorname{argmin}} \sum_{i=1}^{n} -q_i'^T R p_i'$$



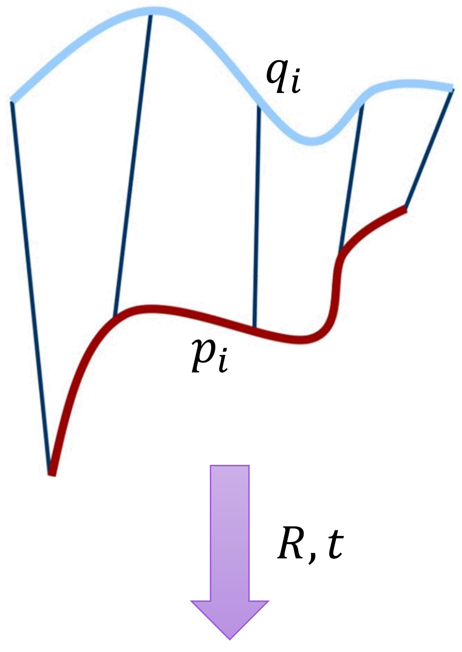
- Minimizing the function is equivalent to maximizing

$$F = \sum_{i=1}^{n} q_i'^T R p_i' = \sum_{i=1}^{n} R q_i'^T p_i' = Trace(RH)$$

, where $H = \sum_{i=1}^{n} q_i'^T p_i'$

Scan-to-Scan Registration

$$H = \sum_{i=1}^{n} {q_i'}^T {p_i}'$$



- we can solve the rotation by the SVD decomposition of H :

$$\underset{R}{\operatorname{argmax}}\, Trace(RH) \Rightarrow H = U\Lambda V^T \Rightarrow R^* = VU^T$$

- Proof:

Lemma:
For any positive definite matrix AA^T
, and any orthonormal matrix B,

$$Trace(AA^T) \geq Trace(BAA^T)$$

Proof of Lemma:
Let a_i be the $\boldsymbol{i}th$ column of A. Then

$$Trace(BAA^T) = Trace(A^T BA) = \sum_i a_i^T (Ba_i)$$

The Cauchy-Schwarz Inequality:

$$a_i^T (Ba_i) \leq \sqrt{(a_i^T a_i)(a_i^T B^T Ba_i)} = a_i^T a_i$$

Hence, $Trace(BAA^T) \leq \sum_i a_i^T a_i = Trace(AA^T)$

SVD decomposition of H :

$$H = U\Lambda V^T$$

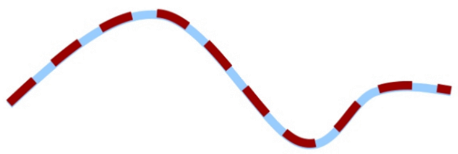
Set $R^* = VU^T$, and we have

$$R^* H = VU^T U\Lambda V^T = V\Lambda V^T$$ (positive definite)

From the Lemma, for ant orthonormal matrix B

$$Trace(R^*H) \geq Trace(\boldsymbol{B}R^*H)$$

Any other rotation



Theorem C.1 **(Cauchy–Schwarz)** *Let V be a linear space with inner product $\langle ., . \rangle$, then for each $\mathbf{a}, \mathbf{b} \in V$ we have:*

$$|\langle \mathbf{a}, \mathbf{b} \rangle|^2 \leq ||\mathbf{a}|| \cdot ||\mathbf{b}||.$$

Proof If $\langle \mathbf{a}, \mathbf{b} \rangle = 0$ then the result is self evident. We therefore assume that $\langle \mathbf{a}, \mathbf{b} \rangle = \alpha \neq 0$, α may of course be complex. We start with the inequality

$$||\mathbf{a} - \lambda \alpha \mathbf{b}||^2 \geq 0$$

where λ is a real number. Now,

$$||\mathbf{a} - \lambda \alpha \mathbf{b}||^2 = \langle \mathbf{a} - \lambda \alpha \mathbf{b}, \mathbf{a} - \lambda \alpha \mathbf{b} \rangle.$$

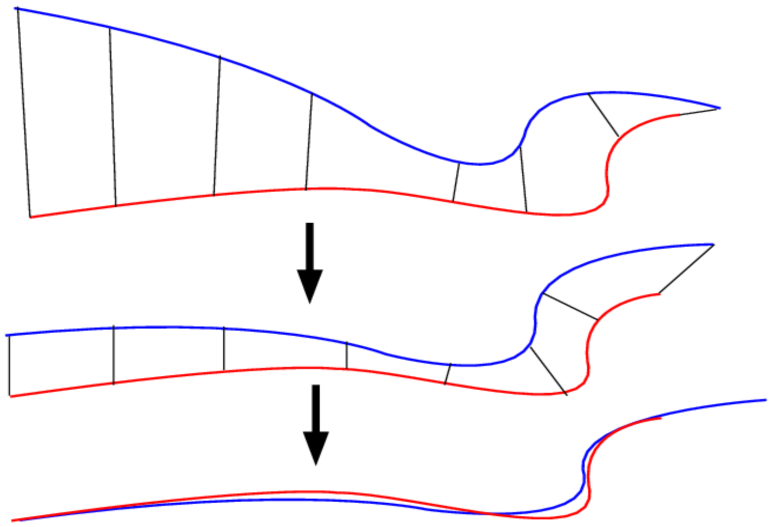
We use the properties of the inner product to expand the right hand side as follows:-

$$\begin{aligned}
\langle \mathbf{a} - \lambda \alpha \mathbf{b}, \mathbf{a} - \lambda \alpha \mathbf{b} \rangle = \langle \mathbf{a}, \mathbf{a} \rangle - \lambda \langle \alpha \mathbf{b}, \mathbf{a} \rangle - \lambda \langle \mathbf{a}, \alpha \mathbf{b} \rangle + \lambda^2 |\alpha|^2 \langle \mathbf{b}, \mathbf{b} \rangle \geq 0 \\
\text{so } ||\mathbf{a}||^2 - \lambda \alpha \langle \mathbf{b}, \mathbf{a} \rangle - \lambda \bar{\alpha} \langle \mathbf{a}, \mathbf{b} \rangle + \lambda^2 |\alpha|^2 ||\mathbf{b}||^2 \geq 0 \\
\text{i.e. } ||\mathbf{a}||^2 - \lambda \alpha \bar{\alpha} - \lambda \bar{\alpha} \alpha + \lambda^2 |\alpha|^2 ||\mathbf{b}||^2 \geq 0 \\
\text{so } ||\mathbf{a}||^2 - 2\lambda |\alpha|^2 + \lambda^2 |\alpha|^2 ||\mathbf{b}||^2 \geq 0.
\end{aligned}$$

Scan-to-Scan Registration

- Iterative Closest Points (ICP) Algorithm

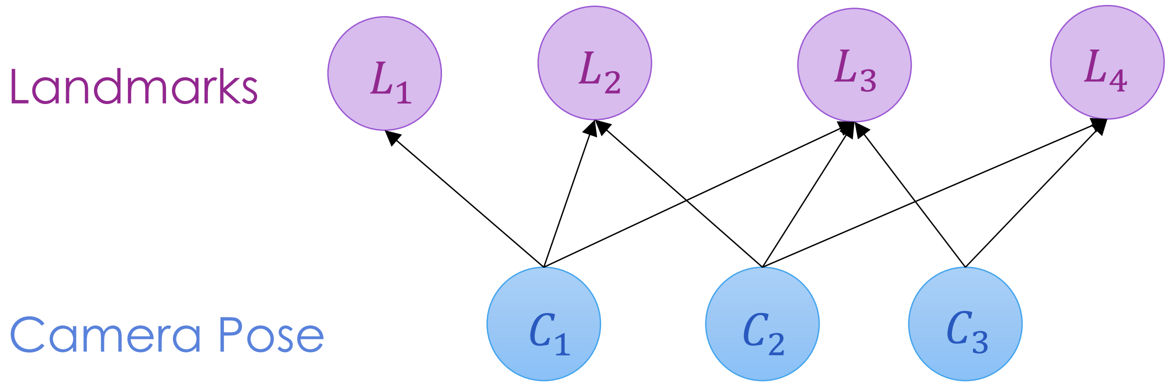
Given two points sets P and Q



Initialize $R_0 = I, t_0 = 0$
Build the kd-tree of Q
Repeat
Transform the points set $\hat{p}_i = R_k p_i + t_k$
Search the nearest points pairs $[q_i, \hat{p}_i]$
Compute mean of points sets and the relative location $\hat{p}_i{}' = \hat{p}_i{}' - \mu_{\hat{p}}$ =and $q_i' = q_i - \mu_q$
SVD Decomposition: $H = U \Lambda V^T$, where $H = \sum_{i=1}^{n} q_i'^T \hat{p}_i{}'$
Get the optimize transformation $R^* = VU^T$ and $t^* = \mu_q - R^* \mu_p$
Update the transformation $R_k = R^* R_{k-1}$ and $t_k = R^* t_{k-1} + t^*$
Until Convergence

Graph Optimization for Map and Pose

- Bundle Adjustment
- The bipartite optimization graph



- Given observation model $z_{ij} = h(C_i, L_j)$, the objective is to minimize the observation error:

$$F = \sum_{ij} \left\| z_{ij}^{obs} - h(C_i, L_j) \right\|^2$$

Sparse Hessian and Marginalization

- The Jacobian matrix of observation error and the approximated Hessian:

$$J_{ij} = \frac{\partial e_{ij}}{\partial \mathrm{x}} = [\underbrace{0, \ldots, 0, \frac{\partial e_{ij}}{\partial C_i}, 0, \ldots,}_{\text{Camera Pose}} 0, \underbrace{0, \ldots, 0, \frac{\partial e_{ij}}{\partial L_j}, 0, \ldots, 0}_{\text{Landmarks}}] \qquad H \cong J^T J = \begin{bmatrix} H_{ii} & H_{ij} \\ H_{ji} & H_{jj} \end{bmatrix}$$

(Arrow-Like Matrix)

- Schur Elimination and Marginalization

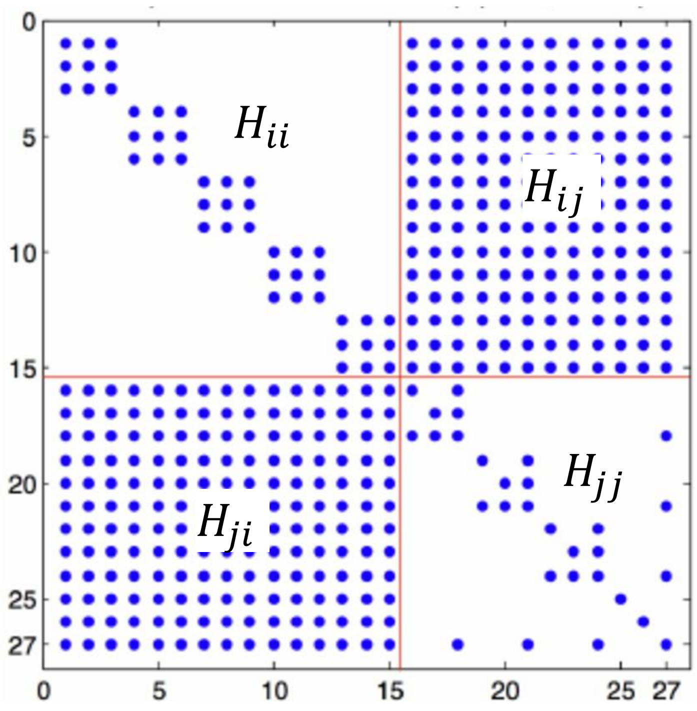
$$H\Delta\mathrm{x} = -b \rightarrow \begin{bmatrix} H_{ii} & H_{ij} \\ H_{ij}^T & H_{jj} \end{bmatrix} \begin{bmatrix} \Delta\mathrm{x}_C \\ \Delta\mathrm{x}_L \end{bmatrix} = \begin{bmatrix} v \\ w \end{bmatrix}$$

$$\begin{bmatrix} I & -H_{ij}H_{jj}^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} H_{ii} & H_{ij} \\ H_{ij}^T & H_{jj} \end{bmatrix} \begin{bmatrix} \Delta\mathrm{x}_C \\ \Delta\mathrm{x}_L \end{bmatrix} = \begin{bmatrix} I & -H_{ij}H_{jj}^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}$$

$$\begin{bmatrix} H_{ii} - H_{ij}H_{jj}^{-1}H_{ij}^T & 0 \\ H_{ij}^T & H_{jj} \end{bmatrix} \begin{bmatrix} \Delta\mathrm{x}_C \\ \Delta\mathrm{x}_L \end{bmatrix} = \begin{bmatrix} v - H_{ij}H_{jj}^{-1}w \\ w \end{bmatrix}$$

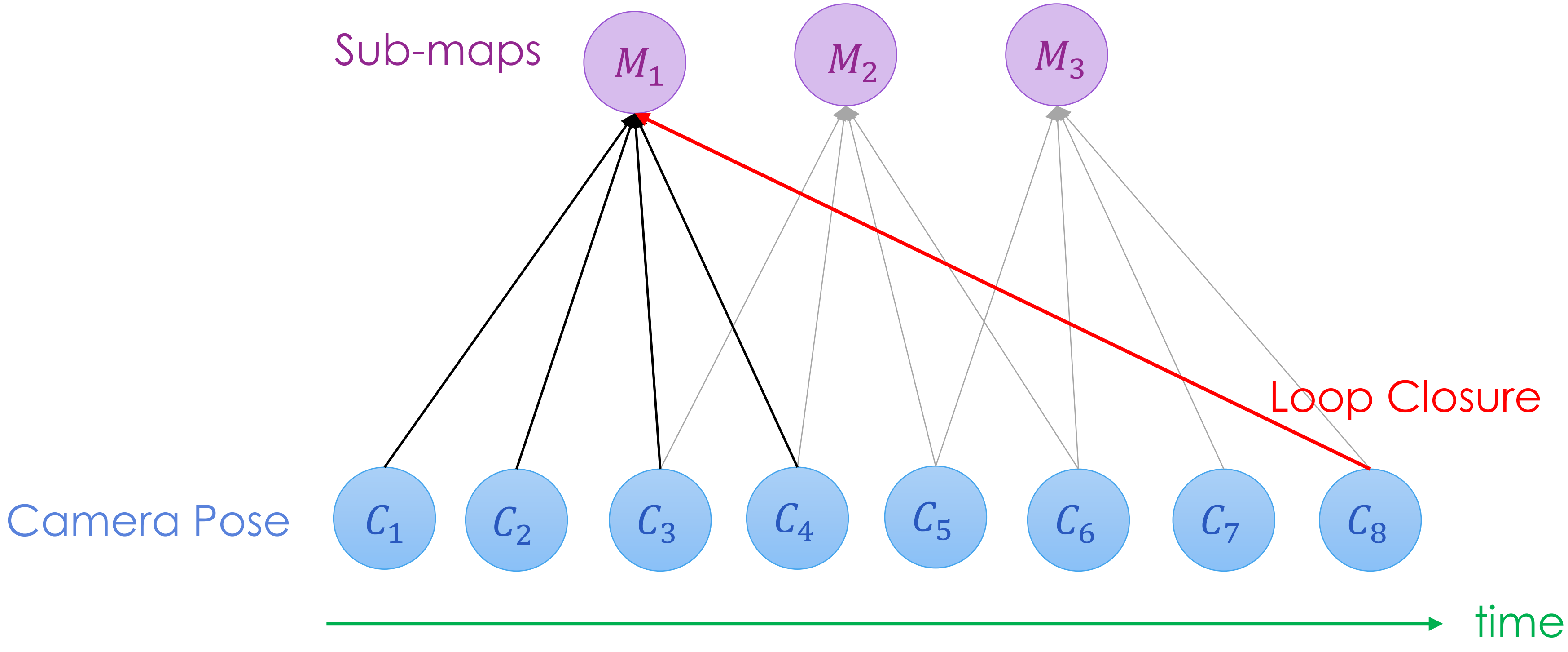
$$[H_{ii} - H_{ij}H_{jj}^{-1}H_{ij}^T]\Delta\mathrm{x}_C = v - H_{ij}H_{jj}^{-1}w$$

Easy to compute !!



Graph Optimization for Grid-based SLAM

- Karto-SLAM (Open-Source) / Cartographer (Google)



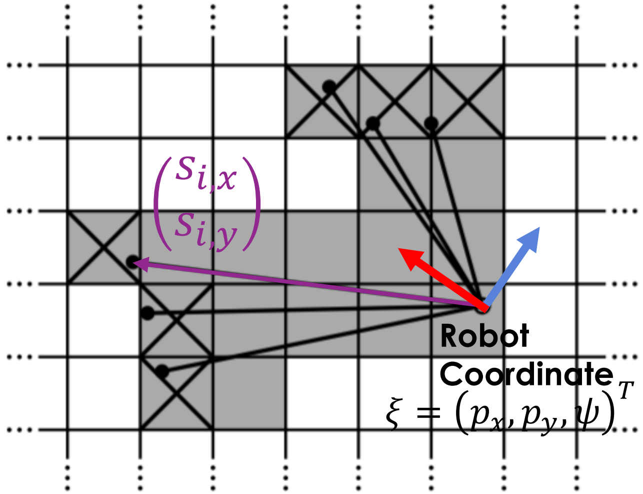
Scan-to-Map Matching

- Define the Robot Pose State $\xi = (p_x, p_y, \psi)^T$ and the Optimization Objective:

$$\xi^* = \text{argmin}_\xi \sum_{i=1}^{n} [1 - M(S_i(\xi))]^2 \text{ , where } S_i(\xi) = \begin{pmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{pmatrix} \begin{pmatrix} s_{i,x} \\ s_{i,y} \end{pmatrix} + \begin{pmatrix} p_x \\ p_y \end{pmatrix}$$

- Apply the 1st order Taylor approximation

$$\sum_{i=1}^{n} [1 - M(S_i(\xi))]^2 \approx \sum_{i=1}^{n} \left[1 - M(S_i(\xi)) - \nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \Delta\xi\right]^2$$



- Partial Derivative to $\Delta\xi$

$$2 \sum_{i=1}^{n} \left[\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi}\right]^T \left[1 - M(S_i(\xi)) - \nabla M(s_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \Delta\xi\right] = 0$$

Scan-to-Map Matching

- Solving the problem by GN methods:

$$2\sum_{i=1}^{n}\left[\nabla M(S_i(\xi))\frac{\partial S_i(\xi)}{\partial \xi}\right]^T\left[1 - M\big(S_i(\xi)\big) - \nabla M(s_i(\xi))\frac{\partial S_i(\xi)}{\partial \xi}\Delta\xi\right] = 0$$

$$\underbrace{\left[\nabla M\big(S_i(\xi)\big)\frac{\partial S_i(\xi)}{\partial \xi}\right]^T\left[\nabla M\big(S_i(\xi)\big)\frac{\partial S_i(\xi)}{\partial \xi}\right]}_{H}\underbrace{\Delta\xi}_{\Delta \mathrm{x}} = \underbrace{\sum_{i=1}^{n}\left[\nabla M(S_i(\xi))\frac{\partial S_i(\xi)}{\partial \xi}\right]^T\left[1 - M(S_i(\xi))\right]}_{-b}$$

$$\Delta\xi = H^{-1}\sum_{i=1}^{n}\left[\nabla M(S_i(\xi))\frac{\partial S_i(\xi)}{\partial \xi}\right]^T\left[1 - M(S_i(\xi))\right]$$

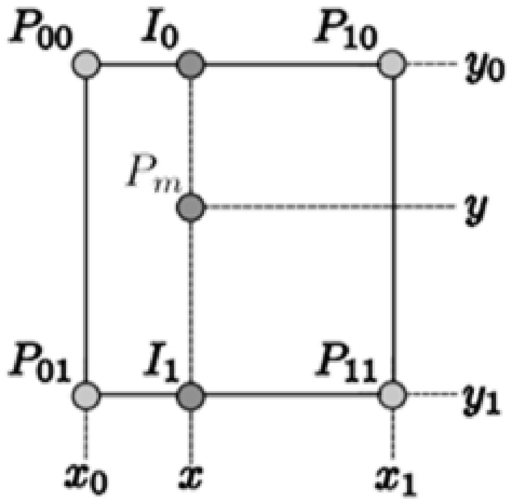
$$\boxed{\frac{\partial S_i(\xi)}{\partial \xi} = \begin{pmatrix} 1 & 0 & -\sin(\psi)\, s_{i,x} - cos(\psi)\, s_{i,y} \\ 0 & 1 & \cos(\psi)\, s_{i,x} - \sin(\psi)\, s_{i,y} \end{pmatrix}}$$

, where $H = \left[\nabla M(S_i(\xi))\frac{\partial S_i(\xi)}{\partial \xi}\right]^T\left[\nabla M(S_i(\xi))\frac{\partial S_i(\xi)}{\partial \xi}\right]$

Scan-to-Map Matching

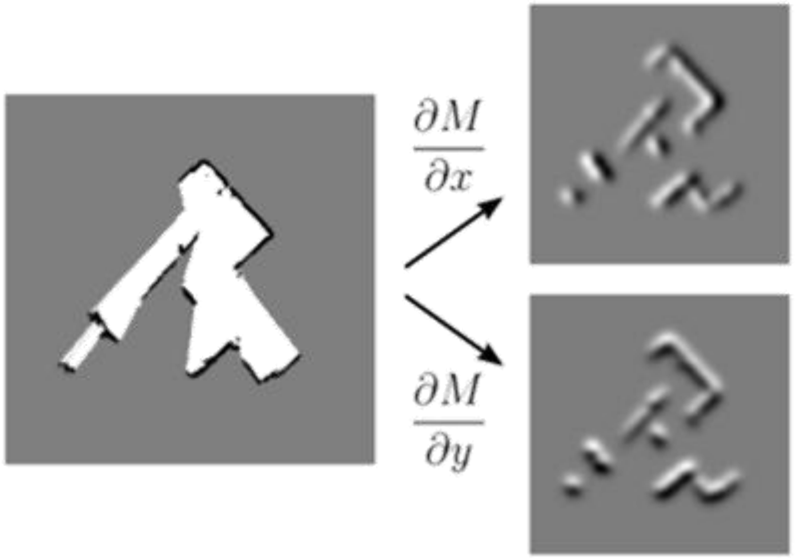
- The derivative of map with respect to location.

$$M(P_m) \approx \frac{y-y_0}{y_1-y_0}\left(\frac{x-x_0}{x_1-x_0}M(P_{11}) + \frac{x_1-x}{x_1-x_0}M(P_{01})\right) + \frac{y_1-y}{y_1-y_0}\left(\frac{x-x_0}{x_1-x_0}M(P_{10}) + \frac{x_1-x}{x_1-x_0}M(P_{00})\right)$$

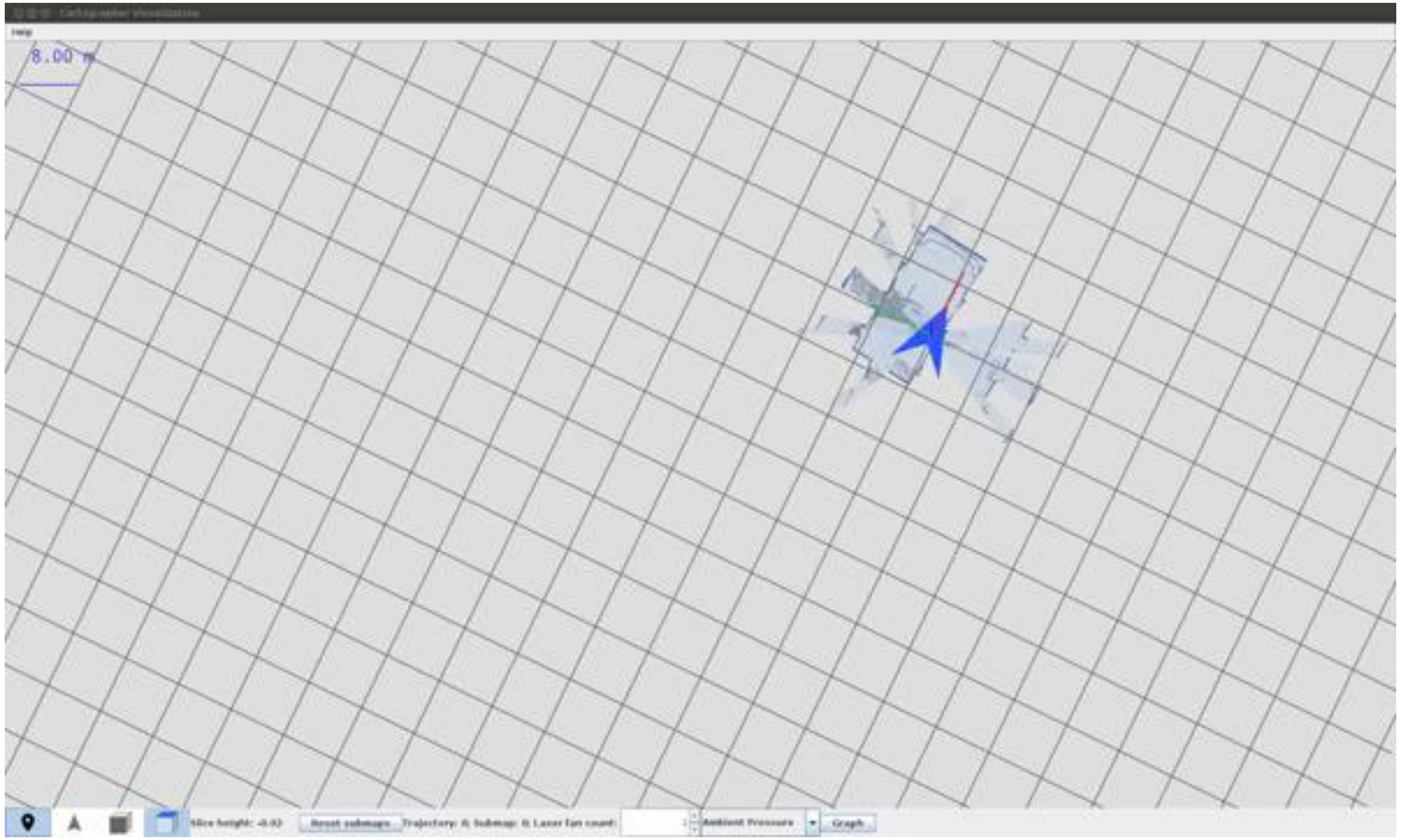


$$\frac{\partial M}{\partial x}(P_m) \approx \frac{y-y_0}{y_1-y_0}(M(P_{11}) - M(P_{01})) + \frac{y_1-y}{y_1-y_0}(M(P_{10}) - M(P_{00}))$$

$$\frac{\partial M}{\partial y}(P_m) \approx \frac{x-x_0}{x_1-x_0}(M(P_{11}) - M(P_{10})) + \frac{x_1-x}{x_1-x_0}(M(P_{01}) - M(P_{00}))$$



Cartographer Demo



Appendix: Non-linear Optimization

Basics of Optimization

Least Squares Problem

m: number of data points

n: number of parameters

Find $\mathbf{x}^*$, a local minimizer for

$$F(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{m}(f_i(\mathbf{x}))^2 ,$$

where $f_i : \mathbb{R}^n \mapsto \mathbb{R},\ i=1,\ldots,m$ are given functions, and $m \geq n$.

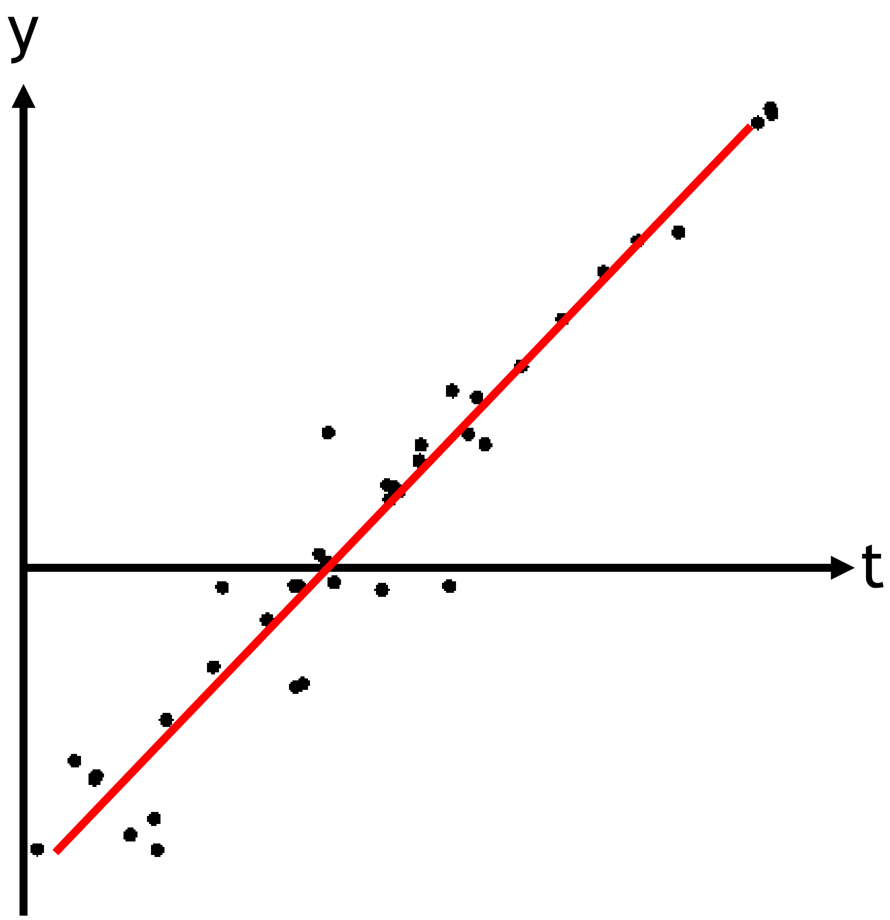
$$\frac{dF}{d\mathbf{x}} = 0$$

Local Minimizer

Given $F : \mathbb{R}^n \mapsto \mathbb{R}$. Find $\mathbf{x}^*$ so that

$$F(\mathbf{x}^*) \leq F(\mathbf{x}) \quad \text{for} \quad \|\mathbf{x} - \mathbf{x}^*\| < \delta .$$

Example: Linear Least Square Fitting



model parameters

$$y(t) = \boxed{M}(t;\boxed{\mathbf{x}}) = x_0 + x_1 t$$

$$f_i(x) = y_i - \boxed{M(t_i;\mathbf{x})}$$

Residual(error) prediction

$M(t;\mathbf{x}) = x_0 + x_1 t + x_2 t^3$ is linear, too.

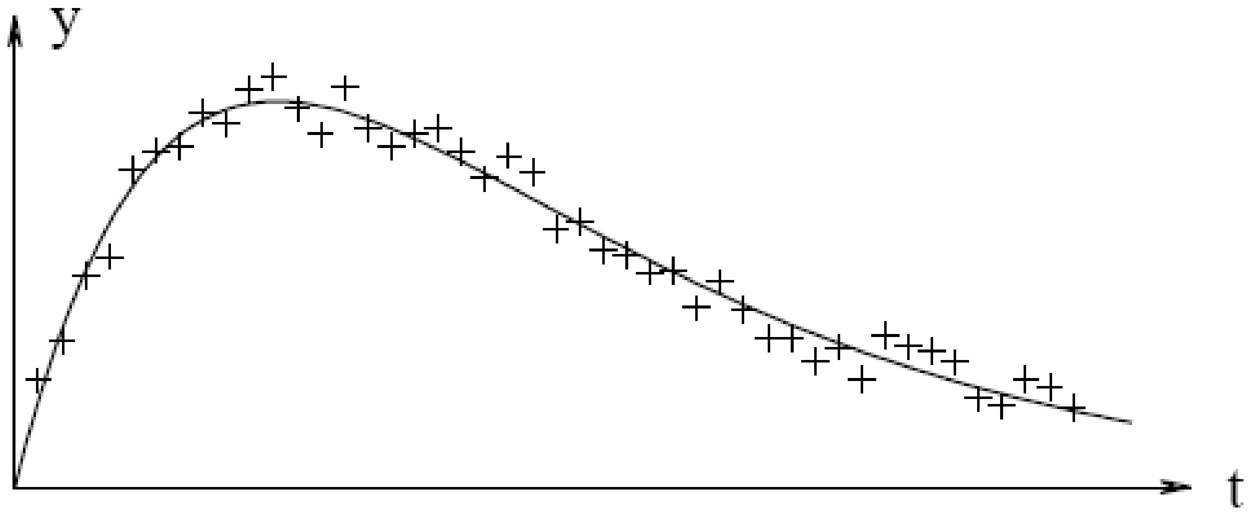
Example: Nonlinear Least Square Fitting

parameters

$$\mathbf{x} = [x_1, x_2, x_3, x_4]^T$$

model

$$M(t;\mathbf{x}) = x_3 e^{x_1 t} + x_4 e^{x_2 t}$$



residuals

$$f_i(\mathbf{x}) = y_i - M(t_i;\mathbf{x})$$
$$= y_i - \left(x_3 e^{x_1 t} + x_4 e^{x_2 t}\right)$$

Function Minimization

Taylor expansion $$F(\mathbf{x}+\mathbf{h}) \approx F(\mathbf{x}) + \boldsymbol{J}(\mathbf{x})^{\boldsymbol{T}}\mathbf{h} + \frac{\mathbf{1}}{\mathbf{2}}\mathbf{h}^{\boldsymbol{T}}\boldsymbol{H}(\mathbf{x})\mathbf{h}$$

$$\boldsymbol{J}(\mathbf{x}) \equiv \boldsymbol{F}'(\mathbf{x}) = \begin{bmatrix} \frac{\partial F}{\partial x_1}(\mathbf{x}) \\ \vdots \\ \frac{\partial F}{\partial x_n}(\mathbf{x}) \end{bmatrix}$$

$$\boldsymbol{H}(\mathbf{x}) \equiv \boldsymbol{F}''(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 F}{\partial x_1^2}(\mathbf{x}) & \frac{\partial^2 F}{\partial x_1 \partial x_2}(\mathbf{x}) & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_n}(\mathbf{x}) \\ \frac{\partial^2 F}{\partial x_2 \partial x_1}(\mathbf{x}) & \frac{\partial^2 F}{\partial x_2^2}(\mathbf{x}) & \cdots & \frac{\partial^2 F}{\partial x_2 \partial x_n}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 F}{\partial x_n \partial x_1}(\mathbf{x}) & \frac{\partial^2 F}{\partial x_n \partial x_2}(\mathbf{x}) & \cdots & \frac{\partial^2 F}{\partial x_n^2}(\mathbf{x}) \end{bmatrix}$$

Function Minimization

Necessary condition for a local minimizer :

$$\boldsymbol{J}(\mathbf{x}^*) \equiv F'(\mathbf{x}) = \mathbf{0}$$

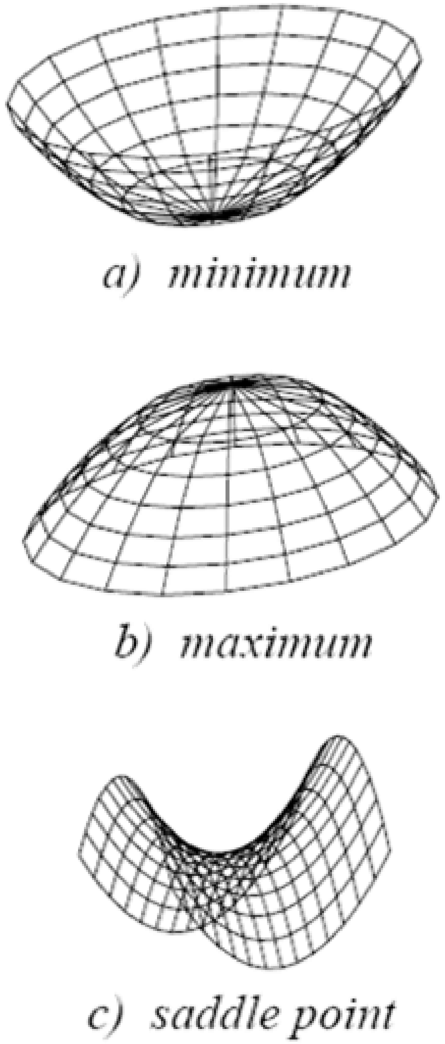
Why?

By definition, if $\mathbf{x}^*$ is a local minimizer,

$$||\mathbf{h}|| \text{ is small enough} \rightarrow F(||\mathbf{x}^* + \mathbf{h}||) > F(\mathbf{x}^*)$$

$$F(\mathbf{x}^* + \mathbf{h}) \approx F(\mathbf{x}^*) + \boldsymbol{J}(\mathbf{x}^*)^{\boldsymbol{T}}\mathbf{h} + \frac{\mathbf{1}}{\mathbf{2}}\mathbf{h}^{\boldsymbol{T}}\boldsymbol{H}(\mathbf{x})\mathbf{h} > F(\mathbf{x}^*)$$

$$F(\mathbf{x}^* - \mathbf{h}) \approx F(\mathbf{x}^*) - \boldsymbol{J}(\mathbf{x}^*)^{\boldsymbol{T}}\mathbf{h} + \frac{\mathbf{1}}{\mathbf{2}}\mathbf{h}^{\boldsymbol{T}}\boldsymbol{H}(\mathbf{x})\mathbf{h} < F(\mathbf{x}^*)$$



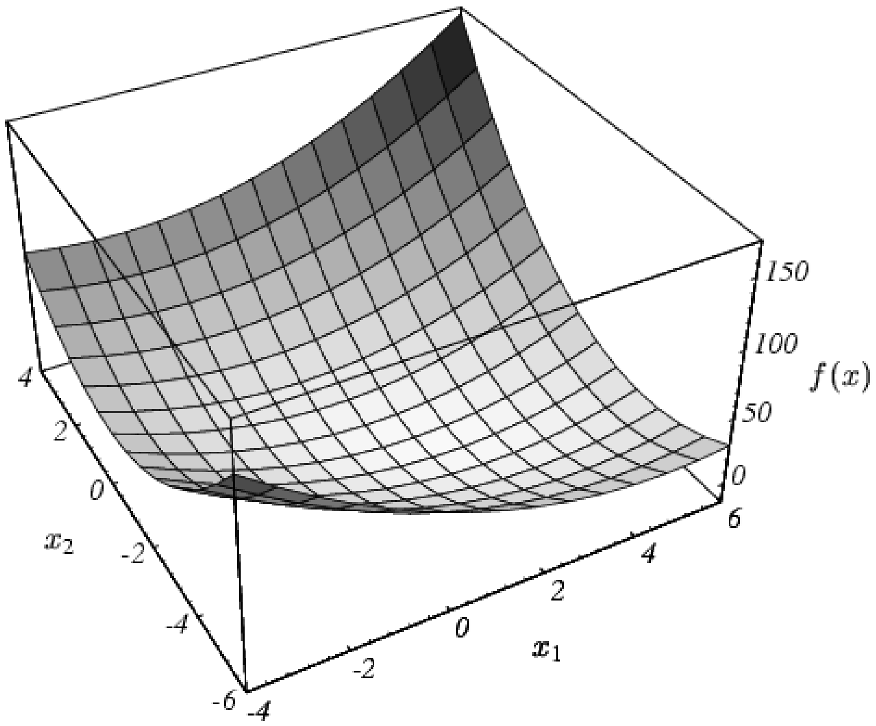
Quadratic Functions

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathbf{T}}\mathbf{A}\mathbf{x} - \mathbf{b}^{\mathbf{T}}\mathbf{x} + \mathbf{c}$$

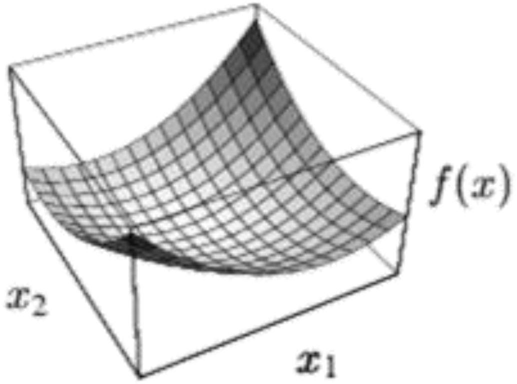
$$\boldsymbol{A} = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}$$

$$\boldsymbol{b} = \begin{bmatrix} 2 \\ -8 \end{bmatrix}$$

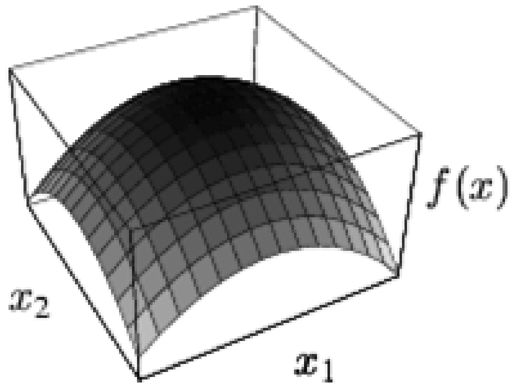
$$\boldsymbol{c} = 0$$



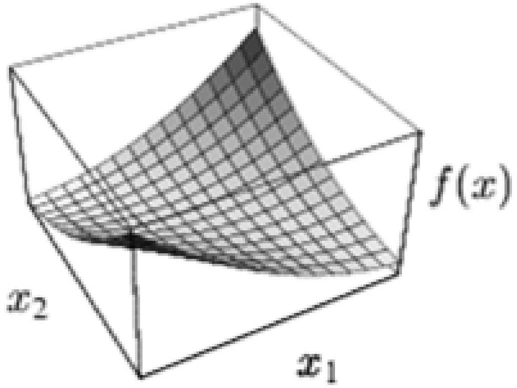
Quadratic Functions



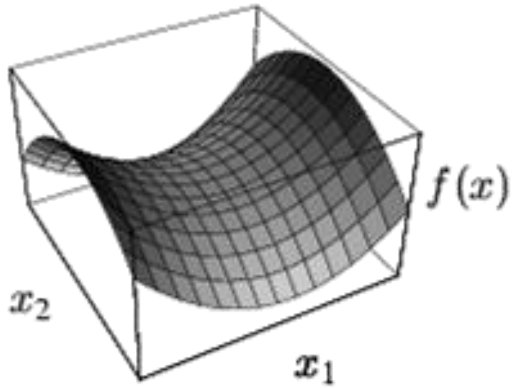
A is positive definite.
All eigenvalues are positive.
For all x, $x^TAx>0$.



A is negative definite.
All eigenvalues are negative.
For all x, $x^TAx<0$.



A is singular



A is indefinite

Descent Methods

$$\mathbf{x}_0,\ \mathbf{x}_1,\ \mathbf{x}_2,\ \ldots\ ,\ \mathbf{x}_k\ \rightarrow\ \mathbf{x}^*\quad \text{for}\quad k \rightarrow \infty$$

Local Minimizer

Given $F:\ \mathbb{R}^n \mapsto \mathbb{R}$. Find $\mathbf{x}^*$ so that

$$F(\mathbf{x}^*) \leq F(\mathbf{x})\quad \text{for}\quad \|\mathbf{x} - \mathbf{x}^*\| < \delta\ .$$

Initialize $\mathbf{x} = \mathbf{x}_0$
For i=0~K
Find $\mathbf{h}$ such that $\|f(\mathbf{x}_i + \boldsymbol{\alpha}\mathbf{h})\|$ can reach the minimum
If $\mathbf{h}$ is smaller than ϵ, stop
else $\mathbf{x} = \mathbf{x}+\alpha\mathbf{h}$

Descent Direction (Line Search Method)

$$\begin{aligned} F(\mathbf{x}+\alpha\mathbf{h}) &= F(\mathbf{x}) + \alpha\mathbf{h}^{\top}\mathbf{F}'(\mathbf{x}) + O(\alpha^2) \\ &\simeq F(\mathbf{x}) + \alpha\mathbf{h}^{\top}\mathbf{F}'(\mathbf{x}) \quad \text{for } \alpha \text{ sufficiently small.} \end{aligned}$$

Definition of descent direction:

$\mathbf{h}$ is a descent direction for F at $\mathbf{x}$ if $\mathbf{hF'(x)} < 0$

Steepest Descent Method

$$F(\mathbf{x}+\alpha\mathbf{h}) = F(\mathbf{x}) + \alpha\mathbf{h}^\top\mathbf{F}'(\mathbf{x}) + O(\alpha^2)$$
$$\simeq F(\mathbf{x}) + \alpha\mathbf{h}^\top\mathbf{F}'(\mathbf{x}) \quad \text{for } \alpha \text{ sufficiently small.}$$

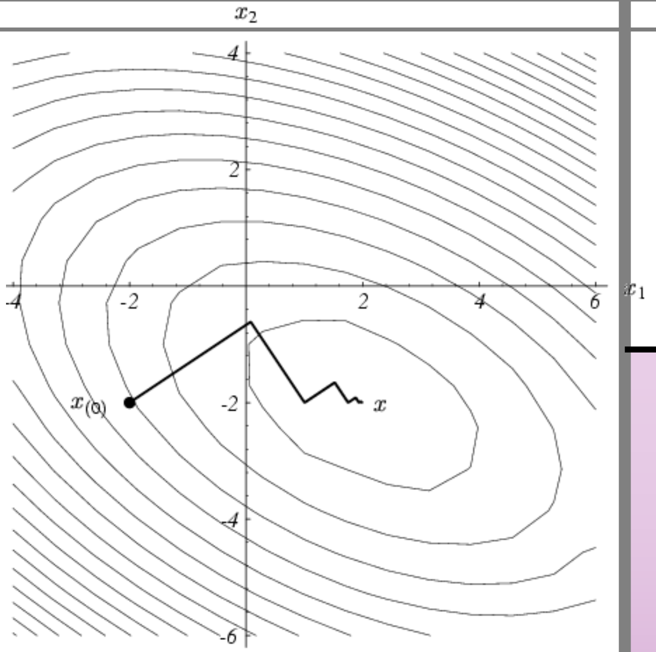
$$\boxed{\frac{F(\mathbf{x}) - F(\mathbf{x}+\alpha\mathbf{h})}{\alpha\|\mathbf{h}\|}} = -\frac{1}{\|\mathbf{h}\|}\mathbf{h}^\top\mathbf{F}'(\mathbf{x}) = -\|\mathbf{F}'(\mathbf{x})\|\cos\theta$$

the decrease of ***F(x)*** per unit along h direction

$$\text{greatest gain rate if } \theta = \pi \quad \rightarrow \quad \mathbf{h}_{\text{sd}} = -\mathbf{F}'(\mathbf{x})$$

h_{sd} is a descent direction because $\mathbf{h}_{sd}^T F'(\mathbf{x}) = -F'(\mathbf{x})^2 < 0$

Steepest Descent Method



$\varphi(\alpha) = F(\mathbf{x} + \alpha\mathbf{h})$, $\mathbf{x}$ and $\mathbf{h}$ are fixed, $\alpha \geq 0$.

Find α so that $\varphi(\alpha) = F(\mathbf{x} + \alpha\mathbf{h})$ is minimum.

$$0 = \frac{\partial\varphi(\alpha)}{\partial\alpha} = \frac{\partial F(\mathbf{x}+\alpha\mathbf{h})}{\partial\alpha} = \frac{\partial F(\mathbf{x}+\alpha\mathbf{h})}{\partial(\mathbf{x}+\alpha\mathbf{h})}\frac{\partial(\mathbf{x}+\alpha\mathbf{h})}{\partial\alpha} = \mathbf{h}^T F'(\mathbf{x} + \alpha\mathbf{h})$$

$$\mathbf{h} = -F'(\mathbf{x})$$

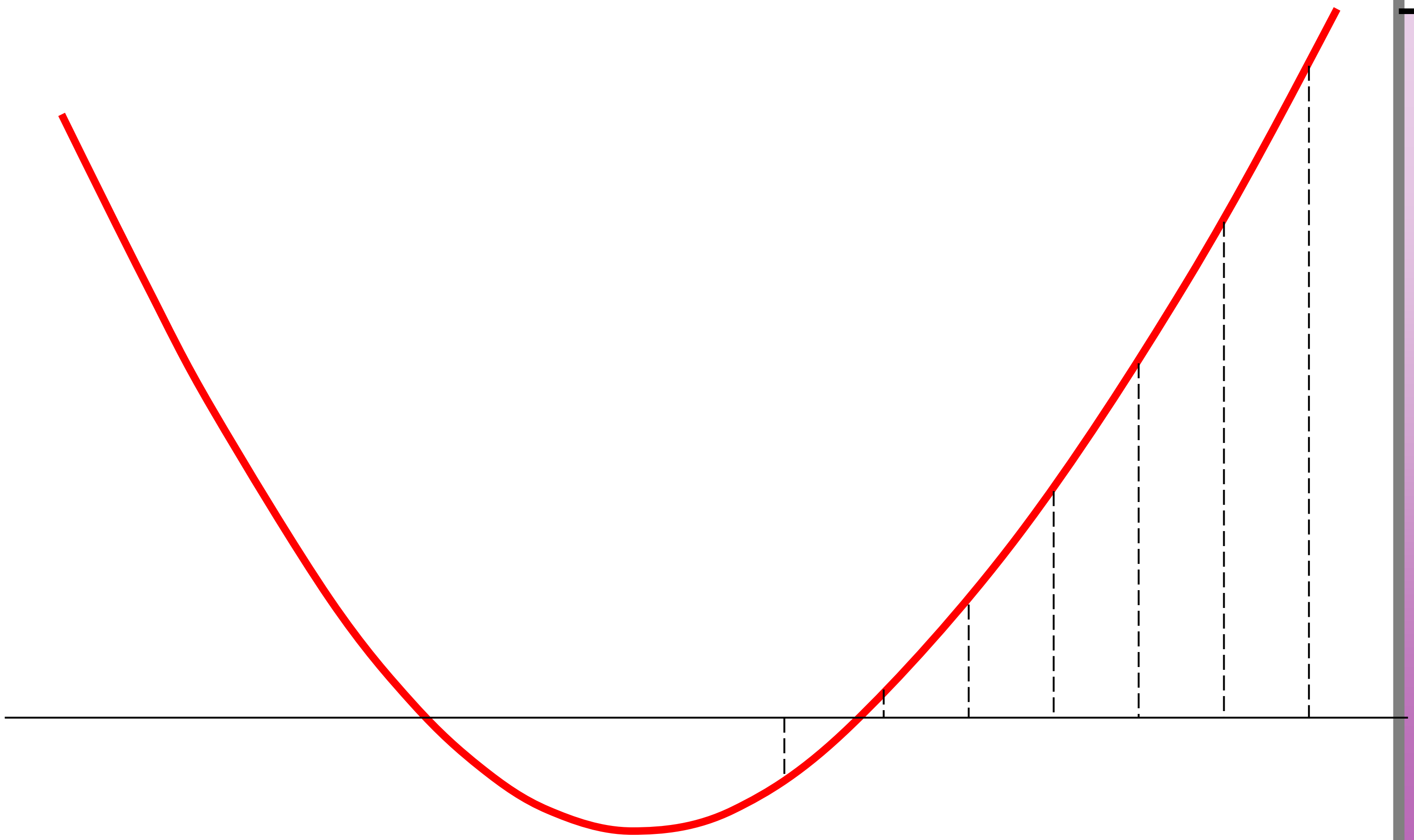
$$=\mathbf{h}^T\ (F'(\mathbf{x}) + \alpha \boldsymbol{F}''(\mathrm{x})^{\boldsymbol{T}}\mathbf{h})=\mathbf{h}^T(-\mathbf{h} + \alpha\mathbf{H}\mathbf{h})$$

$$\alpha = \frac{\mathbf{h}^{\mathrm{T}}\mathbf{h}}{\mathbf{h}^{\mathrm{T}}\mathbf{H}\mathbf{h}}$$

Problem: Has good performance in the initial stages of the iterative process, but converge very slow with a linear rate.

Newton's Method

- Root finding for $f(x)=0$
- March x and test signs
- Determine Δx
  (small→slow; large→ miss)



Newton's Method

- Root finding for $f(x)=0$

Taylor's expansion:

$$f(x_0+\varepsilon)=f(x_0)+f'(x_0)\varepsilon+\frac{1}{2}f''(x_0)\varepsilon^2+\ldots$$

$$0=f(x_0+\varepsilon)\approx f(x_0)+f'(x_0)\varepsilon$$

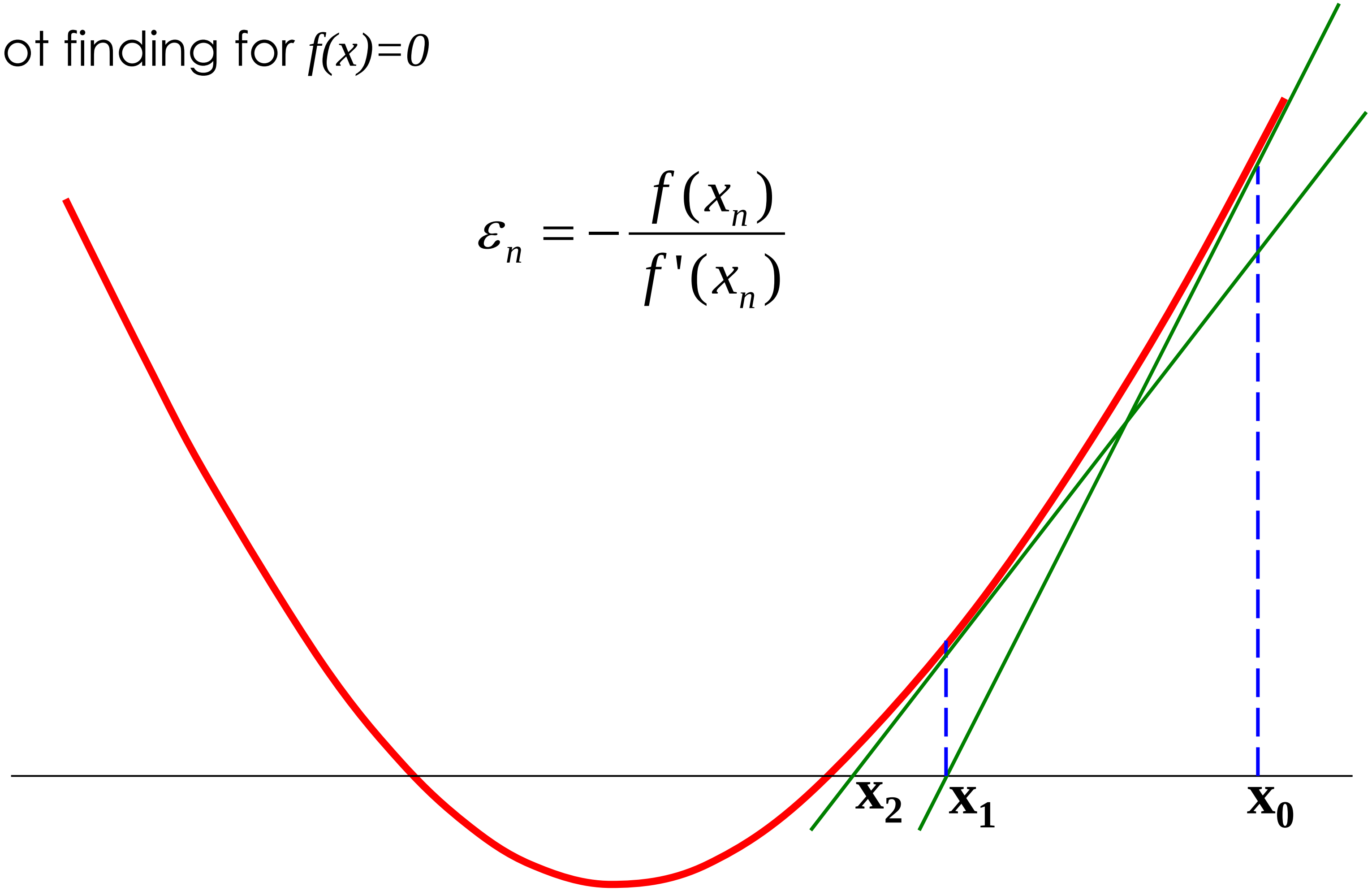
$$\varepsilon=-\frac{f(x_0)}{f'(x_0)}$$

$$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)}$$

Newton's Method

- Root finding for $f(x)=0$

$$\varepsilon_n = -\frac{f(x_n)}{f'(x_n)}$$



Newton's Method

$\mathbf{x}^*$ is a stationary point $\longrightarrow$ it satisfies $\mathbf{F}'(\mathbf{x}^*) = \mathbf{0}$.

$$\begin{aligned} \mathbf{F}'(\mathbf{x}+\mathbf{h}) &= \mathbf{F}'(\mathbf{x}) + \mathbf{F}''(\mathbf{x})\mathbf{h} + O(\|\mathbf{h}\|^2) \\ &\simeq \mathbf{F}'(\mathbf{x}) + \mathbf{F}''(\mathbf{x})\mathbf{h} \quad \text{for } \|\mathbf{h}\| \text{ sufficiently small} \\ &= 0 \end{aligned}$$

$h_n = -\dfrac{F'(x)}{F''(x)}$ $\longrightarrow$ $\mathbf{H}\,\mathbf{h}_n = -\mathbf{F}'(\mathbf{x})$ with $\mathbf{H} = \mathbf{F}''(\mathbf{x})$

$\mathbf{x} := \mathbf{x} + \mathbf{h}_n$

Suppose that $\mathbf{H}$ is positive definite

$\longrightarrow$ $\mathbf{u}^\top \mathbf{H}\,\mathbf{u} > 0$ for all nonzero $\mathbf{u}$.

$\longrightarrow$ $0 < \mathbf{h}_n^\top \mathbf{H}\,\mathbf{h}_n = -\mathbf{h}_n^\top \mathbf{F}'(\mathbf{x})$

$\longrightarrow$ $\mathbf{h}_n$ is a descent direction

Newton's Method

$$\mathbf{H}\mathbf{h} = -F'(\mathbf{x})$$

$$\mathbf{h} = -\mathbf{H}^{-1}\mathbf{J}$$

- It has good performance in the final stage of the iterative process, where x is close to x*.
- It requires solving a linear system and H is not always positive definite.

➔ Use the approximate Hessian $\mathbf{H} \approx \mathbf{J}^{\mathrm{T}}\mathbf{J}$ Gauss-Newton

Gauss-Newton

$$\mathbf{h}^* = \operatorname{argmin} \frac{1}{2} \sum_{i=1}^{m} \|f_i(\mathbf{x}+\mathbf{h})\|^2$$

$$f(\mathbf{x}+\mathbf{h}) \approx f(\mathbf{x}) + \mathbf{J}(\mathbf{x})^T\mathbf{h}$$

$$\frac{1}{2}\|f(\mathbf{x}+\mathbf{h})\|^2 \approx \frac{1}{2}\left\|f(\mathbf{x}) + \mathbf{J}(\mathbf{x})^T\mathbf{h}\right\|^2 = \frac{1}{2}\left(\|f(\mathbf{x})\|^2 + 2f(\mathbf{x})\mathbf{J}(\mathbf{x})^T\mathbf{h} + \mathbf{h}^T\mathbf{J}(\mathbf{x})\mathbf{J}(\mathbf{x})^T\mathbf{h}\right)$$

$$\mathbf{J}(\mathbf{x})f(\mathbf{x})^T + \mathbf{J}(\mathbf{x})\mathbf{J}(\mathrm{x})^T\mathbf{h} = \mathbf{0}$$

$$\underbrace{\mathbf{J}(\mathbf{x})\mathbf{J}(\mathbf{x})^T}_{\mathbf{H}(\mathbf{x})}\mathbf{h} = -\underbrace{\mathbf{J}(\mathbf{x})f(\mathbf{x})^T}_{\mathbf{g}(\mathbf{x})}$$

Newton's Method:

$$\mathbf{Hh} = -F'(\mathbf{x})$$

Levenberg-Marquardt Method (LM)

- LM can be thought of as a combination of steepest descent and the Newton method.
  - When the current solution is far from the correct one, the algorithm behaves like a steepest descent method: slow, but guaranteed to converge.
  - When the current solution is close to the correct solution, it becomes a Newton's method.

$$\begin{array}{l} \textbf{if}\ \ \mathbf{F}''(\mathbf{x})\ \text{is positive definite} \\ \quad \mathbf{h} := \mathbf{h}_{\mathrm{n}} \\ \textbf{else} \\ \quad \mathbf{h} := \mathbf{h}_{\mathrm{sd}} \\ \mathbf{x} := \mathbf{x} + \alpha\mathbf{h} \end{array}$$

true-region method

$$\boldsymbol{\rho} = \frac{\boldsymbol{f}(\mathbf{x}+\mathbf{h}) - \boldsymbol{f}(\mathbf{x})}{\mathbf{J}(\mathbf{x})^{T}\mathbf{h}}$$

This needs to calculate second-order derivative which might not be available.

$f(\mathbf{x}+\mathbf{h}) \approx f(\mathbf{x}) + \mathbf{J}(\mathbf{x})^T\mathbf{h}$

Levenberg-Marquardt Method (LM)

Initialize $\mathbf{x} = \mathbf{x}_0$, $\mu=\mu_0$
For i=0~K
Find $\mathbf{h}$ such that $\min_{\mathbf{h}} \frac{1}{2}\|\mathrm{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})^T\mathbf{h}\|^2$ s.t $\|\mathbf{Dh}\|^2 \leq \mu$
Calculate ρ
If $\rho \geq \frac{3}{4}$
$\mu = 2\mu$
If $\rho < \frac{1}{4}$
$\mu = 0.5\mu$
If $\rho \geq Th$
else $\mathbf{x} = \mathbf{x}+\mathbf{h}$
If $\mathbf{h}$ is smaller than ϵ, stop

$$\mathcal{L}(\mathbf{h},\lambda) = \frac{1}{2}\|\mathrm{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})^T\mathbf{h}\|^2 + \lambda(\|\mathbf{Dh}\|^2 - \mu)$$

$$\nabla\mathcal{L}(\mathbf{h},\lambda) = 0$$

$$\left(\mathbf{J}(\mathbf{x})\mathbf{J}(\mathbf{x})^T + \lambda\mathbf{D}^\mathrm{T}\mathbf{D}\right)\mathbf{h} = -\mathbf{J}(\mathbf{x})f(\mathbf{x})^T$$

$$(\mathbf{H}(\mathbf{x}) + \lambda\mathbf{I})\mathbf{h} = -\mathbf{g}(x)$$