The Oxford Lectures

A Short Course in State Estimation

Timothy D. Barfoot

University of Toronto

UNIVERSITY OF TORONTO

Course Outline

Lecture 1: Linear-Gaussian Estimation

Lecture 2: Nonlinearities and Lie Groups

Lecture 3: Continuous-Time Estimation

Lecture 4: Beyond MAP – Variational Inference

Lecture 1: Linear-Gaussian Estimation

A Short Course in State Estimation

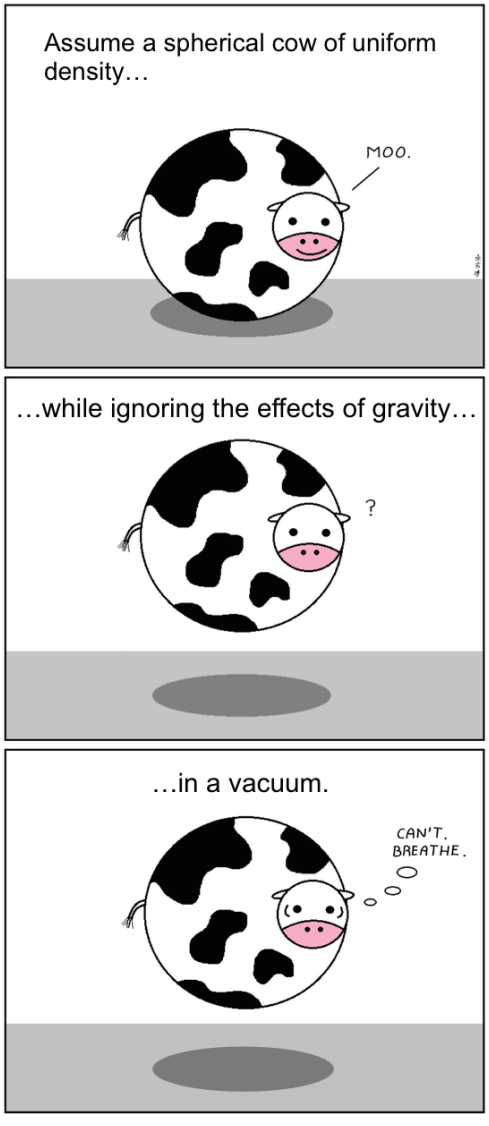
Timothy D. Barfoot

University of Toronto

UNIVERSITY OF TORONTO

Lecture Outline

Why linear-Gaussian?



- most of the classical results are for linear-Gaussian systems
- we can often find results without approximation
- we can get some intuition for real systems by studying them

"Equation (1.2-9) is a second order, nonlinear, vector, differential equation which has defied solution in its present form. It is here therefore we depart from the realities of nature to make some simplifying assumptions..."

– Bate et al., Fundamentals of Astrodynamics (1971)

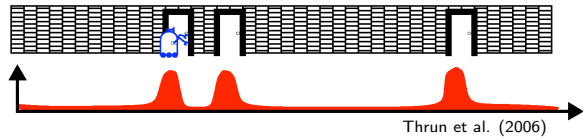
"Linear systems are very important because we can solve them exactly!"

– Sami Mikhail (heard third hand via Raja Mukherji)

UNIVERSITY OF TORONTO

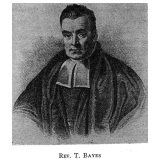
What are the odds? (probability in one slide)

probability density function (PDF) $p(\mathbf{x})$

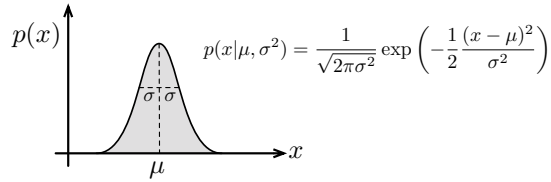


conditional PDF (sensor model) $p(\mathbf{y}|\mathbf{x})$

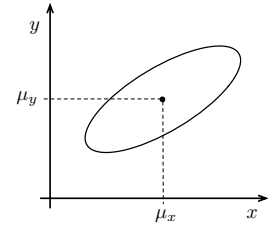
Bayes' rule (inference) $p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}$



Gaussian PDF $p(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{\det(2\pi\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$

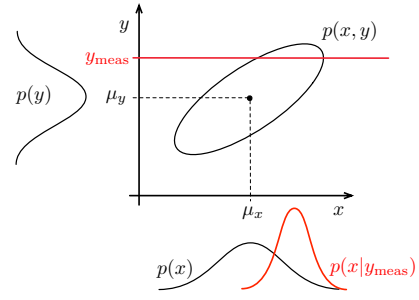


joint Gaussian PDF $p(\mathbf{x}, \mathbf{y}) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_{yy}\end{bmatrix}\right)$



Gaussian marginalization $p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{y})\, d\mathbf{y} = \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_{xx})$

Gaussian conditioning $p(\mathbf{x}|\mathbf{y}) = \mathcal{N}\left(\boldsymbol{\mu}_x + \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}(\mathbf{y}-\boldsymbol{\mu}_y), \boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}\boldsymbol{\Sigma}_{yx}\right)$



UNIVERSITY OF TORONTO

System

– we define our system using the following linear, time-varying models:

$$\text{motion model:} \quad \mathbf{x}_k = \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{v}_k + \mathbf{w}_k, \quad k = 1 \dots K \tag{1a}$$

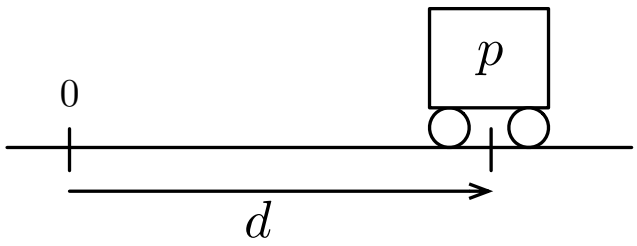
$$\text{observation model:} \quad \mathbf{y}_k = \mathbf{C}_k\mathbf{x}_k + \mathbf{n}_k, \quad k = 0 \dots K \tag{1b}$$

where k is the discrete-time index and K its maximum

– the variables have the following meanings:

$$\begin{aligned}
\text{system state :} \quad & \mathbf{x}_k \in \mathbb{R}^N \\
\text{initial state :} \quad & \mathbf{x}_0 \in \mathbb{R}^N \sim \mathcal{N}\left(\check{\mathbf{x}}_0, \check{\mathbf{P}}_0\right) \\
\text{input :} \quad & \mathbf{v}_k \in \mathbb{R}^N \\
\text{process noise :} \quad & \mathbf{w}_k \in \mathbb{R}^N \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}_k\right) \\
\text{measurement :} \quad & \mathbf{y}_k \in \mathbb{R}^M \\
\text{measurement noise :} \quad & \mathbf{n}_k \in \mathbb{R}^M \sim \mathcal{N}\left(\mathbf{0}, \mathbf{R}_k\right)
\end{aligned}$$

System example



$$\ddot{p}(t) = a(t) + w(t)$$

$$\dot{p}(t) = v(t)$$
$$\dot{v}(t) = a(t) + w(t)$$

$$p_k \approx p_{k-1} + Tv_{k-1} + \frac{1}{2}T^2(a_k + w_k)$$
$$v_k \approx v_{k-1} + T\,(a_k + w_k)$$

$$d_k = p_k + n_k$$

$$\underbrace{d_k}_{\mathbf{y}_k} = \underbrace{\begin{bmatrix}1 & 0\end{bmatrix}}_{\mathbf{C}_k} \underbrace{\begin{bmatrix}p_k \\ v_k\end{bmatrix}}_{\mathbf{x}_k} + \underbrace{n_k}_{\mathbf{n}_k}$$

observation model

$$\underbrace{\begin{bmatrix}p_k \\ v_k\end{bmatrix}}_{\mathbf{x}_k} = \underbrace{\begin{bmatrix}1 & T \\ 0 & 1\end{bmatrix}}_{\mathbf{A}_{k-1}} \underbrace{\begin{bmatrix}p_{k-1} \\ v_{k-1}\end{bmatrix}}_{\mathbf{x}_{k-1}} + \underbrace{\begin{bmatrix}\frac{1}{2}T^2 \\ T\end{bmatrix} a_k}_{\mathbf{v}_k} + \underbrace{\begin{bmatrix}\frac{1}{2}T^2 \\ T\end{bmatrix} w_k}_{\mathbf{w}_k}$$

motion model

What do we know?

- although we want to know the state of the system (at all times), we only have access to the following quantities, and must base our estimate, $\hat{\mathbf{x}}_k$, on just this information:
  - (i) the initial state knowledge, $\check{\mathbf{x}}_0$, and the associated covariance matrix, $\check{\mathbf{P}}_0$; sometimes we do not have this piece of information and must do without
  - (ii) the inputs, $\mathbf{v}_k$, which typically come from the output of our controller and so are known; we also have the associated process noise covariance, $\mathbf{Q}_k$
  - (iii) the measurements, $\mathbf{y}_{k,\text{meas}}$, which are realizations of the associated random variables, $\mathbf{y}_k$, and the associated covariance matrix, $\mathbf{R}_k$
- we will use $\hat{(\cdot)}$ to indicate posterior estimates (incorporating measurements) and $\check{(\cdot)}$ to indicate prior estimates (not incorporating measurements)

Problem statement

– we define the state estimation problem as follows:

Definition

The problem of state estimation is to come up with an estimate, $\hat{\mathbf{x}}_k$, of the true *state* of a system, at one or more timesteps, k, given knowledge of the initial state, $\check{\mathbf{x}}_0$, a sequence of measurements, $\mathbf{y}_{0:K,\text{meas}}$, a sequence of inputs, $\mathbf{v}_{1:K}$, as well as knowledge of the system's motion and observation models.

– our approach will always be to attempt to come up with not only a state estimate, but also to quantify the uncertainty in that estimate

Batch is best

- we will begin by investigating batch linear-Gaussian techniques (sometimes called smoothers)
- the batch solution is very useful for computing state estimates after the fact because it uses all the measurements in the estimation of all the states at once (hence the usage of 'batch')
- a batch method cannot be used online since we cannot employ future measurements to estimate past states
- recursive state estimators (called filters) can be used online

smoothers use all available information to estimate states

$$\overbrace{\underbrace{\check{\mathbf{x}}_0, \mathbf{y}_0, \mathbf{v}_1, \mathbf{y}_1, \mathbf{v}_2, \mathbf{y}_2, \ldots, \mathbf{v}_{k-1}, \mathbf{y}_{k-1}, \mathbf{v}_k, \mathbf{y}_k}_{\hat{\mathbf{x}}_{k,f}}, \mathbf{v}_{k+1}, \mathbf{y}_{k+1}, \ldots, \mathbf{v}_K, \mathbf{y}_K}^{\hat{\mathbf{x}}_k}$$

filters only use past/current information to estimate states

All roads lead to Rome

- we will set up the batch linear-Gaussian estimation problem using two different paradigms:
  - (i) Bayesian inference; here we update a prior density over states (based on the initial state knowledge, inputs, and motion model) with our measurements, to produce a posterior (Gaussian) density over states
  - (ii) maximum a posteriori; here we employ optimization to find the most likely posterior state given the information we have (initial state knowledge, measurements, inputs)
- while these approaches are somewhat different in nature, it turns out that we arrive at the exact same answer for the linear-Gaussian problem
- this is because the full Bayesian posterior is exactly Gaussian and so the optimization approach will find the maximum (i.e., mode) of a Gaussian, and this is the same as the mean

A marriage of convenience

– we will combine the initial state knowledge and inputs in the following way:

$$\mathbf{v} = (\check{\mathbf{x}}_0, \mathbf{v}_{1:K}) = (\check{\mathbf{x}}_0, \mathbf{v}_1, \ldots, \mathbf{v}_K) \tag{2}$$

since they are both related to our prior knowledge of the state

– we will also define

$$\mathbf{y} = \mathbf{y}_{0:K} = (\mathbf{y}_0, \ldots, \mathbf{y}_K) \tag{3}$$

to denote all of our measurements and

$$\mathbf{x} = \mathbf{x}_{0:K} = (\mathbf{x}_0, \ldots, \mathbf{x}_K) \tag{4}$$

for our entire state (all timesteps)

Approach (i): Bayesian inference

– we would like to compute the full Bayesian posterior:

$$p(\mathbf{x}|\mathbf{y},\mathbf{v}) = \frac{p(\mathbf{y}|\mathbf{x},\mathbf{v})p(\mathbf{x}|\mathbf{v})}{p(\mathbf{y}|\mathbf{v})} \tag{5}$$

– we will take the two-step approach of building a joint density over states, inputs, measurements,

$$p(\mathbf{x},\mathbf{y}|\mathbf{v}) = \underbrace{p(\mathbf{y}|\mathbf{x},\mathbf{v})}_{\text{observations}}\underbrace{p(\mathbf{x}|\mathbf{v})}_{\text{prior}} \tag{6}$$

and then factor it the other way, keeping only the bit we want:

$$p(\mathbf{x},\mathbf{y}|\mathbf{v}) = \underbrace{p(\mathbf{x}|\mathbf{y},\mathbf{v})}_{\text{posterior}}\underbrace{p(\mathbf{y}|\mathbf{v})}_{\text{discard}} \tag{7}$$

Prior

– recall our motion model,

$$\mathbf{x}_k = \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{v}_k + \mathbf{w}_k \tag{8}$$

– in lifted matrix form (i.e., whole trajectory), we can write this as

$$\mathbf{x} = \mathbf{A}(\mathbf{v} + \mathbf{w}) \tag{9}$$

where $\mathbf{w}$ is the lifted form of the initial state and process noise, and

$$\mathbf{A} = \begin{bmatrix} \mathbf{1} & & & & \\ \mathbf{A}_0 & \mathbf{1} & & & \\ \mathbf{A}_1\mathbf{A}_0 & \mathbf{A}_1 & \mathbf{1} & & \\ \vdots & \vdots & \vdots & \ddots & \\ \mathbf{A}_{K-2}\cdots\mathbf{A}_0 & \mathbf{A}_{K-2}\cdots\mathbf{A}_1 & \mathbf{A}_{K-2}\cdots\mathbf{A}_2 & \cdots & \mathbf{1} \\ \mathbf{A}_{K-1}\cdots\mathbf{A}_0 & \mathbf{A}_{K-1}\cdots\mathbf{A}_1 & \mathbf{A}_{K-1}\cdots\mathbf{A}_2 & \cdots & \mathbf{A}_{K-1} \quad \mathbf{1} \end{bmatrix} \tag{10}$$

is the lifted transition matrix, which we see is lower-triangular

Prior

- the lifted mean is then

$$\check{\mathbf{x}} = E[\mathbf{x}] = E[\mathbf{A}(\mathbf{v} + \mathbf{w})] = \mathbf{A}\mathbf{v} \tag{11}$$

- the lifted covariance is

$$\check{\mathbf{P}} = E\left[(\mathbf{x} - E[\mathbf{x}])(\mathbf{x} - E[\mathbf{x}])^T\right] = \mathbf{A}\mathbf{Q}\mathbf{A}^T \tag{12}$$

  where $\mathbf{Q} = E[\mathbf{w}\mathbf{w}^T] = \text{diag}(\check{\mathbf{P}}_0, \mathbf{Q}_1, \ldots, \mathbf{Q}_K)$

- our prior can then be neatly expressed as

$$p(\mathbf{x}|\mathbf{v}) = \mathcal{N}\left(\check{\mathbf{x}}, \check{\mathbf{P}}\right) = \mathcal{N}\left(\mathbf{A}\mathbf{v}, \mathbf{A}\mathbf{Q}\mathbf{A}^T\right) \tag{13}$$

Measurements

- we now want to compute $p(\mathbf{x}, \mathbf{y}|\mathbf{v}) = p(\mathbf{y}|\mathbf{x}, \mathbf{v})p(\mathbf{x}|\mathbf{v})$
- we will use the measurement model

$$\mathbf{y}_k = \mathbf{C}_k \mathbf{x}_k + \mathbf{n}_k \tag{14}$$

- this can also be written in lifted form as

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{n} \tag{15}$$

where $\mathbf{n}$ is the lifted form of the measurement noise and

$$\mathbf{C} = \text{diag}\left(\mathbf{C}_0, \mathbf{C}_1, \ldots, \mathbf{C}_K\right) \tag{16}$$

is the lifted observation matrix

Joint Gaussian

- the joint likelihood of the prior lifted state and the measurements can now be written as

$$p(\mathbf{x}, \mathbf{y}|\mathbf{v}) = \mathcal{N}\left(\begin{bmatrix} \check{\mathbf{x}} \\ \mathbf{C}\check{\mathbf{x}} \end{bmatrix}, \begin{bmatrix} \check{\mathbf{P}} & \check{\mathbf{P}}\mathbf{C}^T \\ \mathbf{C}\check{\mathbf{P}} & \mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R} \end{bmatrix}\right) \tag{17}$$

  where $\mathbf{R} = E[\mathbf{n}\mathbf{n}^T] = \text{diag}(\mathbf{R}_0, \mathbf{R}_1, \ldots, \mathbf{R}_K)$

- we can factor this according to

$$p(\mathbf{x}, \mathbf{y}|\mathbf{v}) = p(\mathbf{x}|\mathbf{y}, \mathbf{v})p(\mathbf{y}|\mathbf{v}) \tag{18}$$

- we only care about the first factor, which is the full Bayesian posterior

Posterior

– this can be written, using the Gaussian conditioning formula, as

$$p(\mathbf{x}|\mathbf{v},\mathbf{y}) = \mathcal{N}\Big(\check{\mathbf{x}} + \check{\mathbf{P}}\mathbf{C}^T(\mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R})^{-1}(\mathbf{y} - \mathbf{C}\check{\mathbf{x}}), \\ \check{\mathbf{P}} - \check{\mathbf{P}}\mathbf{C}^T(\mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R})^{-1}\mathbf{C}\check{\mathbf{P}}\Big) \tag{19}$$

– using the Sherman-Morrison-Woodbury identity, this can be manipulated into the following form:

$$p(\mathbf{x}|\mathbf{v},\mathbf{y}) = \mathcal{N}\Big(\underbrace{\left(\check{\mathbf{P}}^{-1} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C}\right)^{-1}\left(\check{\mathbf{P}}^{-1}\check{\mathbf{x}} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{y}\right)}_{\hat{\mathbf{x}},\ \text{mean}}, \\ \underbrace{\left(\check{\mathbf{P}}^{-1} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C}\right)^{-1}}_{\hat{\mathbf{P}},\ \text{covariance}}\Big) \tag{20}$$

– we could use this as is, but there is a better way

Posterior

- we can rearrange the mean expression to arrive at

$$\underbrace{\left(\check{\mathbf{P}}^{-1} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C}\right)}_{\hat{\mathbf{P}}^{-1}} \hat{\mathbf{x}} = \check{\mathbf{P}}^{-1}\check{\mathbf{x}} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{y} \tag{21}$$

  and we see the inverse covariance appearing on the left-hand side

- substituting in $\check{\mathbf{x}} = \mathbf{A}\mathbf{v}$ and $\check{\mathbf{P}}^{-1} = \left(\mathbf{A}\mathbf{Q}\mathbf{A}^T\right)^{-1} = \mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{A}^{-1}$ we can rewrite this as

$$\underbrace{\left(\mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{A}^{-1} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C}\right)}_{\hat{\mathbf{P}}^{-1}} \hat{\mathbf{x}} = \mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{v} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{y} \tag{22}$$

- or more compactly,

$$\left(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}\right) \hat{\mathbf{x}} = \mathbf{H}^T\mathbf{W}^{-1}\mathbf{z} \tag{23}$$

$$\mathbf{z} = \begin{bmatrix}\mathbf{v}\\ \mathbf{y}\end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix}\mathbf{A}^{-1}\\ \mathbf{C}\end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix}\mathbf{Q} & \\ & \mathbf{R}\end{bmatrix} \tag{24}$$

Approach (ii): maximum a posteriori (MAP)

- instead of computing the full Bayesian posterior, what if we just tried to find the most likely state of our system given the initial state knowledge, inputs, and measurements?
- to accomplish this, we could try to solve the following optimization problem:

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{y}, \mathbf{v}) \tag{25}$$

  which is to say that we want to find the best single estimate for the state of the system (at all timesteps), $\hat{\mathbf{x}}$, given the prior information, $\mathbf{v}$, and measurements, $\mathbf{y}$

MAP

– we begin by rewriting the MAP estimate using Bayes' rule:

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{y}, \mathbf{v}) = \arg\max_{\mathbf{x}} \frac{p(\mathbf{y}|\mathbf{x}, \mathbf{v})p(\mathbf{x}|\mathbf{v})}{p(\mathbf{y}|\mathbf{v})}$$
$$= \arg\max_{\mathbf{x}} p(\mathbf{y}|\mathbf{x})p(\mathbf{x}|\mathbf{v}) \quad (26)$$

where we drop the denominator because it does not depend on $\mathbf{x}$

– we also drop $\mathbf{v}$ in $p(\mathbf{y}|\mathbf{x}, \mathbf{v})$ since it does not affect $\mathbf{y}$ in our system if $\mathbf{x}$ is known (see observation model)

MAP

– if we assume that all of the noise variables, $\mathbf{w}_k$ and $\mathbf{n}_k$ for $k = 0 \ldots K$, are uncorrelated, we can use Bayes' rule to factor $p(\mathbf{y}|\mathbf{x})$ in the following way:

$$p(\mathbf{y}|\mathbf{x}) = \prod_{k=0}^{K} p(\mathbf{y}_k \,|\, \mathbf{x}_k) \tag{27}$$

– furthermore, Bayes' rule allows us to factor $p(\mathbf{x}|\mathbf{v})$ as

$$p(\mathbf{x}|\mathbf{v}) = p(\mathbf{x}_0 \,|\, \check{\mathbf{x}}_0) \prod_{k=1}^{K} p(\mathbf{x}_k \,|\, \mathbf{x}_{k-1}, \mathbf{v}_k) \tag{28}$$

Negative log likelihood

- to make the optimization easier, we insert the (negative) logarithm function, which is monotonically decreasing

$$\begin{aligned}\hat{\mathbf{x}} &= \arg\min_{\mathbf{x}} \left(-\ln(p(\mathbf{y}|\mathbf{x})p(\mathbf{x}\,|\mathbf{v}))\right) \\ &= \arg\min_{\mathbf{x}} \Bigg(-\ln p(\mathbf{x}_0 \,|\, \check{\mathbf{x}}_0) - \sum_{k=1}^{K} \ln p(\mathbf{x}_k \,|\, \mathbf{x}_{k-1}, \mathbf{v}_k) \\ &\qquad\qquad - \sum_{k=0}^{K} \ln p(\mathbf{y}_k \,|\, \mathbf{x}_k) \Bigg) \end{aligned} \tag{29}$$

- note, we are now minimizing due to the negative sign, which is more common in optimization

MAP

– the component Gaussian densities are given by

$$p(\mathbf{x}_0|\check{\mathbf{x}}_0) = \frac{1}{\sqrt{(2\pi)^N \det \check{\mathbf{P}}_0}} \exp\left(-\frac{1}{2}(\mathbf{x}_0 - \check{\mathbf{x}}_0)^T \check{\mathbf{P}}_0^{-1}(\mathbf{x}_0 - \check{\mathbf{x}}_0)\right) \tag{30a}$$

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{v}_k) = \frac{1}{\sqrt{(2\pi)^N \det \mathbf{Q}_k}} \exp\left(-\frac{1}{2}(\mathbf{x}_k - \mathbf{A}_{k-1}\mathbf{x}_{k-1} - \mathbf{v}_k)^T \times \mathbf{Q}_k^{-1}(\mathbf{x}_k - \mathbf{A}_{k-1}\mathbf{x}_{k-1} - \mathbf{v}_k)\right) \tag{30b}$$

$$p(\mathbf{y}_k|\mathbf{x}_k) = \frac{1}{\sqrt{(2\pi)^M \det \mathbf{R}_k}} \exp\left(-\frac{1}{2}(\mathbf{y}_k - \mathbf{C}_k\mathbf{x}_k)^T \mathbf{R}_k^{-1}(\mathbf{y}_k - \mathbf{C}_k\mathbf{x}_k)\right) \tag{30c}$$

MAP

– the component negative-log densities are given by

$$-\ln p(\mathbf{x}_0|\check{\mathbf{x}}_0) = \frac{1}{2}(\mathbf{x}_0 - \check{\mathbf{x}}_0)^T \check{\mathbf{P}}_0^{-1}(\mathbf{x}_0 - \check{\mathbf{x}}_0) + \underbrace{\frac{1}{2}\ln\left((2\pi)^N \det \check{\mathbf{P}}_0\right)}_{\text{independent of } \mathbf{x}} \tag{31a}$$

$$-\ln p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{v}_k) = \frac{1}{2}(\mathbf{x}_k - \mathbf{A}_{k-1}\mathbf{x}_{k-1} - \mathbf{v}_k)^T \mathbf{Q}_k^{-1}(\mathbf{x}_k - \mathbf{A}_{k-1}\mathbf{x}_{k-1} - \mathbf{v}_k) + \underbrace{\frac{1}{2}\ln\left((2\pi)^N \det \mathbf{Q}_k\right)}_{\text{independent of } \mathbf{x}} \tag{31b}$$

$$-\ln p(\mathbf{y}_k|\mathbf{x}_k) = \frac{1}{2}(\mathbf{y}_k - \mathbf{C}_k\mathbf{x}_k)^T \mathbf{R}_k^{-1}(\mathbf{y}_k - \mathbf{C}_k\mathbf{x}_k) + \underbrace{\frac{1}{2}\ln\left((2\pi)^M \det \mathbf{R}_k\right)}_{\text{independent of } \mathbf{x}} \tag{31c}$$

MAP as optimization

– from an optimization perspective, we want to solve the following problem:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} J(\mathbf{x}) \tag{32}$$

– our cost function is

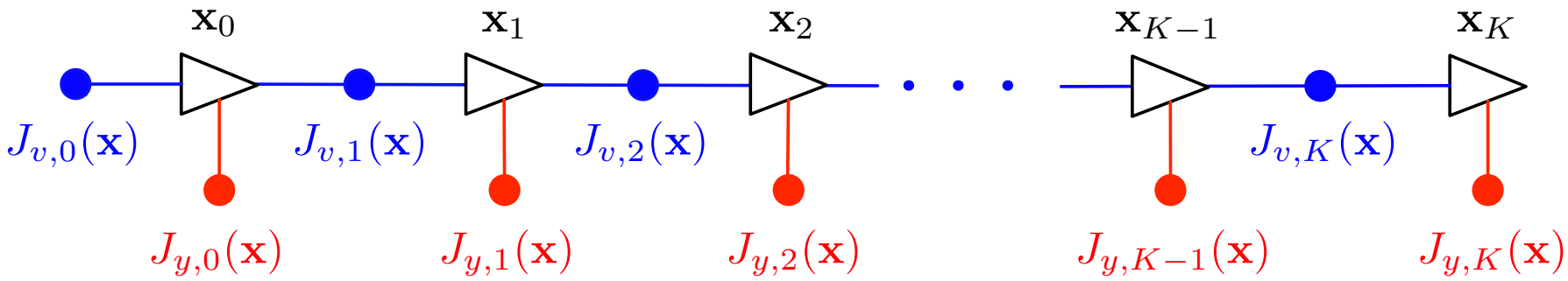
$$J(\mathbf{x}) = \sum_{k=0}^{K} \left(J_{v,k}(\mathbf{x}) + J_{y,k}(\mathbf{x}) \right) \tag{33}$$

with

$$J_{v,k}(\mathbf{x}) = \begin{cases} \frac{1}{2} \left(\mathbf{x}_0 - \check{\mathbf{x}}_0 \right)^T \check{\mathbf{P}}_0^{-1} \left(\mathbf{x}_0 - \check{\mathbf{x}}_0 \right) & k = 0 \\ \frac{1}{2} \left(\mathbf{x}_k - \mathbf{A}_{k-1} \mathbf{x}_{k-1} - \mathbf{v}_k \right)^T & \\ \qquad \times \mathbf{Q}_k^{-1} \left(\mathbf{x}_k - \mathbf{A}_{k-1} \mathbf{x}_{k-1} - \mathbf{v}_k \right) & k = 1 \ldots K \end{cases} \tag{34a}$$

$$J_{y,k}(\mathbf{x}) = \frac{1}{2} \left(\mathbf{y}_k - \mathbf{C}_k \mathbf{x}_k \right)^T \mathbf{R}_k^{-1} \left(\mathbf{y}_k - \mathbf{C}_k \mathbf{x}_k \right) \qquad k = 0 \ldots K \tag{34b}$$

MAP as a factor graph



$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} J(\mathbf{x}) = \arg\min_{\mathbf{x}} \sum_{k=0}^{K} \Big(J_{v,k}(\mathbf{x}) + J_{y,k}(\mathbf{x}) \Big)$$

motion prior measurements

MAP as optimization

– we can express our cost function more compactly as

$$J(\mathbf{x}) = \frac{1}{2}(\mathbf{z} - \mathbf{H}\mathbf{x})^T \mathbf{W}^{-1}(\mathbf{z} - \mathbf{H}\mathbf{x}) \tag{35}$$

recalling our definitions from the full Bayesian approach:

$$\mathbf{z} = \begin{bmatrix} \mathbf{v} \\ \mathbf{y} \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} \mathbf{A}^{-1} \\ \mathbf{C} \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} \mathbf{Q} & \\ & \mathbf{R} \end{bmatrix} \tag{36}$$

– we see that $J(\mathbf{x})$ is exactly quadratic in $\mathbf{x}$

MAP as optimization

- since $J(\mathbf{x})$ is exactly a paraboloid, we can find its minimum in closed form
- simply set the partial derivative with respect to the design variable, $\mathbf{x}$, to zero:

$$\left.\frac{\partial J(\mathbf{x})}{\partial \mathbf{x}^T}\right|_{\hat{\mathbf{x}}} = -\mathbf{H}^T\mathbf{W}^{-1}\left(\mathbf{z} - \mathbf{H}\hat{\mathbf{x}}\right) = \mathbf{0} \tag{37a}$$

- rearranging we have

$$\left(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}\right)\hat{\mathbf{x}} = \mathbf{H}^T\mathbf{W}^{-1}\mathbf{z} \tag{37b}$$

which is the identical linear system we arrived at in the full Bayesian approach

What gives?

- why do the full Bayesian and MAP approaches result in the same linear system if they are trying to do different things?

$$\left(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}\right)\hat{\mathbf{x}} = \mathbf{H}^T\mathbf{W}^{-1}\mathbf{z} \tag{38}$$

- the explanation lies in the fact that the full Bayesian posterior is exactly Gaussian and the mean and mode (i.e., maximum) of a Gaussian are one and the same
- for nonlinear systems, these two approaches will diverge, because the posterior will not be exactly Gaussian

Sparsity

- both our approaches, full Bayesian and MAP, require us to compute $\mathbf{A}^{-1}$
- it turns out this has a beautifully simple form:

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{1} & & & & & \\ -\mathbf{A}_0 & \mathbf{1} & & & & \\ & -\mathbf{A}_1 & \mathbf{1} & & & \\ & & -\mathbf{A}_2 & \ddots & & \\ & & & \ddots & \mathbf{1} & \\ & & & & -\mathbf{A}_{K-1} & \mathbf{1} \end{bmatrix} \tag{39}$$

- the sparsity comes from the fact that the system model obeys the Markov property

Sparsity

– when we plug $\mathbf{A}^{-1}$ into the linear system (via $\mathbf{H}$)

$$\left(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}\right)\hat{\mathbf{x}} = \mathbf{H}^T\mathbf{W}^{-1}\mathbf{z} \tag{40}$$

the left-hand side looks like this:

$$\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H} = \begin{bmatrix} * & * & & & & & \\ * & * & * & & & & \\ & * & * & * & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & * & * & * \\ & & & & * & * \end{bmatrix} \tag{41}$$

where $*$ indicates a non-zero block

– the fact that the LHS is exactly block-tridiagonal means we can solve for $\hat{\mathbf{x}}$ in $O(K)$ time instead of the usual $O(K^3)$ time for solving linear systems

Sparsity

– it turns out we can efficiently factor $\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}$ using a sparse Cholesky factorization into

$$\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H} = \mathbf{L}\mathbf{L}^T \tag{42}$$

where $\mathbf{L}$ is a block-lower-triangular matrix called the Cholesky factor:

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_0 & & & & & \\ \mathbf{L}_{10} & \mathbf{L}_1 & & & & \\ & \mathbf{L}_{21} & \mathbf{L}_2 & & & \\ & & \ddots & \ddots & & \\ & & & \mathbf{L}_{K-1,K-2} & \mathbf{L}_{K-1} & \\ & & & & \mathbf{L}_{K,K-1} & \mathbf{L}_K \end{bmatrix} \tag{43}$$

– the decomposition can be computed in $O(N^3(K+1))$ time

Cholesky solution

- once we have the Cholesky factorization, the solution of our linear system is straightforward
- forward pass: solve

$$\mathbf{L}\mathbf{d} = \mathbf{H}^T\mathbf{W}^{-1}\mathbf{z} \tag{44}$$

  for $\mathbf{d}$
- backward pass: solve

$$\mathbf{L}^T\hat{\mathbf{x}} = \mathbf{d} \tag{45}$$

  for $\hat{\mathbf{x}}$
- both passes can be done in $O(N^3(K+1))$ time through forward/backward substitution owing to the sparse lower-triangular form of $\mathbf{L}$
- thus, the batch equations can be solved in computation time that scales linearly with the size of the state

Cholesky solution

- at the block level, the forward pass, $k = 1 \dots K$, is

$$\begin{aligned}
\mathbf{L}_{k-1}\mathbf{L}_{k-1}^T &= \mathbf{I}_{k-1} + \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \mathbf{A}_{k-1} && (46a)\\
\mathbf{L}_{k-1}\mathbf{d}_{k-1} &= \mathbf{q}_{k-1} - \mathbf{A}_{k-1}^T \mathbf{Q}_k^{-1} \mathbf{v}_k && (46b)\\
\mathbf{L}_{k,k-1}\mathbf{L}_{k-1}^T &= -\mathbf{Q}_k^{-1} \mathbf{A}_{k-1} && (46c)\\
\mathbf{I}_k &= -\mathbf{L}_{k,k-1}\mathbf{L}_{k,k-1}^T + \mathbf{Q}_k^{-1} + \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{C}_k && (46d)\\
\mathbf{q}_k &= -\mathbf{L}_{k,k-1}\mathbf{d}_{k-1} + \mathbf{Q}_k^{-1} \mathbf{v}_k + \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{y}_k && (46e)
\end{aligned}$$

- the backward pass, $k = K \dots 1$, is

$$\mathbf{L}_{k-1}^T \hat{\mathbf{x}}_{k-1} = -\mathbf{L}_{k,k-1}^T \hat{\mathbf{x}}_k + \mathbf{d}_{k-1} \quad (47)$$

- these are initialized with

$$\begin{aligned}
\mathbf{I}_0 &= \check{\mathbf{P}}_0^{-1} + \mathbf{C}_0^T \mathbf{R}_0^{-1} \mathbf{C}_0 && (48a)\\
\mathbf{q}_0 &= \check{\mathbf{P}}_0^{-1} \check{\mathbf{x}}_0 + \mathbf{C}_0^T \mathbf{R}_0^{-1} \mathbf{y}_0 && (48b)\\
\hat{\mathbf{x}}_K &= \mathbf{L}_K^{-T} \mathbf{d}_K && (48c)
\end{aligned}$$

Cholesky solution $\mapsto$ Rauch-Tung-Striebel smoother

- our Cholesky approach will certainly work, and is appealing because it follows easily from the batch setup and the block-tridiagonal sparsity of the LHS
- historically, however, the Rauch-Tung-Striebel (RTS) smoother equations constitute the canonical form
- the forwards pass of the RTS smoother is called the Kalman filter
- the Cholesky and RTS equations are algebraically equivalent, but there is quite a lot of algebra needed to show this (see the book)

RTS smoother

- forward pass (a.k.a., Kalman filter), $k = 1 \dots K$:

$$\check{\mathbf{P}}_{k,f} = \mathbf{A}_{k-1}\hat{\mathbf{P}}_{k-1,f}\mathbf{A}_{k-1}^T + \mathbf{Q}_k \tag{49a}$$

$$\check{\mathbf{x}}_{k,f} = \mathbf{A}_{k-1}\hat{\mathbf{x}}_{k-1,f} + \mathbf{v}_k \tag{49b}$$

$$\mathbf{K}_k = \check{\mathbf{P}}_{k,f}\mathbf{C}_k^T\left(\mathbf{C}_k\check{\mathbf{P}}_{k,f}\mathbf{C}_k^T + \mathbf{R}_k\right)^{-1} \tag{49c}$$

$$\hat{\mathbf{P}}_{k,f} = \left(\mathbf{1} - \mathbf{K}_k\mathbf{C}_k\right)\check{\mathbf{P}}_{k,f} \tag{49d}$$

$$\hat{\mathbf{x}}_{k,f} = \check{\mathbf{x}}_{k,f} + \mathbf{K}_k\left(\mathbf{y}_k - \mathbf{C}_k\check{\mathbf{x}}_{k,f}\right) \tag{49e}$$

- backward pass, $k = K \dots 1$:

$$\hat{\mathbf{x}}_{k-1} = \hat{\mathbf{x}}_{k-1,f} + \left(\hat{\mathbf{P}}_{k-1,f}\mathbf{A}_{k-1}^T\check{\mathbf{P}}_{k,f}^{-1}\right)\left(\hat{\mathbf{x}}_k - \check{\mathbf{x}}_{k,f}\right) \tag{50a}$$

$$\hat{\mathbf{P}}_{k-1} = \hat{\mathbf{P}}_{k-1,f} + \left(\hat{\mathbf{P}}_{k-1,f}\mathbf{A}_{k-1}^T\check{\mathbf{P}}_{k,f}^{-1}\right)\left(\hat{\mathbf{P}}_k - \check{\mathbf{P}}_{k,f}\right) \times \left(\hat{\mathbf{P}}_{k-1,f}\mathbf{A}_{k-1}^T\check{\mathbf{P}}_{k,f}^{-1}\right)^T \tag{50b}$$

- these are initialized with

$$\hat{\mathbf{P}}_{0,f} = (\mathbf{1} - \mathbf{K}_0\mathbf{C}_0)\check{\mathbf{P}}_0, \quad \hat{\mathbf{x}}_{0,f} = \check{\mathbf{x}}_0 + \mathbf{K}_0(\mathbf{y}_0 - \mathbf{C}_0\check{\mathbf{x}}_0) \tag{51a}$$

$$\hat{\mathbf{P}}_K = \hat{\mathbf{P}}_{K,f}, \quad \hat{\mathbf{x}}_K = \hat{\mathbf{x}}_{K,f} \tag{51b}$$

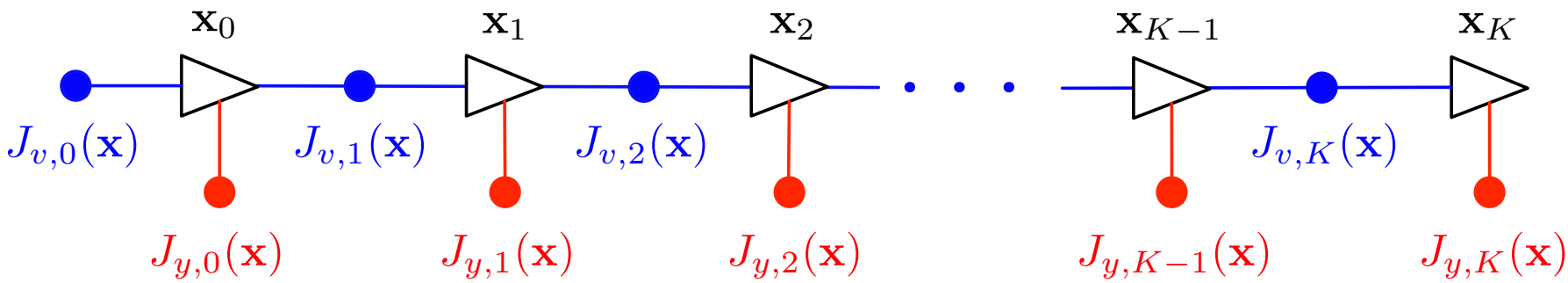
Take-home message

The full Bayesian and MAP approaches result in the same system of linear equations because the Bayesian posterior is exactly Gaussian and the mean and mode (i.e., maximum) of a Gaussian are one and the same.

$$\left(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}\right)\hat{\mathbf{x}} = \mathbf{H}^T\mathbf{W}^{-1}\mathbf{z} \tag{52}$$

Either the Cholesky or RTS smoother equations can be used to solve this system of equations exactly and efficiently. These are also equivalent to Gaussian belief propagation.

Take-home message



$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} J(\mathbf{x}) = \arg\min_{\mathbf{x}} \sum_{k=0}^{K} \Big(J_{v,k}(\mathbf{x}) + J_{y,k}(\mathbf{x}) \Big)$$

motion prior measurements

$$\underbrace{\left(\mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{A}^{-1} + \mathbf{C}^{T}\mathbf{R}^{-1}\mathbf{C}\right)}_{\hat{\mathbf{P}}^{-1}} \hat{\mathbf{x}} = \mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{v} + \mathbf{C}^{T}\mathbf{R}^{-1}\mathbf{y}$$

block-tridiagonal

Existence and uniqueness

- how do we know there will be a solution to our batch equations?

$$\left(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}\right)\hat{\mathbf{x}} = \mathbf{H}^T\mathbf{W}^{-1}\mathbf{z} \tag{53}$$

- in general, linear systems have 0, 1, or infinitely many solutions
- for a unique solution the LHS might be invertible or equivalently

$$\text{rank}\left(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}\right) = N(K+1) \tag{54}$$

  because we have dim $\mathbf{x} = N(K+1)$
- we can drop $\mathbf{W}^{-1}$ as long as it's invertible – it is, it's positive definite by assumption – the test becomes

$$\text{rank}\left(\mathbf{H}^T\right) = N(K+1) \tag{55}$$

Case (i): we have knowledge of the initial state

– we see that

$$\text{rank } \mathbf{H}^T = \text{rank} \left[\begin{array}{ccccc|ccccc} \mathbf{1} & -\mathbf{A}_0^T & & & & \mathbf{C}_0^T & & & & \\ & \mathbf{1} & -\mathbf{A}_1^T & & & & \mathbf{C}_1^T & & & \\ & & \mathbf{1} & \ddots & & & & \mathbf{C}_2^T & & \\ & & & \ddots & -\mathbf{A}_{K-1}^T & & & & \ddots & \\ & & & & \mathbf{1} & & & & & \mathbf{C}_K^T \end{array}\right] \tag{56}$$

– is exactly in block-row-echelon form, meaning there are $N(K+1)$ 'leading ones'

– this means all the rows are linearly independent so the matrix is full rank and our solution is unique

– intuitively, the prior already provides a complete solution and the measurements nudge it

Case (ii): no knowledge of the initial state

- it might be that we have no idea about the initial state of the system
- this removes the first column in $\mathbf{H}^T$ and our condition becomes

$$\text{rank } \mathbf{H}^T = \text{rank} \left[\begin{array}{cccc|ccccc} -\mathbf{A}_0^T & & & & \mathbf{C}_0^T & & & & \\ \mathbf{1} & -\mathbf{A}_1^T & & & & \mathbf{C}_1^T & & & \\ & \mathbf{1} & \ddots & & & & \mathbf{C}_2^T & & \\ & & \ddots & -\mathbf{A}_{K-1}^T & & & & \ddots & \\ & & & \mathbf{1} & & & & & \mathbf{C}_K^T \end{array}\right] \tag{57}$$

- it's no longer obvious that the solution is unique

Case (ii): no knowledge of the initial state

– moving the top block-row to the bottom does not change the rank

$$\begin{aligned}&\text{rank } \mathbf{H}^T \\ &= \text{rank } \left[\begin{array}{cccc|ccc} \mathbf{1} & -\mathbf{A}_1^T & & & \mathbf{C}_1^T & & \\ & \mathbf{1} & \ddots & & & \mathbf{C}_2^T & \\ & & \ddots & -\mathbf{A}_{K-1}^T & & & \ddots \\ & & & \mathbf{1} & & & \mathbf{C}_K^T \\ \hline -\mathbf{A}_0^T & & & & \mathbf{C}_0^T & & \end{array}\right]\end{aligned} \tag{58}$$

– again without changing the rank, we can add to the bottom block-row, $\mathbf{A}_0^T$ times the first block-row, $\mathbf{A}_0^T \mathbf{A}_1^T$ times the second block-row, ..., and $\mathbf{A}_0^T \cdots \mathbf{A}_{K-1}^T$ times the Kth block-row

Case (ii): no knowledge of the initial state

– the result of this last step is

$$\begin{aligned}&\text{rank } \mathbf{H}^T \\ &= \text{rank } \left[\begin{array}{cccc|cccc} \mathbf{1} & -\mathbf{A}_1^T & & & \mathbf{C}_1^T & & & \\ & \mathbf{1} & \ddots & & & \mathbf{C}_1^T & & \\ & & \ddots & -\mathbf{A}_{K-1}^T & & & \ddots & \\ & & & \mathbf{1} & & & & \mathbf{C}_K^T \\ \hline & & & & \mathbf{C}_0^T & \mathbf{A}_0^T\mathbf{C}_1^T & \mathbf{A}_0^T\mathbf{A}_1^T\mathbf{C}_2^T & \cdots & \mathbf{A}_0^T\cdots\mathbf{A}_{K-1}^T\mathbf{C}_K^T \end{array}\right]\end{aligned} \tag{59}$$

– the upper-left block is full rank, NK, since every row has a 'leading one'

– all that remains is to determine if the bottom-right block is full rank:

$$\text{rank } \left[\mathbf{C}_0^T \quad \mathbf{A}_0^T\mathbf{C}_1^T \quad \mathbf{A}_0^T\mathbf{A}_1^T\mathbf{C}_2^T \quad \cdots \quad \mathbf{A}_0^T\cdots\mathbf{A}_{K-1}^T\mathbf{C}_K^T\right] = N \tag{60}$$

Case (ii): no knowledge of the initial state – time-invariant

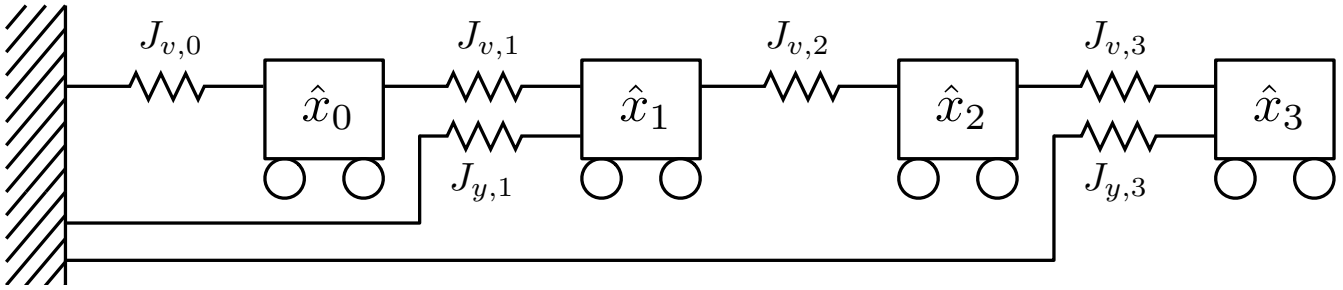
- if we further assume the system is time-invariant such that for all k we have $\mathbf{A}_k = \boldsymbol{A}$ and $\mathbf{C}_k = \boldsymbol{C}$ and we make the not-too-restrictive assumption that $K \gg N$, we may further simplify this condition
- the Cayley-Hamilton theoreom says square matrices satisfy their own characteristic equation, so powers of $\mathbf{A}$ greater than or equal to N can be written as linear combinations of the lower powers
- this let's us keep only the first N block-rows, reducing the test to

$$\text{rank} \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \vdots \\ \boldsymbol{CA}^{(N-1)} \end{bmatrix} = N \tag{61}$$

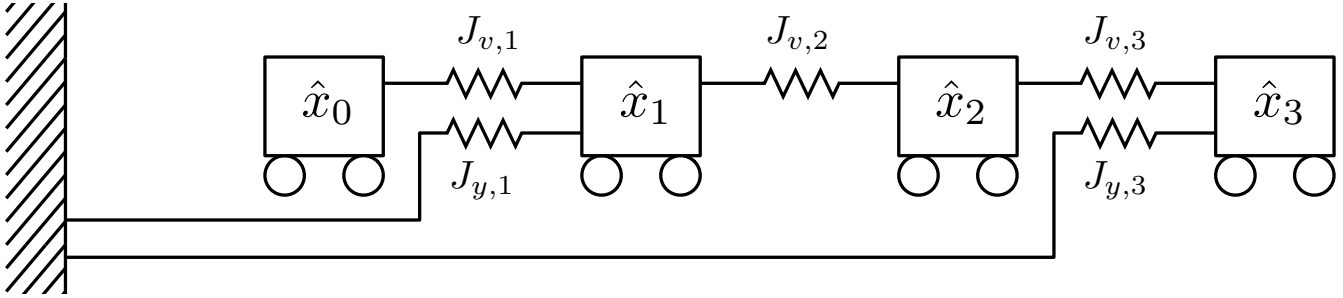
- this is precisely the test for observability – that we can reconstruct the initial state from a finite number of measurements

Mass-spring analogy

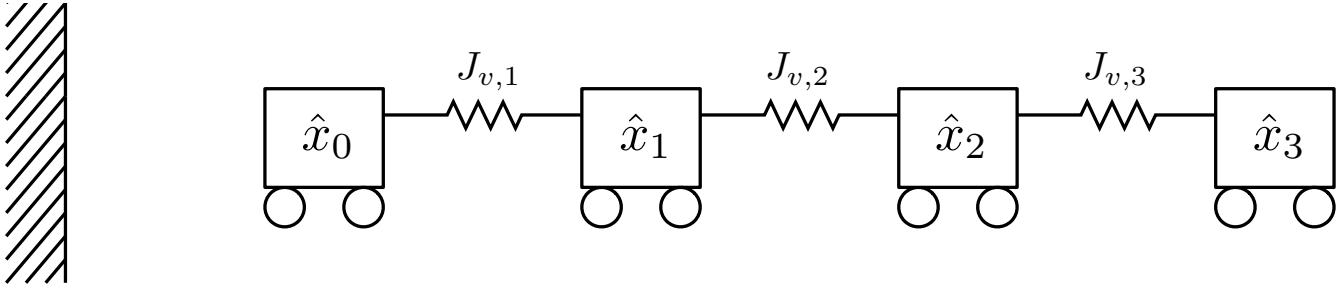
initial state knowledge plus inputs ensures observability



no initial state knowledge can be observable



no initial state knowledge can also be unobservable (in 1D this only happens with no measurements)



Knowing when we don't know

- we said at the beginning of the lecture that we'd like to bookkeep all of the uncertainties in our estimate, but the linear system just tells us the mean, right?
- wrong, we can interpret it in the following way:

$$\underbrace{\left(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}\right)}_{\substack{\text{inverse}\\\text{covariance}}}\underbrace{\hat{\mathbf{x}}}_{\text{mean}} = \underbrace{\mathbf{H}^T\mathbf{W}^{-1}\mathbf{z}}_{\substack{\text{information}\\\text{vector}}} \tag{62}$$

- so the covariance of our estimate is

$$\hat{\mathbf{P}} = \left(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}\right)^{-1} \tag{63}$$

- our (Bayesian) estimate is

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}\left(\hat{\mathbf{x}}, \hat{\mathbf{P}}\right) \tag{64}$$

Knowing how to know when we don't know

– to see this, we can directly take the expectation of the estimate:

$$\mathbf{x} - \underbrace{\left(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{W}^{-1}\mathbf{z}}_{E[\mathbf{x}]} = \left(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{W}^{-1}\underbrace{\left(\mathbf{H}\mathbf{x} - \mathbf{z}\right)}_{\mathbf{s}} \tag{65}$$

where $\mathbf{s} = \begin{bmatrix}\mathbf{w}\\ \mathbf{n}\end{bmatrix}$

– in this case we have

$$\begin{aligned}
\hat{\mathbf{P}} &= E\left[\left(\mathbf{x} - E[\mathbf{x}]\right)\left(\mathbf{x} - E[\mathbf{x}]\right)^T\right] && \text{(66a)}\\
&= \left(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{W}^{-1}\underbrace{E\left[\mathbf{s}\,\mathbf{s}^T\right]}_{\mathbf{W}}\mathbf{W}^{-1}\mathbf{H}\left(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}\right)^{-1} && \text{(66b)}\\
&= \left(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}\right)^{-1} && \text{(66c)}
\end{aligned}$$

Lecture Summary

- we have looked at how to estimate a whole trajectory using initial state knowledge, a sequence of inputs, and a sequence of measurements for linear systems corrupted by Gaussian noise
- our Bayesian method involves solving a linear system of equations for the mean and also builds the covariance
- we know how to solve this linear system very efficiently by exploiting the sparsity of the matrices
- we have a test for when the solution is unique
- the batch approach can only be used offline since it is acausal (it uses future data to estimate past states)
- the forward pass of the RTS smoother is the recursive Kalman filter, which can be used online

- Lecture 2: Nonlinearities and Lie Groups

Lecture 2: Nonlinearities and Lie Groups

A Short Course in State Estimation

Timothy D. Barfoot

University of Toronto

UNIVERSITY OF TORONTO

Lecture Outline

Lecture 1: Linear-Gaussian Estimation

Lecture 2: Nonlinearities and Lie Groups
- Problem Setup
- Gauss-Newton
- Sparsity
- Lie Groups and Algebras
- Perturbations and Optimization
- Example: Point-Cloud Alignment

Lecture 3: Continuous-Time Estimation

Lecture 4: Beyond MAP – Variational Inference

System

– we define our system using the following nonlinear models:

$$\text{motion model:} \quad \mathbf{x}_k = \mathbf{f}\left(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{w}_k\right), \qquad k = 1 \ldots K \tag{67a}$$

$$\text{observation model:} \quad \mathbf{y}_k = \mathbf{g}\left(\mathbf{x}_k, \mathbf{n}_k\right), \qquad k = 0 \ldots K \tag{67b}$$

where k is again the discrete-time index and K its maximum

– the variables have the following meanings:

$$\begin{aligned}
\text{system state :} \quad & \mathbf{x}_k \in \mathbb{R}^N \\
\text{initial state :} \quad & \mathbf{x}_0 \in \mathbb{R}^N \sim \mathcal{N}\left(\check{\mathbf{x}}_0, \check{\mathbf{P}}_0\right) \\
\text{input :} \quad & \mathbf{v}_k \in \mathbb{R}^N \\
\text{process noise :} \quad & \mathbf{w}_k \in \mathbb{R}^N \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}_k\right) \\
\text{measurement :} \quad & \mathbf{y}_k \in \mathbb{R}^M \\
\text{measurement noise :} \quad & \mathbf{n}_k \in \mathbb{R}^M \sim \mathcal{N}\left(\mathbf{0}, \mathbf{R}_k\right)
\end{aligned}$$

Nonlinear errors

- we aim to use the MAP approach to state estimation, but now for nonlinear systems
- we define the errors with respect to the prior and measurements to be

$$\mathbf{e}_{v,k}(\mathbf{x}) = \begin{cases} \check{\mathbf{x}}_0 - \mathbf{x}_0, & k = 0 \\ \mathbf{f}\left(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{0}\right) - \mathbf{x}_k, & k = 1 \dots K \end{cases} \tag{68a}$$

$$\mathbf{e}_{y,k}(\mathbf{x}) = \mathbf{y}_k - \mathbf{g}\left(\mathbf{x}_k, \mathbf{0}\right), \qquad k = 0 \dots K \tag{68b}$$

- as usual, there is one error term for every measurement, every input, and one for the initial state knowledge

MAP cost terms

- the contributions to the objective function are

$$J_{v,k}(\mathbf{x}) = \frac{1}{2}\mathbf{e}_{v,k}(\mathbf{x})^T \mathbf{W}_{v,k}^{-1} \mathbf{e}_{v,k}(\mathbf{x}) \tag{69a}$$

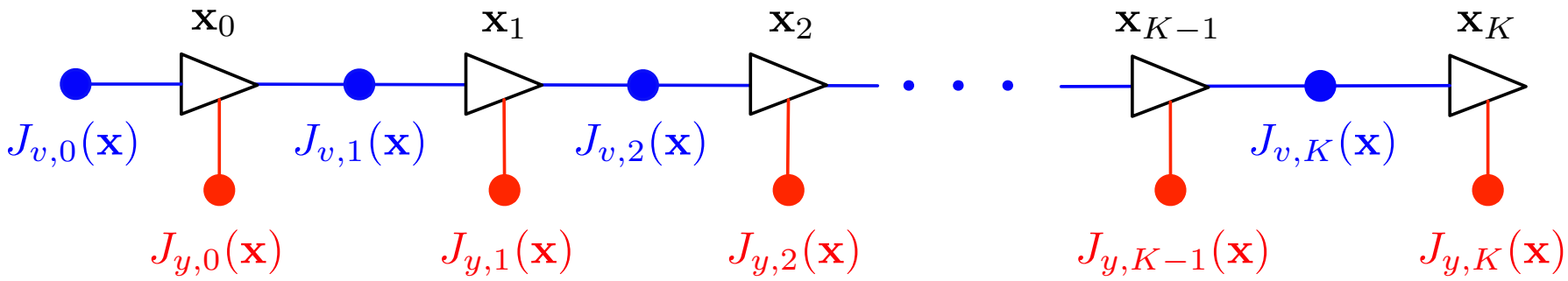
$$J_{y,k}(\mathbf{x}) = \frac{1}{2}\mathbf{e}_{y,k}(\mathbf{x})^T \mathbf{W}_{y,k}^{-1} \mathbf{e}_{y,k}(\mathbf{x}) \tag{69b}$$

- the overall objective function is then

$$J(\mathbf{x}) = \sum_{k=0}^{K} \left(J_{v,k}(\mathbf{x}) + J_{y,k}(\mathbf{x})\right) \tag{70}$$

- by choosing $\mathbf{W}_{v,k}$ and $\mathbf{W}_{y,k}$ to be the inverse covariances of the measurement noises, minimizing the objective function is equivalent to maximizing the joint likelihood of all the data

MAP as a factor graph



$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} J(\mathbf{x}) = \arg\min_{\mathbf{x}} \sum_{k=0}^{K} \Big(J_{v,k}(\mathbf{x}) + J_{y,k}(\mathbf{x}) \Big)$$

motion prior measurements

Stacking things up

– we further define

$$\mathbf{e}(\mathbf{x}) = \begin{bmatrix} \mathbf{e}_v(\mathbf{x}) \\ \mathbf{e}_y(\mathbf{x}) \end{bmatrix}, \quad \mathbf{e}_v(\mathbf{x}) = \begin{bmatrix} \mathbf{e}_{v,0}(\mathbf{x}) \\ \vdots \\ \mathbf{e}_{v,K}(\mathbf{x}) \end{bmatrix}, \quad \mathbf{e}_y(\mathbf{x}) = \begin{bmatrix} \mathbf{e}_{y,0}(\mathbf{x}) \\ \vdots \\ \mathbf{e}_{y,K}(\mathbf{x}) \end{bmatrix} \tag{71a}$$

$$\mathbf{W} = \mathsf{diag}\left(\mathbf{W}_v, \mathbf{W}_y\right), \quad \mathbf{W}_v = \mathsf{diag}\left(\mathbf{W}_{v,0}, \ldots, \mathbf{W}_{v,K}\right), \tag{71b}$$

$$\mathbf{W}_y = \mathsf{diag}\left(\mathbf{W}_{y,0}, \ldots, \mathbf{W}_{y,K}\right) \tag{71c}$$

so that the objective function can be written as

$$J(\mathbf{x}) = \frac{1}{2}\mathbf{e}(\mathbf{x})^T \mathbf{W}^{-1} \mathbf{e}(\mathbf{x}) \tag{72}$$

Standard nonlinear squared error form

- we can further define the modified error term,

$$\mathbf{u}(\mathbf{x}) = \mathbf{L}\,\mathbf{e}(\mathbf{x}) \tag{73}$$

  where $\mathbf{L}^T\mathbf{L} = \mathbf{W}^{-1}$ (i.e., from a Cholesky decomposition since $\mathbf{W}$ is symmetric positive-definite)
- using these definitions, we can write the objective function simply as

$$J(\mathbf{x}) = \frac{1}{2}\mathbf{u}(\mathbf{x})^T\mathbf{u}(\mathbf{x}) \tag{74}$$

- this is precisely in a quadratic form, but not with respect to the design variables, $\mathbf{x}$
- due to the nonlinearities, this is typically a nonconvex function, which can have local minima

Gauss-Newton

- the Gauss-Newton optimization method iteratively approximates the objective function as a quadratic, which can be analytically minimized at each iteration – result is a local minimization of $J(\mathbf{x})$
- a 'short-cut' to the Gauss-Newton method is to start with a first-order Taylor expansion of $\mathbf{u}(\mathbf{x})$:

$$\mathbf{u}(\mathbf{x}_{\text{op}} + \delta\mathbf{x}) \approx \mathbf{u}(\mathbf{x}_{\text{op}}) + \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \bigg|_{\mathbf{x}_{\text{op}}} \right) \delta\mathbf{x} \tag{75}$$

- substituting into J we have

$$\begin{aligned} & J(\mathbf{x}_{\text{op}} + \delta\mathbf{x}) \\ & \approx \frac{1}{2} \left(\mathbf{u}(\mathbf{x}_{\text{op}}) + \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \bigg|_{\mathbf{x}_{\text{op}}} \right) \delta\mathbf{x} \right)^T \left(\mathbf{u}(\mathbf{x}_{\text{op}}) + \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \bigg|_{\mathbf{x}_{\text{op}}} \right) \delta\mathbf{x} \right) \end{aligned} \tag{76}$$

Gauss-Newton

– minimizing with respect to $\delta\mathbf{x}$ gives

$$\frac{\partial J(\mathbf{x}_{\text{op}}+\delta\mathbf{x})}{\partial\,\delta\mathbf{x}} = \left(\mathbf{u}(\mathbf{x}_{\text{op}}) + \left(\frac{\partial\mathbf{u}(\mathbf{x})}{\partial\mathbf{x}}\bigg|_{\mathbf{x}_{\text{op}}}\right)\delta\mathbf{x}^*\right)^T\left(\frac{\partial\mathbf{u}(\mathbf{x})}{\partial\mathbf{x}}\bigg|_{\mathbf{x}_{\text{op}}}\right) = \mathbf{0}$$

$$\Rightarrow \left(\frac{\partial\mathbf{u}(\mathbf{x})}{\partial\mathbf{x}}\bigg|_{\mathbf{x}_{\text{op}}}\right)^T\left(\frac{\partial\mathbf{u}(\mathbf{x})}{\partial\mathbf{x}}\bigg|_{\mathbf{x}_{\text{op}}}\right)\delta\mathbf{x}^* = -\left(\frac{\partial\mathbf{u}(\mathbf{x})}{\partial\mathbf{x}}\bigg|_{\mathbf{x}_{\text{op}}}\right)^T\mathbf{u}(\mathbf{x}_{\text{op}}) \quad (77)$$

– we apply the update to our initial guess using

$$\mathbf{x}_{\text{op}} \leftarrow \mathbf{x}_{\text{op}} + \delta\mathbf{x}^* \quad (78)$$

and iterate to convergence

– modifications such a line search, Levenberg-Marquardt, and trust-region methods can be used to improve convergence

Maximum a posteriori

– getting back to our specific setup, we recall that

$$\mathbf{u}(\mathbf{x}) = \mathbf{L}\,\mathbf{e}(\mathbf{x}) \tag{79}$$

with $\mathbf{L}$ a constant

– we substitute this into the Gauss-Newton update to see that in terms of the error, $\mathbf{e}(\mathbf{x})$, we have

$$\left(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}\right)\delta\mathbf{x}^* = \mathbf{H}^T\mathbf{W}^{-1}\,\mathbf{e}(\mathbf{x}_{\text{op}}) \tag{80}$$

with

$$\mathbf{H} = -\left.\frac{\partial\mathbf{e}(\mathbf{x})}{\partial\mathbf{x}}\right|_{\mathbf{x}_{\text{op}}} \tag{81}$$

and where we have used $\mathbf{L}^T\mathbf{L} = \mathbf{W}^{-1}$

MAP

– at the individual error level, the linearized approximations are:

$$\mathbf{e}_{v,k}(\mathbf{x}_{\text{op}} + \delta\mathbf{x}) \approx \begin{cases} \mathbf{e}_{v,0}(\mathbf{x}_{\text{op}}) - \delta\mathbf{x}_0, & k = 0 \\ \mathbf{e}_{v,k}(\mathbf{x}_{\text{op}}) + \mathbf{F}_{k-1}\delta\mathbf{x}_{k-1} - \delta\mathbf{x}_k, & k = 1 \dots K \end{cases} \tag{82}$$

$$\mathbf{e}_{y,k}(\mathbf{x}_{\text{op}} + \delta\mathbf{x}) \approx \mathbf{e}_{y,k}(\mathbf{x}_{\text{op}}) - \mathbf{G}_k\delta\mathbf{x}_k, \quad k = 0 \dots K \tag{83}$$

where

$$\mathbf{e}_{v,k}(\mathbf{x}_{\text{op}}) \approx \begin{cases} \check{\mathbf{x}}_0 - \mathbf{x}_{\text{op},0}, & k = 0 \\ \mathbf{f}\left(\mathbf{x}_{\text{op},k-1}, \mathbf{v}_k, \mathbf{0}\right) - \mathbf{x}_{\text{op},k}, & k = 1 \dots K \end{cases} \tag{84}$$

$$\mathbf{e}_{y,k}(\mathbf{x}_{\text{op}}) \approx \mathbf{y}_k - \mathbf{g}\left(\mathbf{x}_{\text{op},k}, \mathbf{0}\right), \quad k = 0 \dots K \tag{85}$$

$$\mathbf{F}_{k-1} = \left.\frac{\partial \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{w}_k)}{\partial \mathbf{x}_{k-1}}\right|_{\mathbf{x}_{\text{op},k-1}, \mathbf{v}_k, \mathbf{0}}, \quad \mathbf{G}_k = \left.\frac{\partial \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k)}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_{\text{op},k}, \mathbf{0}} \tag{86}$$

MAP

– plugging these details into the batch Gauss-Newton equations

$$\underbrace{\left(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}\right)}_{\text{block-tridiagonal}}\ \delta\mathbf{x}^* = \mathbf{H}^T\mathbf{W}^{-1}\mathbf{e}(\mathbf{x}_{\text{op}}) \tag{87}$$

we have

$$\delta\mathbf{x} = \begin{bmatrix} \delta\mathbf{x}_0 \\ \delta\mathbf{x}_1 \\ \delta\mathbf{x}_2 \\ \vdots \\ \delta\mathbf{x}_K \end{bmatrix}, \quad \mathbf{H} = \left[\begin{array}{ccccccc} \mathbf{1} & & & & & & \\ -\mathbf{F}_0 & \mathbf{1} & & & & & \\ & -\mathbf{F}_1 & \ddots & & & & \\ & & \ddots & \mathbf{1} & & & \\ & & & -\mathbf{F}_{K-1} & \mathbf{1} & & \\ \hline \mathbf{G}_0 & & & & & & \\ & \mathbf{G}_1 & & & & & \\ & & \mathbf{G}_2 & & & & \\ & & & & \ddots & & \\ & & & & & & \mathbf{G}_K \end{array}\right], \quad \mathbf{e}(\mathbf{x}_{\text{op}}) = \left[\begin{array}{c} \mathbf{e}_{v,0}(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_{v,1}(\mathbf{x}_{\text{op}}) \\ \vdots \\ \mathbf{e}_{v,K}(\mathbf{x}_{\text{op}}) \\ \hline \mathbf{e}_{y,0}(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_{y,1}(\mathbf{x}_{\text{op}}) \\ \vdots \\ \mathbf{e}_{y,K}(\mathbf{x}_{\text{op}}) \end{array}\right], \tag{88}$$

$$\mathbf{W} = \text{diag}\left(\check{\mathbf{P}}_0, \mathbf{Q}'_1, \ldots, \mathbf{Q}'_K, \mathbf{R}'_0, \mathbf{R}'_1, \ldots, \mathbf{R}'_K\right) \tag{89}$$

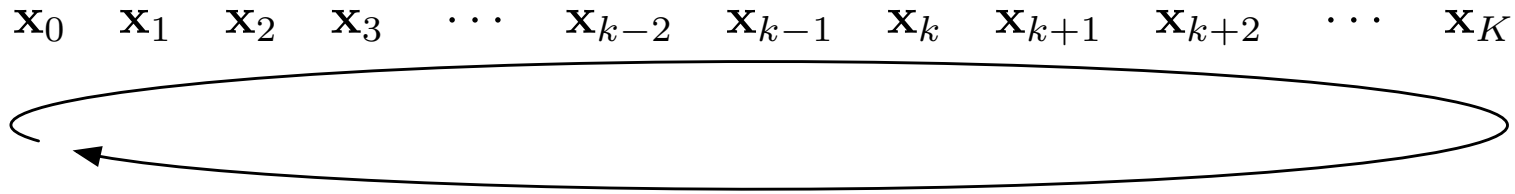
– we solve for $\delta\mathbf{x}^*$, update $\mathbf{x}_{\text{op}}$, and iterate to convergence

Exploiting sparsity

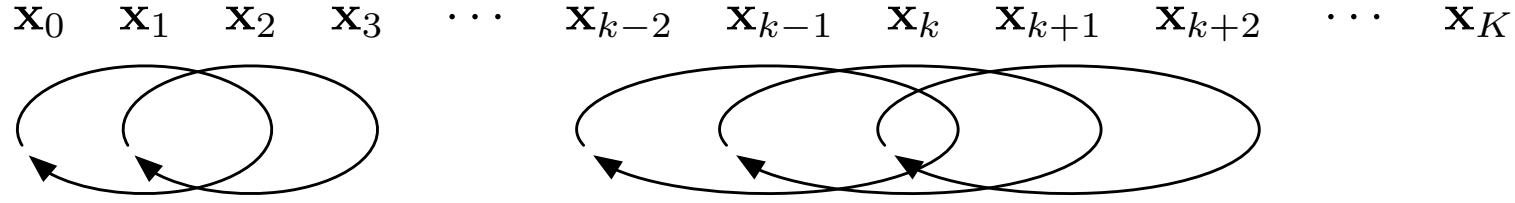
- just as in the linear-Gaussian case, we must solve a large linear system of equations where the left-hand side is block-tridiagonal – only now we must do this at each iteration of Gauss-Newton
- we can modify the Cholesky and RTS smoothers to help us solve this linear system of equations
  - if we use the RTS smoother in a single forward and backward pass where the linearization point is improved at each timestep, this is referred to as the extended RTS smoother – the forward pass of this is the extended Kalman filter
  - if iterate over the whole trajectory, the RTS smoother can efficiently implement the full Gauss-Newton method
  - there are schemes in-between these two extremes such as the iterated EKF and sliding-window filters

Iterating is key

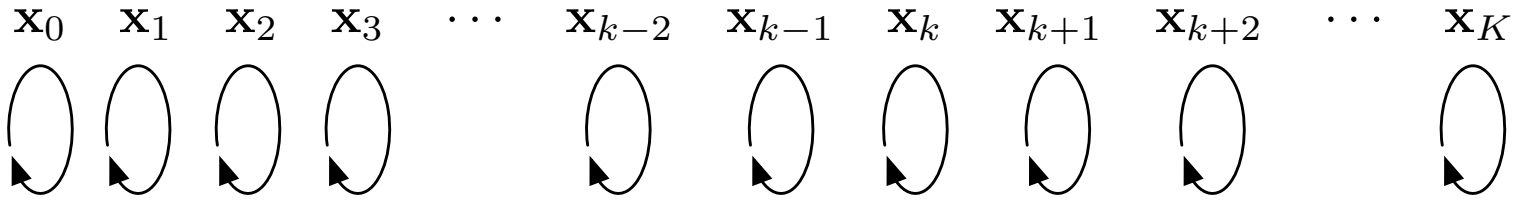
Gauss-Newton iterates over the entire trajectory, but runs offline and not in constant time



Sliding-window filters iterate over several timesteps at once, run online and in constant time

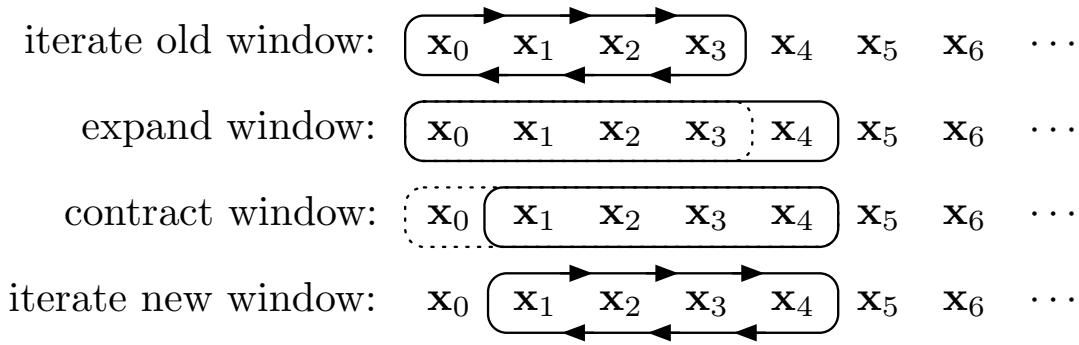


IEKF iterates at only one timestep at a time, but runs online and in constant time



Sliding-window filters

– sliding-window filters are a very common way to strike a nice balance (accuracy vs. computational cost) between full batch methods and basic filters:



– while there are several variants, most SWFs use marginalization in the window-contraction step:

$$\underbrace{\left[\begin{array}{c|ccc} \mathbf{A}_{00} & \mathbf{A}_{10}^T & & \\ \hline \mathbf{A}_{10} & \mathbf{A}_{11} & \mathbf{A}_{21}^T & \\ & \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{32}^T \\ & & \mathbf{A}_{32} & \mathbf{A}_{33} \quad \mathbf{A}_{43}^T \\ & & & \mathbf{A}_{43} \quad \mathbf{A}_{44} \end{array}\right]}_{\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}} \underbrace{\left[\begin{array}{c} \delta\mathbf{x}_0^\star \\ \hline \delta\mathbf{x}_1^\star \\ \delta\mathbf{x}_2^\star \\ \delta\mathbf{x}_3^\star \\ \delta\mathbf{x}_4^\star \end{array}\right]}_{\delta\mathbf{x}^\star} = \underbrace{\left[\begin{array}{c} \mathbf{b}_0 \\ \hline \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \\ \mathbf{b}_4 \end{array}\right]}_{\mathbf{H}^T\mathbf{W}^{-1}\mathbf{e}} \Rightarrow \underbrace{\left[\begin{array}{ccc} \mathbf{A}_{11} - \mathbf{A}_{10}\mathbf{A}_{00}^{-1}\mathbf{A}_{10}^T & \mathbf{A}_{21}^T & \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{32}^T \\ & \mathbf{A}_{32} & \mathbf{A}_{33} \quad \mathbf{A}_{43}^T \\ & & \mathbf{A}_{43} \quad \mathbf{A}_{44} \end{array}\right]}_{\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}} \underbrace{\left[\begin{array}{c} \delta\mathbf{x}_1^\star \\ \delta\mathbf{x}_2^\star \\ \delta\mathbf{x}_3^\star \\ \delta\mathbf{x}_4^\star \end{array}\right]}_{\delta\mathbf{x}^\star} = \underbrace{\left[\begin{array}{c} \mathbf{b}_1 - \mathbf{A}_{10}\mathbf{A}_{00}^{-1}\mathbf{b}_0 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \\ \mathbf{b}_4 \end{array}\right]}_{\mathbf{H}^T\mathbf{W}^{-1}\mathbf{e}}$$

We still need more tools

- consider the problem of aligning two point-clouds, $\boldsymbol{y}_j$ and $\boldsymbol{p}_j$
- we define our error term for each point as

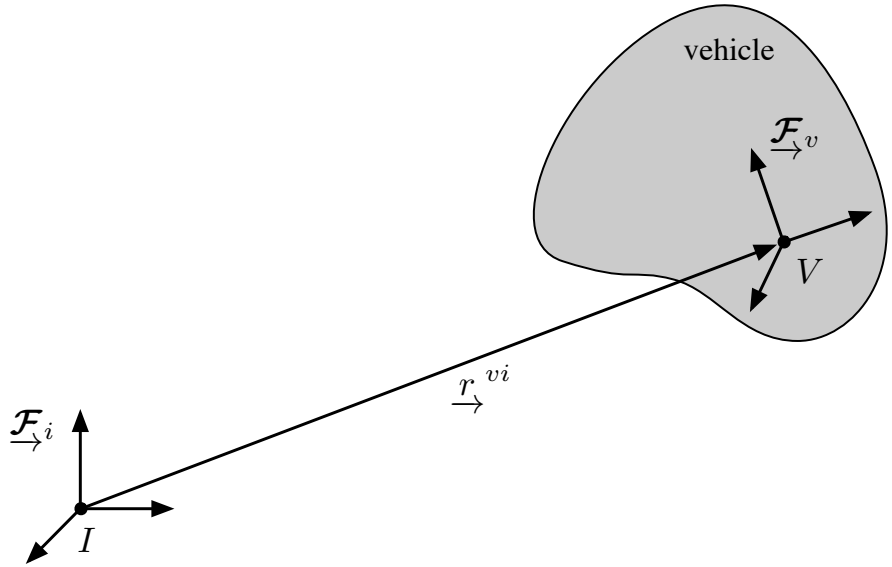
$$\mathbf{e}_j = \boldsymbol{y}_j - \mathbf{T}\boldsymbol{p}_j \tag{90}$$

  and our objective function (with no motion prior) as

$$J(\mathbf{T}) = \frac{1}{2}\sum_{j=1}^{M} w_j \mathbf{e}_j^T \mathbf{e}_j = \frac{1}{2}\sum_{j=1}^{M} w_j \left(\boldsymbol{y}_j - \mathbf{T}\boldsymbol{p}_j\right)^T \left(\boldsymbol{y}_j - \mathbf{T}\boldsymbol{p}_j\right) \tag{91}$$

  where $w_j > 0$ are the usual scalar weights
- our state in this problem, $\mathbf{T}$, is the unknown pose between the two point-clouds
- $\mathbf{T}$ is not a typical vector quantity – it has constraints on its form that we need to account for in the optimization problem

Pose



- the pose of one reference frame with respect to another has six degrees of freedom:
  - three in translation
  - three in rotation
- if we know the pose, $\left\{\mathbf{r}_i^{vi}, \mathbf{C}_{iv}\right\}$, we can transform the coordinates of a point, P, from one frame to another:

$$\mathbf{r}_i^{pi} = \mathbf{C}_{iv}\mathbf{r}_v^{pv} + \mathbf{r}_i^{vi} \tag{92}$$

Transformation matrices

– we can combine the translation and rotation of a pose into a convenient form called the (4×4) transformation matrix:

$$\begin{bmatrix} \mathbf{r}_i^{pi} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{C}_{iv} & \mathbf{r}_i^{vi} \\ \mathbf{0}^T & 1 \end{bmatrix}}_{\mathbf{T}_{iv}} \begin{bmatrix} \mathbf{r}_v^{pv} \\ 1 \end{bmatrix} \tag{93}$$

– we see that this allows us to easily transform points from one frame to another in so-called (4×1) homogenous point representation:

$$\begin{bmatrix} \mathbf{r} \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{94}$$

which just has an extra 1 at the bottom

Transformation matrices

- transformation matrices are 4×4 and always have this special structure:

$$\mathbf{T} = \begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{95}$$

- they have 16 parameters but only 6 degrees of freedom and therefore must have 10 constraints
- 6 constraints come from $\mathbf{C}^T\mathbf{C} = \mathbf{1}$ and the other 4 come from the fact that the bottom row is always $(0, 0, 0, 1)$
- we can compound transformation matrices (just like rotation matrices):

$$\mathbf{T}_{iv} = \mathbf{T}_{ia}\mathbf{T}_{ab}\mathbf{T}_{bv} \tag{96}$$

and the structure always holds (more on this later); order matters

Matrix Lie groups

- rotation and transformation matrices show up in our kinematic (i.e., motion) and sensor (i.e., observation) models
- neither behaves like a vector, yet all of our estimation tools assume the state is a vector
- it turns out that the sets of rotations and poses are not vector spaces, but another type of mathematical object called matrix Lie groups
- we will use the rest of this lecture to learn about this in the hope that it will guide us in the estimation of rotations and poses
- spoiler: we will use properties of the Lie-group structure to maintain our estimation algorithms as unconstrainted optimizations

Special orthogonal group

- the set of rotations is called the special orthogonal group:

$$SO(3) = \left\{ \mathbf{C} \in \mathbb{R}^{3\times 3} | \mathbf{C}\mathbf{C}^T = \mathbf{1}, \det \mathbf{C} = 1 \right\} \tag{97}$$

- the $\mathbf{C}\mathbf{C}^T = \mathbf{1}$ orthogonality condition is needed to impose 6 constraints on the 9-parameter rotation matrix, reducing the degrees of freedom to 3
- noticing that

$$(\det \mathbf{C})^2 = \det\left(\mathbf{C}\mathbf{C}^T\right) = \det \mathbf{1} = 1 \tag{98}$$

  we have that $\det \mathbf{C} = \pm 1$, allowing for two possibilities
- choosing $\det \mathbf{C} = 1$ ensures that we have a proper rotation, and it's this additional property that makes the $O(3)$ orthogonality group special

Special Euclidean group

– the set of transformation matrices representing poses is called the special Euclidean group:

$$SE(3) = \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} \middle| \mathbf{C} \in SO(3), \mathbf{r} \in \mathbb{R}^3 \right\} \tag{99}$$

Matrix Lie groups

- both $SO(3)$ and $SE(3)$ are matrix Lie groups
- to be a group they must have an operator to combine elements that satisfies 4 properties: closure, associativity, identity, invertibility
- to be a Lie group the operator must be 'smooth'
- to be a matrix Lie group the elements must be matrices and the operator matrix multiplication

property	$SO(3)$	$SE(3)$
closure	$\mathbf{C}_1, \mathbf{C}_2 \in SO(3)$ $\Rightarrow \mathbf{C}_1\mathbf{C}_2 \in SO(3)$	$\mathbf{T}_1, \mathbf{T}_2 \in SE(3)$ $\Rightarrow \mathbf{T}_1\mathbf{T}_2 \in SE(3)$
associativity	$\mathbf{C}_1 (\mathbf{C}_2\mathbf{C}_3) = (\mathbf{C}_1\mathbf{C}_2) \mathbf{C}_3$ $= \mathbf{C}_1\mathbf{C}_2\mathbf{C}_3$	$\mathbf{T}_1 (\mathbf{T}_2\mathbf{T}_3) = (\mathbf{T}_1\mathbf{T}_2) \mathbf{T}_3$ $= \mathbf{T}_1\mathbf{T}_2\mathbf{T}_3$
identity	$\mathbf{C}, \mathbf{1} \in SO(3)$ $\Rightarrow \mathbf{C1} = \mathbf{1C} = \mathbf{C}$	$\mathbf{T}, \mathbf{1} \in SE(3)$ $\Rightarrow \mathbf{T1} = \mathbf{1T} = \mathbf{T}$
invertibility	$\mathbf{C} \in SO(3)$ $\Rightarrow \mathbf{C}^{-1} \in SO(3)$	$\mathbf{T} \in SE(3)$ $\Rightarrow \mathbf{T}^{-1} \in SE(3)$

Lie algebras

– to every matrix Lie group there is associated a Lie algebra, which consists of a vector space, $\mathbb{V}$, over some field, $\mathbb{F}$, together with a binary operation, $[\cdot, \cdot]$, called the Lie bracket (of the algebra) that satisfies four properties:

$$\begin{aligned}
\text{closure:} \quad & [\mathbf{X}, \mathbf{Y}] \in \mathbb{V} \\
\text{bilinearity:} \quad & [a\mathbf{X} + b\mathbf{Y}, \mathbf{Z}] = a[\mathbf{X}, \mathbf{Z}] + b[\mathbf{Y}, \mathbf{Z}], \\
& \qquad [\mathbf{Z}, a\mathbf{X} + b\mathbf{Y}] = a[\mathbf{Z}, \mathbf{X}] + b[\mathbf{Z}, \mathbf{Y}] \\
\text{alternating:} \quad & [\mathbf{X}, \mathbf{X}] = \mathbf{0} \\
\text{Jacobi identity:} \quad & [\mathbf{X}, [\mathbf{Y}, \mathbf{Z}]] + [\mathbf{Z}, [\mathbf{Y}, \mathbf{X}]] + [\mathbf{Y}, [\mathbf{Z}, \mathbf{X}]] = \mathbf{0}
\end{aligned}$$

for all $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \in \mathbb{V}$ and $a, b \in \mathbb{F}$

Lie algebra: rotations

– the Lie algebra associated with $SO(3)$ is given by

$$\begin{aligned} \text{vector space:} \quad & \mathfrak{so}(3) = \left\{ \boldsymbol{\Phi} = \boldsymbol{\phi}^{\wedge} \in \mathbb{R}^{3\times 3} | \boldsymbol{\phi} \in \mathbb{R}^3, \right\} \\ \text{field:} \quad & \mathbb{R} \\ \text{Lie bracket:} \quad & [\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2] = \boldsymbol{\Phi}_1 \boldsymbol{\Phi}_2 - \boldsymbol{\Phi}_2 \boldsymbol{\Phi}_1 \end{aligned}$$

where

$$\boldsymbol{\phi}^{\wedge} = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix}^{\wedge} = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 3}, \quad \boldsymbol{\phi} \in \mathbb{R}^3 \tag{100}$$

is the (linear) skew-symmetric operator

Lie algebra: poses

– the Lie algebra associated with $SE(3)$ is given by

$$\begin{aligned} \text{vector space:} \quad & \mathfrak{se}(3) = \left\{ \boldsymbol{\Xi} = \boldsymbol{\xi}^{\wedge} \in \mathbb{R}^{4\times 4} | \boldsymbol{\xi} \in \mathbb{R}^6 \right\} \\ \text{field:} \quad & \mathbb{R} \\ \text{Lie bracket:} \quad & [\boldsymbol{\Xi}_1, \boldsymbol{\Xi}_2] = \boldsymbol{\Xi}_1 \boldsymbol{\Xi}_2 - \boldsymbol{\Xi}_2 \boldsymbol{\Xi}_1 \end{aligned}$$

where

$$\boldsymbol{\xi}^{\wedge} = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix}^{\wedge} = \begin{bmatrix} \boldsymbol{\phi}^{\wedge} & \boldsymbol{\rho} \\ \mathbf{0}^T & 0 \end{bmatrix} \in \mathbb{R}^{4\times 4}, \quad \boldsymbol{\rho}, \boldsymbol{\phi} \in \mathbb{R}^3 \tag{101}$$

– this is an overloading of the $(\cdot)^{\wedge}$ operator from before to take elements of $\mathbb{R}^6$ and turn them into elements of $\mathbb{R}^{4\times 4}$; it is still linear

This is getting exponentially more complicated

- ok, so the sets of rotation and transformation matrices are matrix Lie groups: $SO(3)$ and $SE(3)$
- each one has an associated Lie algebra: $\mathfrak{so}(3)$ and $\mathfrak{se}(3)$
- so what, where is this all going?!
- to get to the next level of understanding, we need a connection between the Lie group and Lie algebra
- that connection is the exponential map given by

$$\exp(\mathbf{A}) = \mathbf{1} + \mathbf{A} + \frac{1}{2!}\mathbf{A}^2 + \frac{1}{3!}\mathbf{A}^3 + \cdots = \sum_{n=0}^{\infty} \frac{1}{n!}\mathbf{A}^n \tag{102}$$

where $\mathbf{A} \in \mathbb{R}^{M \times M}$ is a square matrix

Exponential rotations

- for rotations, we can relate elements of $SO(3)$ to elements of $\mathfrak{so}(3)$ through the exponential map:

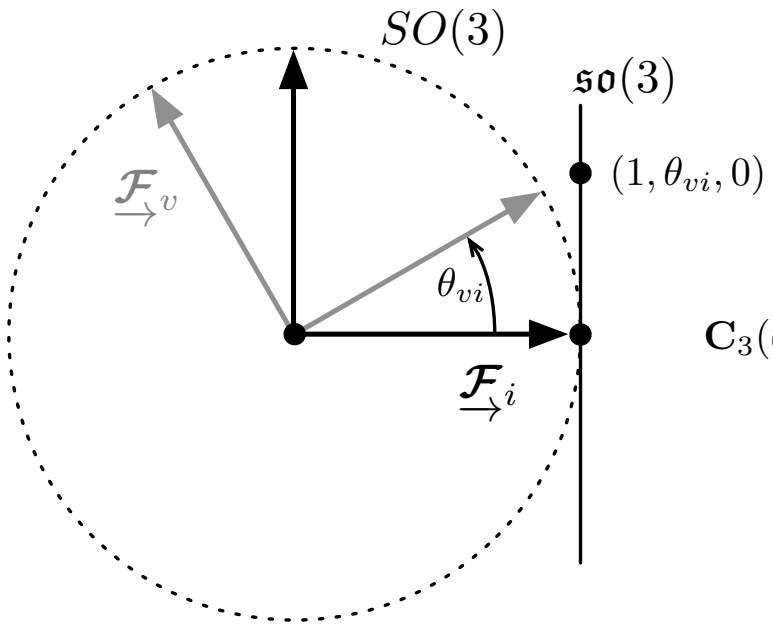
$$\mathbf{C} = \exp\left(\boldsymbol{\phi}^{\wedge}\right) = \sum_{n=0}^{\infty} \frac{1}{n!} \left(\boldsymbol{\phi}^{\wedge}\right)^{n} \tag{103}$$

  where $\mathbf{C} \in SO(3)$ and $\boldsymbol{\phi} \in \mathbb{R}^3$ (and hence $\boldsymbol{\phi}^{\wedge} \in \mathfrak{so}(3)$)
- we can also go in the other direction (but not uniquely) using

$$\boldsymbol{\phi} = \ln\left(\mathbf{C}\right)^{\vee} \tag{104}$$

- the mapping is surjective (or onto), meaning every element of $SO(3)$ can be generated by at least one element of $\mathfrak{so}(3)$
- the non-unique inverse mapping is due to singularities – in this case $\phi + 2\pi m$ with m any integer produces the same $\mathbf{C}$

Tangent space



$$\mathbf{C}_3(\theta_{vi})\,\mathbf{1}_1 = \exp\left(\begin{bmatrix}0\\0\\\theta_{vi}\end{bmatrix}^\wedge\right)\begin{bmatrix}1\\0\\0\end{bmatrix}$$

$$\approx \left(\mathbf{1} + \begin{bmatrix}0\\0\\\theta_{vi}\end{bmatrix}^\wedge\right)\begin{bmatrix}1\\0\\0\end{bmatrix} = \begin{bmatrix}1\\\theta_{vi}\\0\end{bmatrix}$$

– the vector space of a Lie algebra is the tangent space of the associated Lie group at the identity element of the group, and it completely captures the local structure of the group

Exponential poses

- for poses, we can relate elements of $SE(3)$ to elements of $\mathfrak{se}(3)$, again through the exponential map:

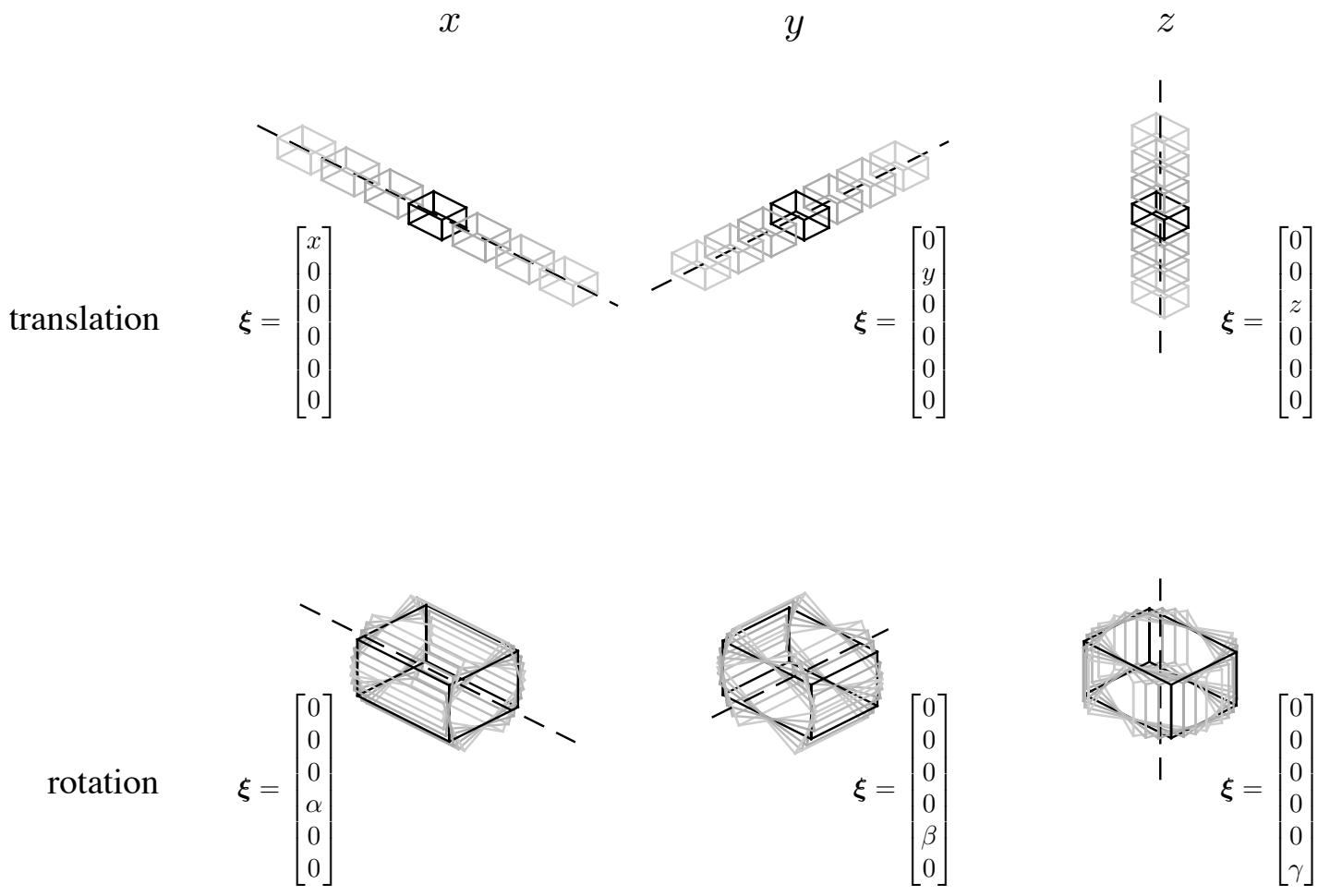
$$\mathbf{T} = \exp\left(\boldsymbol{\xi}^{\wedge}\right) = \sum_{n=0}^{\infty} \frac{1}{n!} \left(\boldsymbol{\xi}^{\wedge}\right)^{n} \tag{105}$$

  where $\mathbf{T} \in SE(3)$ and $\boldsymbol{\xi} \in \mathbb{R}^6$ (and hence $\boldsymbol{\xi}^{\wedge} \in \mathfrak{se}(3)$)
- we can also go in the other direction (again, not uniquely) using

$$\boldsymbol{\xi} = \ln\left(\mathbf{T}\right)^{\vee} \tag{106}$$

- the exponential map from $\mathfrak{se}(3)$ to $SE(3)$ is also surjective: every $\boldsymbol{\xi} \in \mathbb{R}^6$ maps to some $\mathbf{T} \in SE(3)$ (many-to-one) and every $\mathbf{T} \in SE(3)$ can be generated by at least one $\boldsymbol{\xi} \in \mathbb{R}^6$

Pose change



- varying each component of $\boldsymbol{\xi}$ then using $\mathbf{T} = \exp\left(\boldsymbol{\xi}^{\wedge}\right)$ to transform the points comprising the corners of a rectangular prism

Perturbations

- we now introduce the idea of perturbations
- for vectors, we usually perturb like this:

$$\mathbf{x} = \underbrace{\bar{\mathbf{x}}}_{\text{'big'}} + \underbrace{\delta\mathbf{x}}_{\text{'small'}} \tag{107}$$

  but actually this is an arbitrary choice
- in an optimization setting, perturbations are used like this:

$$\mathbf{x} = \underbrace{\bar{\mathbf{x}}}_{\text{initial guess}} + \underbrace{\delta\mathbf{x}}_{\text{optimal update}} \tag{108}$$

- in a probability setting, perturbations are used like this:

$$\mathbf{x} = \underbrace{\bar{\mathbf{x}}}_{\text{deterministic}} + \underbrace{\delta\mathbf{x}}_{\text{random noise}} \tag{109}$$

Rotation perturbations

– for rotations, we will perturb like this:

$$\mathbf{C} = \underbrace{\delta\mathbf{C}}_{\text{'small'}} \underbrace{\bar{\mathbf{C}}}_{\text{'big'}} \tag{110}$$

– we pick the following perturbation

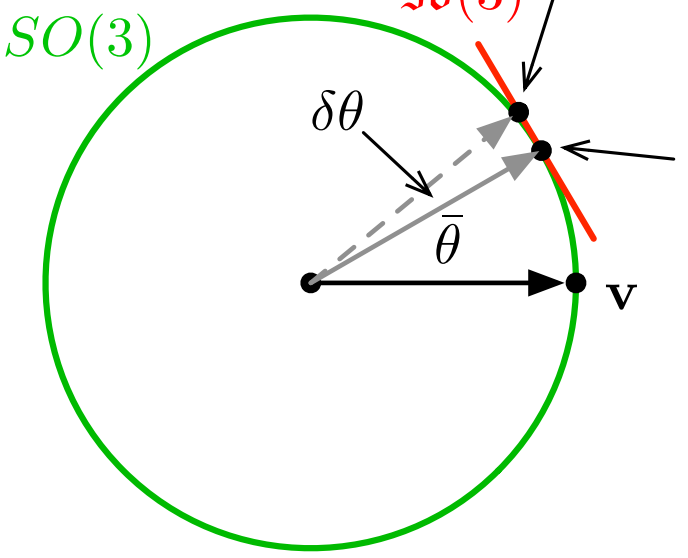
$$\delta\mathbf{C} = \exp\left(\boldsymbol{\psi}^{\wedge}\right) \tag{111}$$

which ensures $\mathbf{C}$ is still a valid rotation (by closure)

– this lets us linearize the product of a rotation and point, $\mathbf{v}$:

$$\mathbf{C}\mathbf{v} = \delta\mathbf{C}\,\bar{\mathbf{C}}\mathbf{v} = \exp\left(\boldsymbol{\psi}^{\wedge}\right)\bar{\mathbf{C}}\mathbf{v} \approx \left(\mathbf{1} + \boldsymbol{\psi}^{\wedge}\right)\bar{\mathbf{C}}\mathbf{v} = \bar{\mathbf{C}}\mathbf{v} - \left(\bar{\mathbf{C}}\mathbf{v}\right)^{\wedge}\boldsymbol{\psi} \tag{112}$$

Rotation perturbations

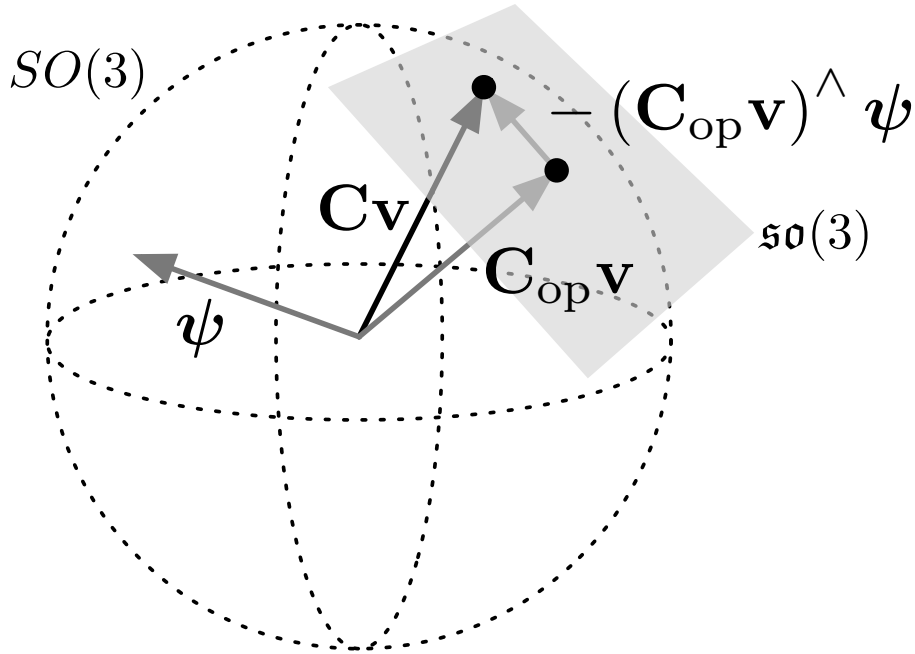


$$\mathbf{C}\mathbf{v} \approx \bar{\mathbf{C}}\mathbf{v} - (\bar{\mathbf{C}}\mathbf{v})^{\wedge}\boldsymbol{\psi} = \begin{bmatrix}\cos\bar{\theta}\\ \sin\bar{\theta}\\ 0\end{bmatrix} + \begin{bmatrix}-\sin\bar{\theta}\\ \cos\bar{\theta}\\ 0\end{bmatrix}\delta\theta$$

$$\bar{\mathbf{C}}\mathbf{v} = \begin{bmatrix}\cos\bar{\theta}\\ \sin\bar{\theta}\\ 0\end{bmatrix}$$

$$\mathbf{v} = \begin{bmatrix}1\\0\\0\end{bmatrix} \quad \boldsymbol{\psi} = \begin{bmatrix}0\\0\\\delta\theta\end{bmatrix} \quad \bar{\mathbf{C}} = \begin{bmatrix}\cos\bar{\theta} & -\sin\bar{\theta} & 0\\ \sin\bar{\theta} & \cos\bar{\theta} & 0\\ 0 & 0 & 1\end{bmatrix}$$

Rotation perturbations



$$\mathbf{C}\mathbf{v} = \exp\left(\boldsymbol{\psi}^{\wedge}\right)\mathbf{C}_{\text{op}}\mathbf{v} \approx \mathbf{C}_{\text{op}}\mathbf{v} - (\mathbf{C}_{\text{op}}\mathbf{v})^{\wedge}\boldsymbol{\psi}$$

Pose perturbations

- for poses, we will perturb like this:

$$\mathbf{T} = \underbrace{\delta\mathbf{T}}_{\text{'small'}} \underbrace{\bar{\mathbf{T}}}_{\text{'big'}} \tag{113}$$

- we pick the following perturbation

$$\delta\mathbf{T} = \exp\left(\boldsymbol{\epsilon}^{\wedge}\right) \tag{114}$$

  which ensures $\mathbf{T}$ is still a valid pose (by closure)

- can now linearize the product of a pose and homogeneous point, $\mathbf{p}$:

$$\mathbf{T}\mathbf{p} = \delta\mathbf{T}\,\bar{\mathbf{T}}\mathbf{p} = \exp\left(\boldsymbol{\epsilon}^{\wedge}\right)\bar{\mathbf{T}}\mathbf{p} \approx \left(\mathbf{1} + \boldsymbol{\epsilon}^{\wedge}\right)\bar{\mathbf{T}}\mathbf{p} = \bar{\mathbf{T}}\mathbf{p} + \left(\bar{\mathbf{T}}\mathbf{p}\right)^{\odot}\boldsymbol{\epsilon} \tag{115}$$

- we have used

$$\boldsymbol{\epsilon}^{\wedge}\mathbf{p} \equiv \mathbf{p}^{\odot}\boldsymbol{\epsilon}, \qquad \mathbf{p}^{\odot} = \begin{bmatrix}\boldsymbol{\rho}\\ \eta\end{bmatrix}^{\odot} = \begin{bmatrix}\eta\mathbf{1} & -\boldsymbol{\rho}^{\wedge}\\ \mathbf{0}^{T} & \mathbf{0}^{T}\end{bmatrix} \tag{116}$$

Rotation optimization

– choose a perturbation scheme,

$$\mathbf{C} = \exp\left(\boldsymbol{\psi}^{\wedge}\right)\mathbf{C}_{\text{op}} \tag{117}$$

where $\boldsymbol{\psi}$ is a small perturbation applied to an initial guess, $\mathbf{C}_{\text{op}}$

– insert this in the function, $u(\mathbf{x})$, to be optimized:

$$\begin{aligned} u\left(\mathbf{C}\mathbf{v}\right) &= u\left(\exp\left(\boldsymbol{\psi}^{\wedge}\right)\mathbf{C}_{\text{op}}\mathbf{v}\right) \approx u\left(\left(\mathbf{1}+\boldsymbol{\psi}^{\wedge}\right)\mathbf{C}_{\text{op}}\mathbf{v}\right) \\ &\approx u(\mathbf{C}_{\text{op}}\mathbf{v}) \underbrace{- \left.\frac{\partial u}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{C}_{\text{op}}\mathbf{v}}}_{\boldsymbol{\delta}^{T}} (\mathbf{C}_{\text{op}}\mathbf{v})^{\wedge}\boldsymbol{\psi} = u(\mathbf{C}_{\text{op}}\mathbf{v}) + \boldsymbol{\delta}^{T}\boldsymbol{\psi} \end{aligned} \tag{118}$$

– then pick a perturbation, $\boldsymbol{\psi}$, to decrease the function

Rotation optimization: gradient descent

- suppose we would like to perform Riemannian gradient descent
- in this case, we would pick the perturbation to be of the form

$$\boldsymbol{\psi} = -\alpha \boldsymbol{\delta} \tag{119}$$

  with $\alpha > 0$ a small step size
- we see the function is reduced by taking this step:

$$u\left(\mathbf{C}\mathbf{v}\right) - u\left(\mathbf{C}_{\mathrm{op}}\mathbf{v}\right) \approx -\underbrace{\alpha\, \boldsymbol{\delta}^T \boldsymbol{\delta}}_{\geq 0} \tag{120}$$

- retraction: apply the perturbation to update the initial guess,

$$\mathbf{C}_{\mathrm{op}} \leftarrow \exp\left(-\alpha \boldsymbol{\delta}^{\wedge}\right) \mathbf{C}_{\mathrm{op}} \tag{121}$$

  so that $\mathbf{C}_{\mathrm{op}} \in SO(3)$ at each iteration; iterate to convergence

Rotation optimization: Gauss-Newton

- gradient descent can be quite slow
- let's look at Gauss-Newton optimization
- suppose we have a general nonlinear, quadratic cost function of a rotation of the form,

$$J(\mathbf{C}) = \frac{1}{2} \sum_m \left(u_m(\mathbf{C}\mathbf{v}_m) \right)^2 \tag{122}$$

  where $u_m(\cdot)$ are scalar nonlinear functions and $\mathbf{v}_m \in \mathbb{R}^3$ are three-dimensional points
- we begin with an initial guess for the optimal rotation, $\mathbf{C}_{\text{op}} \in SO(3)$, and then perturb this (on the left) according to

$$\mathbf{C} = \exp\left(\boldsymbol{\psi}^\wedge\right) \mathbf{C}_{\text{op}} \tag{123}$$

  where $\boldsymbol{\psi}$ is the perturbation

Rotation optimization: Gauss-Newton

– we then apply our perturbation scheme inside each $u_m(\cdot)$ so that

$$u_m\left(\mathbf{C}\mathbf{v}_m\right) = u_m\left(\exp(\boldsymbol{\psi}^\wedge)\mathbf{C}_{\text{op}}\mathbf{v}_m\right) \approx u_m\left(\left(\mathbf{1}+\boldsymbol{\psi}^\wedge\right)\mathbf{C}_{\text{op}}\mathbf{v}_m\right)$$
$$\approx \underbrace{u_m(\mathbf{C}_{\text{op}}\mathbf{v}_m)}_{\beta_m} \underbrace{- \left.\frac{\partial u_m}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{C}_{\text{op}}\mathbf{v}_m} (\mathbf{C}_{\text{op}}\mathbf{v}_m)^\wedge \boldsymbol{\psi}}_{\boldsymbol{\delta}_m^T} \tag{124}$$

is a linearized version of $u_m(\cdot)$ in terms of our perturbation, $\boldsymbol{\psi}$

– inserting this back into our cost function we have

$$J(\mathbf{C}) \approx \frac{1}{2}\sum_m \left(\boldsymbol{\delta}_m^T\boldsymbol{\psi} + \beta_m\right)^2 \tag{125}$$

which is exactly quadratic in $\boldsymbol{\psi}$

Rotation optimization: Gauss-Newton

– taking the derivative of J with respect to $\boldsymbol{\psi}$ we have

$$\frac{\partial J}{\partial \boldsymbol{\psi}^T} = \sum_m \boldsymbol{\delta}_m \left(\boldsymbol{\delta}_m^T \boldsymbol{\psi} + \beta_m \right) \tag{126}$$

– set the derivative to zero to find the optimal perturbation, $\boldsymbol{\psi}^\star$:

$$\left(\sum_m \boldsymbol{\delta}_m \boldsymbol{\delta}_m^T \right) \boldsymbol{\psi}^\star = - \sum_m \beta_m \boldsymbol{\delta}_m \tag{127}$$

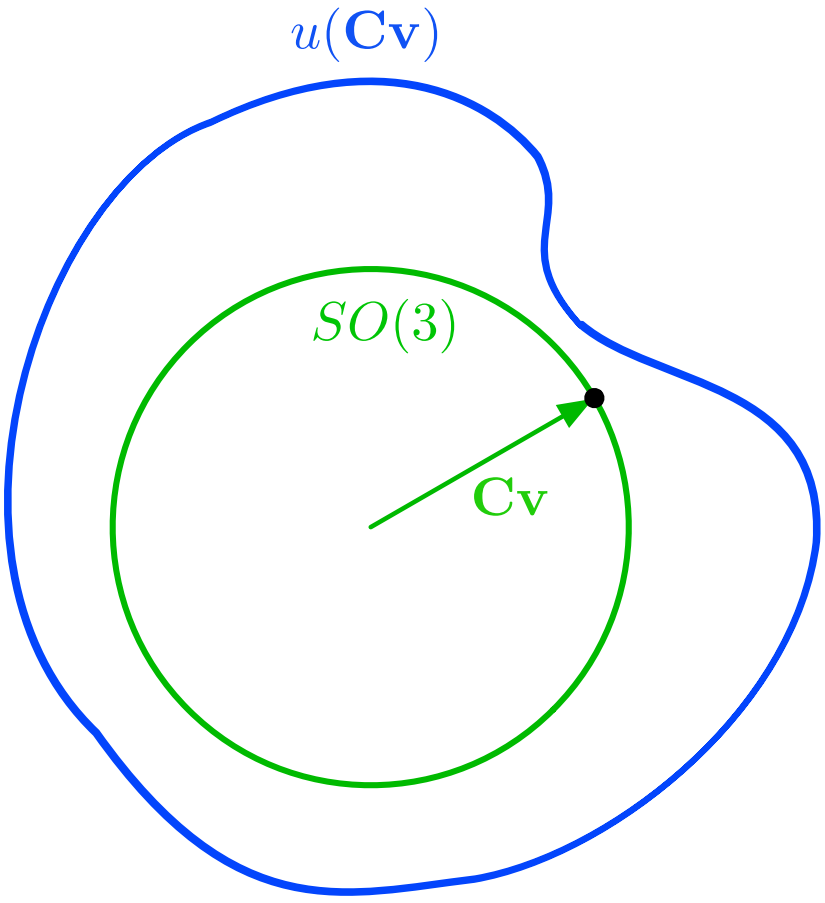
– this is a linear system of equations, which we can solve for $\boldsymbol{\psi}^\star$

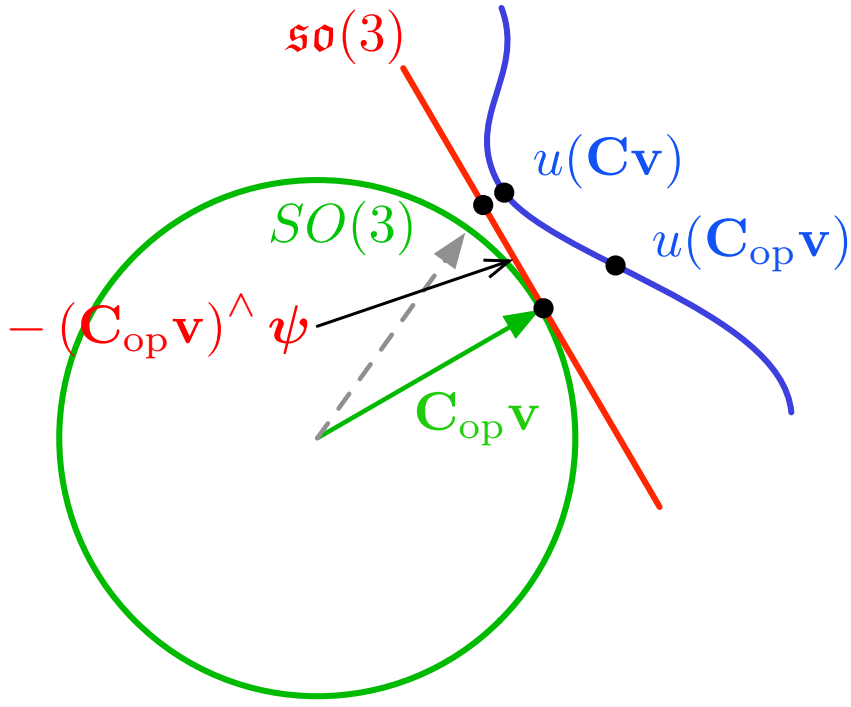
– retraction: apply this optimal perturbation to our initial guess,

$$\mathbf{C}_{\text{op}} \leftarrow \exp\left(\boldsymbol{\psi}^{\star\wedge} \right) \mathbf{C}_{\text{op}}, \tag{128}$$

so that $\mathbf{C}_{\text{op}} \in SO(3)$ at each iteration; iterate to convergence

Rotation optimization





Rotation optimization commentary

- we have adapted classic optimization algorithms to work with the matrix Lie group, $SO(3)$, by exploiting the surjective property of the exponential map to define an appropriate perturbation scheme

$$\mathbf{C} = \exp\left(\boldsymbol{\psi}^{\wedge}\right)\mathbf{C}_{\text{op}} \tag{129}$$

- we are essentially assuming that at each iteration the update, $\boldsymbol{\psi}$, will be small and so have mapped the optimization problem from the Lie group up into the Lie algebra, $\mathfrak{so}(3)$
- this approach has three major advantages:
  - we are storing our rotation in a singularity-free format, $\mathbf{C}_{\text{op}}$
  - at each iteration we are performing unconstrained optimization
  - our manipulations occur at the matrix level
- we get away with this because the perturbation always becomes very small as we converge to the optimum

Pose optimization

- the same concepts can also be applied to poses
- suppose we have a general nonlinear, quadratic cost function of a transformation of the form

$$J(\mathbf{T}) = \frac{1}{2} \sum_m \left(u_m(\mathbf{T}\mathbf{p}_m) \right)^2 \tag{130}$$

  where $u_m(\cdot)$ are nonlinear functions and $\mathbf{p}_m \in \mathbb{R}^4$ are three-dimensional points expressed in homogeneous coordinates
- we begin with an initial guess for the optimal transformation, $\mathbf{T}_{\text{op}} \in SE(3)$, and then perturb this (on the left) according to

$$\mathbf{T} = \exp\left(\boldsymbol{\epsilon}^\wedge\right) \mathbf{T}_{\text{op}} \tag{131}$$

  where $\boldsymbol{\epsilon}$ is the perturbation

Pose optimization

– we then apply our perturbation scheme inside each $u_m(\cdot)$ so that

$$u_m\left(\mathbf{T}\mathbf{p}_m\right) = u_m\left(\exp(\boldsymbol{\epsilon}^\wedge)\mathbf{T}_{\text{op}}\mathbf{p}_m\right) \approx u_m\left(\left(\mathbf{1}+\boldsymbol{\epsilon}^\wedge\right)\mathbf{T}_{\text{op}}\mathbf{p}_m\right)$$
$$\approx \underbrace{u_m(\mathbf{T}_{\text{op}}\mathbf{p}_m)}_{\beta_m} + \underbrace{\left.\frac{\partial u_m}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{T}_{\text{op}}\mathbf{p}_m} (\mathbf{T}_{\text{op}}\mathbf{p}_m)^\odot}_{\boldsymbol{\delta}_m^T}\boldsymbol{\epsilon} \tag{132}$$

is a linearized version of $u_m(\cdot)$ in terms of our perturbation, $\boldsymbol{\epsilon}$

– inserting this back into our cost function we have

$$J(\mathbf{T}) = \frac{1}{2}\sum_m \left(\boldsymbol{\delta}_m^T\boldsymbol{\epsilon} + \beta_m\right)^2 \tag{133}$$

which is exactly quadratic in $\boldsymbol{\epsilon}$

Pose optimization

– taking the derivative of J with respect to $\boldsymbol{\epsilon}$ we have

$$\frac{\partial J}{\partial \boldsymbol{\epsilon}^T} = \sum_m \boldsymbol{\delta}_m \left(\boldsymbol{\delta}_m^T \boldsymbol{\epsilon} + \beta_m \right) \tag{134}$$

– set the derivative to zero to find the optimal perturbation, $\boldsymbol{\epsilon}^\star$:

$$\left(\sum_m \boldsymbol{\delta}_m \boldsymbol{\delta}_m^T \right) \boldsymbol{\epsilon}^\star = -\sum_m \beta_m \boldsymbol{\delta}_m \tag{135}$$

– this is a linear system of equations, which we can solve for $\boldsymbol{\epsilon}^\star$

– retraction: apply this optimal perturbation to our initial guess,

$$\mathbf{T}_{\text{op}} \leftarrow \exp\left(\boldsymbol{\epsilon}^{\star\wedge} \right) \mathbf{T}_{\text{op}} \tag{136}$$

so that $\mathbf{T}_{\text{op}} \in SE(3)$ at each iteration; iterate to convergence

Point-cloud alignment

- consider the problem of aligning two point-clouds, $\boldsymbol{y}_j$ and $\boldsymbol{p}_j$, which are in homogeneous-point form and $j = 1 \dots J$
- we define our error term for each point pair as

$$\mathbf{e}_j = \boldsymbol{y}_j - \mathbf{T}\boldsymbol{p}_j \tag{137}$$

- we define our objective function as

$$J(\mathbf{T}) = \frac{1}{2}\sum_{j=1}^{M} w_j \mathbf{e}_j^T \mathbf{e}_j = \frac{1}{2}\sum_{j=1}^{M} w_j \left(\boldsymbol{y}_j - \mathbf{T}\boldsymbol{p}_j\right)^T \left(\boldsymbol{y}_j - \mathbf{T}\boldsymbol{p}_j\right) \tag{138}$$

  where $w_j > 0$ are scalar weights
- we seek to minimize J with respect to $\mathbf{T} \in SE(3)$; we want to know the pose between the two point-clouds

Point-cloud alignment

– we use our $SE(3)$-sensitive perturbation scheme

$$\mathbf{T} = \exp\left(\boldsymbol{\epsilon}^{\wedge}\right)\mathbf{T}_{\text{op}} \approx \left(\mathbf{1} + \boldsymbol{\epsilon}^{\wedge}\right)\mathbf{T}_{\text{op}} \tag{139}$$

where $\mathbf{T}_{\text{op}}$ is some initial guess and $\boldsymbol{\epsilon}$ is a small perturbation

– inserting this into the objective function we then have

$$J(\mathbf{T}) \approx \frac{1}{2}\sum_{j=1}^{M} w_j \left((\boldsymbol{y}_j - \boldsymbol{z}_j) - \boldsymbol{z}_j^{\odot}\,\boldsymbol{\epsilon}\right)^T \left((\boldsymbol{y}_j - \boldsymbol{z}_j) - \boldsymbol{z}_j^{\odot}\,\boldsymbol{\epsilon}\right) \tag{140}$$

where $\boldsymbol{z}_j = \mathbf{T}_{\text{op}}\boldsymbol{p}_j$ and we have used that

$$\boldsymbol{\epsilon}^{\wedge}\,\boldsymbol{z}_j = \boldsymbol{z}_j^{\odot}\,\boldsymbol{\epsilon} \tag{141}$$

– the objective function is now exactly quadratic in $\boldsymbol{\epsilon}$

Point-cloud alignment

- we can carry out a simple, unconstrained optimization for $\boldsymbol{\epsilon}$
- taking the derivative we find

$$\frac{\partial J}{\partial \boldsymbol{\epsilon}^T} = -\sum_{j=1}^{M} w_j \boldsymbol{z}_j^{\odot^T} \left((\boldsymbol{y}_j - \boldsymbol{z}_j) - \boldsymbol{z}_j^{\odot} \boldsymbol{\epsilon} \right) \tag{142}$$

- setting this to zero, we have the following system of equations for the optimal $\boldsymbol{\epsilon}^\star$:

$$\left(\frac{1}{w} \sum_{j=1}^{M} w_j \boldsymbol{z}_j^{\odot^T} \boldsymbol{z}_j^{\odot} \right) \boldsymbol{\epsilon}^\star = \frac{1}{w} \sum_{j=1}^{M} w_j \boldsymbol{z}_j^{\odot^T} (\boldsymbol{y}_j - \boldsymbol{z}_j) \tag{143}$$

- retraction: apply the optimal perturbation and iterate to convergence:

$$\mathbf{T}_{\text{op}} \leftarrow \exp\left(\boldsymbol{\epsilon}^{\star\wedge} \right) \mathbf{T}_{\text{op}} \tag{144}$$

Pose optimization commentary

- we have adapted classic optimization algorithms to work with the matrix Lie group, $SE(3)$, by exploiting the surjective property of the exponential map to define an appropriate perturbation scheme

$$\mathbf{T} = \exp\left(\boldsymbol{\epsilon}^{\wedge}\right) \mathbf{T}_{\text{op}} \tag{145}$$

- we are essentially assuming that at each iteration the update, $\boldsymbol{\epsilon}$, will be small and so have mapped the optimization problem from the Lie group up into the Lie algebra, $\mathfrak{se}(3)$
- this approach has three major advantages:
  - we are storing our pose in a singularity-free format, $\mathbf{T}_{\text{op}}$
  - at each iteration we are performing unconstrained optimization
  - our manipulations occur at the matrix level
- we get away with this because the perturbation always becomes very small as we converge to the optimum

Lecture Summary

- we have seen that despite nonlinearities in our motion and observation models and even state variables that are members of Lie groups, we can still carry out MAP estimation very similar to the linear-Gaussian case
- the difference is that we now need to iterate to convergence, which we do using some variant of Gauss-Newton
- many of the usual algorithms differ only based on which variables are held fixed and which variables are optimized at each iteration (e.g., full batch, extended RTS smoother, sliding-window filter, (iterated) EKF)
- Lecture 3: Continuous-Time Estimation

Lecture 3: Continuous-Time Estimation

A Short Course in State Estimation

Timothy D. Barfoot

University of Toronto

Lecture Outline

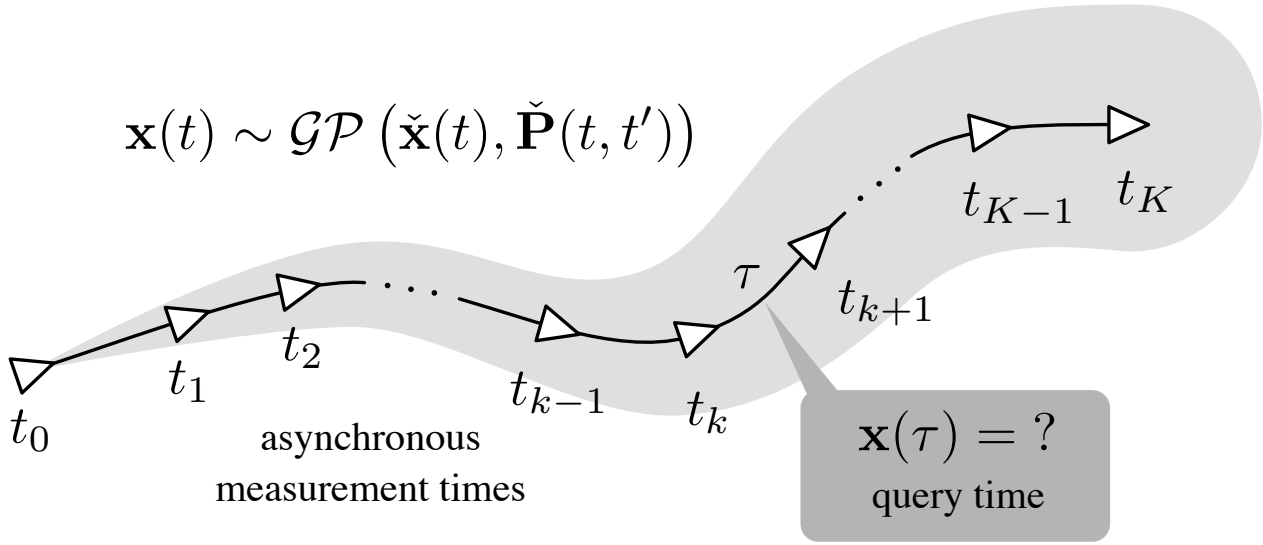
Lecture 1: Linear-Gaussian Estimation

Lecture 2: Nonlinearities and Lie Groups

Lecture 3: Continuous-Time Estimation
- Problem Setup
- Motion Prior
- GP Regression
- Lie Groups
- STEAM

Lecture 4: Beyond MAP – Variational Inference

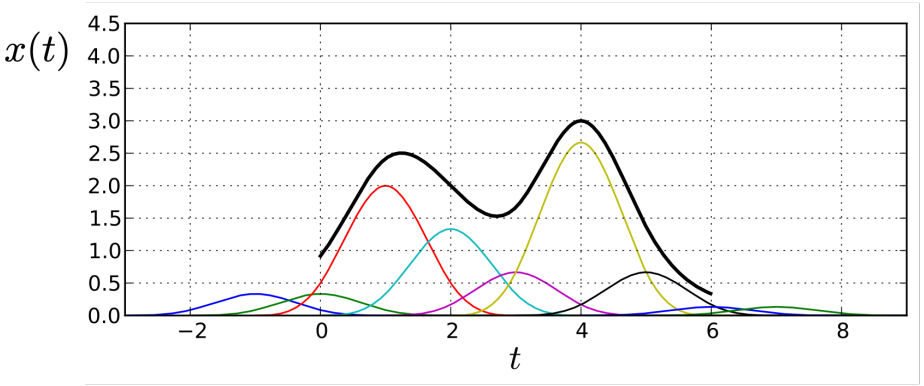
Motivation



- there are situations where it is beneficial to think of the robot's trajectory as a continuous function of time
- e.g., asynchronous measurements, scanning-while-moving sensors

Two camps

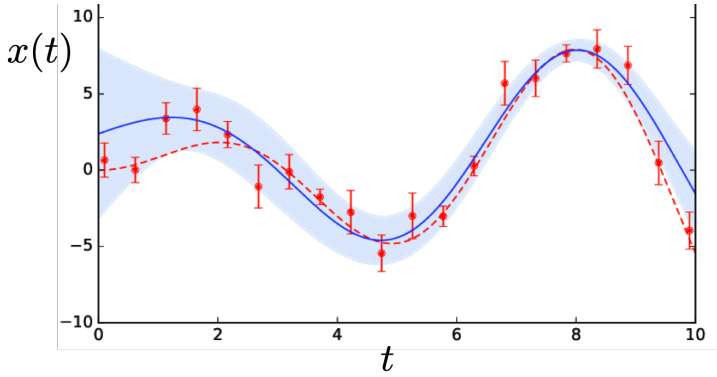
$$\mathbf{x}(t) = \mathbf{\Psi}(t)\mathbf{c}$$



Parametric methods:

(i) hard to decide what basis functions and how many
(ii) don't encode motion priors explicitly
(iii) computations are typically fast

$$\mathbf{x}(t) \sim \mathcal{GP}(\check{\mathbf{x}}(t), \check{\mathbf{P}}(t, t'))$$



Nonparametric methods:

(i) no need to pick number of basis functions
(ii) can encode motion priors explicitly
(iii) can be computationally expensive

- two main camps: parametric (e.g., spline) and nonparametric (e.g., Gaussian process)
- we will explore the latter in this lecture

System

– we define our system using the following linear, time-varying models:

$$\text{motion model:} \quad \dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{v}(t) + \mathbf{L}(t)\mathbf{w}(t) \tag{146a}$$

$$\text{observation model:} \quad \mathbf{y}(t_k) = \mathbf{C}(t_k)\mathbf{x}(t_k) + \mathbf{n}(t_k), \quad k = 0 \dots K \tag{146b}$$

where k is again the discrete-time index and K its maximum

– the variables have the following meanings:

$$\begin{aligned}
\text{system state :} \quad & \mathbf{x}(t) \in \mathbb{R}^N \\
\text{initial state :} \quad & \mathbf{x}_0 \in \mathbb{R}^N \sim \mathcal{N}\left(\check{\mathbf{x}}_0, \check{\mathbf{P}}_0\right) \\
\text{input :} \quad & \mathbf{v}(t) \in \mathbb{R}^N \\
\text{process noise :} \quad & \mathbf{w}(t) \in \mathbb{R}^N \sim \mathcal{GP}\left(\mathbf{0}, \boldsymbol{Q}\delta(t - t')\right) \\
\text{measurement :} \quad & \mathbf{y}(t_k) \in \mathbb{R}^M \\
\text{measurement noise :} \quad & \mathbf{n}(t_k) \in \mathbb{R}^M \sim \mathcal{N}\left(\mathbf{0}, \mathbf{R}(t_k)\right)
\end{aligned}$$

Motion prior

– to connect to Gaussian process regression, we will convert our motion model

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{v}(t) + \mathbf{L}(t)\mathbf{w}(t) \tag{147}$$

into a kernel function

– after stochastic integration we have a GP motion prior

$$\mathbf{x}(t) \sim \mathcal{GP}\Bigg(\underbrace{\boldsymbol{\Phi}(t,t_0)\check{\mathbf{x}}_0 + \int_{t_0}^{t} \boldsymbol{\Phi}(t,s)\mathbf{v}(s)\,ds}_{\check{\mathbf{x}}(t)}, \underbrace{\boldsymbol{\Phi}(t,t_0)\check{\mathbf{P}}_0\boldsymbol{\Phi}(t',t_0)^T + \int_{t_0}^{\min(t,t')} \boldsymbol{\Phi}(t,s)\mathbf{L}(s)\boldsymbol{Q}\mathbf{L}(s)^T\boldsymbol{\Phi}(t',s)^T\,ds}_{\check{\mathbf{P}}(t,t')}\Bigg) \tag{148}$$

where $\boldsymbol{\Phi}(t,s)$ is known as the transition function

Motion prior

– at the measurement times $t_0 < t_1 < \cdots < t_K$ we have

$$\mathbf{x} \sim \mathcal{N}(\check{\mathbf{x}}, \check{\mathbf{P}}) = \mathcal{N}\Big(\mathbf{A}\mathbf{v},\ \underbrace{\mathbf{A}\mathbf{Q}\mathbf{A}^T}_{\text{kernel matrix}}\Big) \tag{149}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{1} & & & & & \\ \boldsymbol{\Phi}(t_1,t_0) & \mathbf{1} & & & & \\ \boldsymbol{\Phi}(t_2,t_0) & \boldsymbol{\Phi}(t_2,t_1) & \mathbf{1} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ \boldsymbol{\Phi}(t_{K-1},t_0) & \boldsymbol{\Phi}(t_{K-1},t_1) & \boldsymbol{\Phi}(t_{K-1},t_2) & \cdots & \mathbf{1} & \\ \boldsymbol{\Phi}(t_K,t_0) & \boldsymbol{\Phi}(t_K,t_1) & \boldsymbol{\Phi}(t_K,t_2) & \cdots & \boldsymbol{\Phi}(t_K,t_{K-1}) & \mathbf{1} \end{bmatrix},$$

$$\mathbf{v} = \begin{bmatrix} \check{\mathbf{x}}_0 \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_K \end{bmatrix}, \quad \mathbf{v}_k = \int_{t_{k-1}}^{t_k} \boldsymbol{\Phi}(t_k,s)\mathbf{v}(s)\,ds, \qquad k = 1 \ldots K,$$

$$\mathbf{Q} = \text{diag}\left(\check{\mathbf{P}}_0, \mathbf{Q}_1, \mathbf{Q}_2, \ldots, \mathbf{Q}_K\right), \quad \mathbf{Q}_k = \int_{t_{k-1}}^{t_k} \boldsymbol{\Phi}(t_k,s)\mathbf{L}(s)\boldsymbol{Q}\mathbf{L}(s)^T\boldsymbol{\Phi}(t_k,s)^T\,ds, \quad k = 1 \ldots K$$

– this can often be precalculated analytically

Estimate at measurement times

– if we only want to solve at the measurement times, things are basically identical to the linear-Gaussian case:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \tfrac{1}{2}(\underbrace{\mathbf{A}\mathbf{v}}_{\check{\mathbf{x}}} - \mathbf{x})^T \underbrace{\mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{A}^{-1}}_{\check{\mathbf{P}}^{-1}}(\underbrace{\mathbf{A}\mathbf{v}}_{\check{\mathbf{x}}} - \mathbf{x}) + \tfrac{1}{2}(\mathbf{y} - \mathbf{C}\mathbf{x})^T \mathbf{R}^{-1}(\mathbf{y} - \mathbf{C}\mathbf{x}) \tag{150}$$

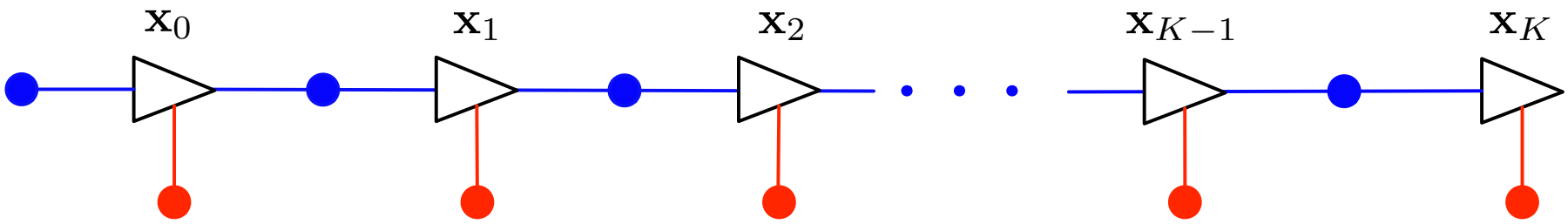
$$\Rightarrow \quad \underbrace{\left(\mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{A}^{-1} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C}\right)}_{\text{block-tridiagonal}} \hat{\mathbf{x}} = \mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{v} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{y} \tag{151}$$

where

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{x}}(t_0) \\ \vdots \\ \hat{\mathbf{x}}(t_K) \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} \mathbf{y}(t_0) \\ \vdots \\ \mathbf{y}(t_K) \end{bmatrix}, \quad \mathbf{C} = \text{diag}\left(\mathbf{C}(t_0), \ldots, \mathbf{C}(t_K)\right), \quad \mathbf{R} = \text{diag}\left(\mathbf{R}(t_0), \ldots, \mathbf{R}(t_K)\right)$$

– only now our motion prior has been constructed from a stochastic differential equation

Still a sparse factor graph



motion prior measurements

$$\underbrace{\left(\mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{A}^{-1} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C}\right)}_{\hat{\mathbf{P}}^{-1}} \hat{\mathbf{x}} = \mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{v} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{y}$$

block-tridiagonal

Querying at other times

- suppose that we'd like to query the trajectory as some other times $\tau_0 < \tau_1 < \ldots < \tau_J$
- the joint density between the state (at the query times) and the measurements (at the measurement times) is written as

$$p\left(\begin{bmatrix}\mathbf{x}_\tau \\ \mathbf{y}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\check{\mathbf{x}}_\tau \\ \mathbf{C}\check{\mathbf{x}}\end{bmatrix}, \begin{bmatrix}\check{\mathbf{P}}_{\tau\tau} & \check{\mathbf{P}}_\tau \mathbf{C}^T \\ \mathbf{C}\check{\mathbf{P}}_\tau^T & \mathbf{R} + \mathbf{C}\check{\mathbf{P}}\mathbf{C}^T\end{bmatrix}\right) \tag{152}$$

where

$$\check{\mathbf{x}} = \begin{bmatrix}\check{\mathbf{x}}(t_0) \\ \vdots \\ \check{\mathbf{x}}(t_K)\end{bmatrix}, \quad \mathbf{x}_\tau = \begin{bmatrix}\mathbf{x}(\tau_0) \\ \vdots \\ \mathbf{x}(\tau_J)\end{bmatrix}, \quad \check{\mathbf{x}}_\tau = \begin{bmatrix}\check{\mathbf{x}}(\tau_0) \\ \vdots \\ \check{\mathbf{x}}(\tau_J)\end{bmatrix}, \quad \check{\mathbf{P}}_\tau = \left[\check{\mathbf{P}}(\tau_i, t_j)\right]_{ij}, \quad \check{\mathbf{P}}_{\tau\tau} = \left[\check{\mathbf{P}}(\tau_i, \tau_j)\right]_{ij}$$

- using the Gaussian conditioning formula we have

$$p(\mathbf{x}_\tau|\mathbf{y}) = \mathcal{N}\Big(\underbrace{\check{\mathbf{x}}_\tau + \check{\mathbf{P}}_\tau \mathbf{C}^T(\mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R})^{-1}(\mathbf{y} - \mathbf{C}\check{\mathbf{x}})}_{\hat{\mathbf{x}}_\tau,\ \text{mean}}, \underbrace{\check{\mathbf{P}}_{\tau\tau} - \check{\mathbf{P}}_\tau \mathbf{C}^T(\mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R})^{-1}\mathbf{C}\check{\mathbf{P}}_\tau^T}_{\hat{\mathbf{P}}_{\tau\tau},\ \text{covariance}}\Big) \tag{153}$$

Querying at other times

- since we have already solved at the measurement times, we can use $\hat{\mathbf{x}}$ and $\hat{\mathbf{P}}$ to simplify the solution at the query times:

$$\hat{\mathbf{x}}_\tau = \check{\mathbf{x}}_\tau + \left(\check{\mathbf{P}}_\tau \check{\mathbf{P}}^{-1}\right)\left(\hat{\mathbf{x}} - \check{\mathbf{x}}\right) \tag{154a}$$

$$\hat{\mathbf{P}}_{\tau\tau} = \check{\mathbf{P}}_{\tau\tau} + \left(\check{\mathbf{P}}_\tau \check{\mathbf{P}}^{-1}\right)\left(\hat{\mathbf{P}} - \check{\mathbf{P}}\right)\left(\check{\mathbf{P}}_\tau \check{\mathbf{P}}^{-1}\right)^T \tag{154b}$$

  which are the standard GP interpolation equations

- our inverse kernel matrix, $\check{\mathbf{P}}^{-1}$, is block-tridiagonal, so interpolation is extremely efficient compared to generic GP regression

Single query is $O(1)$

- consider a single query time, $t_k \leq \tau < t_{k+1}$
- we can show that

$$\hat{\mathbf{x}}(\tau) = \check{\mathbf{x}}(\tau) + \begin{bmatrix} \boldsymbol{\Lambda}(\tau) & \boldsymbol{\Psi}(\tau) \end{bmatrix} \left(\begin{bmatrix} \hat{\mathbf{x}}_k \\ \hat{\mathbf{x}}_{k+1} \end{bmatrix} - \begin{bmatrix} \check{\mathbf{x}}(t_k) \\ \check{\mathbf{x}}(t_{k+1}) \end{bmatrix} \right) \tag{155a}$$

$$\hat{\mathbf{P}}(\tau,\tau) = \check{\mathbf{P}}(\tau,\tau) + \begin{bmatrix} \boldsymbol{\Lambda}(\tau) & \boldsymbol{\Psi}(\tau) \end{bmatrix} \left(\begin{bmatrix} \hat{\mathbf{P}}_{k,k} & \hat{\mathbf{P}}_{k,k+1} \\ \hat{\mathbf{P}}_{k+1,k} & \hat{\mathbf{P}}_{k+1,k+1} \end{bmatrix} - \begin{bmatrix} \check{\mathbf{P}}(t_k,t_k) & \check{\mathbf{P}}(t_k,t_{k+1}) \\ \check{\mathbf{P}}(t_{k+1},t_k) & \check{\mathbf{P}}(t_{k+1},t_{k+1}) \end{bmatrix} \right) \begin{bmatrix} \boldsymbol{\Lambda}(\tau)^T \\ \boldsymbol{\Psi}(\tau)^T \end{bmatrix} \tag{155b}$$

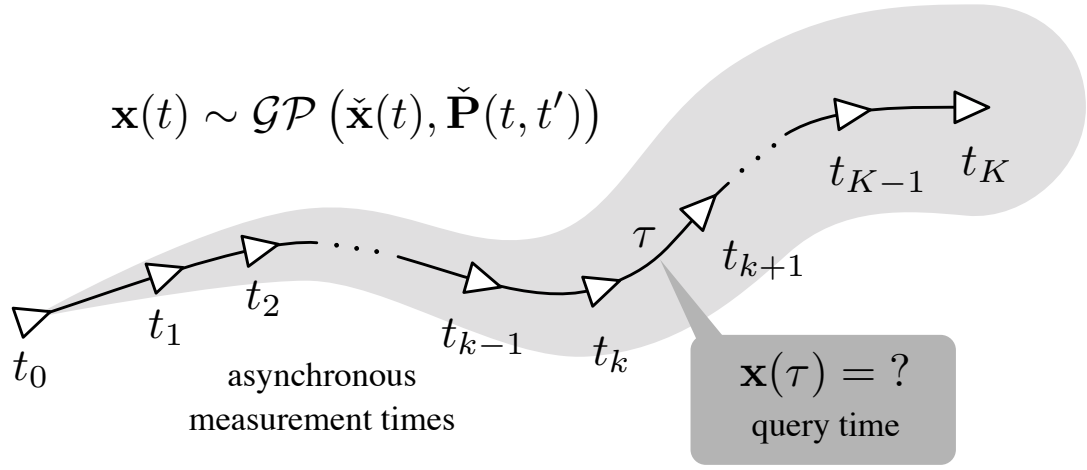
where

$$\boldsymbol{\Lambda}(\tau) = \boldsymbol{\Phi}(\tau,t_k) - \mathbf{Q}_\tau \boldsymbol{\Phi}(t_{k+1},\tau)^T \mathbf{Q}_{k+1}^{-1} \boldsymbol{\Phi}(t_{k+1},t_k), \quad \boldsymbol{\Psi}(\tau) = \mathbf{Q}_\tau \boldsymbol{\Phi}(t_{k+1},\tau)^T \mathbf{Q}_{k+1}^{-1}, \tag{156}$$

$$\mathbf{Q}_\tau = \int_{t_k}^{\tau} \boldsymbol{\Phi}(\tau,s) \mathbf{L}(s) \boldsymbol{Q} \mathbf{L}(s)^T \boldsymbol{\Phi}(\tau,s)^T \, ds \tag{157}$$

- in other words, we only need to use the posterior solution at the two measurement times that bracket the query time

GP regression summary



- solve at the measurement times $t_0 < t_1 < \cdots < t_K$ in $O(K)$ time
- solve at each query time, τ, in $O(1)$ time
- principled approach to interpolation since we're using the motion model not some other scheme
- we can also use this GP interpolation to reduce the number of state variables while accounting for all measurement timestamps – GP inducing points

Example: white-noise on acceleration prior

- consider the case $\ddot{\mathbf{p}}(t) = \mathbf{w}(t)$, where $\mathbf{p}(t)$ corresponds to position and $\mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \boldsymbol{Q}\,\delta(t - t'))$ is white noise as before
- we can cast this in the form

$$\dot{\mathbf{x}}(t) = \boldsymbol{A}\mathbf{x}(t) + \mathbf{v}(t) + \boldsymbol{L}\mathbf{w}(t) \tag{158}$$

by taking

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{p}(t) \\ \dot{\mathbf{p}}(t) \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \mathbf{v}(t) = \mathbf{0}, \quad \boldsymbol{L} = \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix} \tag{159}$$

- importantly, the Markovian state is now both position and velocity
- in the absence of measurements to the contrary, this motion prior tries to keep the velocity constant

Example: white-noise on acceleration prior

– the transition function for this linear, time-invariant example is

$$\mathbf{\Phi}(t_k, t_{k-1}) = \begin{bmatrix} \mathbf{1} & \Delta t_{k:k-1}\mathbf{1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{160}$$

where $\Delta t_{k:k-1} = t_k - t_{k-1}$

– since there is no input, $\mathbf{v}(t) = \mathbf{0}$, the prior becomes

$$\check{\mathbf{x}} = \mathbf{A}\mathbf{v}, \quad \check{\mathbf{P}} = \mathbf{A}\mathbf{Q}\mathbf{A}^T, \tag{161}$$

$$\mathbf{v} = \begin{bmatrix} \check{\mathbf{x}}_0 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \quad \mathbf{Q} = \mathsf{diag}(\check{\mathbf{P}}_0, \mathbf{Q}_1, \ldots, \mathbf{Q}_K), \quad \mathbf{Q}_k = \begin{bmatrix} \frac{1}{3}\Delta t_{k:k-1}^3 \boldsymbol{Q} & \frac{1}{2}\Delta t_{k:k-1}^2 \boldsymbol{Q} \\ \frac{1}{2}\Delta t_{k:k-1}^2 \boldsymbol{Q} & \Delta t_{k:k-1} \boldsymbol{Q} \end{bmatrix} \tag{162}$$

Example: white-noise on acceleration prior

- after solving at the measurement times, the interpolation equations for $t_k \leq \tau < t_{k+1}$ become

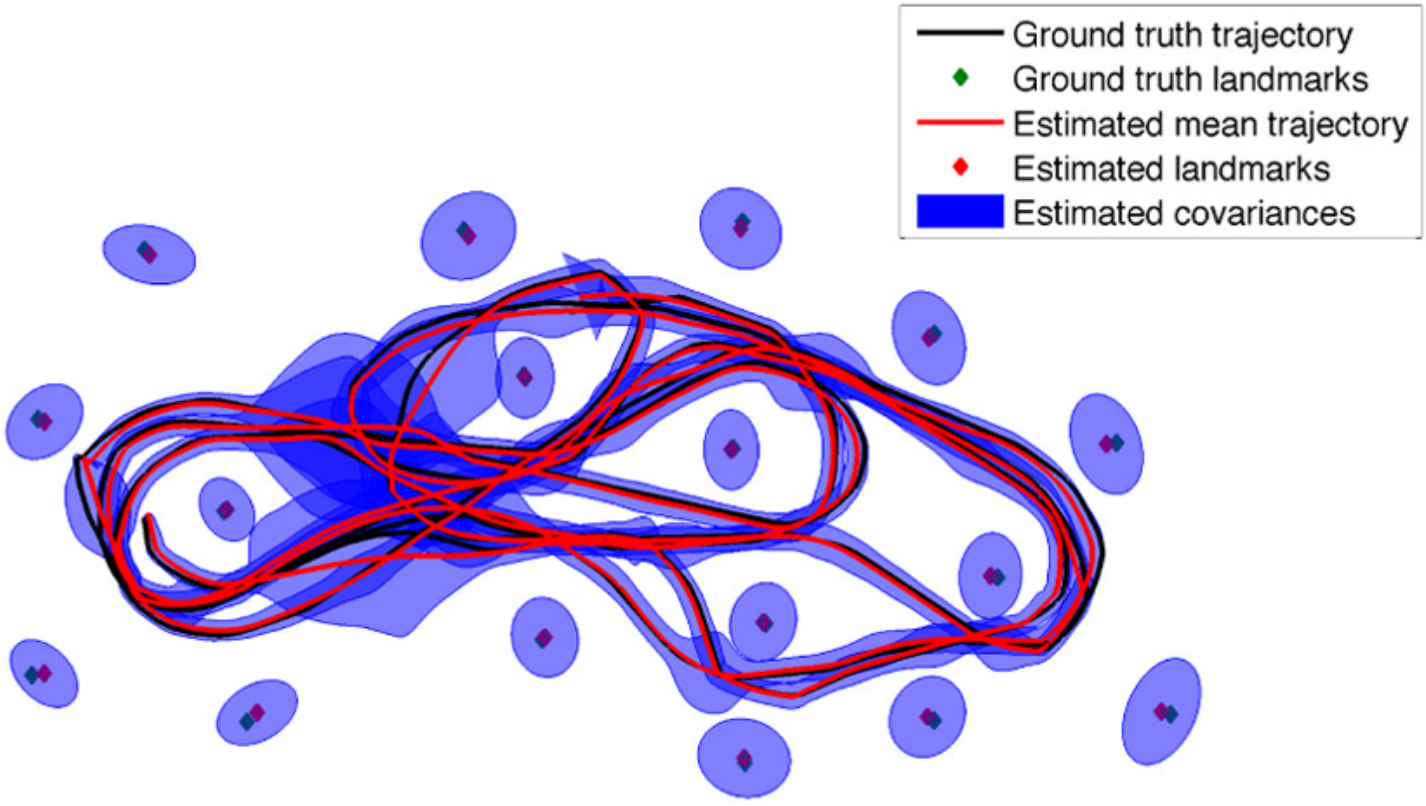
$$\hat{\mathbf{x}}_\tau = \check{\mathbf{x}}_\tau + \begin{bmatrix} (1 - 3\alpha^2 + 2\alpha^3)\mathbf{1} & T(\alpha - 2\alpha^2 + \alpha^3)\mathbf{1} \\ \frac{1}{T}6(-\alpha + \alpha^2)\mathbf{1} & (1 - 4\alpha + 3\alpha^2)\mathbf{1} \end{bmatrix} (\hat{\mathbf{x}}_k - \check{\mathbf{x}}_k) + \begin{bmatrix} (3\alpha^2 - 2\alpha^3)\mathbf{1} & T(-\alpha^2 + \alpha^3)\mathbf{1} \\ \frac{1}{T}6(\alpha - \alpha^2)\mathbf{1} & (-2\alpha + 3\alpha^2)\mathbf{1} \end{bmatrix} (\hat{\mathbf{x}}_{k+1} - \check{\mathbf{x}}_{k+1}) \quad (163)$$

  where

$$\check{\mathbf{x}}_\tau = \boldsymbol{\Phi}(\tau, t_k)\check{\mathbf{x}}_k, \quad \alpha = \frac{\tau - t_k}{t_{k+1} - t_k} \in [0, 1], \quad T = \Delta t_{k+1:k} = t_{k+1} - t_k$$

- remarkably, the top row (corresponding to position) is precisely a cubic Hermite polynomial interpolation – bottom row (corresponding to velocity) is derivative of this
- this falls out of the math for free simply by picking the WNOA motion prior

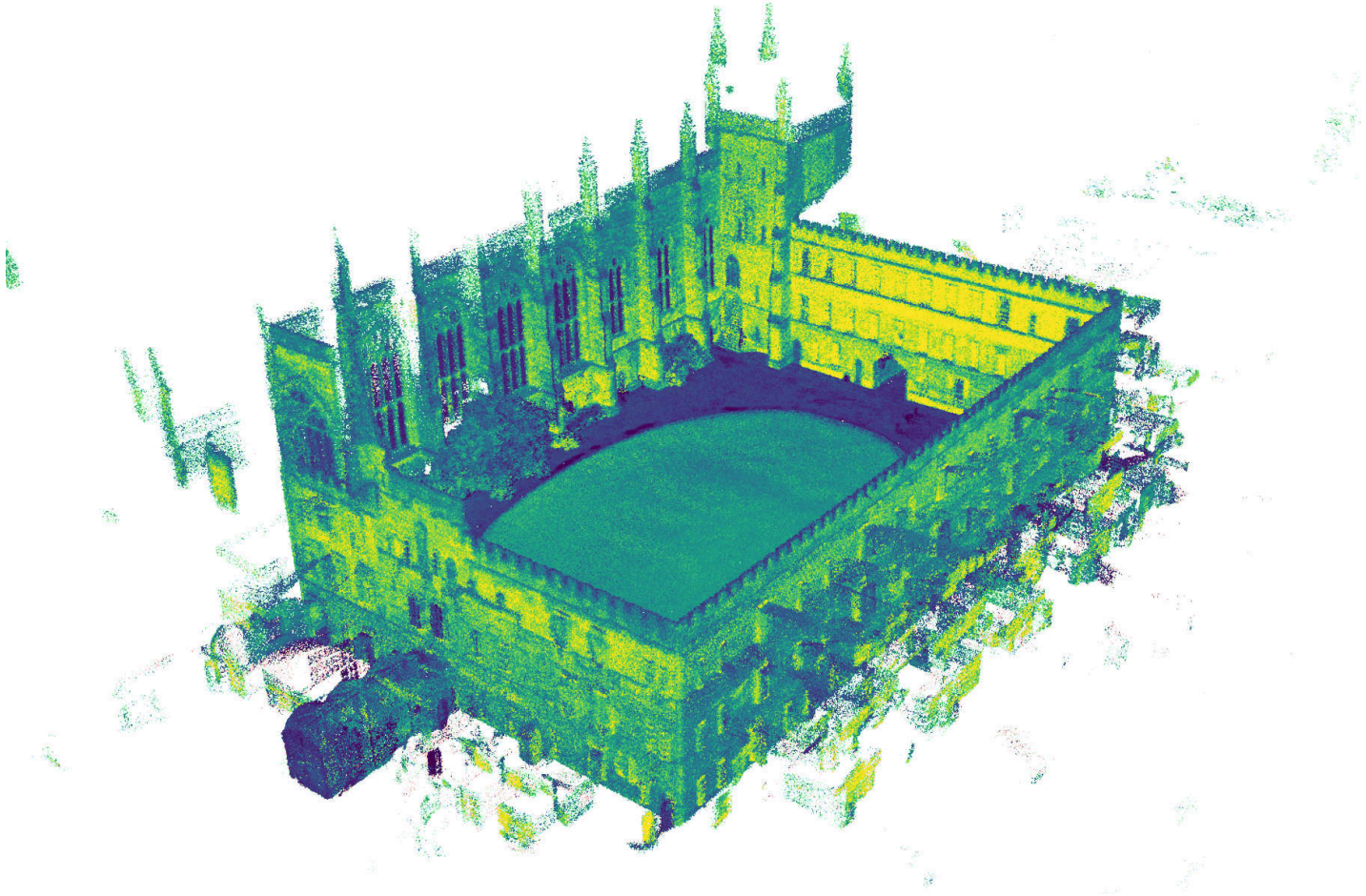
Example: white-noise on acceleration prior



– example continuous-time estimation using only the WNOA motion prior plus landmark measurements – no odometry!

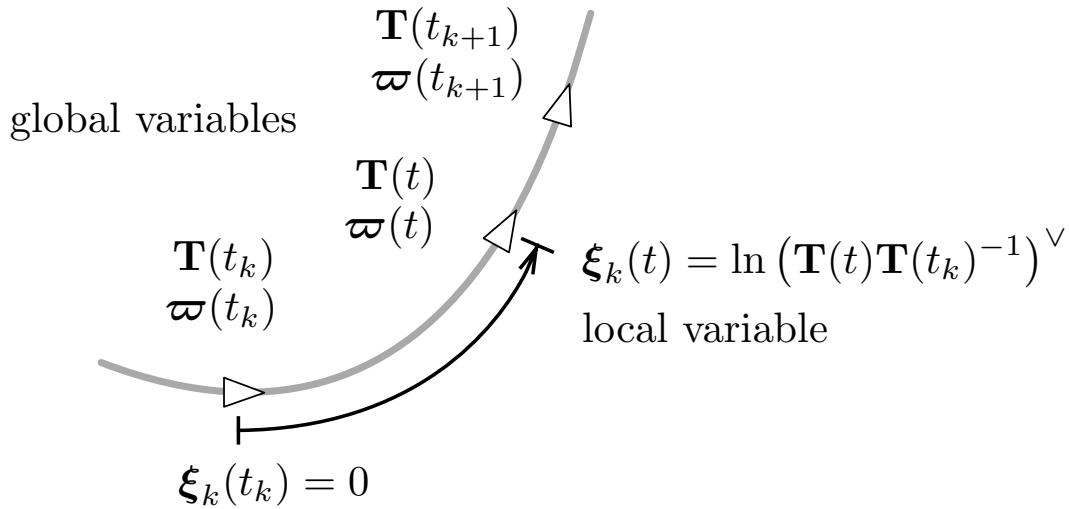
What about 3D?

– we would like to do continuous-time estimation with Lie-group state variables so we can estimate 3D trajectories



Special Euclidean group

– in order to recycle our linear continuous-time tools, we will stitch together many local GPs



– we introduce local variables, $\boldsymbol{\xi}_k(t)$, in the Lie algebra of $SE(3)$ and then define a WNOA motion prior over these

$$\ddot{\boldsymbol{\xi}}_k(t) = \mathbf{w}_k(t), \quad \mathbf{w}_k(t) \sim \mathcal{GP}(\mathbf{0}, \boldsymbol{Q}(t - t')) \tag{164}$$

Motion prior in Lie algebra

– we reorganize the WNOA motion prior into the requisite form

$$\frac{d}{dt}\begin{bmatrix}\boldsymbol{\xi}_k(t)\\ \boldsymbol{\psi}_k(t)\end{bmatrix} = \begin{bmatrix}\mathbf{0} & \mathbf{1}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\underbrace{\begin{bmatrix}\boldsymbol{\xi}_k(t)\\ \boldsymbol{\psi}_k(t)\end{bmatrix}}_{\boldsymbol{\gamma}_k(t)} + \begin{bmatrix}\mathbf{0}\\ \mathbf{1}\end{bmatrix}\mathbf{w}_k(t) \tag{165}$$

– we integrate this stochastically to obtain our kernel function

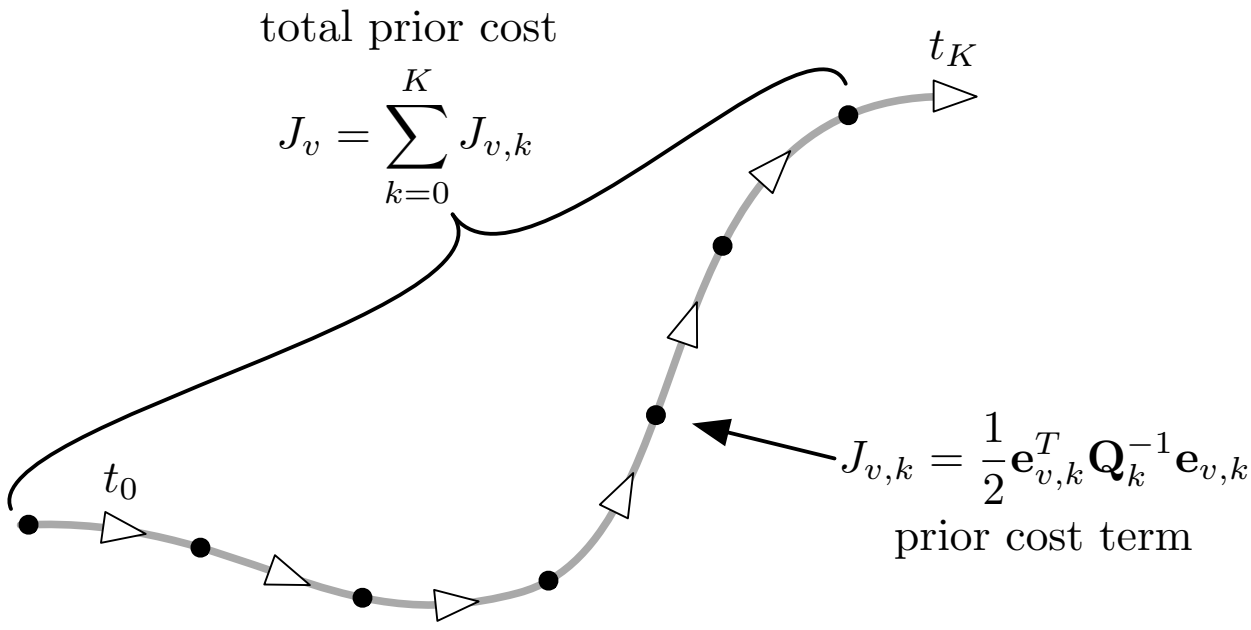
$$\boldsymbol{\gamma}_k(t) \sim \mathcal{GP}\Big(\underbrace{\boldsymbol{\Phi}(t,t_k)\check{\boldsymbol{\gamma}}_k(t_k)}_{\text{mean function}}, \underbrace{\boldsymbol{\Phi}(t,t_k)\check{\mathbf{P}}(t_k)\boldsymbol{\Phi}(t,t_k)^T + \mathbf{Q}(t-t_k)}_{\text{covariance function}}\Big) \tag{166}$$

where

$$\boldsymbol{\Phi}(t,t') = \begin{bmatrix}\mathbf{1} & (t-t')\mathbf{1}\\ \mathbf{0} & \mathbf{1}\end{bmatrix}, \quad \mathbf{Q}(t-t') = \begin{bmatrix}\frac{1}{3}(t-t')^3\boldsymbol{Q} & \frac{1}{2}(t-t')^2\boldsymbol{Q}\\ \frac{1}{2}(t-t')^2\boldsymbol{Q} & (t-t')\boldsymbol{Q}\end{bmatrix}, \quad t \geq t' \tag{167}$$

and $\check{\boldsymbol{\gamma}}_k(t_k)$ and $\check{\mathbf{P}}(t_k)$ are the initial mean and covariance at $t = t_k$

Motion prior in Lie algebra

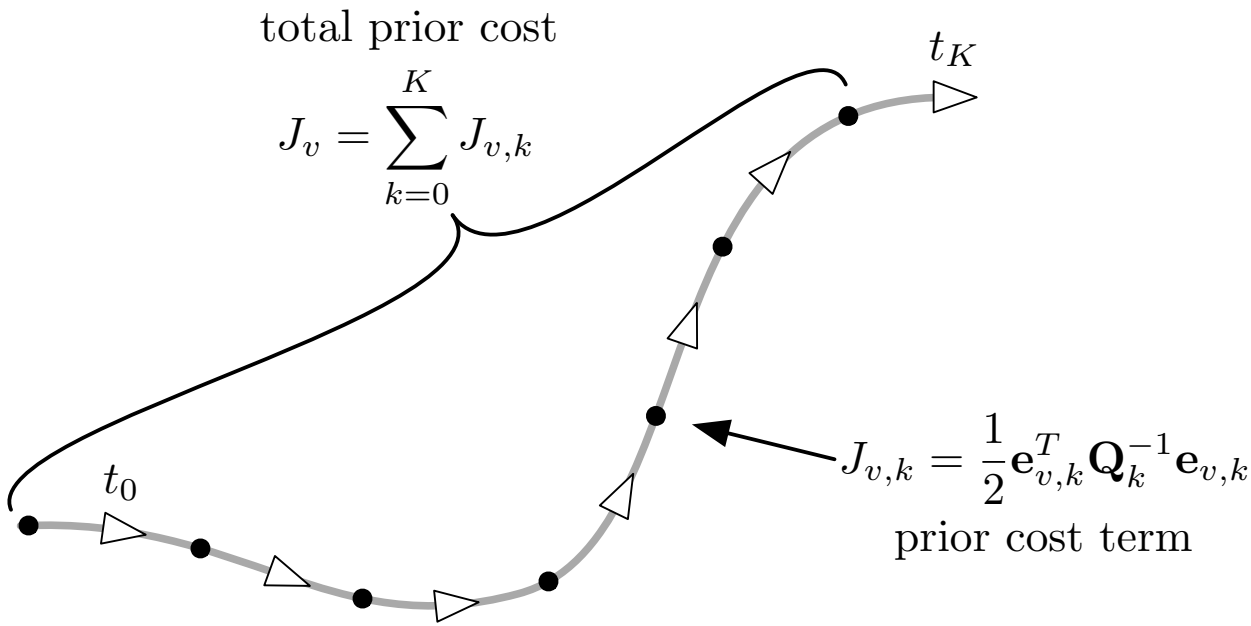


– build errors and assemble them into MAP cost terms

$$\mathbf{e}_{v,k} = \begin{cases} \begin{bmatrix} -\ln\left(\mathbf{T}(t_0)\check{\mathbf{T}}_0^{-1}\right)^\vee \\ \check{\boldsymbol{\varpi}}_0 - \boldsymbol{\varpi}(t_0) \end{bmatrix} & k = 0 \\ \boldsymbol{\Phi}(t_k, t_{k-1})\left(\boldsymbol{\gamma}_{k-1}(t_{k-1}) - \check{\boldsymbol{\gamma}}_{k-1}(t_{k-1})\right) & \\ \qquad - \left(\boldsymbol{\gamma}_{k-1}(t_k) - \check{\boldsymbol{\gamma}}_{k-1}(t_k)\right) & k > 0 \end{cases} \tag{168}$$

– note $\mathbf{Q}_0 = \check{\mathbf{P}}_0$ and $\mathbf{Q}_k = \mathbf{Q}(t_k - t_{k-1})$ for $k = 1 \ldots K$

Motion prior in Lie algebra



– we back-substitute for the errors in terms of the global variables

$$\mathbf{e}_{v,k} = \begin{cases} \begin{bmatrix} -\ln\left(\mathbf{T}(t_0)\check{\mathbf{T}}_0^{-1}\right)^\vee \\ \check{\boldsymbol{\varpi}}_0 - \boldsymbol{\varpi}(t_0) \end{bmatrix} & k = 0 \\ \begin{bmatrix} (t_k - t_{k-1})\,\boldsymbol{\varpi}(t_{k-1}) - \ln\left(\mathbf{T}(t_k)\mathbf{T}(t_{k-1})^{-1}\right)^\vee \\ \boldsymbol{\varpi}(t_{k-1}) - \boldsymbol{\mathcal{J}}\left(\ln\left(\mathbf{T}(t_k)\mathbf{T}(t_{k-1})^{-1}\right)^\vee\right)^{-1}\boldsymbol{\varpi}(t_k) \end{bmatrix} & k > 0 \end{cases} \tag{169}$$

Lie group aside

- we need a few more Lie-group tools at this point
- the $SE(3)$ Jacobian, $\boldsymbol{\mathcal{J}} \in \mathbb{R}^{6\times 6}$, allows us to make the following approximation when $\boldsymbol{\xi}_1$ small

$$\exp\left(\boldsymbol{\xi}_1^{\wedge}\right)\exp\left(\boldsymbol{\xi}_2^{\wedge}\right) \approx \exp\left(\boldsymbol{\mathcal{J}}(\boldsymbol{\xi}_2)^{-1}\,\boldsymbol{\xi}_1 + \boldsymbol{\xi}_2\right) \tag{170}$$

- the adjoint of a pose, $\boldsymbol{\mathcal{T}} = \mathrm{Ad}(\mathbf{T}) \in \mathbb{R}^{6\times 6}$, is an inner automorphism operator that allows

$$(\boldsymbol{\mathcal{T}}\mathbf{x})^{\wedge} = \mathbf{T}\mathbf{x}^{\wedge}\mathbf{T}^{-1} \tag{171}$$

Motion prior in Lie algebra

– with these Lie group tools we can perturb our to-be-estimated state variables according to

$$\mathbf{T}_k = \exp(\boldsymbol{\epsilon}_k^{\wedge})\mathbf{T}_{\mathrm{op},k}, \quad \boldsymbol{\varpi}_k = \boldsymbol{\varpi}_{\mathrm{op},k} + \boldsymbol{\eta}_k \tag{172}$$

– we then linearize our motion prior errors terms so that

$$\mathbf{e}_{v,k}(\mathbf{x}) \approx \mathbf{e}_{v,k}(\mathbf{x}_{\mathrm{op}}) + \mathbf{F}_{k-1}\boldsymbol{\varepsilon}_{k-1} - \mathbf{E}_k\boldsymbol{\varepsilon}_k \tag{173}$$

where $\boldsymbol{\varepsilon}_k = \begin{bmatrix} \boldsymbol{\epsilon}_k \\ \boldsymbol{\eta}_k \end{bmatrix}$ and

$$\mathbf{F}_{k-1} = \begin{cases} \mathbf{0} & k = 0 \\ \begin{bmatrix} \boldsymbol{\mathcal{J}}_{\mathrm{op},k,k-1}^{-1}\boldsymbol{\mathcal{T}}_{\mathrm{op},k,k-1} & (t_k - t_{k-1})\mathbf{1} \\ \frac{1}{2}\boldsymbol{\varpi}_{\mathrm{op},k}^{\curlywedge}\boldsymbol{\mathcal{J}}_{\mathrm{op},k,k-1}^{-1}\boldsymbol{\mathcal{T}}_{\mathrm{op},k,k-1} & \mathbf{1} \end{bmatrix} & k > 0 \end{cases}, \quad \mathbf{E}_k = \begin{cases} \begin{bmatrix} \boldsymbol{\mathcal{J}}\left(\ln\left(\mathbf{T}_{\mathrm{op},0}\check{\mathbf{T}}_0^{-1}\right)^{\vee}\right)^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} & k = 0 \\ \begin{bmatrix} \boldsymbol{\mathcal{J}}_{\mathrm{op},k,k-1}^{-1} & \mathbf{0} \\ \frac{1}{2}\boldsymbol{\varpi}_{\mathrm{op},k}^{\curlywedge}\boldsymbol{\mathcal{J}}_{\mathrm{op},k,k-1}^{-1} & \boldsymbol{\mathcal{J}}_{\mathrm{op},k,k-1}^{-1} \end{bmatrix} & k > 0 \end{cases}$$

Motion prior in Lie algebra

– our quadratic approximation of the motion prior term at our current guess is therefore

$$J_v \approx \frac{1}{2}\left(\mathbf{e}_v(\mathbf{x}_{\text{op}}) - \mathbf{F}^{-1}\,\delta\mathbf{x}_1\right)^T \mathbf{Q}^{-1}\left(\mathbf{e}_v(\mathbf{x}_{\text{op}}) - \mathbf{F}^{-1}\,\delta\mathbf{x}_1\right) \tag{174}$$

where

$$\mathbf{e}_v(\mathbf{x}_{\text{op}}) = \begin{bmatrix} \mathbf{e}_{v,0}(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_{v,1}(\mathbf{x}_{\text{op}}) \\ \vdots \\ \mathbf{e}_{v,K}(\mathbf{x}_{\text{op}}) \end{bmatrix}, \quad \mathbf{F}^{-1} = \begin{bmatrix} \mathbf{E}_0 & & & & \\ -\mathbf{F}_0 & \mathbf{E}_1 & & & \\ & -\mathbf{F}_1 & \mathbf{E}_2 & & \\ & & \ddots & \ddots & \\ & & & -\mathbf{F}_{K-1} & \mathbf{E}_K \end{bmatrix}, \quad \delta\mathbf{x}_1 = \begin{bmatrix} \boldsymbol{\varepsilon}_0 \\ \boldsymbol{\varepsilon}_1 \\ \boldsymbol{\varepsilon}_2 \\ \vdots \\ \boldsymbol{\varepsilon}_K \end{bmatrix},$$

$$\mathbf{Q} = \text{diag}\left(\check{\mathbf{P}}_0, \mathbf{Q}_1, \ldots, \mathbf{Q}_K\right)$$

– this is essentially all we need to carry out MAP estimation via Gauss-Newton once we incorporate some measurement terms

STEAM

- let's look at simultaneous trajectory estimation and mapping
- the state to be estimated is now

$$\mathbf{x} = \Big\{ \underbrace{\mathbf{T}_0, \boldsymbol{\varpi}_0, \ldots, \mathbf{T}_K, \boldsymbol{\varpi}_K}_{\mathbf{x}_1}, \underbrace{\mathbf{p}_1, \ldots, \mathbf{p}_M}_{\mathbf{x}_2} \Big\} \tag{175}$$

  which includes poses, generalized velocities, and landmarks, $\mathbf{p}_j$
- the measurements at times $t_0, t_1, \ldots, t_K$ are

$$\mathbf{y} = \{\mathbf{y}_{10}, \ldots \mathbf{y}_{M0}, \ldots, \mathbf{y}_{1K}, \ldots, \mathbf{y}_{MK}\} \tag{176}$$

- each measurement is of the form

$$\mathbf{y}_{jk} = \mathbf{s}(\mathbf{T}_k \mathbf{p}_j) + \mathbf{n}_{jk}, \quad \mathbf{n}_{jk} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{jk}) \tag{177}$$

  where $\mathbf{s}(\cdot)$ is a nonlinear sensor model (e.g., a stereo camera)

Landmarks and measurement error term

– our landmarks are in homogeneous coordinates so we perturb according to

$$\mathbf{p}_j = \mathbf{p}_{\mathrm{op},j} + \mathbf{D}\boldsymbol{\zeta}_j, \quad \mathbf{D} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{178}$$

where $\mathbf{p}_{\mathrm{op},j}$ is the operating point and $\boldsymbol{\zeta}_j$ is the perturbation

– the error associated with a measurement is then

$$\mathbf{e}_{y,jk}(\mathbf{x}) = \mathbf{y}_{jk} - \mathbf{s}\left(\mathbf{T}_k \mathbf{p}_j\right) \tag{179}$$

– all the errors can be stacked up into a tall error vector as

$$\mathbf{e}_y(\mathbf{x}_{\mathrm{op}}) = \begin{bmatrix} \mathbf{e}_{y,10}(\mathbf{x}_{\mathrm{op}}) \\ \mathbf{e}_{y,20}(\mathbf{x}_{\mathrm{op}}) \\ \vdots \\ \mathbf{e}_{y,MK}(\mathbf{x}_{\mathrm{op}}) \end{bmatrix} \tag{180}$$

Linearizing $\mathbf{Tp}$

- it is worth a small aside to see how to linearize the product $\mathbf{Tp}$
- we insert our various perturbation schemes so that

$$\begin{aligned}
\mathbf{Tp} &= \exp\left(\boldsymbol{\epsilon}^{\wedge}\right)\mathbf{T}_{\mathrm{op}}\left(\mathbf{p}_{\mathrm{op}}+\mathbf{D}\boldsymbol{\zeta}\right) \\
&\approx \left(\mathbf{1}+\boldsymbol{\epsilon}^{\wedge}\right)\mathbf{T}_{\mathrm{op}}\left(\mathbf{p}_{\mathrm{op}}+\mathbf{D}\boldsymbol{\zeta}\right) \\
&\approx \mathbf{T}_{\mathrm{op}}\mathbf{p}_{\mathrm{op}}+\boldsymbol{\epsilon}^{\wedge}\mathbf{T}_{\mathrm{op}}\mathbf{p}_{\mathrm{op}}+\mathbf{T}_{\mathrm{op}}\mathbf{D}\boldsymbol{\zeta}+\underbrace{\boldsymbol{\epsilon}^{\wedge}\mathbf{T}_{\mathrm{op}}\mathbf{D}\boldsymbol{\zeta}}_{\text{very small}} \\
&\approx \mathbf{T}_{\mathrm{op}}\mathbf{p}_{\mathrm{op}}+\left(\mathbf{T}_{\mathrm{op}}\mathbf{p}_{\mathrm{op}}\right)^{\odot}\boldsymbol{\epsilon}+\mathbf{T}_{\mathrm{op}}\mathbf{D}\boldsymbol{\zeta} \\
&= \mathbf{T}_{\mathrm{op}}\mathbf{p}_{\mathrm{op}}+\begin{bmatrix}\left(\mathbf{T}_{\mathrm{op}}\mathbf{p}_{\mathrm{op}}\right)^{\odot} & \mathbf{0} & \mathbf{T}_{\mathrm{op}}\mathbf{D}\end{bmatrix}\begin{bmatrix}\boldsymbol{\epsilon}\\ \boldsymbol{\eta}\\ \boldsymbol{\zeta}\end{bmatrix}
\end{aligned}$$

- then we can use the chain rule to write

$$\boldsymbol{s}\left(\mathbf{Tp}\right)\approx\boldsymbol{s}\left(\mathbf{T}_{\mathrm{op}}\mathbf{p}_{\mathrm{op}}\right)+\mathbf{S}\begin{bmatrix}\left(\mathbf{T}_{\mathrm{op}}\mathbf{p}_{\mathrm{op}}\right)^{\odot} & \mathbf{0} & \mathbf{T}_{\mathrm{op}}\mathbf{D}\end{bmatrix}\begin{bmatrix}\boldsymbol{\epsilon}\\ \boldsymbol{\eta}\\ \boldsymbol{\zeta}\end{bmatrix} \tag{182}$$

with $\mathbf{S}$ the Jacobian of the sensor model, $\mathbf{s}(\cdot)$

Linearized error term

– our linearized error can then be written as

$$\mathbf{e}_y(\mathbf{x}) \approx \mathbf{e}_y(\mathbf{x}_{\text{op}}) - \mathbf{G}_1\,\delta\mathbf{x}_1 - \mathbf{G}_2\,\delta\mathbf{x}_2 \tag{183}$$

where

$$\delta\mathbf{x}_1 = \begin{bmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_K \end{bmatrix}, \quad \delta\mathbf{x}_2 = \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \vdots \\ \zeta_M \end{bmatrix}, \quad \mathbf{G}_1 = \begin{bmatrix} \mathbf{G}_{1,10} & & & & \\ \vdots & & & & \\ \mathbf{G}_{1,M0} & & & & \\ & \mathbf{G}_{1,11} & & & \\ & \vdots & & & \\ & \mathbf{G}_{1,M1} & & & \\ & & \ddots & & \\ & & \ddots & & \\ & & & \mathbf{G}_{1,1K} & \\ & & & \vdots & \\ & & & \mathbf{G}_{1,MK} \end{bmatrix}, \quad \mathbf{G}_2 = \begin{bmatrix} \mathbf{G}_{2,10} & & \\ & \ddots & \\ & & \mathbf{G}_{2,M0} \\ \mathbf{G}_{2,11} & & \\ & \ddots & \\ & & \mathbf{G}_{2,M1} \\ & \vdots & \\ & \vdots & \\ \mathbf{G}_{2,1K} & & \\ & \ddots & \\ & & \mathbf{G}_{2,MK} \end{bmatrix},$$

$$\mathbf{G}_{1,jk} = \begin{bmatrix} \mathbf{S}_{jk}\left(\mathbf{T}_{\text{op},k}\mathbf{p}_{\text{op},j}\right)^{\odot} & \mathbf{0} \end{bmatrix}, \quad \mathbf{G}_{2,jk} = \mathbf{S}_{jk}\mathbf{T}_{\text{op},k}\mathbf{D}, \quad \mathbf{R} = \text{diag}\left(\mathbf{R}_{10}, \mathbf{R}_{20}, \ldots, \mathbf{R}_{MK}\right)$$

– our quadratic approximation of the measurement term at our current guess is therefore

$$J_y(\mathbf{x}) \approx \tfrac{1}{2}\left(\mathbf{e}_y(\mathbf{x}_{\text{op}}) - \mathbf{G}_1\,\delta\mathbf{x}_1 - \mathbf{G}_2\,\delta\mathbf{x}_2\right)^T \mathbf{R}^{-1}\left(\mathbf{e}_y(\mathbf{x}_{\text{op}}) - \mathbf{G}_1\,\delta\mathbf{x}_1 - \mathbf{G}_2\,\delta\mathbf{x}_2\right) \tag{184}$$

MAP cost

- MAP cost is then $J(\mathbf{x}) = J_v(\mathbf{x}) + J_y(\mathbf{x})$, which can be quadratically approximated as

$$J(\mathbf{x}) = J_v(\mathbf{x}) + J_y(\mathbf{x}) \approx J(\mathbf{x}_{\text{op}}) - \mathbf{e}(\mathbf{x}_{\text{op}})^T \mathbf{W}^{-1} \mathbf{H}\, \delta\mathbf{x} + \delta\mathbf{x}^T \mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}\, \delta\mathbf{x} \quad (185)$$

with

$$\mathbf{e}(\mathbf{x}_{\text{op}}) = \begin{bmatrix} \mathbf{e}_v(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_y(\mathbf{x}_{\text{op}}) \end{bmatrix}, \quad \delta\mathbf{x} = \begin{bmatrix} \delta\mathbf{x}_1 \\ \delta\mathbf{x}_2 \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} \mathbf{F}^{-1} & \mathbf{0} \\ \mathbf{G}_1 & \mathbf{G}_2 \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{bmatrix}$$

- the minimizing perturbation, $\delta\mathbf{x}^\star$, is the solution to

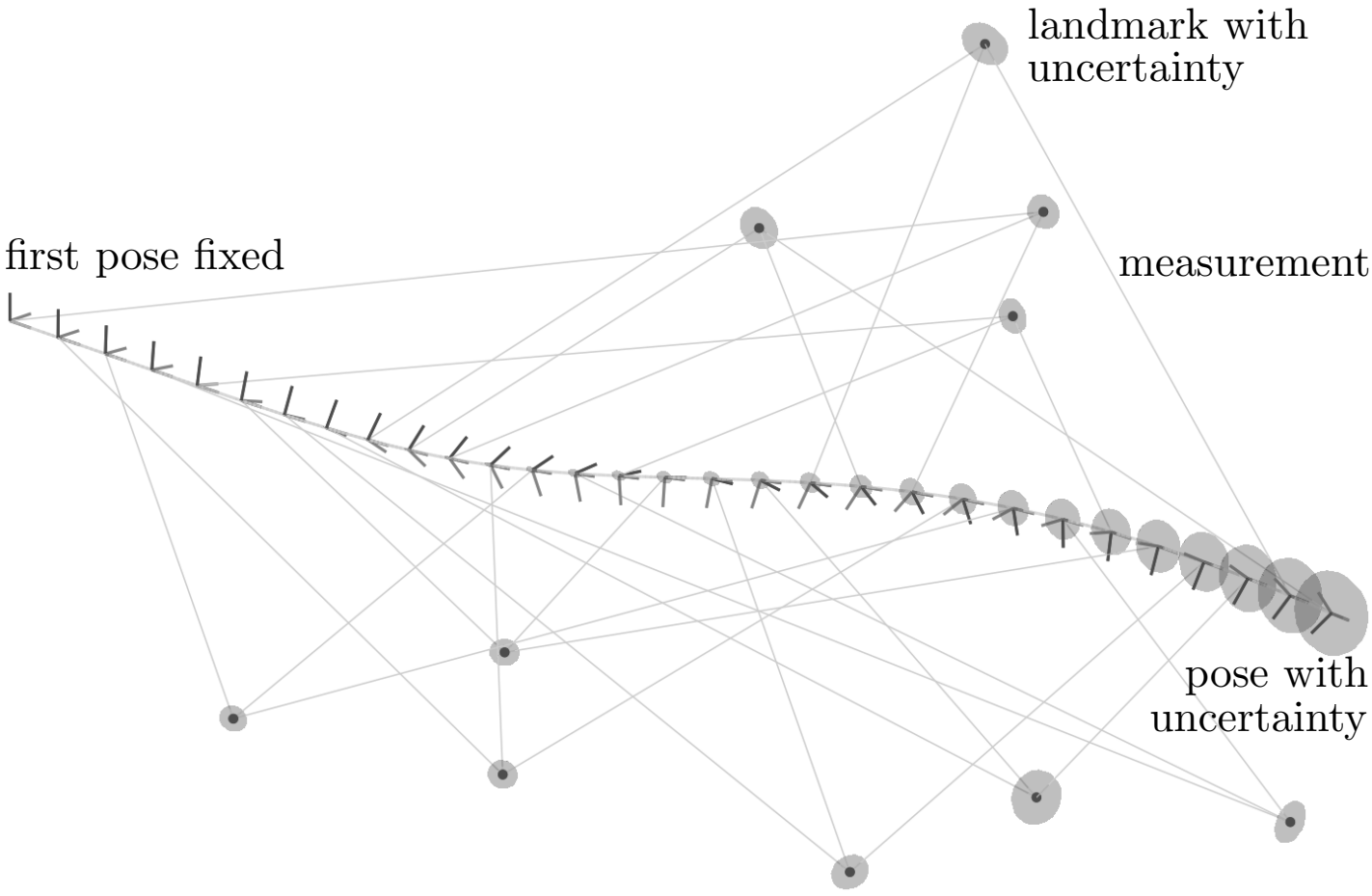
$$\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H}\, \delta\mathbf{x}^\star = \mathbf{H}^T \mathbf{W}^{-1} \mathbf{e}(\mathbf{x}_{\text{op}}) \quad (186)$$

- we solve for $\delta\mathbf{x}^\star$, then apply the optimal perturbations

$$\mathbf{T}_{\text{op},k} \leftarrow \exp\left(\boldsymbol{\epsilon}_k^{\star\wedge}\right) \mathbf{T}_{\text{op},k}, \quad \boldsymbol{\varpi}_{\text{op},k} \leftarrow \boldsymbol{\varpi}_{\text{op},k} + \boldsymbol{\eta}_k^\star, \quad \mathbf{p}_{\text{op},j} \leftarrow \mathbf{p}_{\text{op},j} + \mathbf{D}\boldsymbol{\zeta}_j^\star$$

and iterate to convergence

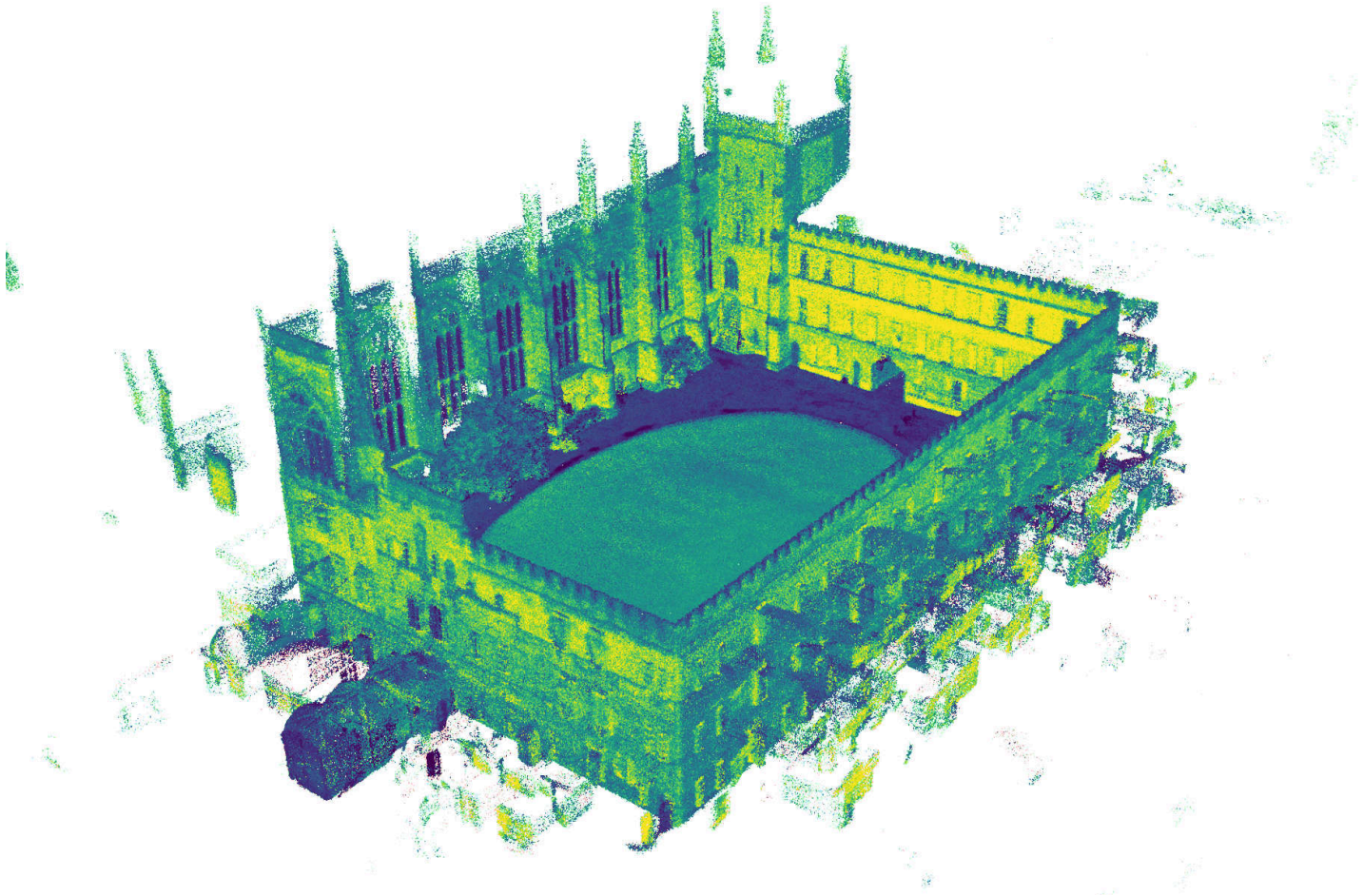
STEAM example



– with only one landmark measurement per pose, the motion prior is necessary to solve

UNIVERSITY OF TORONTO

STEAM example



– these continuous-time methods can be used as a sliding-window filter and run in realtime

STEAM sparsity

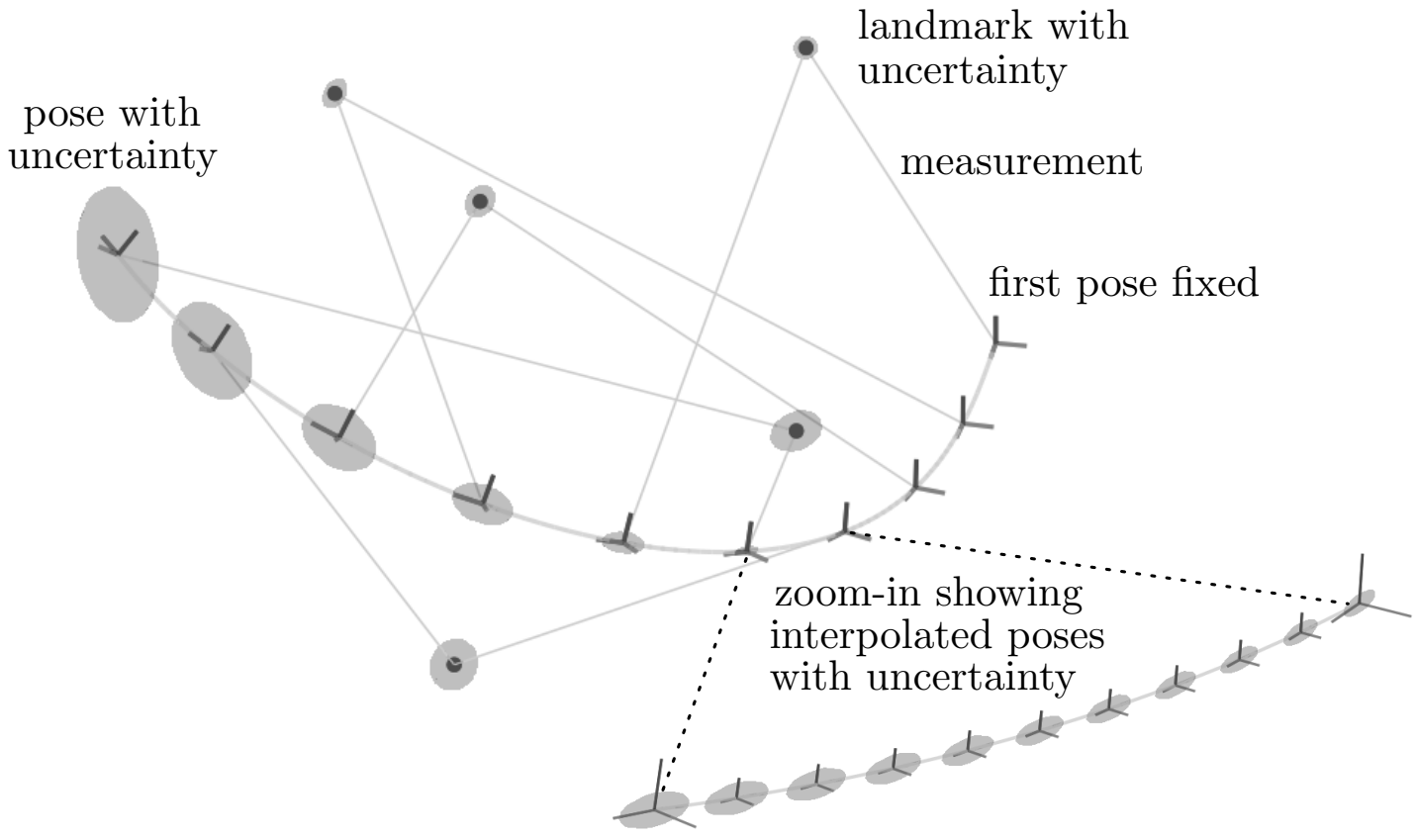
- the left-hand side in our linear system of equations has the form

$$\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H} = \begin{bmatrix} \mathbf{F}^{-T}\mathbf{Q}^{-1}\mathbf{F}^{-1} + \mathbf{G}_1^T\mathbf{R}^{-1}\mathbf{G}_1 & \mathbf{G}_1^T\mathbf{R}^{-1}\mathbf{G}_2 \\ \mathbf{G}_2^T\mathbf{R}^{-1}\mathbf{G}_1 & \mathbf{G}_2^T\mathbf{R}^{-1}\mathbf{G}_2 \end{bmatrix} \tag{187}$$

- this preserves the usual SLAM arrowhead form
  - the top-left block is block-tridiagonal
  - the bottom-right block is block-diagonal

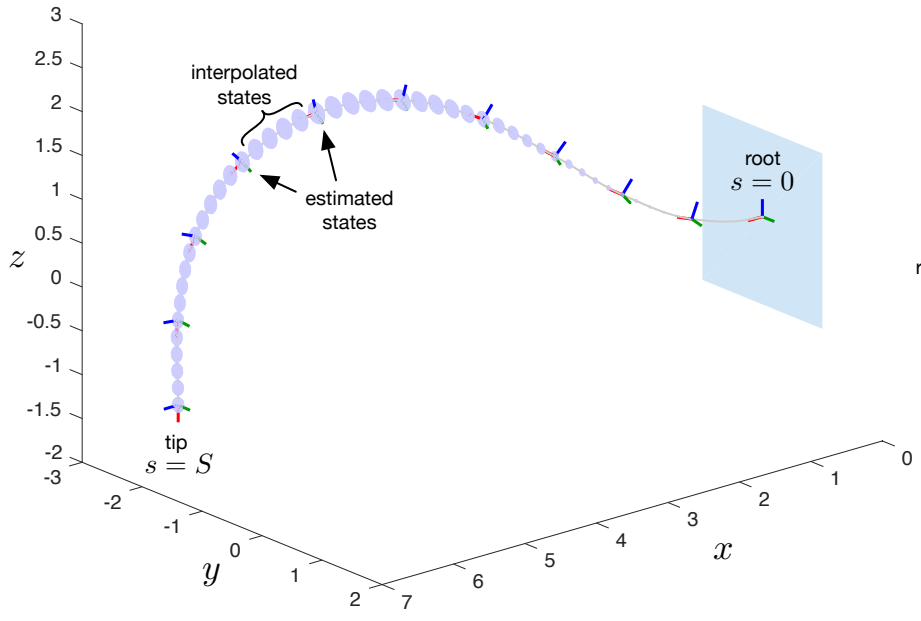
- a sparse Cholesky decomposition or Schur complement can be used to solve the linear system efficiently

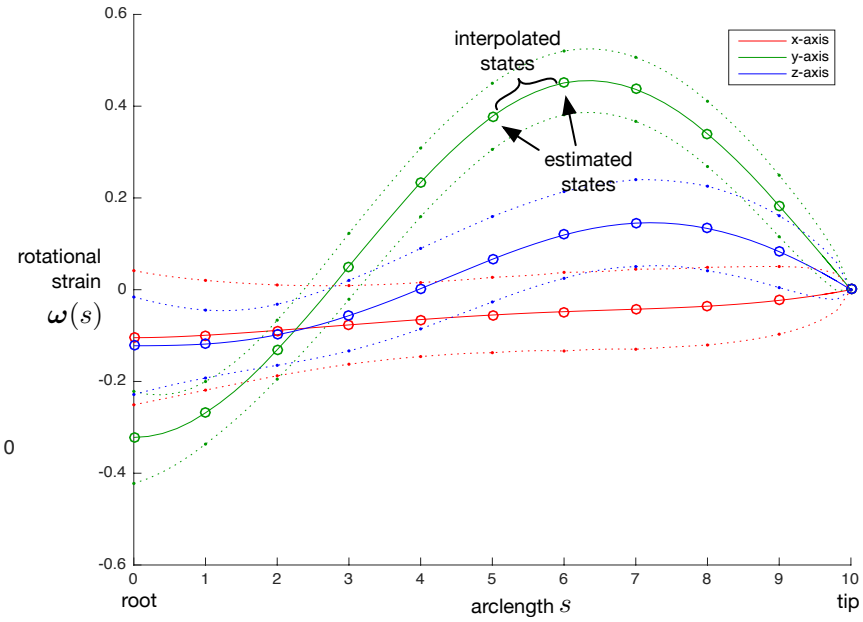
STEAM interpolation



– we can still use GP interpolation after the main solve – see book for details

Continuum robot example





- we can use these tools to estimate the shape of snake-like robots
- we simply replace time, t, with arclength, s

Lecture Summary

- we can build useful motion priors starting from physically motivated stochastic differential equations
- we then reframe estimation as Gaussian process regression
- still very efficient due to our particular motion priors / kernels – $O(K)$ for main solve, $O(1)$ for a query – reason is the Markov property
- ideas can be applied even when the state variables are members of Lie groups
- ideas can be used within a SLAM setup – STEAM
- Lecture 4: Beyond MAP – Variational Inference

Lecture 4: Beyond MAP – Variational Inference

A Short Course in State Estimation

Timothy D. Barfoot

University of Toronto

UNIVERSITY OF TORONTO

Lecture Outline

Lecture 1: Linear-Gaussian Estimation

Lecture 2: Nonlinearities and Lie Groups

Lecture 3: Continuous-Time Estimation

Lecture 4: Beyond MAP – Variational Inference

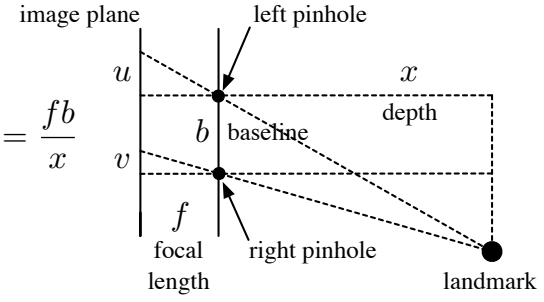
- Motivation
- Variational Inference
- Sparsity
- Derivative-free
- Parameter Learning

Motivation

- we have primarily presented an MAP approach to state estimation, and this certainly has its place
- but, is it possible to do better than that? can we be better Bayesians?
- consider a one-dimensional example where we get a single disparity measurement from a stereo camera:

$$y = \frac{fb}{x} + n, \quad n \sim \mathcal{N}(0, R)$$

$$\underbrace{y = u - v}_{\text{disparity}} = \frac{fb}{x}$$



(188)

- if we have a prior on position, $p(x) = \mathcal{N}\left(\check{x}, \check{P}\right)$, what is the Bayesian posterior once we incorporate this measurement?

Stereo camera example

– to perform Bayesian inference,

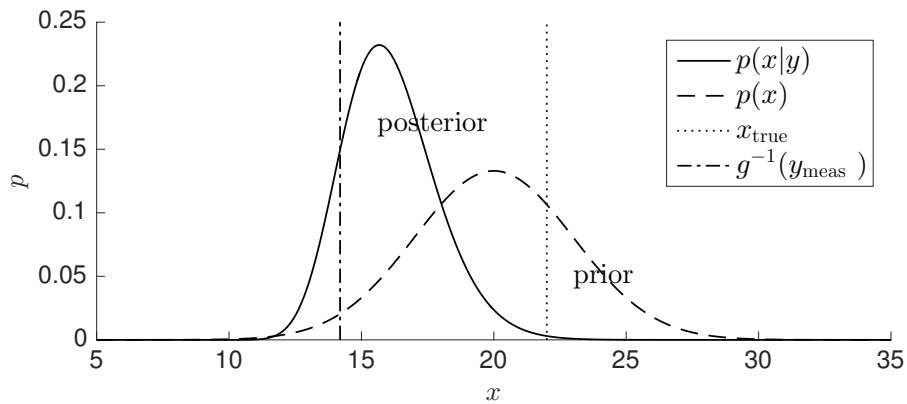
$$p(x|y) = \frac{p(y|x)p(x)}{\int_{-\infty}^{\infty} p(y|x)p(x)\,dx} \tag{189}$$

we require expressions for $p(y|x)$ and $p(x)$

$$p(y|x) = \mathcal{N}\left(\frac{fb}{x}, R\right) = \frac{1}{\sqrt{2\pi R}} \exp\left(-\frac{1}{2R}\left(y - \frac{fb}{x}\right)^2\right), \quad p(x) = \mathcal{N}\left(\check{x}, \check{P}\right) = \frac{1}{\sqrt{2\pi\check{P}}} \exp\left(-\frac{1}{2\check{P}}\left(x - \check{x}\right)^2\right)$$

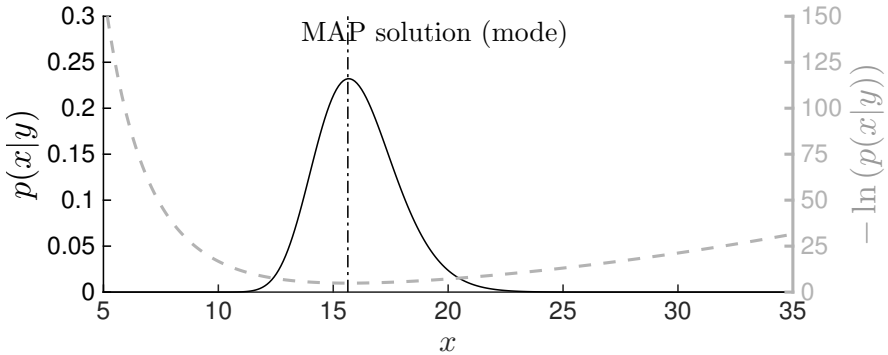
$$\check{x} = 20 \text{ [m]}, \quad \check{P} = 9 \text{ [m}^2\text{]}, \quad f = 400 \text{ [pixel]}, \quad b = 0.1 \text{ [m]}, \quad R = 0.09 \text{ [pixel}^2\text{]}, \quad x_{\text{true}} = 22 \text{ [m]}, \quad y_{\text{meas}} = \frac{fb}{x_{\text{true}}} + 1 \text{ [pixel]}$$

– the Bayesian posterior then looks like this (computed numerically)

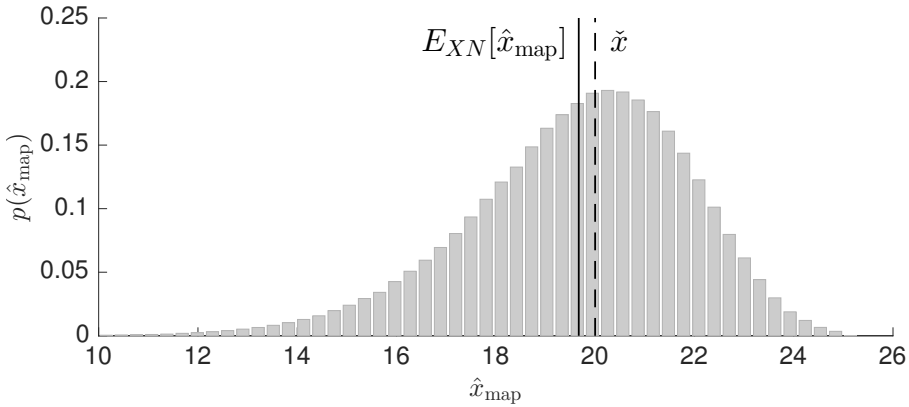


Stereo camera example

- clearly the Bayesian posterior is not Gaussian
- the MAP solution will find the mode of the Bayesian posterior



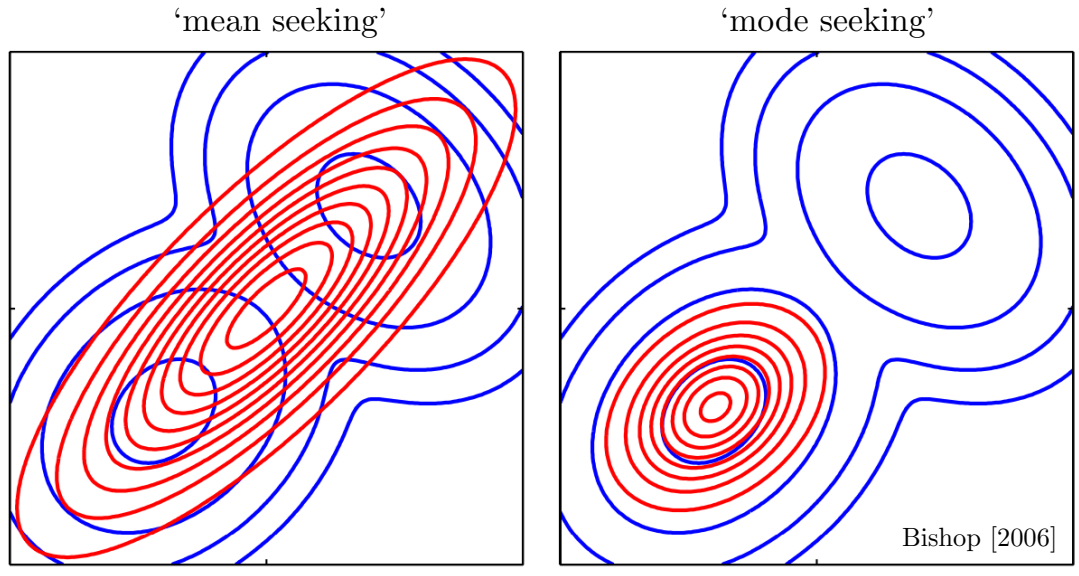
- is this what we want? we end up with a -33.0 cm average bias



- we might prefer an algorithm that can estimate the mean of the posterior if this is the metric we care about

Variational inference

- we could try variational (Bayesian) inference
- find a (Gaussian) estimate, $q(\mathbf{x})$, that is 'closest' to the Bayesian posterior, $p(\mathbf{x}|\mathbf{z})$, in terms of the Kullback-Leibler divergence



$$\mathrm{KL}(p||q) = -\int_{-\infty}^{\infty} p(\mathbf{x}|\mathbf{z}) \ln\left(\frac{q(\mathbf{x})}{p(\mathbf{x}|\mathbf{z})}\right) d\mathbf{x} = E_p\left[\ln p(\mathbf{x}|\mathbf{z}) - \ln q(\mathbf{x})\right]$$

$$\mathrm{KL}(q||p) = -\int_{-\infty}^{\infty} q(\mathbf{x}) \ln\left(\frac{p(\mathbf{x}|\mathbf{z})}{q(\mathbf{x})}\right) d\mathbf{x} = E_q\left[\ln q(\mathbf{x}) - \ln p(\mathbf{x}|\mathbf{z})\right],$$

- we pick the KL divergence on the right since easier to compute

Gaussian variational inference

- we assume our estimate is Gaussian

$$q(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^N |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{190}$$

- our chosen KL divergence then simplifies to

$$\mathrm{KL}(q||p) = E_q[-\ln p(\mathbf{x}, \mathbf{z})] - \underbrace{\frac{1}{2}\ln\left((2\pi e)^N |\boldsymbol{\Sigma}|\right)}_{\text{entropy}} + \underbrace{\ln p(\mathbf{z})}_{\text{constant}} \tag{191}$$

- dropping the constant term and defining $\phi(\mathbf{x}) = -\ln p(\mathbf{x}, \mathbf{z})$ we can define a functional as

$$V(q) = E_q[\phi(\mathbf{x})] + \frac{1}{2}\ln\left(|\boldsymbol{\Sigma}^{-1}|\right) \tag{192}$$

Gaussian variational inference

– our functional is

$$V(q) = E_q[\phi(\mathbf{x})] + \frac{1}{2}\ln\left(|\boldsymbol{\Sigma}^{-1}|\right) \tag{193}$$

– the derivatives with respect to the unknown mean, $\boldsymbol{\mu}$, and inverse covariance, $\boldsymbol{\Sigma}^{-1}$, are

$$\begin{aligned}
\frac{\partial V(q)}{\partial \boldsymbol{\mu}^T} &= \boldsymbol{\Sigma}^{-1} E_q[(\mathbf{x}-\boldsymbol{\mu})\phi(\mathbf{x})] \\
\frac{\partial^2 V(q)}{\partial \boldsymbol{\mu}^T \partial \boldsymbol{\mu}} &= \boldsymbol{\Sigma}^{-1} E_q[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T\phi(\mathbf{x})]\boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1}\, E_q[\phi(\mathbf{x})] \\
\frac{\partial V(q)}{\partial \boldsymbol{\Sigma}^{-1}} &= -\frac{1}{2} E_q[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T\phi(\mathbf{x})] + \frac{1}{2}\boldsymbol{\Sigma}\, E_q[\phi(\mathbf{x})] + \frac{1}{2}\boldsymbol{\Sigma}
\end{aligned}$$

Gaussian variational inference

– with these derivatives we can approximate the functional as

$$V\left(q^{(i+1)}\right) \approx V\left(q^{(i)}\right) + \left(\frac{\partial V(q)}{\partial \boldsymbol{\mu}^T}\bigg|_{q^{(i)}}\right)^T \delta\boldsymbol{\mu} + \frac{1}{2}\delta\boldsymbol{\mu}^T \left(\frac{\partial^2 V(q)}{\partial \boldsymbol{\mu}^T \partial \boldsymbol{\mu}}\bigg|_{q^{(i)}}\right) \delta\boldsymbol{\mu} + \text{tr}\left(\frac{\partial V(q)}{\partial \boldsymbol{\Sigma}^{-1}}\bigg|_{q^{(i)}} \delta\boldsymbol{\Sigma}^{-1}\right) \tag{195}$$

which we now want to iteratively minimize with respect to $\delta\boldsymbol{\mu} = \boldsymbol{\mu}^{(i+1)} - \boldsymbol{\mu}^{(i)}$ and $\delta\boldsymbol{\Sigma}^{-1} = \left(\boldsymbol{\Sigma}^{-1}\right)^{(i+1)} - \left(\boldsymbol{\Sigma}^{-1}\right)^{(i)}$

– clearly we want $\delta\boldsymbol{\mu}$ to be the solution to

$$\underbrace{\left(\frac{\partial^2 V(q)}{\partial \boldsymbol{\mu}^T \partial \boldsymbol{\mu}}\bigg|_{q^{(i)}}\right)}_{\left(\boldsymbol{\Sigma}^{-1}\right)^{(i+1)}} \delta\boldsymbol{\mu} = -\left(\frac{\partial V(q)}{\partial \boldsymbol{\mu}^T}\bigg|_{q^{(i)}}\right) \tag{196}$$

– the indicated inverse-covariance update results from setting $\frac{\partial V(q)}{\partial \boldsymbol{\Sigma}^{-1}}$ to zero and noticing the equivalence

$$\frac{\partial^2 V(q)}{\partial \boldsymbol{\mu}^T \partial \boldsymbol{\mu}} = \boldsymbol{\Sigma}^{-1} - 2\boldsymbol{\Sigma}^{-1}\frac{\partial V(q)}{\partial \boldsymbol{\Sigma}^{-1}}\boldsymbol{\Sigma}^{-1} \tag{197}$$

Natural gradient descent interpretation

- it turns out this update scheme is equivalent to natural gradient descent

$$\delta\boldsymbol{\alpha} = -\boldsymbol{\mathcal{I}}_{\boldsymbol{\alpha}}^{-1}\frac{\partial V(q)}{\partial \boldsymbol{\alpha}^T} \tag{198}$$

where

$$\boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{\mu} \\ \text{vec}\left(\boldsymbol{\Sigma}^{-1}\right) \end{bmatrix}, \quad \delta\boldsymbol{\alpha} = \begin{bmatrix} \delta\boldsymbol{\mu} \\ \text{vec}\left(\delta\boldsymbol{\Sigma}^{-1}\right) \end{bmatrix}, \quad \frac{\partial V(q)}{\partial \boldsymbol{\alpha}^T} = \begin{bmatrix} \frac{\partial V(q)}{\partial \boldsymbol{\mu}^T} \\ \text{vec}\left(\frac{\partial V(q)}{\partial \boldsymbol{\Sigma}^{-1}}\right) \end{bmatrix}, \quad \boldsymbol{\mathcal{I}}_{\boldsymbol{\alpha}} = \begin{bmatrix} \boldsymbol{\Sigma}^{-1} & \mathbf{0} \\ \mathbf{0} & \frac{1}{2}\left(\boldsymbol{\Sigma} \otimes \boldsymbol{\Sigma}\right) \end{bmatrix}$$

- importantly, $\boldsymbol{\mathcal{I}}_{\boldsymbol{\alpha}}$ is the Fisher information matrix for the variational parameter, $\boldsymbol{\alpha}$
- NGD is essentially a quasi-Newton method that uses the FIM as an approximation of the Hessian

Stein's lemma to the rescue

– we can then employ Stein's lemma

$$E_q[(\mathbf{x}-\boldsymbol{\mu})\phi(\mathbf{x})] = \boldsymbol{\Sigma}\, E_q\left[\frac{\partial \phi(\mathbf{x})}{\partial \mathbf{x}^T}\right] \tag{199}$$

to clean up the update further so that it becomes

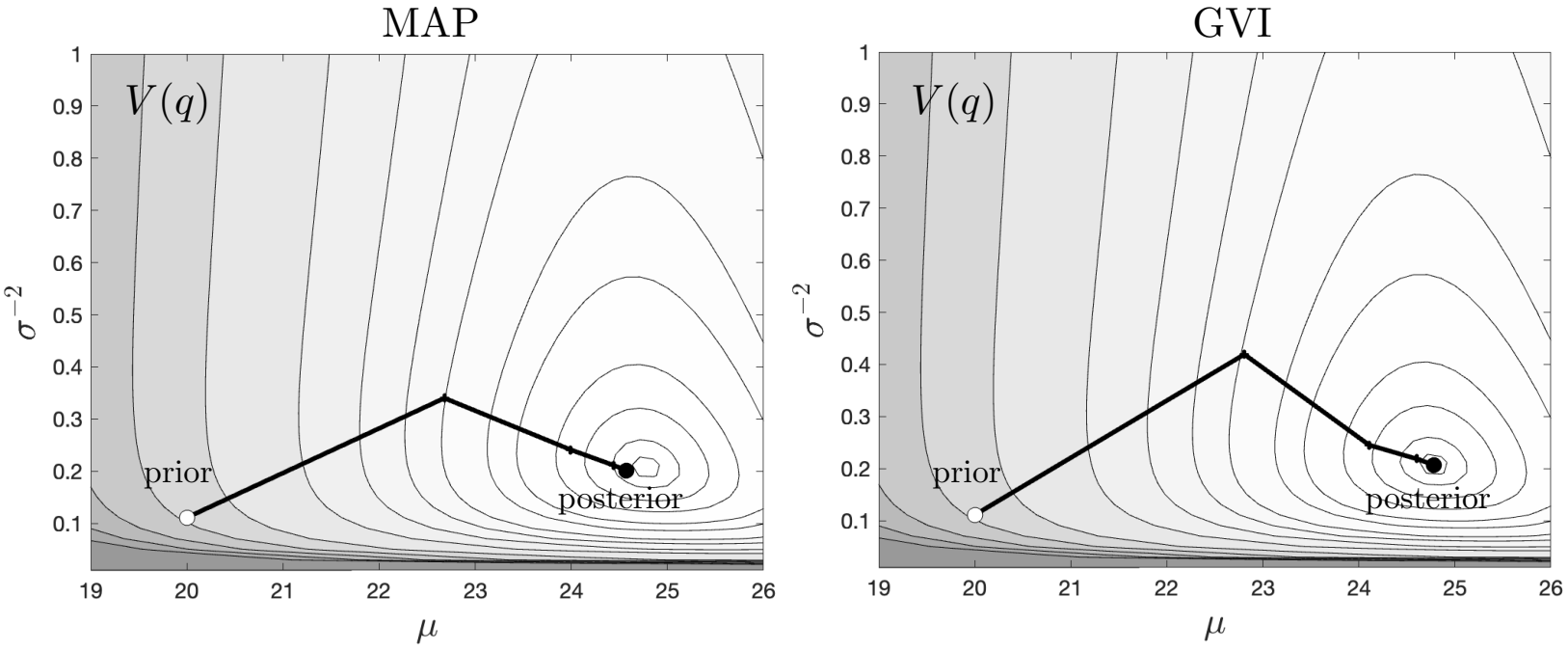
$$\begin{aligned}
\left(\boldsymbol{\Sigma}^{-1}\right)^{(i+1)} &= E_{q^{(i)}}\left[\frac{\partial^2}{\partial \mathbf{x}^T \partial \mathbf{x}}\phi(\mathbf{x})\right] && (200a)\\
\left(\boldsymbol{\Sigma}^{-1}\right)^{(i+1)}\,\delta\boldsymbol{\mu} &= -E_{q^{(i)}}\left[\frac{\partial}{\partial \mathbf{x}^T}\phi(\mathbf{x})\right] && (200b)\\
\boldsymbol{\mu}^{(i+1)} &= \boldsymbol{\mu}^{(i)} + \delta\boldsymbol{\mu} && (200c)
\end{aligned}$$

where i is the iteration index

– this looks just like an MAP update via Newton's method – but now we have expectations wrapping the derivatives of $\phi(\mathbf{x})$

Stereo camera example

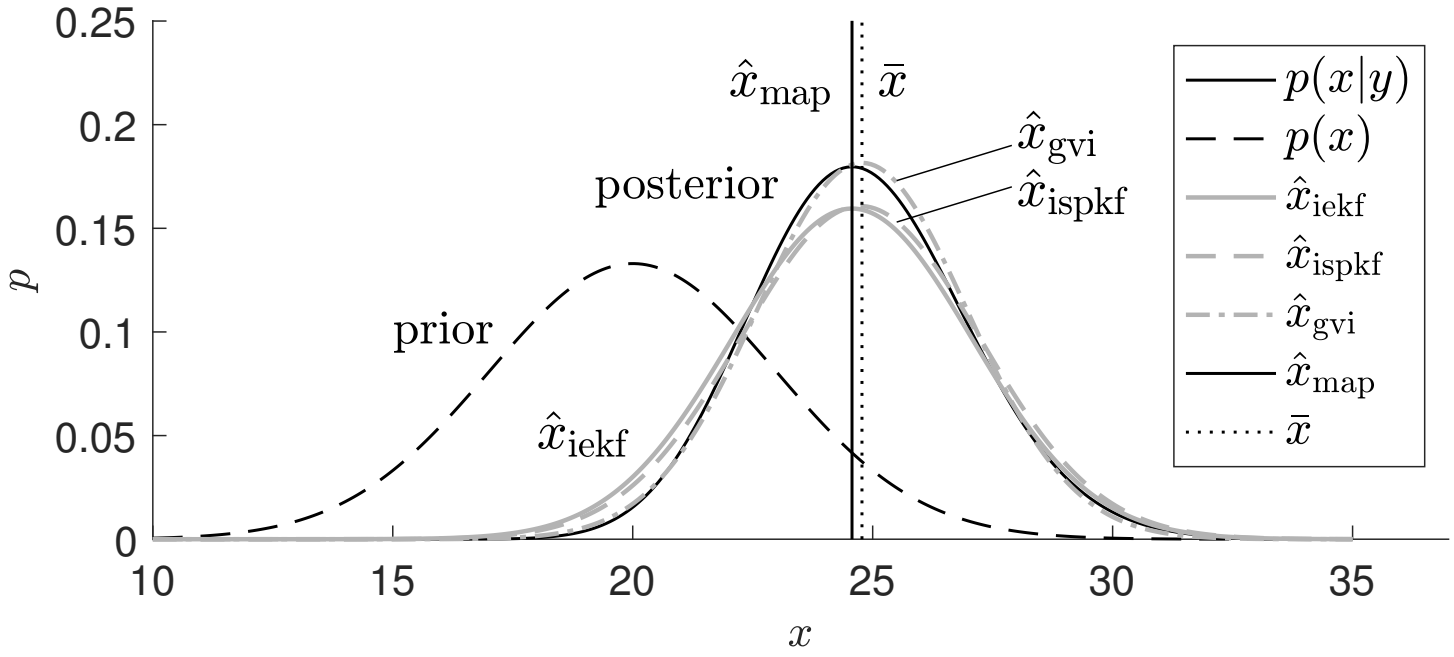
– if we return to our stereo camera example, we can compare the performance of MAP and GVI



– MAP approximates the expectations, $E_q[\cdot]$, only at the mean and as a result does not get to the minimum of $V(q)$

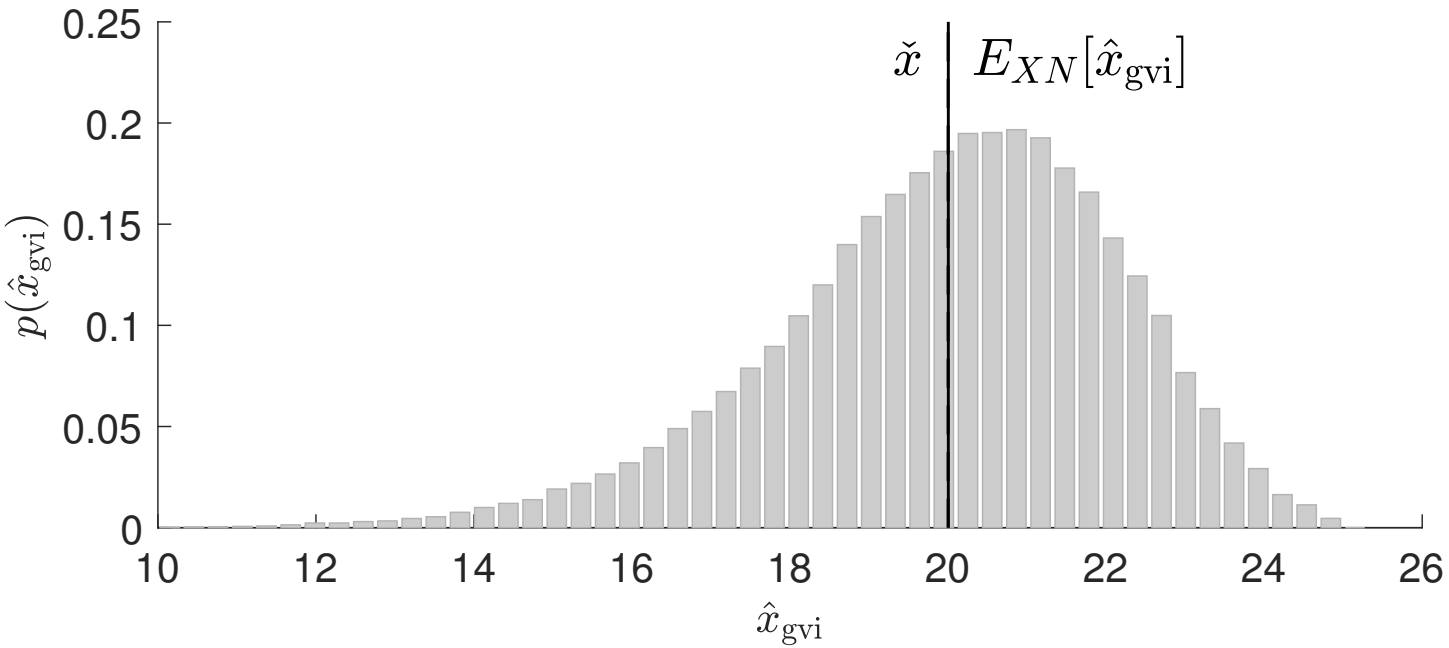
Stereo camera example

– we can see that GVI also converges to a point where its mean looks closer to the mean of the true posterior



Stereo camera example

– histogram of the errors over many trials shows GVI has an average error of 0.28 cm whereas MAP was -33.0 cm



– GVI seems to help remove the bias in this controlled example

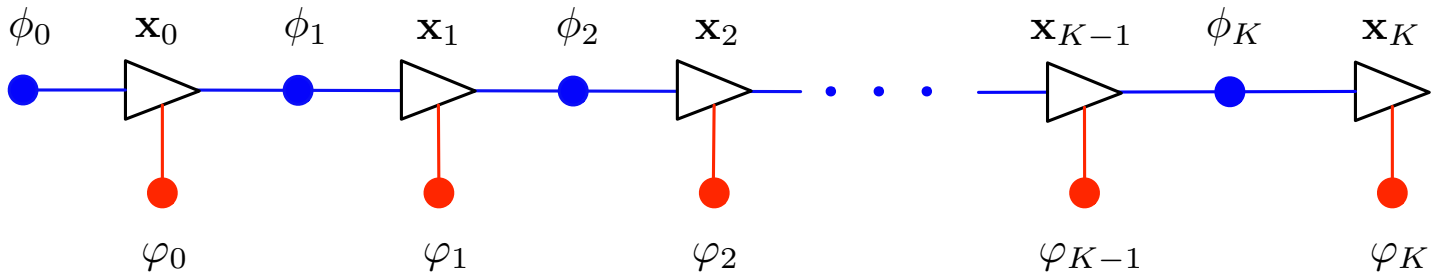
Can we scale up?

- there are some expectations in our update scheme that look computationally expensive

$$\underbrace{E_q[\phi(\mathbf{x})]}_{\text{scalar}}, \quad \underbrace{E_q\left[\frac{\partial}{\partial \mathbf{x}^T}\phi(\mathbf{x})\right]}_{\text{column}}, \quad \underbrace{E_q\left[\frac{\partial^2}{\partial \mathbf{x}^T \partial \mathbf{x}}\phi(\mathbf{x})\right]}_{\text{matrix}} \tag{201}$$

- however, if we assume there is some sparsity (in the usual factor graph way) then our negative log-likelihood decomposes as

$$\phi(\mathbf{x}) = \sum_{k=1}^{K} \phi_k(\mathbf{x}_k) \tag{202}$$



Can we scale up?

- our negative log-likelihood decomposes as

$$\phi(\mathbf{x}) = \sum_{k=1}^{K} \phi_k(\mathbf{x}_k) \tag{203}$$

- inserting this we see for the first expectation that

$$E_q[\phi(\mathbf{x})] = E_q\left[\sum_{k=1}^{K} \phi_k(\mathbf{x}_k)\right] = \sum_{k=1}^{K} E_q[\phi_k(\mathbf{x}_k)] = \sum_{k=1}^{K} E_{q_k}[\phi_k(\mathbf{x}_k)] \tag{204}$$

  where the expectations are now only over $q_k(\mathbf{x}_k) = \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_{kk})$, the marginal for the variables involved in $\phi_k(\mathbf{x}_k)$

- similar simplifications occur for the other two expectations

$$E_q\left[\frac{\partial}{\partial \mathbf{x}^T}\phi(\mathbf{x})\right] = \sum_{k=1}^{K} \mathbf{P}_k^T E_{q_k}\left[\frac{\partial}{\partial \mathbf{x}_k^T}\phi_k(\mathbf{x}_k)\right] \tag{205a}$$

$$E_q\left[\frac{\partial^2}{\partial \mathbf{x}^T \partial \mathbf{x}}\phi(\mathbf{x})\right] = \sum_{k=1}^{K} \mathbf{P}_k^T E_{q_k}\left[\frac{\partial^2}{\partial \mathbf{x}_k^T \partial \mathbf{x}_k}\phi_k(\mathbf{x}_k)\right] \mathbf{P}_k \tag{205b}$$

  where $\mathbf{P}_k$ is a selection matrix so that $\mathbf{x}_k = \mathbf{P}_k\mathbf{x}$

Can we scale up?

- with this simplification of the expectations, we call our scheme exactly sparse Gaussian variational inference (ESGVI) – we only need to compute expectations over the marginals associated with the variables in each factor
- this means that we only need the blocks of the covariance, $\boldsymbol{\Sigma}$, associated with the non-zero blocks of the inverse covariance, $\boldsymbol{\Sigma}^{-1}$
- Takahashi et al. [1973] shows how to compute a set of covariance entries that is just what we need, plus some extra blocks that come from fill-in – e.g., Cholesky decomposition $\boldsymbol{\Sigma}^{-1} = \mathbf{L}\mathbf{L}^T$
- finding a variable order that minimizes fill-in is known to be an NP-hard problem in general
- we piggyback finding the required covariance entries onto the usual MAP solve, so ESGVI has the same big-O complexity as MAP, but the expectations still make it slower

Fill-in

– typical fill-in '$+$' in for our estimation problems – we efficiently compute blocks of $\mathbf{\Sigma}$ where $\mathbf{L}$ is non-zero

basic sparsity constraint
(note fill in at $(5,3)$ in $\mathbf{L}$)

$$\mathbf{\Sigma}^{-1} = \begin{bmatrix} * & & * & & * & \\ & * & & & & \\ * & & * & & & \\ & & & * & & \\ * & & & & * & \\ & & & & & * \end{bmatrix}$$

$$\mathbf{L} = \begin{bmatrix} * & & & & & \\ & * & & & & \\ * & & * & & & \\ & & & * & & \\ * & & + & & * & \\ & & & & & * \end{bmatrix}$$

trajectory example
(6 robot poses)

$$\mathbf{\Sigma}^{-1} = \begin{bmatrix} * & * & & & & \\ * & * & * & & & \\ & * & * & * & & \\ & & * & * & * & \\ & & & * & * & * \\ & & & & * & * \end{bmatrix}$$

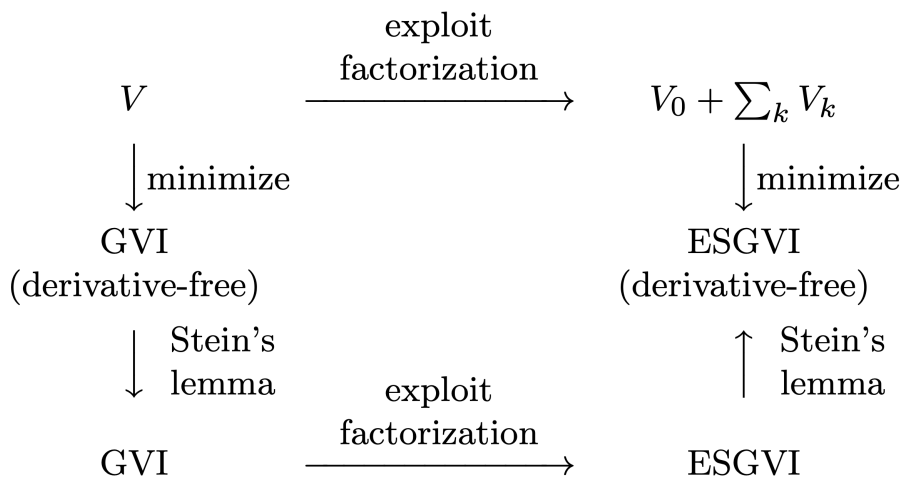
$$\mathbf{L} = \begin{bmatrix} * & & & & & \\ * & * & & & & \\ & * & * & & & \\ & & * & * & & \\ & & & * & * & \\ & & & & * & * \end{bmatrix}$$

SLAM example
(3 poses, 3 landmarks)

$$\mathbf{\Sigma}^{-1} = \left[\begin{array}{ccc|ccc} * & * & & * & * & * \\ * & * & * & * & * & * \\ & * & * & * & * & * \\ \hline * & * & * & * & & \\ * & * & * & & * & \\ * & * & * & & & * \end{array}\right]$$

$$\mathbf{L} = \left[\begin{array}{ccc|ccc} * & & & & & \\ * & * & & & & \\ & * & * & & & \\ \hline * & * & * & * & & \\ * & * & * & + & * & \\ * & * & * & + & + & * \end{array}\right]$$

Derivative-free version



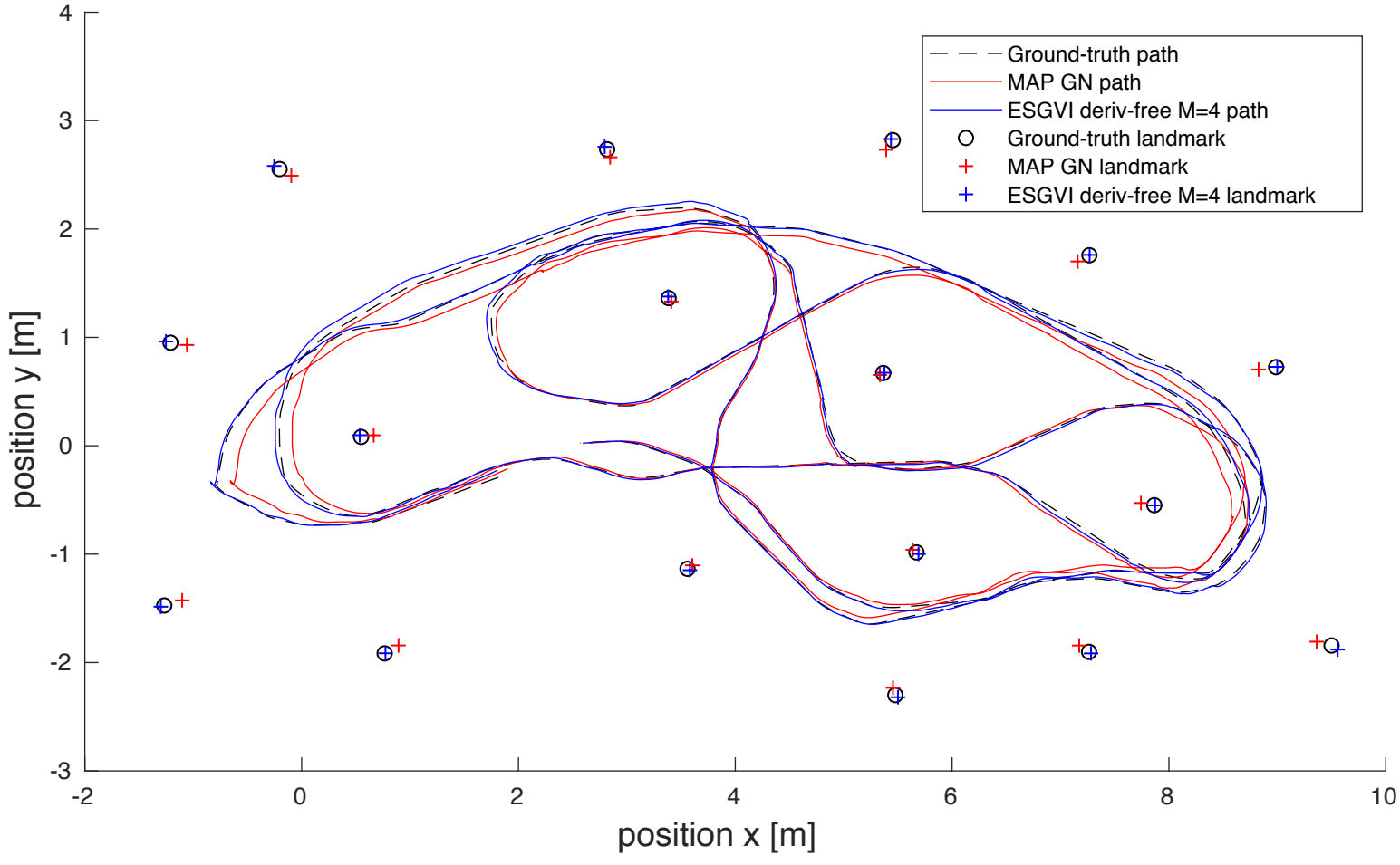
– we can also use Stein's lemma (opposite direction) at the individual factor level to compute all the small expectations derivative-free

$$\text{e.g.,} \quad E_{q_k}\left[\frac{\partial}{\partial \mathbf{x}_k^T}\phi_k(\mathbf{x}_k)\right] = \boldsymbol{\Sigma}_{kk}^{-1} E_{q_k}\left[(\mathbf{x}_k - \boldsymbol{\mu}_k)\phi_k(\mathbf{x}_k)\right] \tag{206}$$

– this means we can do batch state estimation without derivatives!

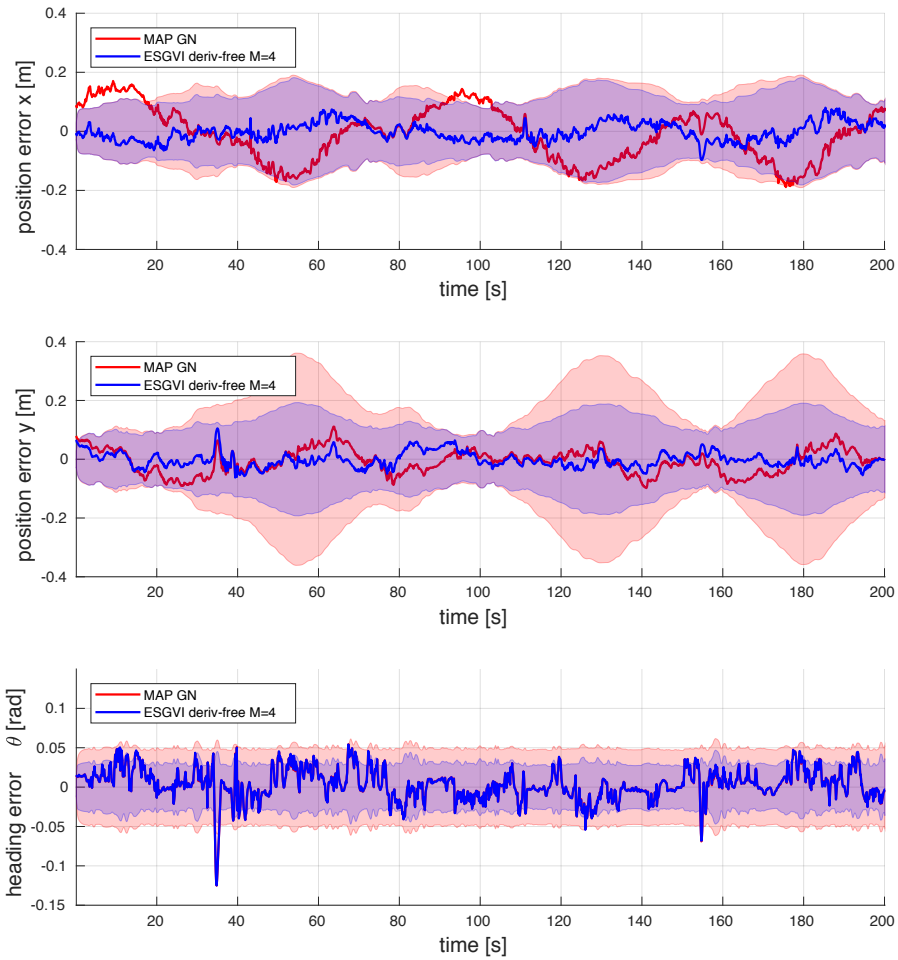
Example: bearing-only SLAM

– we can find challenging estimation cases where GVI outperforms MAP – the gains are usually small



Example: bearing-only SLAM

– we can find challenging estimation cases where GVI outperforms MAP – the gains are usually small but measurable



UNIVERSITY OF TORONTO

Parameter learning

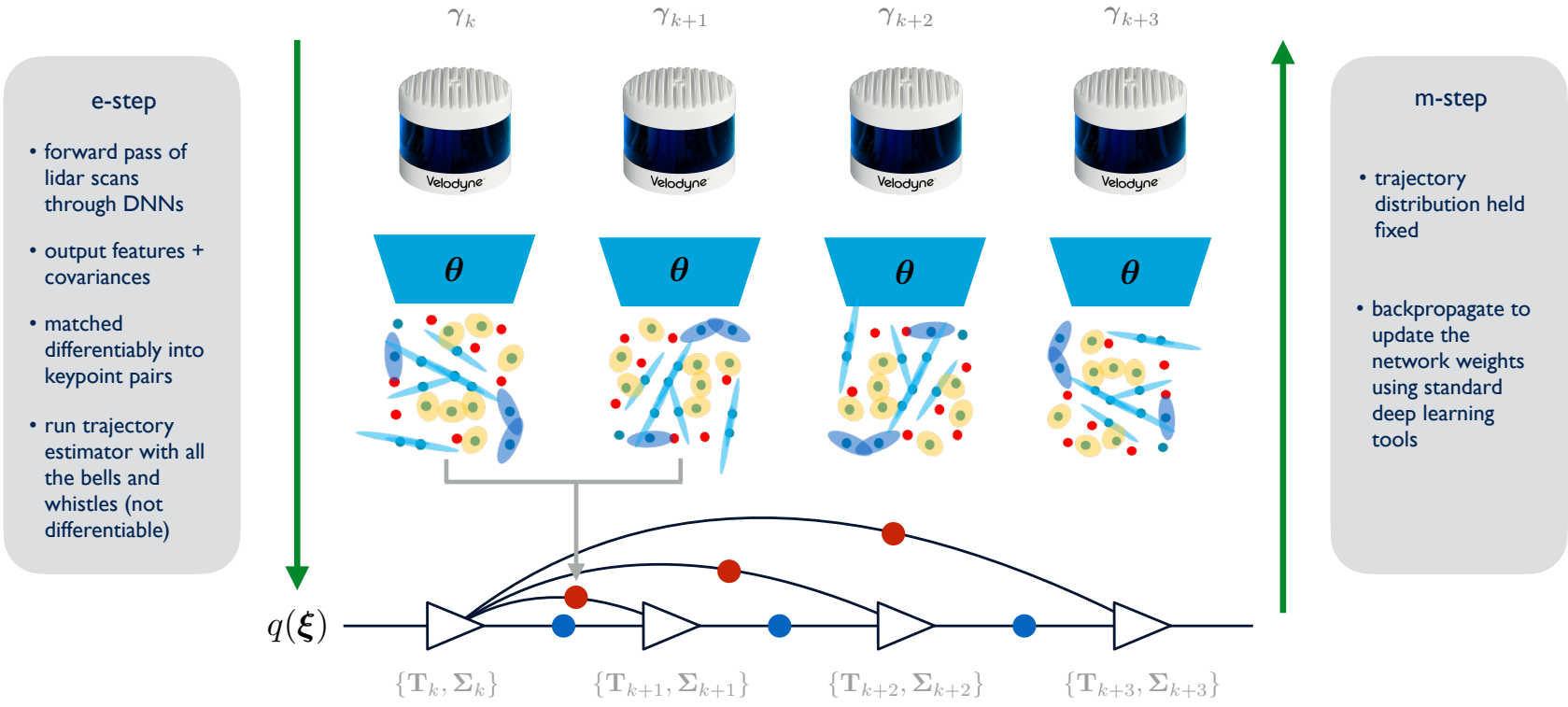
- an added bonus of this framework is that we can also learn parameters, $\boldsymbol{\theta}$
- the negative log-likelihood of our data (given the parameters) can be written as

$$-\ln(p(\mathbf{z}|\boldsymbol{\theta}) = \underbrace{\int_{-\infty}^{\infty} q(\mathbf{x}) \ln\left(\frac{p(\mathbf{x}|\mathbf{z},\boldsymbol{\theta})}{q(\mathbf{x})}\right) d\mathbf{x}}_{-\mathrm{KL}(q||p)\leq 0} \underbrace{- \int_{-\infty}^{\infty} q(\mathbf{x}) \ln\left(\frac{p(\mathbf{x},\mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{x})}\right) d\mathbf{x}}_{V(q|\boldsymbol{\theta})} \tag{207}$$

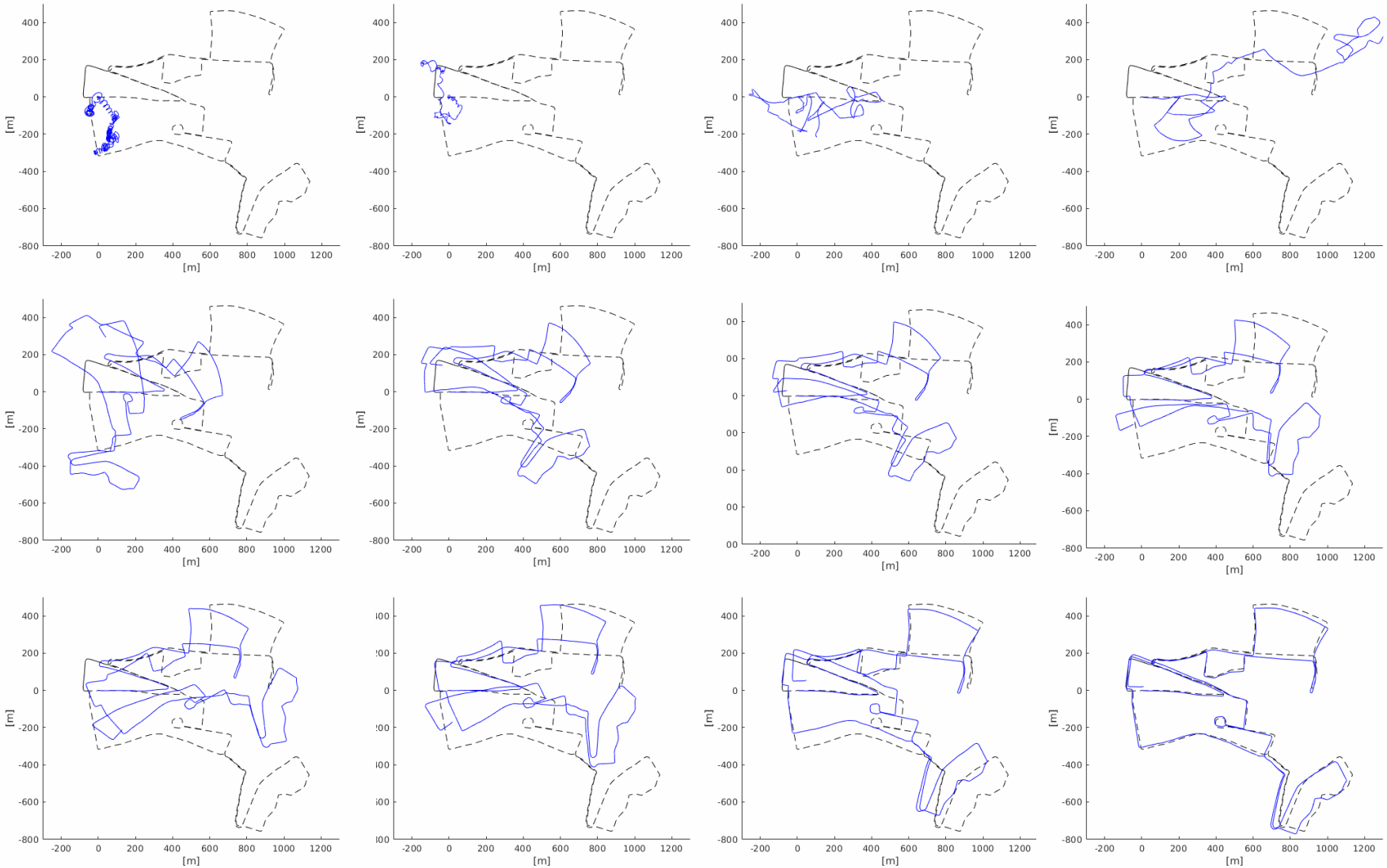
- we can use expectation minimization [Neal and Hinton, 1998]:
  - e-step: hold $\boldsymbol{\theta}$ fixed, minimize $\mathbf{V}(q|\boldsymbol{\theta})$ w.r.t. q
  - m-step: hold q fixed, minimize $\mathbf{V}(q|\boldsymbol{\theta})$ w.r.t. $\boldsymbol{\theta}$
- parameters can be anything – measurement covariances, calibrations, network weights for measurement model

Example: parameter learning

– we can do unsupervised parameter learning with our full non-differentiable state estimator



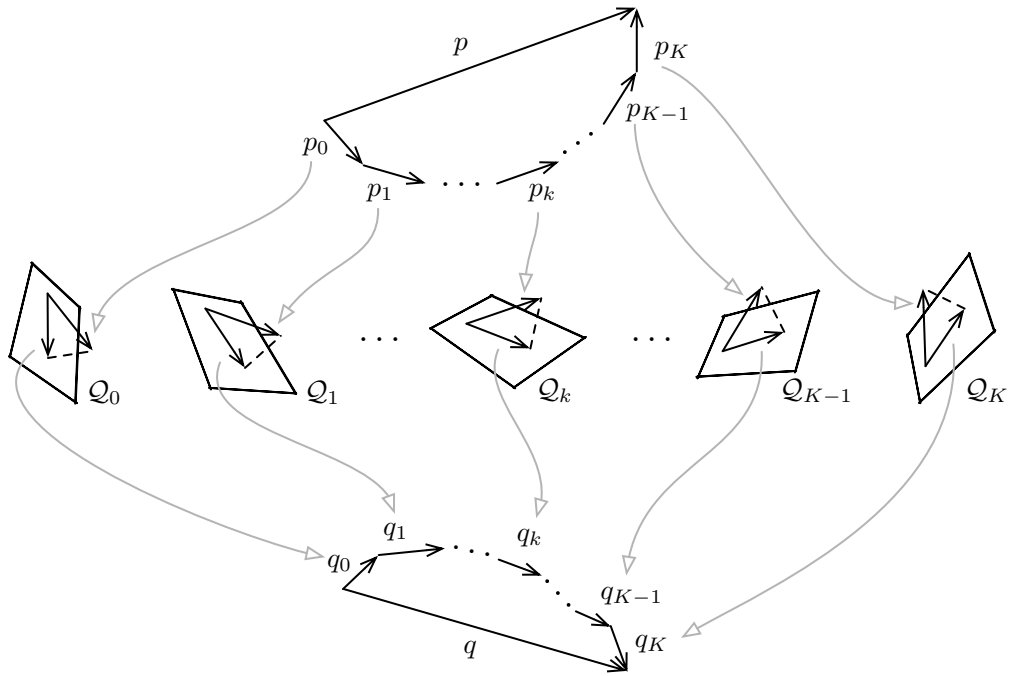
Example: parameter learning



UNIVERSITY OF TORONTO

Geometric interpretation

– can also view ESGVI through the lens of Bayes space – a high-dimensional geometric space where entire PDFs can be viewed as a single vector



– inference is then iterative projection of the full Bayesian posterior onto a subspace containing Gaussians (Barfoot and D'Eleuterio, 2023)

Lecture Summary

- we have seen that Gaussian variational inference can be used to produce a 'better' solution than MAP
- we can still exploit sparsity to make ESGVI have the same big-O complexity as MAP – but compute time is still a large factor slower
- we can avoid even computing derivatives by using Stein's lemma and Gaussian cubature to evaluate expectations
- in practice we can run MAP first since it's faster, then refine the answer with ESGVI for a few iterations
- can fold in parameter learning via expectation minimization
- there is a way to also enforce that $\boldsymbol{\Sigma}$ remains positive definite by exploiting the connection to NGD [Goudar et al., 2022]
- although not discussed, can make the ideas work on Lie groups
- ESGVI is related to but distinct from Stein variational inference

Course Conclusion

- looking for next steps?
- estimation, solvers:
  - general overview (Thrun et al., 2006; Barfoot, 2024)
  - incremental smoothing and mapping (iSAM2) (Kaess et al., 2012)
  - certifiably optimal estimation (Yang and Carlone, 2022)
- Lie groups:
  - invariant EKF (Bonnabel et al., 2008)
  - equivariant EKF (Mahony and Trumpf, 2021)
- variational inference:
  - Stein variational inference (Liu and Wang, 2016; Maken et al., 2022)

References

Barfoot, T. D., *State Estimation for Robotics*, Cambridge University Press, 2nd edition, 2024.

Barfoot, T. D. and D'Eleuterio, G. M. T., "Variational Inference as Iterative Projection in a Bayesian Hilbert Space with Application to Robotic State Estimation," *Robotica*, 41(2):632–667, 2023.

Bishop, C. M., *Pattern Recognition and Machine Learning*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

Bonnabel, S., Martin, P., and Rouchon, P., "Symmetry-preserving observers," *IEEE Transactions on Automatic Control*, 53(11):2514–2526, 2008.

Boumal, N., *An introduction to optimization on smooth manifolds*, Cambridge University Press, 2023.

Goudar, A., Zhao, W., Barfoot, T. D., and Schoellig, A. P., "Gaussian Variational Inference with Covariance Constraints Applied to Range-Only Localization," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robot Systems (IROS)*, Kyoto, Japan, 2022.

Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., and Dellaert, F., "iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree," *IJRR*, 31(2):217–236, 2012.

Liu, Q. and Wang, D., "Stein variational gradient descent: A general purpose bayesian inference algorithm," *Advances in neural information processing systems*, 29, 2016.

Mahony, R. and Trumpf, J., "Equivariant filter design for kinematic systems on lie groups," *IFAC-PapersOnLine*, 54(9):253–260, 2021.

Maken, F. A., Ramos, F., and Ott, L., "Stein particle filter for nonlinear, non-Gaussian state estimation," *IEEE Robotics and Automation Letters*, 7(2):5421–5428, 2022.

Rasmussen, C. E. and Williams, C. K. I., *Gaussian Processes for Machine Learning*, MIT Press, Cambridge, MA, 2006.

Takahashi, K., Fagan, J., and Chen, M.-S., "A Sparse Bus Impedance Matrix and its Application to Short Circuit Study," in *Proceedings of the PICA Conference*, 1973.

Thrun, S., Burgard, W., and Fox, D., *Probabilistic Robotics*, MIT Press, 2006.

Yang, H. and Carlone, L., "Certifiably optimal outlier-robust geometric perception: Semidefinite relaxations and scalable global optimization," *IEEE transactions on pattern analysis and machine intelligence*, 45(3):2816–2834, 2022.