Lecture 7: Batch Nonlinear Non-Gaussian Estimation

AER1513: State Estimation

Timothy D. Barfoot

University of Toronto

UNIVERSITY OF TORONTO

Outline

Motivation

- we've seen several ways of building a recursive solution to the nonlinear, non-Gaussian estimation problem
- these were all some approximation of the Bayes filter
- in this lecture, we return to the idea of batch methods that we first saw in the linear-Gaussian context

smoothers use all available information to estimate states

$$\hat{\mathbf{x}}_k$$

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

filters only use past/current information to estimate states

System

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

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

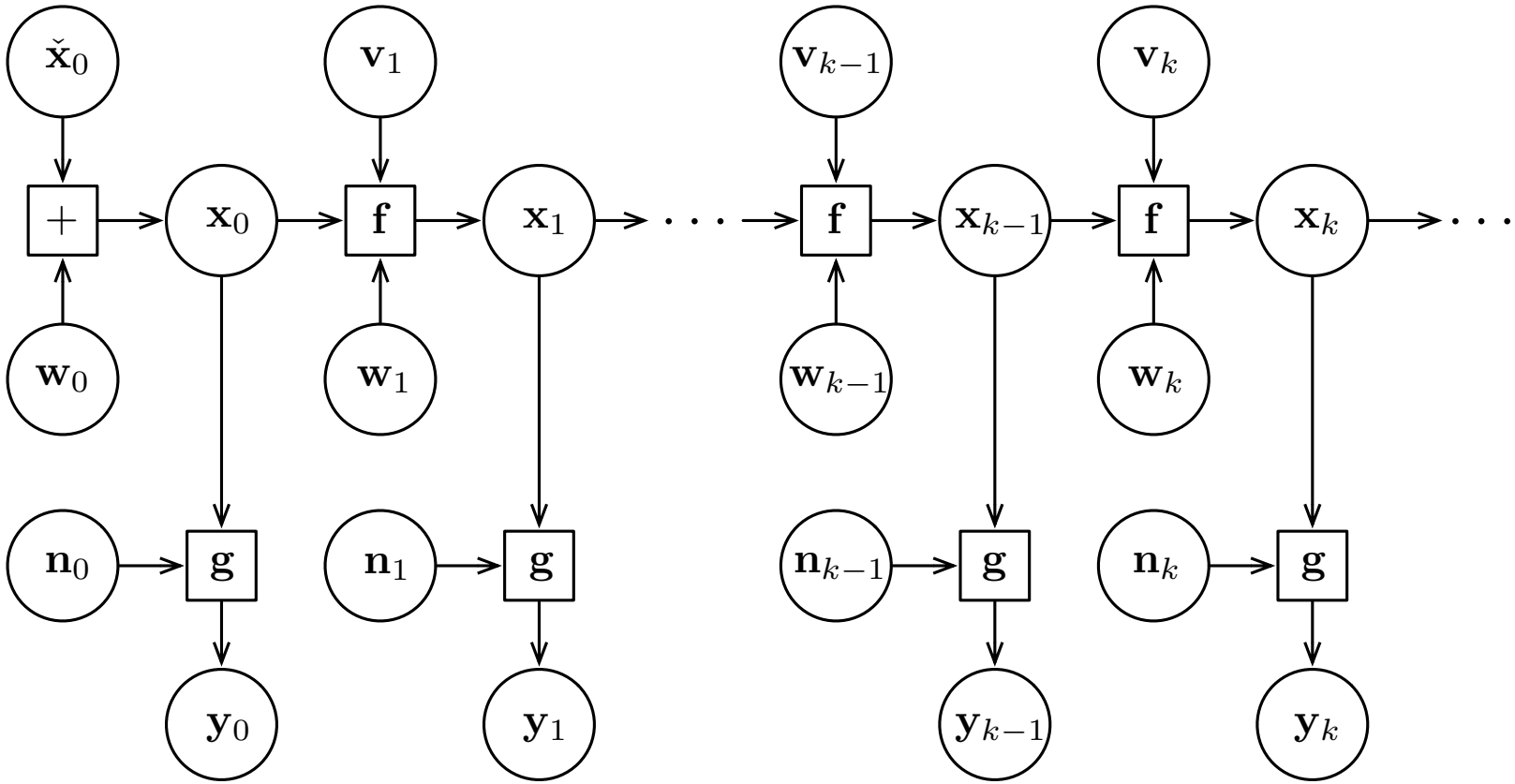
$$\text{observation model:} \quad \mathbf{y}_k = \mathbf{g}\left(\mathbf{x}_k, \mathbf{n}_k\right), \qquad k = 0 \ldots K \tag{1b}$$

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

– the variables have the following meanings:

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

Markov property



Problem statement

– our state estimation problem is still the following:

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.

Batch optimization

– we seek to construct an objective function that we will minimize with respect to

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_K \end{bmatrix} \tag{2}$$

which represents the entire trajectory that we want to estimate

– recall that the linear-Gaussian objective function took the form of a squared Mahalanobis distance and was proportional to the negative log likelihood of all the measurements

Nonlinear errors

– we define the errors with respect to the prior and measurements to be

$$\mathbf{e}_{v,k}(\mathbf{x}) = \begin{cases} \check{\mathbf{x}}_0 - \mathbf{x}_0, & k = 0 \\ \mathbf{f}\left(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{0}\right) - \mathbf{x}_k, & k = 1 \dots K \end{cases} \tag{3a}$$

$$\mathbf{e}_{y,k}(\mathbf{x}) = \mathbf{y}_k - \mathbf{g}\left(\mathbf{x}_k, \mathbf{0}\right), \qquad k = 0 \dots K \tag{3b}$$

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

Cost terms

- the contributions to the objective function are

$$J_{v,k}(\mathbf{x}) = \frac{1}{2}\mathbf{e}_{v,k}(\mathbf{x})^T \mathbf{W}_{v,k}^{-1} \mathbf{e}_{v,k}(\mathbf{x}) \tag{4a}$$

$$J_{y,k}(\mathbf{x}) = \frac{1}{2}\mathbf{e}_{y,k}(\mathbf{x})^T \mathbf{W}_{y,k}^{-1} \mathbf{e}_{y,k}(\mathbf{x}) \tag{4b}$$

- the overall objective function is then

$$J(\mathbf{x}) = \sum_{k=0}^{K} \left(J_{v,k}(\mathbf{x}) + J_{y,k}(\mathbf{x})\right) \tag{5}$$

- 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

Stacking things up

– we further define

$$\mathbf{e}(\mathbf{x}) = \begin{bmatrix} \mathbf{e}_v(\mathbf{x}) \\ \mathbf{e}_y(\mathbf{x}) \end{bmatrix}, \quad \mathbf{e}_v(\mathbf{x}) = \begin{bmatrix} \mathbf{e}_{v,0}(\mathbf{x}) \\ \vdots \\ \mathbf{e}_{v,K}(\mathbf{x}) \end{bmatrix}, \quad \mathbf{e}_y(\mathbf{x}) = \begin{bmatrix} \mathbf{e}_{y,0}(\mathbf{x}) \\ \vdots \\ \mathbf{e}_{y,K}(\mathbf{x}) \end{bmatrix} \tag{6a}$$

$$\mathbf{W} = \mathsf{diag}\left(\mathbf{W}_v, \mathbf{W}_y\right), \quad \mathbf{W}_v = \mathsf{diag}\left(\mathbf{W}_{v,0}, \ldots, \mathbf{W}_{v,K}\right), \tag{6b}$$

$$\mathbf{W}_y = \mathsf{diag}\left(\mathbf{W}_{y,0}, \ldots, \mathbf{W}_{y,K}\right) \tag{6c}$$

so that the objective function can be written as

$$J(\mathbf{x}) = \frac{1}{2}\mathbf{e}(\mathbf{x})^T \mathbf{W}^{-1} \mathbf{e}(\mathbf{x}) \tag{7}$$

Standard nonlinear squared error form

- we can further define the modified error term,

$$\mathbf{u}(\mathbf{x}) = \mathbf{L}\,\mathbf{e}(\mathbf{x}) \tag{8}$$

  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

$$J(\mathbf{x}) = \frac{1}{2}\mathbf{u}(\mathbf{x})^T\mathbf{u}(\mathbf{x}) \tag{9}$$

- this is precisely in a quadratic form, but not with respect to the design variables, $\mathbf{x}$

Optimization perspective

- our goal is to determine the optimum design parameter, $\hat{\mathbf{x}}$, that minimizes the objective function:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} J(\mathbf{x}) \tag{10}$$

- in the linear-Gaussian case, we could find the optimum in closed form, but this is no longer true in the nonlinear, non-Gaussian case
- there are many nonlinear optimization techniques that we can apply to minimize this expression due to its quadratic nature
- typical technique to use is Gauss-Newton optimization, but there are many other possibilities
- we'll derive Gauss-Newton via Newton's method

Newton's method

- Newton's method works by iteratively approximating the (differentiable) objective function by a quadratic function and then jumping to (or moving towards) the minimum
- suppose we have an initial guess, or operating point, for the design parameter, $\mathbf{x}_{\text{op}}$
- we use a three-term Taylor-series expansion to approximate J as a quadratic function,

$$J(\mathbf{x}_{\text{op}}+\delta\mathbf{x}) \approx J(\mathbf{x}_{\text{op}})+\underbrace{\left(\left.\frac{\partial J(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_{\text{op}}}\right)}_{\text{Jacobian}}\delta\mathbf{x}+\frac{1}{2}\delta\mathbf{x}^T\underbrace{\left(\left.\frac{\partial^2 J(\mathbf{x})}{\partial \mathbf{x}\partial \mathbf{x}^T}\right|_{\mathbf{x}_{\text{op}}}\right)}_{\text{Hessian}}\delta\mathbf{x} \tag{11}$$

  of $\delta\mathbf{x}$, a 'small' change to the initial guess, $\mathbf{x}_{\text{op}}$

Newton's method

– we can find the minimizing value of $\delta\mathbf{x}$ by taking the derivative with respect to $\delta\mathbf{x}$ and setting to zero to find a critical point:

$$\frac{\partial J(\mathbf{x}_{\text{op}} + \delta\mathbf{x})}{\partial\, \delta\mathbf{x}} = \left(\frac{\partial J(\mathbf{x})}{\partial \mathbf{x}}\bigg|_{\mathbf{x}_{\text{op}}} \right) + \delta\mathbf{x}^{*^T} \left(\frac{\partial^2 J(\mathbf{x})}{\partial \mathbf{x} \partial \mathbf{x}^T}\bigg|_{\mathbf{x}_{\text{op}}} \right) = \mathbf{0}$$

$$\Rightarrow \quad \left(\frac{\partial^2 J(\mathbf{x})}{\partial \mathbf{x} \partial \mathbf{x}^T}\bigg|_{\mathbf{x}_{\text{op}}} \right) \delta\mathbf{x}^* = -\left(\frac{\partial J(\mathbf{x})}{\partial \mathbf{x}}\bigg|_{\mathbf{x}_{\text{op}}} \right)^T \tag{12}$$

– the last line is just a linear system of equations and can be solved when the Hessian is invertible

– we may then update our operating point according to:

$$\mathbf{x}_{\text{op}} \leftarrow \mathbf{x}_{\text{op}} + \delta\mathbf{x}^* \tag{13}$$

– this procedure iterates until $\delta\mathbf{x}^*$ becomes sufficiently small

Newton's method commentary

- Newton's method is locally convergent, which means the successive approximations are guaranteed to converge to a solution when the initial guess is already close enough to the solution
- for a complex nonlinear objective function, this is really the best we can expect (i.e., global convergence is difficult to achieve)
- the rate of convergence is quadratic (i.e., it converges much faster than simple gradient descent)
- it can be difficult to implement because the Hessian must be computed
- the Gauss-Newton method approximates Newton's method further, in the case of a special form of objective function

Gauss-Newton

– our objective function has a squared structure:

$$J(\mathbf{x}) = \frac{1}{2}\mathbf{u}(\mathbf{x})^T\mathbf{u}(\mathbf{x}) \tag{14}$$

– in this case, the Jacobian and Hessian matrices are

Jacobian:
$$\left.\frac{\partial J(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_{\text{op}}} = \mathbf{u}(\mathbf{x}_{\text{op}})^T\left(\left.\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_{\text{op}}}\right) \tag{15a}$$

Hessian:
$$\left.\frac{\partial^2 J(\mathbf{x})}{\partial \mathbf{x}\partial \mathbf{x}^T}\right|_{\mathbf{x}_{\text{op}}} = \left(\left.\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_{\text{op}}}\right)^T\left(\left.\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_{\text{op}}}\right) + \sum_{i=1}^{M} u_i(\mathbf{x}_{\text{op}})\left(\left.\frac{\partial^2 u_i(\mathbf{x})}{\partial \mathbf{x}\partial \mathbf{x}^T}\right|_{\mathbf{x}_{\text{op}}}\right) \tag{15b}$$

where $\mathbf{u}(\mathbf{x}) = \big(u_1(\mathbf{x}), \ldots, u_i(\mathbf{x}), \ldots, u_M(\mathbf{x})\big)$

Gauss-Newton

- near the optimum we should have $u_i(\mathbf{x})$ small (and ideally zero)
- we thus approximate the Hessian according to

$$\left.\frac{\partial^2 J(\mathbf{x})}{\partial \mathbf{x} \partial \mathbf{x}^T}\right|_{\mathbf{x}_{\text{op}}} \approx \left(\left.\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_{\text{op}}}\right)^T \left(\left.\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_{\text{op}}}\right) \tag{16}$$

  which does not involve any second derivatives

- the Newton update becomes

$$\left(\left.\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_{\text{op}}}\right)^T \left(\left.\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_{\text{op}}}\right) \delta \mathbf{x}^* = -\left(\left.\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_{\text{op}}}\right)^T \mathbf{u}(\mathbf{x}_{\text{op}}) \tag{17}$$

  which we now call Gauss-Newton

- again, this is iterated to convergence

Gauss-Newton: alternate derivation

- the other way to think about the Gauss-Newton method is to start with a Taylor expansion of $\mathbf{u}(\mathbf{x})$, instead of $J(\mathbf{x})$
- the approximation in this case is

$$\mathbf{u}(\mathbf{x}_{\text{op}} + \delta\mathbf{x}) \approx \mathbf{u}(\mathbf{x}_{\text{op}}) + \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \bigg|_{\mathbf{x}_{\text{op}}} \right) \delta\mathbf{x} \tag{18}$$

- substituting into J we have

$$\begin{aligned} & J(\mathbf{x}_{\text{op}} + \delta\mathbf{x}) \\ & \approx \frac{1}{2} \left(\mathbf{u}(\mathbf{x}_{\text{op}}) + \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \bigg|_{\mathbf{x}_{\text{op}}} \right) \delta\mathbf{x} \right)^T \left(\mathbf{u}(\mathbf{x}_{\text{op}}) + \left(\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}} \bigg|_{\mathbf{x}_{\text{op}}} \right) \delta\mathbf{x} \right) \end{aligned} \tag{19}$$

Gauss-Newton: alternate derivation

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

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

$$\Rightarrow \left(\frac{\partial\mathbf{u}(\mathbf{x})}{\partial\mathbf{x}}\bigg|_{\mathbf{x}_{\text{op}}}\right)^T\left(\frac{\partial\mathbf{u}(\mathbf{x})}{\partial\mathbf{x}}\bigg|_{\mathbf{x}_{\text{op}}}\right)\delta\mathbf{x}^* = -\left(\frac{\partial\mathbf{u}(\mathbf{x})}{\partial\mathbf{x}}\bigg|_{\mathbf{x}_{\text{op}}}\right)^T\mathbf{u}(\mathbf{x}_{\text{op}}) \quad (20)$$

  which is the same update as before

- we will employ this shortcut to the Gauss-Newton method in a later chapter when confronted with dealing with nonlinearities in the form of rotations

Gauss-Newton: patch 1

- since the Gauss-Newton method is not guaranteed to converge (owing to the approximate Hessian matrix), we can make two practical patches to help with convergence
- first, once the optimal update is computed, $\delta\mathbf{x}^*$, we perform the actual update according to

$$\mathbf{x}_{\text{op}} \leftarrow \mathbf{x}_{\text{op}} + \alpha\,\delta\mathbf{x}^* \tag{21}$$

  where $\alpha \in [0, 1]$ is a user-definable parameter
- performing a line search line search for the best value of α works well in practice
- this works because $\delta\mathbf{x}^*$ is a descent direction; we are just adjusting how far we step in this direction to be a bit more conservative towards robustness (rather than speed)

Gauss-Newton: patch 2

- second, we can use the Levenberg-Marquardt modification to the Gauss-Newton method:

$$\left(\left(\left.\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_{\text{op}}}\right)^T\left(\left.\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_{\text{op}}}\right)+\lambda\mathbf{D}\right)\delta\mathbf{x}^* = -\left(\left.\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_{\text{op}}}\right)^T\mathbf{u}(\mathbf{x}_{\text{op}}) \tag{22}$$

  where $\mathbf{D}$ is a positive diagonal matrix

- when $\mathbf{D}=\mathbf{1}$, we can see that as λ becomes very big the Hessian is relatively small and we have

$$\delta\mathbf{x}^* \approx -\frac{1}{\lambda}\left(\left.\frac{\partial \mathbf{u}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_{\text{op}}}\right)^T\mathbf{u}(\mathbf{x}_{\text{op}}) \tag{23}$$

  which says to take a small step in the direction of the gradient

- when $\lambda = 0$ we recover the usual Gauss-Newton update
- LM can work well in situations when the Hessian is near singular

Maximum a posteriori

– getting back to our specific setup, we recall that

$$\mathbf{u}(\mathbf{x}) = \mathbf{L}\,\mathbf{e}(\mathbf{x}) \tag{24}$$

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

$$\left(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}\right)\delta\mathbf{x}^* = \mathbf{H}^T\mathbf{W}^{-1}\,\mathbf{e}(\mathbf{x}_{\text{op}}) \tag{25}$$

with

$$\mathbf{H} = -\left.\frac{\partial\mathbf{e}(\mathbf{x})}{\partial\mathbf{x}}\right|_{\mathbf{x}_{\text{op}}} \tag{26}$$

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

MAP

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

$$\mathbf{e}_{v,k}(\mathbf{x}_{\text{op}} + \delta\mathbf{x}) \approx \begin{cases} \mathbf{e}_{v,0}(\mathbf{x}_{\text{op}}) - \delta\mathbf{x}_0, & k = 0 \\ \mathbf{e}_{v,k}(\mathbf{x}_{\text{op}}) + \mathbf{F}_{k-1}\delta\mathbf{x}_{k-1} - \delta\mathbf{x}_k, & k = 1 \dots K \end{cases} \tag{27}$$

$$\mathbf{e}_{y,k}(\mathbf{x}_{\text{op}} + \delta\mathbf{x}) \approx \mathbf{e}_{y,k}(\mathbf{x}_{\text{op}}) - \mathbf{G}_k\delta\mathbf{x}_k, \quad k = 0 \dots K \tag{28}$$

where

$$\mathbf{e}_{v,k}(\mathbf{x}_{\text{op}}) \approx \begin{cases} \check{\mathbf{x}}_0 - \mathbf{x}_{\text{op},0}, & k = 0 \\ \mathbf{f}\left(\mathbf{x}_{\text{op},k-1}, \mathbf{v}_k, \mathbf{0}\right) - \mathbf{x}_{\text{op},k}, & k = 1 \dots K \end{cases} \tag{29}$$

$$\mathbf{e}_{y,k}(\mathbf{x}_{\text{op}}) \approx \mathbf{y}_k - \mathbf{g}\left(\mathbf{x}_{\text{op},k}, \mathbf{0}\right), \quad k = 0 \dots K \tag{30}$$

$$\mathbf{F}_{k-1} = \left.\frac{\partial \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{w}_k)}{\partial \mathbf{x}_{k-1}}\right|_{\mathbf{x}_{\text{op},k-1}, \mathbf{v}_k, \mathbf{0}}, \quad \mathbf{G}_k = \left.\frac{\partial \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k)}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_{\text{op},k}, \mathbf{0}} \tag{31}$$

MAP

– plugging these details into the batch Gauss-Newton equations

$$\underbrace{\left(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}\right)}_{\text{block-tridiagonal}}\ \delta\mathbf{x}^* = \mathbf{H}^T\mathbf{W}^{-1}\mathbf{e}(\mathbf{x}_{\text{op}}) \tag{32}$$

we have

$$\delta\mathbf{x} = \begin{bmatrix} \delta\mathbf{x}_0 \\ \delta\mathbf{x}_1 \\ \delta\mathbf{x}_2 \\ \vdots \\ \delta\mathbf{x}_K \end{bmatrix}, \quad \mathbf{H} = \left[\begin{array}{ccccccc} \mathbf{1} & & & & & & \\ -\mathbf{F}_0 & \mathbf{1} & & & & & \\ & -\mathbf{F}_1 & \ddots & & & & \\ & & \ddots & \mathbf{1} & & & \\ & & & -\mathbf{F}_{K-1} & \mathbf{1} & & \\ \hline \mathbf{G}_0 & & & & & & \\ & \mathbf{G}_1 & & & & & \\ & & \mathbf{G}_2 & & & & \\ & & & & \ddots & & \\ & & & & & & \mathbf{G}_K \end{array}\right], \quad \mathbf{e}(\mathbf{x}_{\text{op}}) = \left[\begin{array}{c} \mathbf{e}_{v,0}(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_{v,1}(\mathbf{x}_{\text{op}}) \\ \vdots \\ \mathbf{e}_{v,K}(\mathbf{x}_{\text{op}}) \\ \hline \mathbf{e}_{y,0}(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_{y,1}(\mathbf{x}_{\text{op}}) \\ \vdots \\ \mathbf{e}_{y,K}(\mathbf{x}_{\text{op}}) \end{array}\right], \tag{33}$$

$$\mathbf{W} = \text{diag}\left(\check{\mathbf{P}}_0, \mathbf{Q}'_1, \ldots, \mathbf{Q}'_K, \mathbf{R}'_0, \mathbf{R}'_1, \ldots, \mathbf{R}'_K\right) \tag{34}$$

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

Bayesian inference

- we can also get to the same batch update equations from a Bayesian-inference perspective
- assume we begin with an initial guess for the entire trajectory, $\mathbf{x}_{\text{op}}$
- we can linearize the motion model about this guess and construct a prior over the whole trajectory using all the inputs
- the linearized motion model is

$$\mathbf{x}_k \approx \mathbf{f}(\mathbf{x}_{\text{op},k-1}, \mathbf{v}_k, \mathbf{0}) + \mathbf{F}_{k-1}\left(\mathbf{x}_{k-1} - \mathbf{x}_{\text{op},k-1}\right) + \mathbf{w}'_k \tag{35}$$

where the Jacobian, $\mathbf{F}_{k-1}$, is the same as in the batch MAP case

Bayesian inference: prior

– we can write the prior in lifted form as

$$\mathbf{x} = \mathbf{F}\left(\boldsymbol{\nu} + \mathbf{w}'\right) \tag{36}$$

where $\mathbf{w}' \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}'\right)$ and

$$\boldsymbol{\nu} = \begin{bmatrix} \check{\mathbf{x}}_0 \\ \mathbf{f}(\mathbf{x}_{\mathrm{op},0}, \mathbf{v}_1, \mathbf{0}) - \mathbf{F}_0 \mathbf{x}_{\mathrm{op},0} \\ \mathbf{f}(\mathbf{x}_{\mathrm{op},1}, \mathbf{v}_2, \mathbf{0}) - \mathbf{F}_1 \mathbf{x}_{\mathrm{op},1} \\ \vdots \\ \mathbf{f}(\mathbf{x}_{\mathrm{op},K-1}, \mathbf{v}_K, \mathbf{0}) - \mathbf{F}_{K-1} \mathbf{x}_{\mathrm{op},K-1} \end{bmatrix} \tag{37a}$$

$$\mathbf{F} = \begin{bmatrix} \mathbf{1} & & & & \\ \mathbf{F}_0 & \mathbf{1} & & & \\ \mathbf{F}_1\mathbf{F}_0 & \mathbf{F}_1 & \mathbf{1} & & \\ \vdots & \vdots & \vdots & \ddots & \\ \mathbf{F}_{K-2}\cdots\mathbf{F}_0 & \mathbf{F}_{K-2}\cdots\mathbf{F}_1 & \mathbf{F}_{K-2}\cdots\mathbf{F}_2 & \cdots & \mathbf{1} \\ \mathbf{F}_{K-1}\cdots\mathbf{F}_0 & \mathbf{F}_{K-1}\cdots\mathbf{F}_1 & \mathbf{F}_{K-1}\cdots\mathbf{F}_2 & \cdots & \mathbf{F}_{K-1} \quad \mathbf{1} \end{bmatrix} \tag{37b}$$

$$\mathbf{Q}' = \mathrm{diag}\left(\check{\mathbf{P}}_0, \mathbf{Q}'_1, \mathbf{Q}'_2, \ldots, \mathbf{Q}'_K\right) \tag{37c}$$

Bayesian inference: prior

- for the mean of the prior, $\check{\mathbf{x}}$, we then simply have

$$\check{\mathbf{x}} = E[\mathbf{x}] = E\left[\mathbf{F}\left(\boldsymbol{\nu} + \mathbf{w}'\right)\right] = \mathbf{F}\boldsymbol{\nu} \tag{38}$$

- for the covariance of the prior, $\check{\mathbf{P}}$, we have

$$\check{\mathbf{P}} = E\left[(\mathbf{x} - E[\mathbf{x}])(\mathbf{x} - E[\mathbf{x}])^T\right] = \mathbf{F}\, E\left[\mathbf{w}'\mathbf{w}'^T\right]\mathbf{F}^T = \mathbf{F}\mathbf{Q}'\mathbf{F}^T \tag{39}$$

- thus, the prior can be summarized as

$$\mathbf{x} \sim \mathcal{N}\left(\mathbf{F}\boldsymbol{\nu}, \mathbf{F}\mathbf{Q}'\mathbf{F}^T\right) \tag{40}$$

Bayesian inference: measurement

– the linearized observation model is

$$\mathbf{y}_k \approx \mathbf{g}\left(\mathbf{x}_{\mathrm{op},k}, \mathbf{0}\right) + \mathbf{G}_k\left(\mathbf{x}_k - \mathbf{x}_{\mathrm{op},k}\right) + \mathbf{n}'_k \tag{41}$$

which can be written in lifted form as

$$\mathbf{y} = \mathbf{y}_{\mathrm{op}} + \mathbf{G}\left(\mathbf{x} - \mathbf{x}_{\mathrm{op}}\right) + \mathbf{n}' \tag{42}$$

where $\mathbf{n}' \sim \mathcal{N}(\mathbf{0}, \mathbf{R}')$ and

$$\mathbf{y}_{\mathrm{op}} = \begin{bmatrix} \mathbf{g}(\mathbf{x}_{\mathrm{op},0}, \mathbf{0}) \\ \mathbf{g}(\mathbf{x}_{\mathrm{op},1}, \mathbf{0}) \\ \vdots \\ \mathbf{g}(\mathbf{x}_{\mathrm{op},K}, \mathbf{0}) \end{bmatrix} \tag{43a}$$

$$\mathbf{G} = \mathrm{diag}\left(\mathbf{G}_0, \mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_K\right) \tag{43b}$$

$$\mathbf{R} = \mathrm{diag}\left(\mathbf{R}'_0, \mathbf{R}'_1, \mathbf{R}'_2, \ldots, \mathbf{R}'_K\right) \tag{43c}$$

Bayesian inference: measurement, joint density

– the mean, covariance, and cross-covariance with the state are

$$
\begin{aligned}
E\left[\mathbf{y}\right] &= \mathbf{y}_{\mathrm{op}} + \mathbf{G}\left(\check{\mathbf{x}} - \mathbf{x}_{\mathrm{op}}\right) && \text{(44a)} \\
E\left[(\mathbf{y} - E[\mathbf{y}])(\mathbf{y} - E[\mathbf{y}])^T\right] &= \mathbf{G}\check{\mathbf{P}}\mathbf{G}^T + \mathbf{R}' && \text{(44b)} \\
E\left[(\mathbf{y} - E[\mathbf{y}])(\mathbf{x} - E[\mathbf{x}])^T\right] &= \mathbf{G}\check{\mathbf{P}} && \text{(44c)}
\end{aligned}
$$

– we can now write the joint density for the lifted trajectory and measurements as

$$
p(\mathbf{x}, \mathbf{y}|\mathbf{v}) = \mathcal{N}\left(\begin{bmatrix} \check{\mathbf{x}} \\ \mathbf{y}_{\mathrm{op}} + \mathbf{G}\left(\check{\mathbf{x}} - \mathbf{x}_{\mathrm{op}}\right) \end{bmatrix}, \begin{bmatrix} \check{\mathbf{P}} & \check{\mathbf{P}}\mathbf{G}^T \\ \mathbf{G}\check{\mathbf{P}} & \mathbf{G}\check{\mathbf{P}}\mathbf{G}^T + \mathbf{R}' \end{bmatrix}\right) \tag{45}
$$

which is quite similar to the expression for the IEKF, but now for the whole trajectory rather than just one timestep

Bayesian inference: posterior

– using the usual two-step approach, we can immediately write the Gaussian posterior from the components of the joint density as

$$p(\mathbf{x}|\mathbf{v},\mathbf{y}) = \mathcal{N}\left(\hat{\mathbf{x}}, \hat{\mathbf{P}}\right) \tag{46}$$

where

$$\begin{aligned}
\mathbf{K} &= \check{\mathbf{P}}\mathbf{G}^T\left(\mathbf{G}\check{\mathbf{P}}\mathbf{G}^T + \mathbf{R}'\right)^{-1} && \text{(47a)}\\
\hat{\mathbf{P}} &= \left(\mathbf{1} - \mathbf{K}\mathbf{G}\right)\check{\mathbf{P}} && \text{(47b)}\\
\hat{\mathbf{x}} &= \check{\mathbf{x}} + \mathbf{K}\left(\mathbf{y} - \mathbf{y}_{\text{op}} - \mathbf{G}(\check{\mathbf{x}} - \mathbf{x}_{\text{op}})\right) && \text{(47c)}
\end{aligned}$$

– again, this looks a lot like the IEKF correction step, but it's now for the whole trajectory at once

Bayesian inference: posterior

– using the SMW identity, we can rearrange the equation for the posterior mean to be

$$\left(\check{\mathbf{P}}^{-1} + \mathbf{G}^T \mathbf{R'}^{-1} \mathbf{G}\right) \delta\mathbf{x}^* = \check{\mathbf{P}}^{-1} \left(\check{\mathbf{x}} - \mathbf{x}_{\text{op}}\right) + \mathbf{G}^T \mathbf{R'}^{-1} \left(\mathbf{y} - \mathbf{y}_{\text{op}}\right) \tag{48}$$

where $\delta\mathbf{x}^* = \hat{\mathbf{x}} - \mathbf{x}_{\text{op}}$

– inserting the details of the prior this becomes

$$\underbrace{\left(\mathbf{F}^{-T} \mathbf{Q'}^{-1} \mathbf{F}^{-1} + \mathbf{G}^T \mathbf{R'}^{-1} \mathbf{G}\right)}_{\text{block-tridiagonal}} \delta\mathbf{x}^*$$

$$= \mathbf{F}^{-T} \mathbf{Q'}^{-1} \left(\boldsymbol{\nu} - \mathbf{F}^{-1} \mathbf{x}_{\text{op}}\right) + \mathbf{G}^T \mathbf{R'}^{-1} \left(\mathbf{y} - \mathbf{y}_{\text{op}}\right) \tag{49}$$

Bayesian inference: posterior

– then, under the definitions

$$\mathbf{H} = \begin{bmatrix} \mathbf{F}^{-1} \\ \mathbf{G} \end{bmatrix}, \quad \mathbf{W} = \text{diag}\left(\mathbf{Q}', \mathbf{R}'\right), \quad \mathbf{e}(\mathbf{x}_{\text{op}}) = \begin{bmatrix} \boldsymbol{\nu} - \mathbf{F}^{-1}\mathbf{x}_{\text{op}} \\ \mathbf{y} - \mathbf{y}_{\text{op}} \end{bmatrix} \tag{50}$$

we can rewrite this as

$$\underbrace{\left(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}\right)}_{\text{block-tridiagonal}} \delta\mathbf{x}^* = \mathbf{H}^T\mathbf{W}^{-1}\mathbf{e}(\mathbf{x}_{\text{op}}) \tag{51}$$

which is identical to the update equation from the MAP approach

– as usual, we iterate to convergence

Bayesian inference commentary

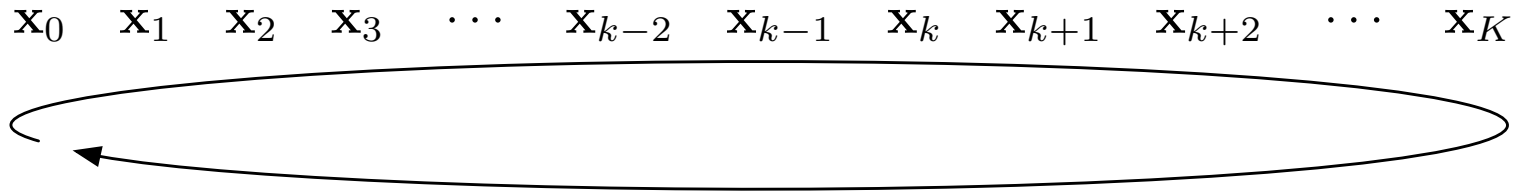
- the difference between the Bayesian and MAP approaches basically comes down to on which side of the SMW identity one begins
- the Bayesian approach produces a covariance explicitly, although we have shown the same thing can be extracted from the MAP approach
- it was our choice to iteratively relinearize about the mean of the best estimate so far that caused the Bayesian approach to have the same 'mean' as the MAP solution (like in the IEKF)
- we could also imagine making different choices than linearization in the batch case (e.g., particles, sigmapoints) to compute the required moments for the update equations, but we will not explore these possibilities here

Nonlinear discussion

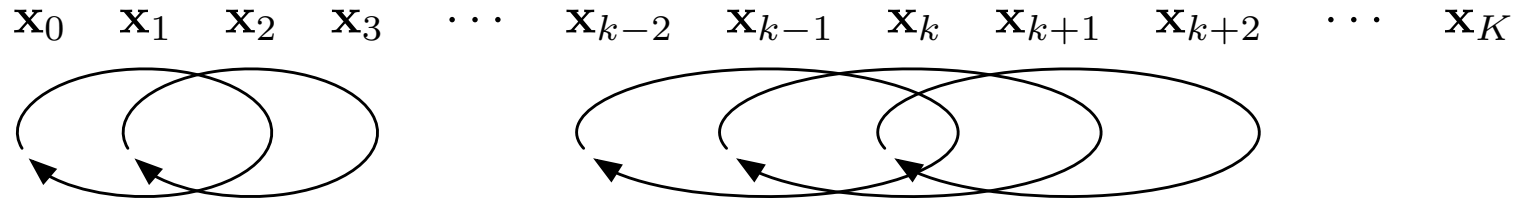
- if we think of the EKF as an approximation of the full nonlinear Gauss-Newton (or even Newton) method applied to our estimation problem, we can see that it is really quite inferior
- the Jacobians are evaluated only once (at the best estimate so far)
- in truth, the EKF can do better than just one iteration of Gauss-Newton because it does not evaluate all the Jacobians at once, but the lack of iteration is its main downfall
- this is obvious from an optimization perspective; we need to iterate to converge

Iterating is key

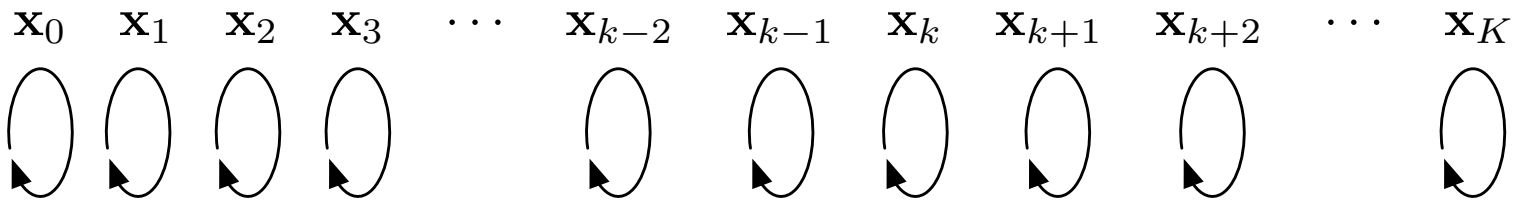
Gauss-Newton iterates over the entire trajectory, but runs offline and not in constant time



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



Achilles heel

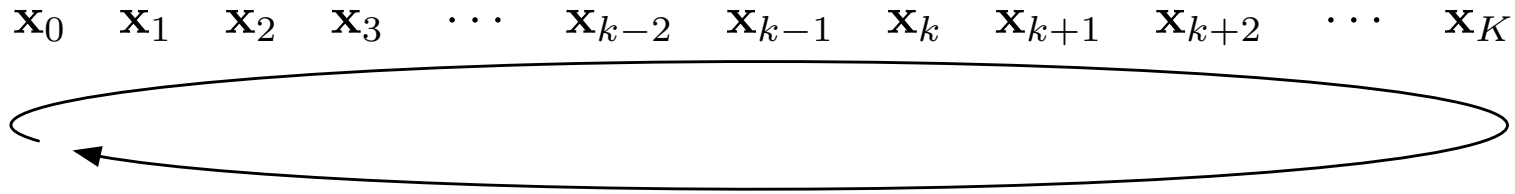
- we derived the EKF from the Bayes filter, which used something called the Markov assumption to achieve its recursive form
- the problem with the Markov assumption is that once this is built into the estimator, we cannot get rid of it; it is a fundamental constraint that cannot be overcome
- there have been many attempts to patch the EKF, including the Iterated EKF described earlier in this chapter
- however, for very nonlinear systems, these may not help much
- the problem with the IEKF is that it still clings to the Markov assumption
- it is iterating at a single time-step, not over the whole trajectory at once

Middle ground

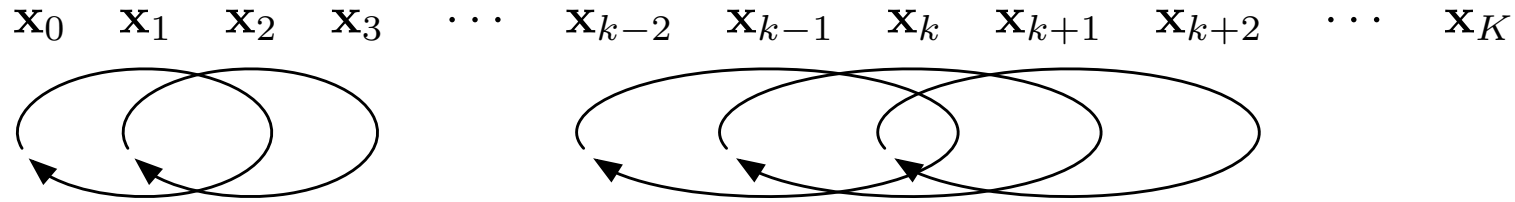
- batch estimation via the Gauss-Newton method has its own problems
- in particular, it must be run offline and is not a constant-time algorithm, whereas the EKF is both online and a constant-time method (although there are some ways of incrementally updating the batch solution)
- so-called sliding-window filters (SWFs) seek to get the best of both worlds by iterating over a window of time-steps and sliding this window along to allow for online/constant-time implementation
- SWFs are really still an active area of research, but when viewed from an optimization perspective, it is hard to imagine that they do not offer a drastic improvement over the EKF and its variants

Iterating is key

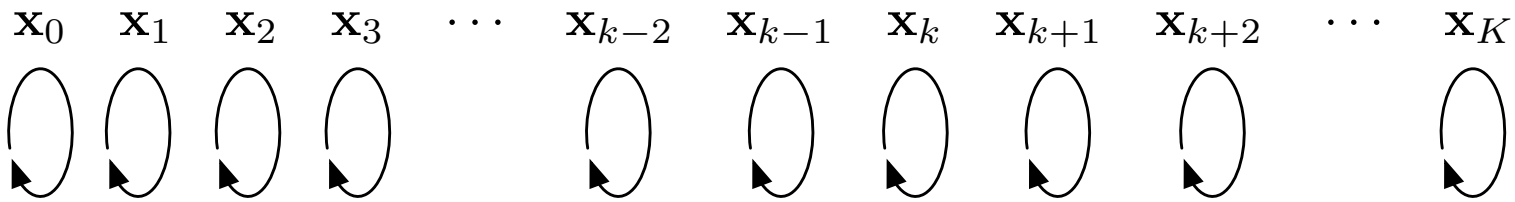
Gauss-Newton iterates over the entire trajectory, but runs offline and not in constant time



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



Summary

- we derived batch estimators for the nonlinear, non-Gaussian situation
- these can be viewed from an optimization perspective – we're just minimizing a sum-of-squares cost function
- iteration is key to converging to the minimum