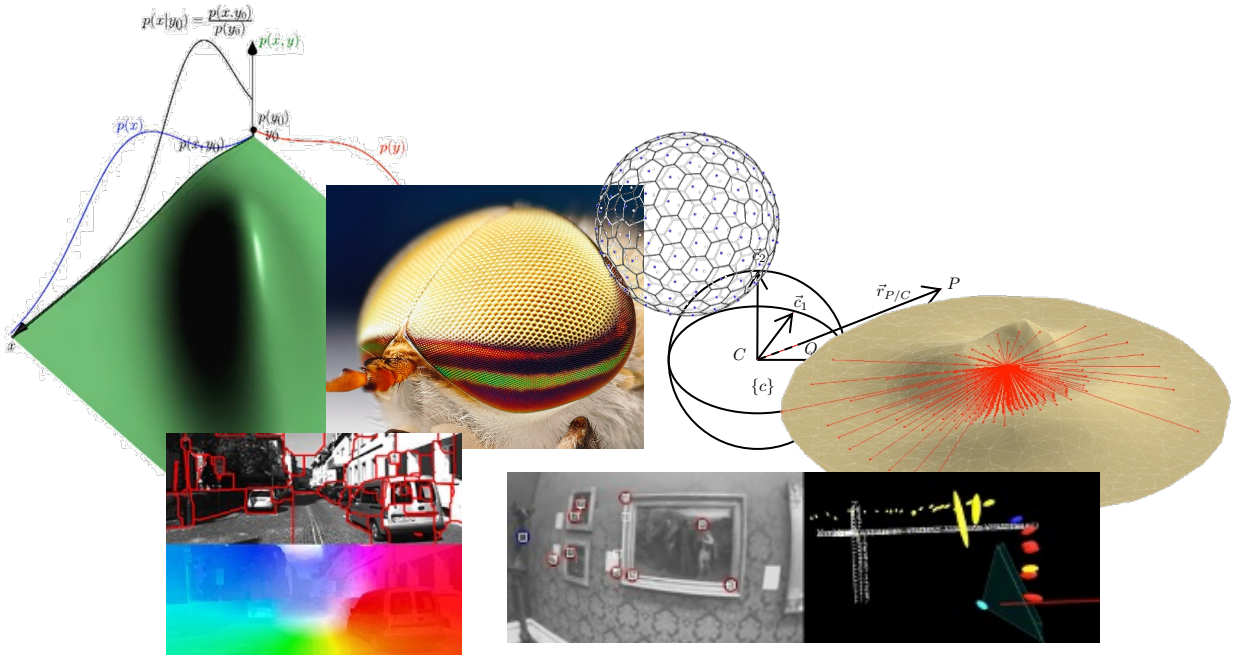
MCHA4400

Vision-based navigation



Dr Chris Renton

School of Engineering

Gaussian filter

If we assume that the initial state and the process and the measurement models are Gaussian

$$p(\mathbf{x}_1) = \mathcal{N}(\mathbf{x}_1; \boldsymbol{\mu}_{1|0}, \mathbf{P}_{1|0})$$
$$p(\mathbf{x}_{k+1}|\mathbf{x}_k) = \mathcal{N}(\mathbf{x}_k; \mathbf{f}_k(\mathbf{x}), \mathbf{Q}_k)$$
$$p(\mathbf{y}_k|\mathbf{x}_k) = \mathcal{N}(\mathbf{y}_k; \mathbf{h}_k(\mathbf{x}), \mathbf{R}_k)$$

then the Bayes filter

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) = \frac{p(\mathbf{y}_k|\mathbf{x}_k)\, p(\mathbf{x}_k|\mathbf{y}_{1:k-1})}{p(\mathbf{y}_k|\mathbf{y}_{1:k-1})}$$

$$p(\mathbf{x}_{k+1}|\mathbf{y}_{1:k}) = \int_{\mathcal{X}} p(\mathbf{x}_{k+1}|\mathbf{x}_k)\, p(\mathbf{x}_k|\mathbf{y}_{1:k})\, \mathrm{d}\mathbf{x}_k,$$

can result in non-Gaussian state distributions due to the nonlinearity in the process and measurement mean functions, $\mathbf{f}_k(\mathbf{x})$ and $\mathbf{h}_k(\mathbf{x})$.

Gaussian filter

In a Gaussian filter, we *assert* that the filtered and predicted state distributions are Gaussian at each time step, i.e.,

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) \approx \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k}, \mathbf{P}_{k|k})$$
$$p(\mathbf{x}_{k+1}|\mathbf{y}_{1:k}) \approx \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k+1|k}, \mathbf{P}_{k+1|k})$$

If we accept these assertions,

- How can we obtain the "best" Gaussian approximation to the filtered state distribution?
- Can we extend the filter to handle a non-Gaussian measurement likelihood function?

We will address both questions by finding the "best" Gaussian approximation to the filtered state distribution:

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) \propto p(\mathbf{y}_k|\mathbf{x}_k)\, p(\mathbf{x}_k|\mathbf{y}_{1:k-1}) = p(\mathbf{y}_k|\mathbf{x}_k)\, \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}, \mathbf{P}_{k|k-1})$$

Laplace approximation

Laplace approximation

Laplace (1774) developed a method to approximate integrals of well-behaved unimodal functions by approximating them with a Gaussian distribution.

Suppose we have a positive square-integrable function $f : \mathcal{X} \to \mathbb{R}^+$, which achieves its maximum at $\mathbf{x}^*$ and we want to compute the following integral:

$$\mathcal{I} = \int_{\mathcal{D}} f(\mathbf{x})\,\mathrm{d}\mathbf{x} \qquad \int_{\mathcal{D}} f(\mathbf{x})^2\,\mathrm{d}\mathbf{x} < \infty$$

Let $\ell(\mathbf{x}) = \log f(\mathbf{x})$ so that we have

$$\mathcal{I} = \int_{\mathcal{D}} \exp \ell(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

Laplace approximation

Replace $\ell(\mathbf{x})$ with its 2nd order Taylor series approximation about $\mathbf{x} = \mathbf{x}^*$,

$$\mathcal{I} \approx \int_{\mathcal{D}} \exp\left(\ell(\mathbf{x}^*) + \left.\frac{\partial^\mathsf{T}\ell}{\partial\mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^*}(\mathbf{x}-\mathbf{x}^*) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^*)^\mathsf{T}\left.\frac{\partial^2\ell}{\partial\mathbf{x}^2}\right|_{\mathbf{x}=\mathbf{x}^*}(\mathbf{x}-\mathbf{x}^*)\right)\mathrm{d}\mathbf{x}$$

Since $\mathbf{x}^*$ is a maximiser of $\ell(\mathbf{x})$, $\left.\frac{\partial\ell}{\partial\mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^*} = \mathbf{0}$. Let $\mathbf{P} = -\left(\left.\frac{\partial^2\ell}{\partial\mathbf{x}^2}\right|_{\mathbf{x}=\mathbf{x}^*}\right)^{-1}$

Then,
$$\begin{aligned}\mathcal{I} &\approx \int_{\mathcal{D}} \exp\left(\ell(\mathbf{x}^*) - \frac{1}{2}(\mathbf{x}-\mathbf{x}^*)^\mathsf{T}\mathbf{P}^{-1}(\mathbf{x}-\mathbf{x}^*)\right)\mathrm{d}\mathbf{x}\\ &= \exp\ell(\mathbf{x}^*)\int_{\mathcal{D}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{x}^*)^\mathsf{T}\mathbf{P}^{-1}(\mathbf{x}-\mathbf{x}^*)\right)\mathrm{d}\mathbf{x}\\ &= f(\mathbf{x}^*)\sqrt{\det 2\pi\mathbf{P}}\int_{\mathcal{D}}\mathcal{N}(\mathbf{x};\mathbf{x}^*,\mathbf{P})\,\mathrm{d}\mathbf{x}.\end{aligned}$$

If $\mathcal{D} = \mathcal{X}$, then the integral is equal to one.

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Laplace approximation

The Laplace approximation of the integral of $f(\mathbf{x})$ is equal to the value of the function evaluated at its mode multiplied by a constant that depends on the curvature of f,

$$\mathcal{I} = \int_{\mathcal{X}} f(\mathbf{x})\,\mathrm{d}\mathbf{x} \approx f(\mathbf{x}^*)\sqrt{\det 2\pi\mathbf{P}}$$

where

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} f(\mathbf{x})$$

$$\mathbf{P} = -\left(\frac{\partial^2}{\partial \mathbf{x}^2} \log f(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}^*} \right)^{-1}$$

The Laplace approximation replaces the problem of integrating a function with the problem of maximising it—an optimisation problem.

It is typically easier to maximise a function using modern optimisation tools than it is to integrate one.

Laplace approximation

We can also use this idea to find the Laplace approximation of a pdf $p(\mathbf{x})$.
Let us approximate $\log p(\mathbf{x})$ by its second-order Taylor series about its mode,

$$\log p(\mathbf{x}) \approx \log p(\boldsymbol{\mu}) + \frac{\partial^{\mathsf{T}}}{\partial \mathbf{x}} \log p(\mathbf{x})\Big|_{\mathbf{x}=\boldsymbol{\mu}} (\mathbf{x}-\boldsymbol{\mu}) + \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}} \frac{\partial^2}{\partial \mathbf{x}^2} \log p(\mathbf{x})\Big|_{\mathbf{x}=\boldsymbol{\mu}} (\mathbf{x}-\boldsymbol{\mu}).$$

where the mode is given by

$$\boldsymbol{\mu} = \arg\max_{\mathbf{x}} p(\mathbf{x})$$

Since $\log p(\mathbf{x})$ achieves its maximum at $\boldsymbol{\mu}$, its gradient is zero at this point and

$$\log p(\mathbf{x}) \approx \log p(\boldsymbol{\mu}) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\mathbf{P}^{-1}(\mathbf{x}-\boldsymbol{\mu})$$

where

$$\mathbf{P} = -\left(\frac{\partial^2}{\partial \mathbf{x}^2} \log p(\mathbf{x})\Big|_{\mathbf{x}=\boldsymbol{\mu}}\right)^{-1}$$

Laplace approximation

Let
$$f(\mathbf{x}) = \exp\left(\log p(\boldsymbol{\mu}) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\mathbf{P}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$
$$= p(\boldsymbol{\mu})\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\mathbf{P}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

which is non-negative for all $\mathbf{x}$, and let Z be the normalising constant,

$$Z = \int_{\mathcal{X}} f(\mathbf{x})\,\mathrm{d}\mathbf{x} = p(\boldsymbol{\mu})\int_{\mathcal{X}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\mathbf{P}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)\,\mathrm{d}\mathbf{x}$$
$$= p(\boldsymbol{\mu})\sqrt{\det 2\pi\mathbf{P}}$$

Then,
$$p(\mathbf{x}) \approx \frac{1}{Z}f(\mathbf{x}) = \frac{1}{\sqrt{\det 2\pi\mathbf{P}}}\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\mathbf{P}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$
$$= \mathcal{N}(\mathbf{x};\boldsymbol{\mu},\mathbf{P})$$

Laplace approximation

Lemma (Laplace approximation of a pdf): Given a pdf $p(\mathbf{x})$, where $\mathbf{x} \in \mathcal{X}$, the Laplace approximation at the MAP of $p(\mathbf{x})$ is given by

$$p(\mathbf{x}) \approx \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \mathbf{P}),$$

where

$$\begin{aligned} \boldsymbol{\mu} &= \arg\min_{\mathbf{x}} \mathcal{V}(\mathbf{x}), \\ \mathbf{P}^{-1} &= \left. \frac{\partial^2 \mathcal{V}}{\partial \mathbf{x}^2} \right|_{\mathbf{x}=\boldsymbol{\mu}}, \\ \mathcal{V}(\mathbf{x}) &= -\log p(\mathbf{x}). \end{aligned}$$

Laplace approximation

Let us apply the Laplace approximation to obtain a Gaussian approximation to the filtered density

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) \propto p(\mathbf{y}_k|\mathbf{x}_k)\,\mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}, \mathbf{P}_{k|k-1})$$

which yields

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) \approx \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k}, \mathbf{P}_{k|k})$$

where

$$\boldsymbol{\mu}_{k|k} = \arg\min_{\mathbf{x}_k} \mathcal{V}(\mathbf{x}_k), \qquad \mathbf{P}_{k|k} = \left(\frac{\partial^2 \mathcal{V}}{\partial \mathbf{x}_k^2}\bigg|_{\mathbf{x}_k = \boldsymbol{\mu}_{k|k}} \right)^{-1}$$

and

$$\begin{aligned}\mathcal{V}(\mathbf{x}_k) &= -\log \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}, \mathbf{P}_{k|k-1}) - \log p(\mathbf{y}_k|\mathbf{x}_k) \\ &= \frac{1}{2}(\mathbf{x}_k - \boldsymbol{\mu}_{k|k-1})^{\mathsf{T}} \mathbf{P}_{k|k-1}^{-1} (\mathbf{x}_k - \boldsymbol{\mu}_{k|k-1}) + \frac{1}{2} \log\det 2\pi \mathbf{P}_{k|k-1} - \log p(\mathbf{y}_k|\mathbf{x}_k)\end{aligned}$$

Square-root implementation of cost

Given an upper triangular matrix $\mathbf{S}_{k|k\text{-}1}$ such that $\mathbf{S}_{k|k-1}^{\mathsf{T}}\mathbf{S}_{k|k-1} = \mathbf{P}_{k|k-1}$ we can evaluate the cost as follows:

$$\begin{aligned}
\mathcal{V}(\mathbf{x}_k) &= \frac{1}{2}(\mathbf{x}_k - \boldsymbol{\mu}_{k|k-1})^{\mathsf{T}} \left(\mathbf{S}_{k|k-1}^{\mathsf{T}}\mathbf{S}_{k|k-1}\right)^{-1} (\mathbf{x}_k - \boldsymbol{\mu}_{k|k-1}) + \frac{1}{2}\log(2\pi)^n \\
&\quad + \frac{1}{2}\log\det \mathbf{S}_{k|k-1}^{\mathsf{T}}\mathbf{S}_{k|k-1} - \log p(\mathbf{y}_k|\mathbf{x}_k) \\
&= \frac{1}{2}(\mathbf{x}_k - \boldsymbol{\mu}_{k|k-1})^{\mathsf{T}}\mathbf{S}_{k|k-1}^{-1}\mathbf{S}_{k|k-1}^{-\mathsf{T}}(\mathbf{x}_k - \boldsymbol{\mu}_{k|k-1}) + \frac{n}{2}\log 2\pi \\
&\quad + \log|\det \mathbf{S}_{k|k-1}| - \log p(\mathbf{y}_k|\mathbf{x}_k) \\
&= \frac{1}{2}\left(\mathbf{S}_{k|k-1}^{-\mathsf{T}}(\mathbf{x}_k - \boldsymbol{\mu}_{k|k-1})\right)^{\mathsf{T}}\mathbf{S}_{k|k-1}^{-\mathsf{T}}(\mathbf{x}_k - \boldsymbol{\mu}_{k|k-1}) + \frac{n}{2}\log 2\pi \\
&\quad + \log\left|\prod_{i=1}^{n}(\mathbf{S}_{k|k-1})_{ii}\right| - \log p(\mathbf{y}_k|\mathbf{x}_k)
\end{aligned}$$

$$\mathcal{V}(\mathbf{x}_k) = \frac{1}{2}\left\|\mathbf{S}_{k|k-1}^{-\mathsf{T}}(\mathbf{x}_k - \boldsymbol{\mu}_{k|k-1})\right\|^2 + \frac{n}{2}\log 2\pi + \sum_{i=1}^{n}\log|\mathbf{S}_{k|k-1}|_{ii} - \log p(\mathbf{y}_k|\mathbf{x}_k)$$

where $|\cdot|_{ii}$ denotes the absolute value of the i^{th} diagonal element of its argument.

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Unconstrained optimisation

Unconstrained optimisation

Consider the following unconstrained optimisation problem:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x})$$

Some methods for unconstrained optimisation:

- Gradient descent
- Newton's method
- quasi-Newton methods

Newton's method

Given the k^{th} iterate $\mathbf{x}_k$, Newton's method assumes the change in cost due to a perturbation in the decision variables, $\mathbf{p}$, is well approximated to second order, i.e.,

$$f(\mathbf{x}_k + \mathbf{p}) \approx m(\mathbf{x}_k + \mathbf{p}) = f_k + \mathbf{g}_k^\mathsf{T}\mathbf{p} + \frac{1}{2}\mathbf{p}^\mathsf{T}\mathbf{H}_k\mathbf{p},$$

where

$$\begin{aligned} f_k &= f(\mathbf{x}_k), \\ \mathbf{g}_k &= \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_k}, \\ \mathbf{H}_k &= \left.\frac{\partial^2 f}{\partial \mathbf{x}^2}\right|_{\mathbf{x}=\mathbf{x}_k}. \end{aligned}$$

Newton's method

The gradient of the perturbed cost is given by

$$\mathbf{g}(\mathbf{x}_k + \mathbf{p}) \approx \mathbf{g}_k + \mathbf{H}_k \mathbf{p}$$

Setting this gradient to zero at $\mathbf{p} = \mathbf{p}_k$ yields the Newton step,

$$\mathbf{p}_k = -\mathbf{H}_k^{-1} \mathbf{g}_k$$

which can be used to determine the next iterate,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k$$

Newton's method

At the solution, we expect $\|\mathbf{g}(\mathbf{x}^*)\| = 0$, so we could terminate when $\|\mathbf{g}(\mathbf{x}_k)\| < \delta$ for some k where δ is a small tolerance. The problem with this idea is that the tolerance depends on the scale of the cost function and decision variables.

$$\|\mathbf{g}(\mathbf{x})\| = \sqrt{\mathbf{g}(\mathbf{x})^\mathsf{T}\mathbf{g}(\mathbf{x})}$$

To overcome this, we consider weighting the norm of the gradient by the inverse Hessian, which yields the *Newton decrement*,

$$\lambda(\mathbf{x}) = \|\mathbf{g}(\mathbf{x})\|_{\mathbf{H}(\mathbf{x})^{-1}} = \sqrt{\mathbf{g}(\mathbf{x})^\mathsf{T}\mathbf{H}(\mathbf{x})^{-1}\mathbf{g}(\mathbf{x})}$$

and terminate when

$$\frac{\lambda(\mathbf{x}_k)^2}{2} \leq \epsilon$$

for some iteration k, where ϵ is the machine epsilon.

The Newton decrement is invariant to an affine change of coordinates.

Newton's method

If the Hessian is not positive definite, taking the Newton step can lead to an increase in the cost, since Newton's method only seeks a stationary point.

We can monitor the quality of the Newton step by comparing the actual reduction in cost to the predicted reduction in cost.

Let

$$\rho_k = \frac{f(\mathbf{x}_k) - f(\mathbf{x}_k + \mathbf{p}_k)}{m(\mathbf{x}_k) - m(\mathbf{x}_k + \mathbf{p}_k)}$$

where $\mathbf{p}_k$ is the Newton step and $m(\cdot)$ is the quadratic model of the cost.

If $\rho_k \approx 1$, then the cost function is well approximated by the quadratic model.

If $\rho_k < 0$, then we should not trust the Newton step.

Trust-region optimisation

When the quadratic model is a poor approximation to the cost function, we should not take the Newton step.

Instead of taking the Newton step, we can employ a trust-region method to determine the step,

$$\mathbf{p}_k = \arg\min_{\mathbf{p}} \mathbf{g}_k^\mathsf{T}\mathbf{p} + \frac{1}{2}\mathbf{p}^\mathsf{T}\mathbf{H}_k\mathbf{p}$$
$$\text{subject to } \|\mathbf{p}\| \leq \Delta_k$$

where Δ_k is the trust region radius that is determined based on the how closely the cost function behaves like the quadratic model.

For example,

- if $\rho_k < 0.1$, then set $\Delta_{k+1} = 0.25\ \|\mathbf{p}_k\|$
- if $\rho_k > 0.75$ and $\|\mathbf{p}_k\| > 0.8\ \Delta_k$, then set $\Delta_{k+1} = 2\ \Delta_k$.

Trust-region subproblem

The trust-region subproblem is an inequality constrained optimisation problem,

$$\mathbf{p}_k = \arg\min_{\mathbf{p}} \mathbf{g}_k^\mathsf{T}\mathbf{p} + \frac{1}{2}\mathbf{p}^\mathsf{T}\mathbf{H}_k\mathbf{p}$$

$$\text{subject to } \|\mathbf{p}\| \leq \Delta_k$$

To solve this, we can examine the solutions for the two cases where the constraint is inactive or active at the solution $\mathbf{p}_k$, i.e.,

- $\|\mathbf{p}_k\| < \Delta_k$, which has solution given by the Newton step $\mathbf{p}_k = -\mathbf{H}_k^{-1}\mathbf{g}_k$
- $\|\mathbf{p}_k\| = \Delta_k$, which we can solve by the augmented Lagrangian method.

Trust-region subproblem

To solve the active constraint case, we can solve the equality constrained optimisation problem,

$$\mathbf{p}_k = \arg\min_{\mathbf{p}} \mathbf{g}_k^\mathsf{T}\mathbf{p} + \frac{1}{2}\mathbf{p}^\mathsf{T}\mathbf{H}_k\mathbf{p}, \quad \text{subject to } \|\mathbf{p}\| = \Delta_k$$

by minimising the following Lagrangian:

$$\begin{aligned}\mathcal{L}(\mathbf{p}, \lambda) &= \mathbf{g}_k^\mathsf{T}\mathbf{p} + \frac{1}{2}\mathbf{p}^\mathsf{T}\mathbf{H}_k\mathbf{p} + \frac{1}{2}\lambda(\mathbf{p}^\mathsf{T}\mathbf{p} - \Delta_k^2) \\ &= \mathbf{g}_k^\mathsf{T}\mathbf{p} + \frac{1}{2}\mathbf{p}^\mathsf{T}(\mathbf{H}_k + \lambda\mathbf{I})\mathbf{p} - \frac{1}{2}\lambda\Delta_k^2\end{aligned}$$

where λ is the Lagrangian multiplier.

Trust-region subproblem

Setting the gradient of the Lagrangian w.r.t. $\mathbf{p}$ at the solution to zero yields

$$\left.\frac{\partial \mathcal{L}}{\partial \mathbf{p}}\right|_{\substack{\mathbf{p}=\mathbf{p}_k \\ \lambda=\lambda_k}} = \mathbf{g}_k + (\mathbf{H}_k + \lambda_k \mathbf{I})\mathbf{p}_k = \mathbf{0}$$

$$\mathbf{p}_k = -(\mathbf{H}_k + \lambda_k \mathbf{I})^{-1}\mathbf{g}_k \qquad (*)$$

The gradient of the Lagrangian w.r.t. λ simply yields the constraint,

$$\left.\frac{\partial \mathcal{L}}{\partial \lambda}\right|_{\substack{\mathbf{p}=\mathbf{p}_k \\ \lambda=\lambda_k}} = \frac{1}{2}(\mathbf{p}_k^\mathsf{T}\mathbf{p}_k - \Delta_k^2) = 0$$

Substituting the former into the latter yields

$$\mathbf{g}_k^\mathsf{T}(\mathbf{H}_k + \lambda_k \mathbf{I})^{-2}\mathbf{g}_k = \Delta_k^2$$

which can be solved for λ_k and then substituted into (*) to obtain $\mathbf{p}_k$.

Trust-region subproblem

More and Sorensen (1983) solve λ_k by performing a Newton-Raphson search on the so-called *secular equation*,

$$\frac{1}{\Delta_k} - \frac{1}{\|(\mathbf{H}_k + \lambda \mathbf{I})^{-1}\mathbf{g}_k\|} = 0$$

for λ while ensuring that $\lambda \geq 0$ and $\lambda \geq \lambda_{\min}$, where $\lambda_{\min}$ is the minimum eigenvalue of $\mathbf{H}_k$.

The secular equation is used since it is almost linear in λ for $\lambda \geq \lambda_{\min}$ and the solution typically converges in 5 or fewer iterations.

This relies on the fact that if λ is a scalar such that $\mathbf{H}_k + \lambda\mathbf{I}$ is positive semi-definite, then the solution of

$$(\mathbf{H}_k + \lambda \mathbf{I})\mathbf{p}_k = -\mathbf{g}_k$$

for $\mathbf{p}_k$ is a solution to the trust-region subproblem, if $\lambda = 0$ and $\|\mathbf{p}_k\| < \Delta_k$ or $\lambda \geq 0$ and $\|\mathbf{p}_k\| = \Delta_k$.

Require: Twice differentiable cost function $f\colon \mathcal{X} \to \mathbb{R}$
Require: Initial point $\mathbf{x}_0$

function FMINNEWTONTRUST($f(\cdot)$, $\mathbf{x}_0$)
  $k \leftarrow 0$
  $(f_k, \mathbf{g}_k, \mathbf{H}_k) \leftarrow (f(\mathbf{x}_k), \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_k}, \left.\frac{\partial^2 f}{\partial \mathbf{x}^2}\right|_{\mathbf{x}=\mathbf{x}_k})$
  $\Delta_k \leftarrow 1$ ▷ Initial trust region radius
  repeat
    $\mathbf{p}_k \leftarrow \arg\min_{\mathbf{p}} \mathbf{g}_k^\mathsf{T}\mathbf{p} + \frac{1}{2}\mathbf{p}^\mathsf{T}\mathbf{H}_k\mathbf{p}$, s.t. $\|\mathbf{p}\| \leq \Delta_k$ ▷ Solve TRS
    if $-\mathbf{p}_k^\mathsf{T}\mathbf{g}_k < 2\epsilon$ **then** ▷ If Newton decrement converged
      return $\mathbf{x}_k$
    end if
    $\tilde{\mathbf{x}} \leftarrow \mathbf{x}_k + \mathbf{p}_k$ ▷ Trial step
    $(\tilde{f}, \tilde{\mathbf{g}}, \tilde{\mathbf{H}}) \leftarrow (f(\tilde{\mathbf{x}}), \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}=\tilde{\mathbf{x}}}, \left.\frac{\partial^2 f}{\partial \mathbf{x}^2}\right|_{\mathbf{x}=\tilde{\mathbf{x}}})$ ▷ Trial cost, gradient, Hessian
    $\rho_k \leftarrow \frac{f_k - \tilde{f}}{-\mathbf{p}_k^\mathsf{T}\mathbf{g}_k - \frac{1}{2}\mathbf{p}_k^\mathsf{T}\mathbf{H}_k\mathbf{p}_k}$ ▷ Trial step quality
    if $\rho_k < 0.25$ **then**
      $\Delta_{k+1} \leftarrow \frac{1}{4}\|\mathbf{p}_k\|$ ▷ Decrease trust region radius
    else if $\rho_k > 0.75$ and $\|\mathbf{p}_k\| > 0.8\Delta_k$ **then**
      $\Delta_{k+1} \leftarrow 2\Delta_k$ ▷ Increase trust region radius
    else
      $\Delta_{k+1} \leftarrow \Delta_k$ ▷ Preserve trust region radius
    end if
    if $\rho_k > 0.001$ **then**
      $(\mathbf{x}_{k+1}, f_{k+1}, \mathbf{g}_{k+1}, \mathbf{H}_{k+1}) \leftarrow (\tilde{\mathbf{x}}, \tilde{f}, \tilde{\mathbf{g}}, \tilde{\mathbf{H}})$ ▷ Accept trial step
    else
      $(\mathbf{x}_{k+1}, f_{k+1}, \mathbf{g}_{k+1}, \mathbf{H}_{k+1}) \leftarrow (\mathbf{x}_k, f_k, \mathbf{g}_k, \mathbf{H}_k)$ ▷ Reject trial step
    end if
    $k \leftarrow k + 1$
  until death
end function

Square-root trust-region subproblem

Recall the standard trust-region subproblem:

$$\mathbf{p}_k = \arg\min_{\mathbf{p}} \mathbf{g}_k^\mathsf{T}\mathbf{p} + \frac{1}{2}\mathbf{p}^\mathsf{T}\mathbf{H}_k\mathbf{p}, \text{ subject to } \|\mathbf{p}\| \leq \Delta_k$$

Assume we have a square-root factorisation of the inverse Hessian,

$$\mathbf{H}_k^{-1} = \mathbf{S}_k^\mathsf{T}\mathbf{S}_k$$

First, we modify the shape of the trust region as follows:

$$\mathbf{p}_k = \arg\min_{\mathbf{p}} \mathbf{g}_k^\mathsf{T}\mathbf{p} + \frac{1}{2}\mathbf{p}^\mathsf{T}\mathbf{S}_k^{-1}\mathbf{S}_k^{-\mathsf{T}}\mathbf{p}, \text{ subject to } \|\mathbf{S}_k^{-\mathsf{T}}\mathbf{p}\| \leq \Delta_k$$

Then, let

$$\mathbf{z} = \mathbf{S}_k^{-\mathsf{T}}\mathbf{p}$$

$$\tilde{\mathbf{g}}_k = \mathbf{S}_k\mathbf{g}_k$$

and rewrite the modified TRS as follows:

$$\mathbf{z}_k = \arg\min_{\mathbf{z}} \tilde{\mathbf{g}}_k^\mathsf{T}\mathbf{z} + \frac{1}{2}\mathbf{z}^\mathsf{T}\mathbf{z}, \text{ subject to } \|\mathbf{z}\| \leq \Delta_k$$

Square-root trust-region subproblem

$$\mathbf{z}_k = \arg\min_{\mathbf{z}} \tilde{\mathbf{g}}_k^{\mathsf{T}}\mathbf{z} + \frac{1}{2}\mathbf{z}^{\mathsf{T}}\mathbf{z}, \text{ subject to } \|\mathbf{z}\| \leq \Delta_k$$

To solve this, we can examine the solutions for the two cases where the constraint is inactive or active at the solution $\mathbf{z}_k$, i.e.,

- $\|\mathbf{z}_k\| < \Delta_k$, which has solution given by the Newton step $\mathbf{z}_k = -\tilde{\mathbf{g}}_k$
- $\|\mathbf{z}_k\| = \Delta_k$, which we can solve by the augmented Lagrangian method.

For the second case, consider the following Lagrangian:

$$\begin{aligned}\mathcal{L}(\mathbf{z}, \lambda) &= \tilde{\mathbf{g}}_k^{\mathsf{T}}\mathbf{z} + \frac{1}{2}\mathbf{z}^{\mathsf{T}}\mathbf{z} + \frac{1}{2}\lambda(\mathbf{z}^{\mathsf{T}}\mathbf{z} - \Delta_k^2) \\ &= \tilde{\mathbf{g}}_k^{\mathsf{T}}\mathbf{z} + \frac{1}{2}(1+\lambda)\mathbf{z}^{\mathsf{T}}\mathbf{z} - \frac{1}{2}\lambda\Delta_k^2\end{aligned}$$

where λ is the Lagrangian multiplier.

Square-root trust-region subproblem

Setting the gradients of the Lagrangian to zero at the solution yields

$$\left.\frac{\partial \mathcal{L}}{\partial \mathbf{z}}\right|_{\substack{\mathbf{z}=\mathbf{z}_k \\ \lambda=\lambda_k}} = \tilde{\mathbf{g}}_k + (1+\lambda_k)\mathbf{z}_k = \mathbf{0}$$

$$\mathbf{z}_k = -\frac{1}{1+\lambda_k}\tilde{\mathbf{g}}_k$$

$$\left.\frac{\partial \mathcal{L}}{\partial \lambda}\right|_{\substack{\mathbf{z}=\mathbf{z}_k \\ \lambda=\lambda_k}} = \frac{1}{2}(\mathbf{z}_k^\mathsf{T}\mathbf{z}_k - \Delta_k^2) = 0$$

Therefore, $$1+\lambda = \pm\frac{1}{\Delta_k}\sqrt{\tilde{\mathbf{g}}^\mathsf{T}\tilde{\mathbf{g}}} = \pm\frac{\|\tilde{\mathbf{g}}\|}{\Delta_k}$$

Selecting the positive solution yields $\mathbf{z}_k = -\frac{\Delta_k}{\|\tilde{\mathbf{g}}\|}\tilde{\mathbf{g}}_k$

Square-root trust-region subproblem

Assembling the solutions from the two cases yields the TRS solution in $\mathbf{z}$-coordinates:

$$\mathbf{z}_k = -\min\left(1, \frac{\Delta_k}{\|\tilde{\mathbf{g}}\|}\right)\tilde{\mathbf{g}}_k$$

Transforming back to $\mathbf{p}$-coordinates yields

$$\begin{aligned}
\mathbf{S}_k^{-\mathsf{T}}\mathbf{p}_k &= -\min\left(1, \frac{\Delta_k}{\|\tilde{\mathbf{g}}\|}\right)\tilde{\mathbf{g}}_k \\
\mathbf{p}_k &= -\min\left(1, \frac{\Delta_k}{\|\tilde{\mathbf{g}}\|}\right)\mathbf{S}_k^{\mathsf{T}}\tilde{\mathbf{g}}_k \\
&= -\min\left(1, \frac{\Delta_k}{\|\mathbf{S}_k\mathbf{g}_k\|}\right)\mathbf{S}_k^{\mathsf{T}}\mathbf{S}_k\mathbf{g}_k
\end{aligned}$$

Quasi-Newton optimisation

In practice, it may be difficult or expensive to compute the Hessian needed in each iteration of Newton's optimisation method and the Hessian at the minimiser needed to evaluate the posterior covariance using the Laplace approximation.

To overcome this difficulty, quasi-Newton methods, such as BFGS and SR1, use a numerical approximation to update the Hessian based on the change in gradient at each iteration of Newton's method that satisfies the secant equation,

$$\mathbf{g}_{k+1} = \mathbf{g}_k + \mathbf{H}_{k+1}\mathbf{p}_k$$

where $\mathbf{g}_{k+1} = \mathbf{g}(\mathbf{x}_k + \mathbf{p}_k)$

Since the secant equation is undetermined in $\mathbf{H}_{k+1}$, we need some other criterion to uniquely determine the Hessian update. At each iteration, we only learn about curvature information in one direction, so we can expect that $\mathbf{H}_{k+1}$ differs from $\mathbf{H}_k$ by a low rank matrix.

BFGS Hessian update

The Broyden–Fletcher–Goldfarb–Shanno (BFGS) update is given by

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{\Delta\mathbf{g}_k \Delta\mathbf{g}_k^\mathsf{T}}{\Delta\mathbf{g}_k^\mathsf{T}\mathbf{p}_k} - \frac{\mathbf{H}_k\mathbf{p}_k\mathbf{p}_k^\mathsf{T}\mathbf{H}_k^\mathsf{T}}{\mathbf{p}_k^\mathsf{T}\mathbf{H}_k\mathbf{p}_k}$$

where

$$\Delta\mathbf{g}_k = \mathbf{g}_{k+1} - \mathbf{g}_k = \mathbf{H}_{k+1}\mathbf{p}_k$$

which is known to guarantee positive definite $\mathbf{H}_{k+1}$ (not obvious).

Using the matrix inversion lemma, the BFGS update can also be written for the inverse of the Hessian,

$$\mathbf{H}_{k+1}^{-1} = \left(\mathbf{I} - \frac{\Delta\mathbf{g}_k\mathbf{p}_k^\mathsf{T}}{\Delta\mathbf{g}_k^\mathsf{T}\mathbf{p}_k}\right)^\mathsf{T} \mathbf{H}_k^{-1} \left(\mathbf{I} - \frac{\Delta\mathbf{g}_k\mathbf{p}_k^\mathsf{T}}{\Delta\mathbf{g}_k^\mathsf{T}\mathbf{p}_k}\right) + \frac{\mathbf{p}_k\mathbf{p}_k^\mathsf{T}}{\Delta\mathbf{g}_k^\mathsf{T}\mathbf{p}_k}$$

Square-root inverse Hessian BFGS update

Given $\mathbf{H}_k^{-1} = \mathbf{S}_k^\mathsf{T}\mathbf{S}_k$, the square-root inverse Hessian BFGS update becomes

$$\begin{aligned}
\mathbf{S}_{k+1}^\mathsf{T}\mathbf{S}_{k+1} &= \left(\mathbf{I} - \frac{\Delta\mathbf{g}_k\mathbf{p}_k^\mathsf{T}}{\Delta\mathbf{g}_k^\mathsf{T}\mathbf{p}_k}\right)^\mathsf{T} \mathbf{S}_k^\mathsf{T}\mathbf{S}_k \left(\mathbf{I} - \frac{\Delta\mathbf{g}_k\mathbf{p}_k^\mathsf{T}}{\Delta\mathbf{g}_k^\mathsf{T}\mathbf{p}_k}\right) + \frac{\mathbf{p}_k\mathbf{p}_k^\mathsf{T}}{\Delta\mathbf{g}_k^\mathsf{T}\mathbf{p}_k} \\
&= \begin{bmatrix} \frac{\mathbf{p}_k}{\sqrt{\Delta\mathbf{g}_k^\mathsf{T}\mathbf{p}_k}} & \left(\mathbf{I} - \frac{\Delta\mathbf{g}_k\mathbf{p}_k^\mathsf{T}}{\Delta\mathbf{g}_k^\mathsf{T}\mathbf{p}_k}\right)^\mathsf{T} \mathbf{S}_k^\mathsf{T} \end{bmatrix} \begin{bmatrix} \frac{\mathbf{p}_k^\mathsf{T}}{\sqrt{\Delta\mathbf{g}_k^\mathsf{T}\mathbf{p}_k}} \\ \mathbf{S}_k\left(\mathbf{I} - \frac{\Delta\mathbf{g}_k\mathbf{p}_k^\mathsf{T}}{\Delta\mathbf{g}_k^\mathsf{T}\mathbf{p}_k}\right) \end{bmatrix} \\
&= \begin{bmatrix} \frac{\mathbf{p}_k^\mathsf{T}}{\sqrt{\Delta\mathbf{g}_k^\mathsf{T}\mathbf{p}_k}} \\ \mathbf{S}_k\left(\mathbf{I} - \frac{\Delta\mathbf{g}_k\mathbf{p}_k^\mathsf{T}}{\Delta\mathbf{g}_k^\mathsf{T}\mathbf{p}_k}\right) \end{bmatrix}^\mathsf{T} \begin{bmatrix} \frac{\mathbf{p}_k^\mathsf{T}}{\sqrt{\Delta\mathbf{g}_k^\mathsf{T}\mathbf{p}_k}} \\ \mathbf{S}_k\left(\mathbf{I} - \frac{\Delta\mathbf{g}_k\mathbf{p}_k^\mathsf{T}}{\Delta\mathbf{g}_k^\mathsf{T}\mathbf{p}_k}\right) \end{bmatrix} \\
\mathbf{S}_{k+1} &= \mathrm{qr}\left(\begin{bmatrix} \frac{\mathbf{p}_k^\mathsf{T}}{\sqrt{\Delta\mathbf{g}_k^\mathsf{T}\mathbf{p}_k}} \\ \mathbf{S}_k\left(\mathbf{I} - \frac{\Delta\mathbf{g}_k\mathbf{p}_k^\mathsf{T}}{\Delta\mathbf{g}_k^\mathsf{T}\mathbf{p}_k}\right) \end{bmatrix}\right)
\end{aligned}$$

The term under the square root is always positive, since

$$\Delta\mathbf{g}_k^\mathsf{T}\mathbf{p}_k = \mathbf{p}_k^\mathsf{T}\mathbf{H}_{k+1}\mathbf{p}_k > 0$$

Require: Differentiable cost function $f\colon \mathcal{X} \to \mathbb{R}$
Require: Initial point $\mathbf{x}_0$ and upper triangular matrix $\mathbf{S}_0$ such that $\mathbf{S}_0^\mathsf{T}\mathbf{S}_0$ is the initial inverse Hessian

function FMINBFGSTRUSTSQRTINV($f(\cdot)$, $\mathbf{x}_0$, $\mathbf{S}_0$)
  $k \leftarrow 0$
  $(f_k, \mathbf{g}_k) \leftarrow (f(\mathbf{x}_k), \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_k})$ ▷ Initial cost and gradient
  $\Delta_k \leftarrow 1$ ▷ Initial trust region radius
  repeat
    $\mathbf{p}_k \leftarrow \arg\min_{\mathbf{p}} \mathbf{g}_k^\mathsf{T}\mathbf{p} + \frac{1}{2}\mathbf{p}^\mathsf{T}\mathbf{S}_k^{-1}\mathbf{S}_k^{-\mathsf{T}}\mathbf{p}$, s.t. $\|\mathbf{S}_k^{-\mathsf{T}}\mathbf{p}\| \leq \Delta_k$ ▷ Solve TRS
    if $-\mathbf{p}_k^\mathsf{T}\mathbf{g}_k < 2\epsilon$ **then** ▷ If Newton decrement converged
      return $\mathbf{x}_k$, $\mathbf{S}_k$
    end if
    $\tilde{\mathbf{x}} \leftarrow \mathbf{x}_k + \mathbf{p}_k$ ▷ Trial step
    $(\tilde{f}, \tilde{\mathbf{g}}) \leftarrow (f(\tilde{\mathbf{x}}), \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}=\tilde{\mathbf{x}}})$ ▷ Trial cost and gradient
    $\Delta\mathbf{g}_k \leftarrow \tilde{\mathbf{g}} - \mathbf{g}_k$ ▷ Trial gradient change
    $\rho_k \leftarrow \frac{f_k - \tilde{f}}{-\mathbf{p}_k^\mathsf{T}\mathbf{g}_k - \frac{1}{2}\mathbf{p}_k^\mathsf{T}\mathbf{S}_k^{-1}\mathbf{S}_k^{-\mathsf{T}}\mathbf{p}_k}$ ▷ Trial step quality
    if $\rho_k < 0.25$ **then**
      $\Delta_{k+1} \leftarrow \frac{1}{4}\|\mathbf{S}_k^{-\mathsf{T}}\mathbf{p}_k\|$ ▷ Decrease trust region radius
    else if $\rho_k > 0.75$ and $\|\mathbf{S}_k^{-\mathsf{T}}\mathbf{p}_k\| > 0.8\Delta_k$ **then**
      $\Delta_{k+1} \leftarrow 2\Delta_k$ ▷ Increase trust region radius
    else
      $\Delta_{k+1} \leftarrow \Delta_k$ ▷ Preserve trust region radius
    end if
    if $\rho_k > 0.001$ **then**
      $(\mathbf{x}_{k+1}, f_{k+1}, \mathbf{g}_{k+1}) \leftarrow (\tilde{\mathbf{x}}, \tilde{f}, \tilde{\mathbf{g}})$ ▷ Accept the trial step
    else
      $(\mathbf{x}_{k+1}, f_{k+1}, \mathbf{g}_{k+1}) \leftarrow (\mathbf{x}_k, f_k, \mathbf{g}_k)$ ▷ Reject the trial step
    end if
    if $\Delta\mathbf{g}_k^\mathsf{T}\mathbf{p}_k \geq \sqrt{\epsilon}\|\Delta\mathbf{g}_k\|\|\mathbf{p}_k\|$ **then**
      $$\begin{bmatrix}\mathbf{R}_1 \\ \mathbf{0}\end{bmatrix} = \operatorname{qr}\left(\begin{bmatrix}\frac{\mathbf{p}_k^\mathsf{T}}{\sqrt{\Delta\mathbf{g}_k^\mathsf{T}\mathbf{p}_k}} \\ \mathbf{S}_k\left(\mathbf{I} - \frac{\Delta\mathbf{g}_k\mathbf{p}_k^\mathsf{T}}{\Delta\mathbf{g}_k^\mathsf{T}\mathbf{p}_k}\right)\end{bmatrix}\right)$$
      $\mathbf{S}_{k+1} \leftarrow \mathbf{R}_1$ ▷ Update square-root inverse Hessian
    else
      $\mathbf{S}_{k+1} = \mathbf{S}_k$ ▷ Preserve square-root inverse Hessian
    end if
    $k \leftarrow k + 1$
  until death
end function

Laplace filter

Laplace filter

1. Form the prediction density $p(\mathbf{x}_k|\mathbf{y}_{1:k-1})$ by propagating $p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})$ through the transformation

$$\boldsymbol{\Phi}: \begin{bmatrix} \mathbf{x}_{k-1} \\ \Delta\mathbf{w}_{k-1} \end{bmatrix} \mapsto \mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}, \Delta\mathbf{w}_{k-1})$$

$$\mathcal{T}\{\boldsymbol{\Phi}\}: \left(\begin{bmatrix} \boldsymbol{\mu}_{k-1|k-1} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{P}_{k-1|k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}\,\Delta t_{k-1} \end{bmatrix} \right) \mapsto (\boldsymbol{\mu}_{k|k-1}, \mathbf{P}_{k|k-1})$$

2. Compute the filtered pdf $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ from the Laplace approximation

$$\mathcal{V}(\mathbf{x}_k) = -\log \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}, \mathbf{P}_{k|k-1}) - \log p(\mathbf{y}_k|\mathbf{x}_k)$$

$$\boldsymbol{\mu}_{k|k} = \arg\min_{\mathbf{x}_k} \mathcal{V}(\mathbf{x}_k)$$

$$\mathbf{P}_{k|k} = \left(\frac{\partial^2 \mathcal{V}}{\partial \mathbf{x}_k^2}\bigg|_{\mathbf{x}_k = \boldsymbol{\mu}_{k|k}} \right)^{-1}$$

Square-root Laplace filter

1. Form the prediction density $p(\mathbf{x}_k|\mathbf{y}_{1:k-1})$ by propagating $p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})$ through the transformation

$$\boldsymbol{\Phi}: \begin{bmatrix} \mathbf{x}_{k-1} \\ \Delta\mathbf{w}_{k-1} \end{bmatrix} \mapsto \mathbf{x}_k$$

$$\mathcal{T}^{\frac{1}{2}}\{\boldsymbol{\Phi}\}: \left(\begin{bmatrix} \boldsymbol{\mu}_{k-1|k-1} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{S}_{k-1|k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{S_Q}\sqrt{\Delta t_k} \end{bmatrix} \right) \mapsto (\boldsymbol{\mu}_{k|k-1}, \mathbf{S}_{k|k-1})$$

2. Compute the filtered pdf $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ from the Laplace approximation

$$\mathcal{V}(\mathbf{x}_k) = -\log \mathcal{N}^{\frac{1}{2}}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}, \mathbf{S}_{k|k-1}) - \log p(\mathbf{y}_k|\mathbf{x}_k)$$

$$(\boldsymbol{\mu}_{k|k}, \mathbf{S}_{k|k}) = \textsc{fminBFGSTrustSqrtInv}(\mathcal{V}(\cdot), \boldsymbol{\mu}_{k|k-1}, \mathbf{S}_{k|k-1})$$

Laplace filter for visual navigation

The Laplace filter presented can handle any second-differentiable measurement log-likelihood that results in a convex cost $\mathcal{V}(\mathbf{x}_k)$. A convex $\mathcal{V}(\mathbf{x}_k)$ is equivalent to the Bayes posterior being unimodal.

What sort of measurement likelihood should we use? Ideally, we should choose and fit a likelihood function to match calibration data (Farnworth, 2021).

For measurements on an image, $\mathbf{y}_k \in \mathbb{R}^2$, start with a Gaussian distribution, which is the maximum entropy distribution on $\mathbb{R}^n$ for a given mean and covariance,

$$p(\mathbf{y}_k|\mathbf{x}_k) = \mathcal{N}(\mathbf{y}_k; \mathbf{h}_k(\mathbf{x}_k), \mathbf{R}_k) = (\det 2\pi\mathbf{R}_k)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{y}_k - \mathbf{h}_k(\mathbf{x}_k))^\mathsf{T}\mathbf{R}_k^{-1}(\mathbf{y}_k - \mathbf{h}_k(\mathbf{x}_k))\right)$$

For measurements on the view sphere, $\mathbf{y}_k \in \mathsf{S}^2$, start with a von-Mises Fisher distribution, which is the maximum entropy distribution on S^n for a given mean direction and concentration parameter,

$$p(\mathbf{y}_k|\mathbf{x}_k) = \mathcal{VMF}(\mathbf{y}_k; \mathbf{h}_k(\mathbf{x}_k), \kappa_k) \propto \exp\left(\kappa\mathbf{y}_k^\mathsf{T}\mathbf{h}_k(\mathbf{x}_k)\right)$$

Gaussian likelihood

For brevity, we omit the time index.

$$\log p(\mathbf{y}|\mathbf{x}) = \log \mathcal{N}(\mathbf{y}; \mathbf{h}(\mathbf{x}), \mathbf{R}) = -\frac{1}{2}\big(\mathbf{y} - \mathbf{h}(\mathbf{x})\big)^{\mathsf{T}} \mathbf{R}^{-1} \big(\mathbf{y} - \mathbf{h}(\mathbf{x})\big) - \frac{1}{2} \log \det 2\pi\mathbf{R}$$

$$= -\frac{1}{2} \sum_m \sum_n (y_m - h_m(\mathbf{x}))(\mathbf{R}^{-1})_{mn}(y_n - h_n(\mathbf{x})) - \frac{1}{2} \log \det 2\pi\mathbf{R}$$

The i^{th} component of the gradient of the log-likelihood is given by taking the partial derivative w.r.t. x_i,

$$\frac{\partial}{\partial x_i} \log p(\mathbf{y}|\mathbf{x}) = \frac{\partial}{\partial x_i} \log \mathcal{N}(\mathbf{y}; \mathbf{h}(\mathbf{x}), \mathbf{R}) = \sum_m \sum_n \frac{\partial h_m}{\partial x_i} (\mathbf{R}^{-1})_{mn}(y_n - h_n(\mathbf{x}))$$

The gradient of the log-likelihood can be written as a column vector,

$$\frac{\partial}{\partial \mathbf{x}} \log p(\mathbf{y}|\mathbf{x}) = \frac{\partial}{\partial \mathbf{x}} \log \mathcal{N}(\mathbf{y}; \mathbf{h}(\mathbf{x}), \mathbf{R}) = \left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right)^{\mathsf{T}} \mathbf{R}^{-1}(\mathbf{y} - \mathbf{h}(\mathbf{x}))$$

Gaussian likelihood

The ij^{th} component of the Hessian of the log-likelihood is given by

$$\frac{\partial^2}{\partial x_i \partial x_j} \log p(\mathbf{y}|\mathbf{x}) = \sum_m \sum_n \frac{\partial^2 h_m}{\partial x_i \partial x_j} (\mathbf{R}^{-1})_{mn} (y_n - h_n(\mathbf{x})) - \frac{\partial h_m}{\partial x_i} (\mathbf{R}^{-1})_{mn} \frac{\partial h_n}{\partial x_j}$$

If $\mathbf{h}$ is a linear function of $\mathbf{x}$, the second derivatives of $\mathbf{h}$ are zero, in which case we can write the Hessian matrix as follows:

$$\frac{\partial^2}{\partial \mathbf{x}^2} \log p(\mathbf{y}|\mathbf{x}) = \frac{\partial^2}{\partial \mathbf{x}^2} \log \mathcal{N}(\mathbf{y}; \mathbf{h}(\mathbf{x}), \mathbf{R}) = -\left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right)^{\mathsf{T}} \mathbf{R}^{-1} \frac{\partial \mathbf{h}}{\partial \mathbf{x}}$$

If the second derivatives of $\mathbf{h}$ are nonzero, we can't write the Hessian in such a compact way due to the presence of a third-order tensor.

Gaussian LF vs EKF

How is the LF with a Gaussian likelihood different from the EKF?

Consider the cost function:

$$\begin{aligned}\mathcal{V}(\mathbf{x}_k) &= -\log \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}, \mathbf{P}_{k|k-1}) - \log p(\mathbf{y}_k|\mathbf{x}_k) \\ &= -\log \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}, \mathbf{P}_{k|k-1}) - \log \mathcal{N}(\mathbf{y}_k; \mathbf{h}_k(\mathbf{x}_k), \mathbf{R}_k) \\ &= \frac{1}{2}(\mathbf{x}_k - \boldsymbol{\mu}_{k|k-1})^\mathsf{T}\mathbf{P}_{k|k-1}^{-1}(\mathbf{x}_k - \boldsymbol{\mu}_{k|k-1}) + \frac{1}{2}(\mathbf{y}_k - \mathbf{h}_k(\mathbf{x}_k))^\mathsf{T}\mathbf{R}_k^{-1}(\mathbf{y}_k - \mathbf{h}_k(\mathbf{x}_k)) + \text{const}\end{aligned}$$

The gradient of the cost is given by:

$$\frac{\partial \mathcal{V}}{\partial \mathbf{x}_k} = \mathbf{P}_{k|k-1}^{-1}(\mathbf{x}_k - \boldsymbol{\mu}_{k|k-1}) - \left(\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}\right)^\mathsf{T} \mathbf{R}^{-1}(\mathbf{y}_k - \mathbf{h}_k(\mathbf{x}_k))$$

The minimiser of $\mathcal{V}(\mathbf{x}_k)$ occurs when

$$\left.\frac{\partial \mathcal{V}}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_k = \boldsymbol{\mu}_{k|k}} = \mathbf{P}_{k|k-1}^{-1}(\boldsymbol{\mu}_{k|k} - \boldsymbol{\mu}_{k|k-1}) - \left.\frac{\partial^\mathsf{T} \mathbf{h}_k}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_k = \boldsymbol{\mu}_{k|k}} \mathbf{R}^{-1}(\mathbf{y}_k - \mathbf{h}_k(\boldsymbol{\mu}_{k|k})) = \mathbf{0}$$

Gaussian LF vs EKF

If we assume

$$\mathbf{C}_k = \left.\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_k=\boldsymbol{\mu}_{k|k-1}} \approx \left.\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_k=\boldsymbol{\mu}_{k|k}}$$

$$\mathbf{h}_k(\boldsymbol{\mu}_{k|k}) \approx \mathbf{h}_k(\boldsymbol{\mu}_{k|k-1}) + \mathbf{C}_k(\boldsymbol{\mu}_{k|k} - \boldsymbol{\mu}_{k|k-1})$$

then,

$$\mathbf{P}_{k|k-1}^{-1}(\boldsymbol{\mu}_{k|k} - \boldsymbol{\mu}_{k|k-1}) - \mathbf{C}_k^{\mathsf{T}}\mathbf{R}^{-1}\left(\mathbf{y}_k - \mathbf{h}_k(\boldsymbol{\mu}_{k|k-1}) - \mathbf{C}_k(\boldsymbol{\mu}_{k|k} - \boldsymbol{\mu}_{k|k-1})\right) = \mathbf{0}$$

$$\begin{aligned}
(\mathbf{P}_{k|k-1}^{-1} + \mathbf{C}_k^{\mathsf{T}}\mathbf{R}_k^{-1}\mathbf{C}_k)\boldsymbol{\mu}_{k|k} &= (\mathbf{P}_{k|k-1}^{-1} + \mathbf{C}_k^{\mathsf{T}}\mathbf{R}_k^{-1}\mathbf{C}_k)\boldsymbol{\mu}_{k|k-1} + \mathbf{C}_k^{\mathsf{T}}\mathbf{R}^{-1}(\mathbf{y}_k - \mathbf{h}_k(\boldsymbol{\mu}_{k|k-1})) \\
\boldsymbol{\mu}_{k|k} &= \boldsymbol{\mu}_{k|k-1} + (\mathbf{P}_{k|k-1}^{-1} + \mathbf{C}_k^{\mathsf{T}}\mathbf{R}_k^{-1}\mathbf{C}_k)^{-1}\mathbf{C}_k^{\mathsf{T}}\mathbf{R}^{-1}(\mathbf{y}_k - \mathbf{h}_k(\boldsymbol{\mu}_{k|k-1})) \\
\boldsymbol{\mu}_{k|k} &= \boldsymbol{\mu}_{k|k-1} + \mathbf{P}_{k|k-1}\mathbf{C}_k^{\mathsf{T}}(\mathbf{C}_k\mathbf{P}_{k|k-1}\mathbf{C}_k^{\mathsf{T}} + \mathbf{R}_k)^{-1}(\mathbf{y}_k - \mathbf{h}_k(\boldsymbol{\mu}_{k|k-1}))
\end{aligned}$$

which corresponds to the posterior mean of the EKF.

Gaussian LF vs EKF

In addition, if we neglect the second derivatives of $\mathbf{h}$,

$$\frac{\partial^2 \mathbf{h}}{\partial x_i \partial x_j} \approx \mathbf{0}, \quad \forall i, j$$

then the Hessian of the cost is given by

$$\frac{\partial^2 \mathcal{V}}{\partial \mathbf{x}_k^2} = \mathbf{P}_{k|k-1}^{-1} + \left(\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}\right)^{\mathsf{T}} \mathbf{R}^{-1} \left(\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}\right)$$

The posterior covariance is given by

$$\mathbf{P}_{k|k} = \left(\frac{\partial^2 \mathcal{V}}{\partial \mathbf{x}_k^2}\bigg|_{\mathbf{x}_k = \boldsymbol{\mu}_{k|k}} \right)^{-1}$$

From our previous assumptions, this evaluates to

$$\mathbf{P}_{k|k} = \left(\mathbf{P}_{k|k-1}^{-1} + \mathbf{C}_k^{\mathsf{T}} \mathbf{R}_k^{-1} \mathbf{C}_k\right)^{-1} = \mathbf{P}_{k|k-1} - \mathbf{P}_{k|k-1} \mathbf{C}_k^{\mathsf{T}} (\mathbf{C}_k \mathbf{P}_{k|k-1} \mathbf{C}_k^{\mathsf{T}} + \mathbf{R}_k)^{-1} \mathbf{C}_k \mathbf{P}_{k|k-1}$$

which corresponds to the posterior covariance of the EKF.

Gaussian LF vs EKF

To summarise, the LF with a Gaussian likelihood agrees with the EKF under the following assumptions:

$$\left.\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_k=\boldsymbol{\mu}_{k|k-1}} \approx \left.\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_k=\boldsymbol{\mu}_{k|k}}$$

$$\frac{\partial^2 \mathbf{h}_k}{\partial x_i \partial x_j} \approx \mathbf{0}, \quad \forall i, j$$

$$\mathbf{h}_k(\boldsymbol{\mu}_{k|k}) \approx \mathbf{h}_k(\boldsymbol{\mu}_{k|k-1}) + \mathbf{C}_k(\boldsymbol{\mu}_{k|k} - \boldsymbol{\mu}_{k|k-1})$$

These approximations are exact if $\mathbf{h}$ is a linear function of $\mathbf{x}_k$, in which case the KF is also the solution.

Gaussian LF vs EKF

For Gaussian likelihood with nonlinear mean function $\mathbf{h}(\cdot)$, the LF differs from the EKF in two important ways:

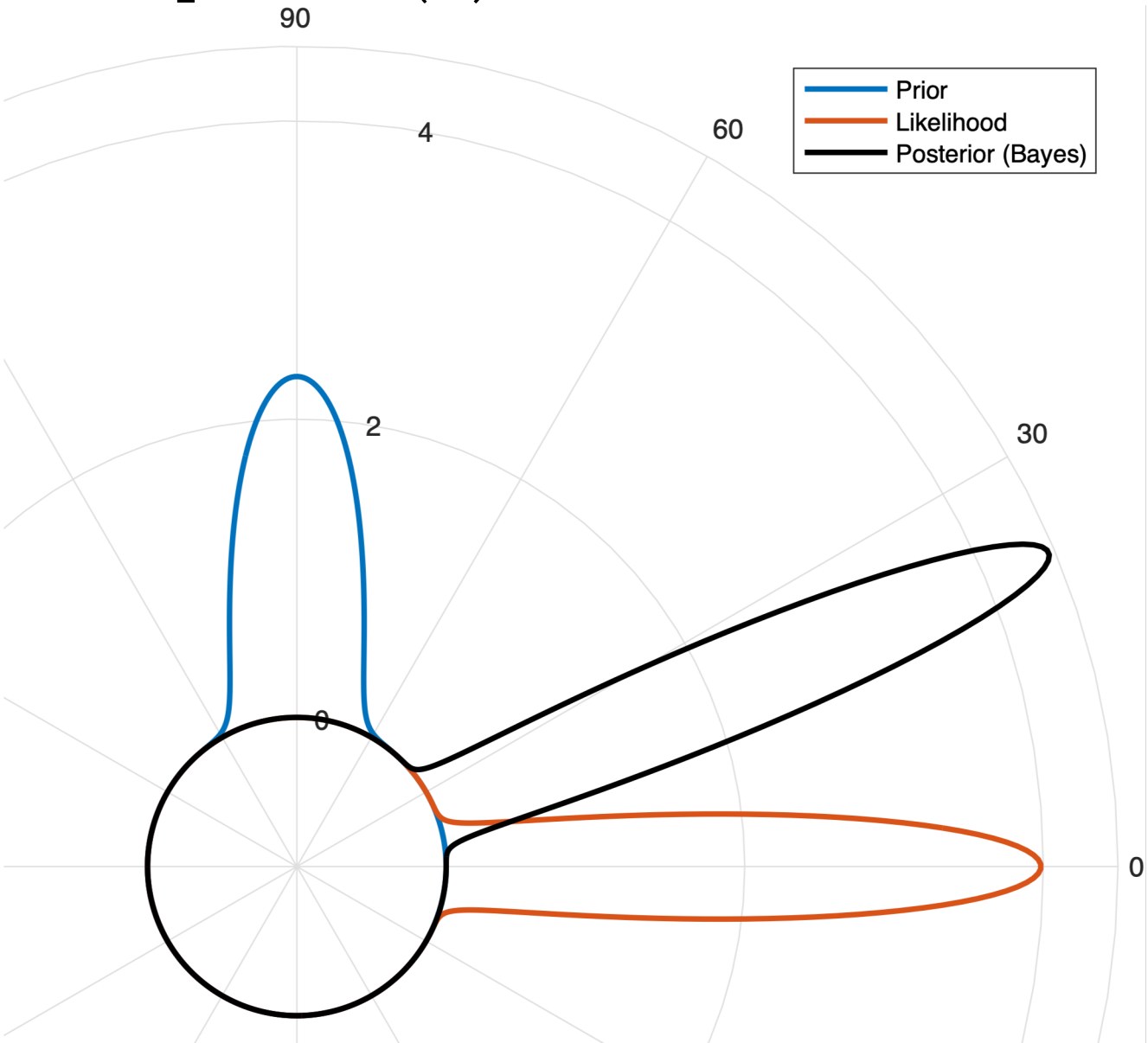
1. The LF evaluates the gradient of $\mathbf{h}(\cdot)$ at the posterior mean rather than the prior mean,
2. The LF accounts for the effect of the curvature of $\mathbf{h}(\cdot)$ on the posterior covariance.

If $\mathbf{h}(\cdot)$ is linear, the iterated measurement correction terminates in one iteration with the same solution as the EKF/KF, thus there is no penalty in applying the LF.

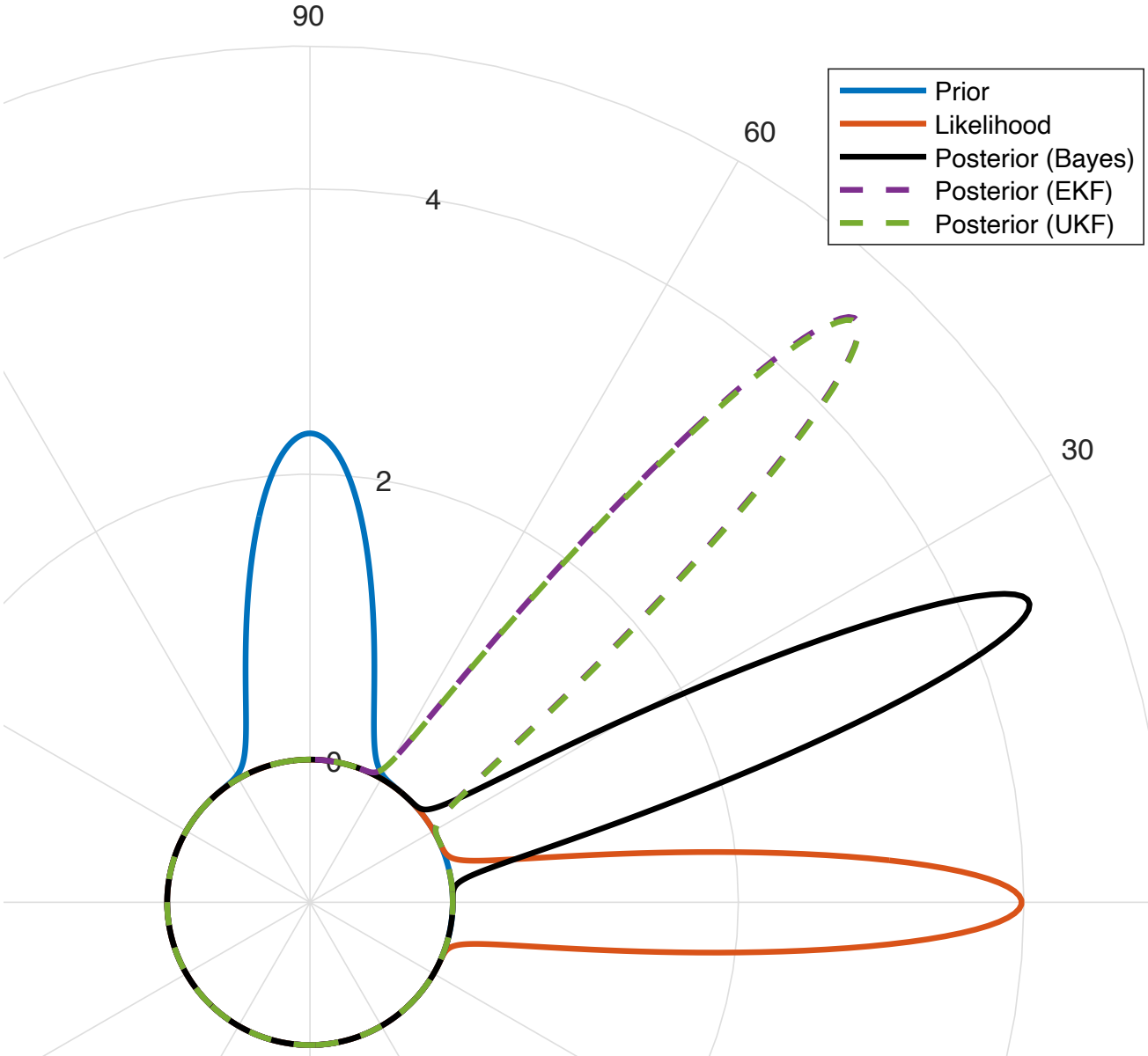
Note that $\mathbf{h}(\cdot)$ is always nonlinear. Especially in vision-based navigation.

EKF-SLAM is known to suffer from inconsistency due to over-confidence in the posterior.

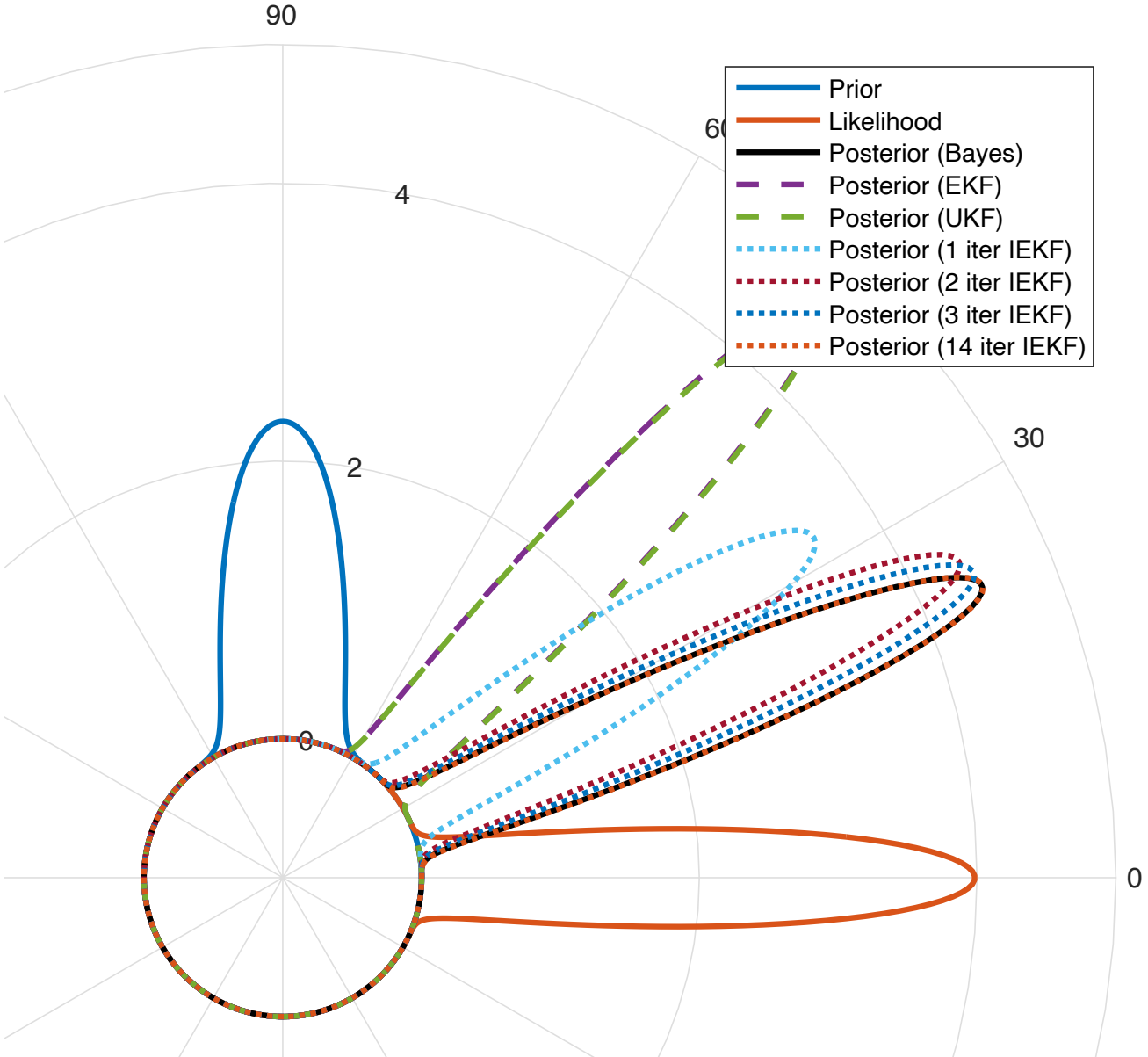
Example: SO(2) state and S^1 measurement



Example: EKF and UKF



Example: Iterated EKF



Automatic differentiation

Computing derivatives

There are several ways to implement the derivatives required by optimisation routines:

- Symbolic differentiation
  - By hand
  - Using a CAS tool

$$f(\mathbf{x}) = \|\mathbf{x}\|$$
$$\frac{\partial f}{\partial \mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|}$$
$$\frac{\partial^2 f}{\partial \mathbf{x}^2} = \frac{1}{\|\mathbf{x}\|}\mathbf{I} - \frac{1}{\|\mathbf{x}\|^3}\mathbf{x}\mathbf{x}^\mathsf{T}$$

- Numerical differentiation (finite difference approximations)

$$\frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$
$$\frac{\partial^2 f}{\partial x^2} = \lim_{h \to 0} \frac{f(x+2h) - 2f(x+h) + f(x)}{h^2}$$

- Automatic differentiation

Automatic differentiation

Automatic differentiation is a set of numerical methods to compute derivatives of a function *already implemented* in a computer program.

The idea is to replace the scalar type (e.g., `float` or `double`) with another type that has overloaded operators for the elementary arithmetic operators (+, –, ×, ÷) and functions (exp, log, sin, cos, etc.).

There are two types of automatic differentiation

- Forward mode
- Reverse mode

Forward-mode autodifferentiation

Replace the scalar type with a *dual number,*

$$x \mapsto x + x'\varepsilon$$

where ε is an infinitesimal value such that

$$\varepsilon^2 = 0$$

Then we can derive the rules for each operator.

Addition:

$$(x + x'\varepsilon) + (y + y'\varepsilon) = x + y + (x' + y')\varepsilon$$

Multiplication:

$$(x + x'\varepsilon)(y + y'\varepsilon) = xy + (xy' + yx')\varepsilon + x'y'\varepsilon^2 = xy + (xy' + yx')\varepsilon$$

Forward-mode autodifferentiation

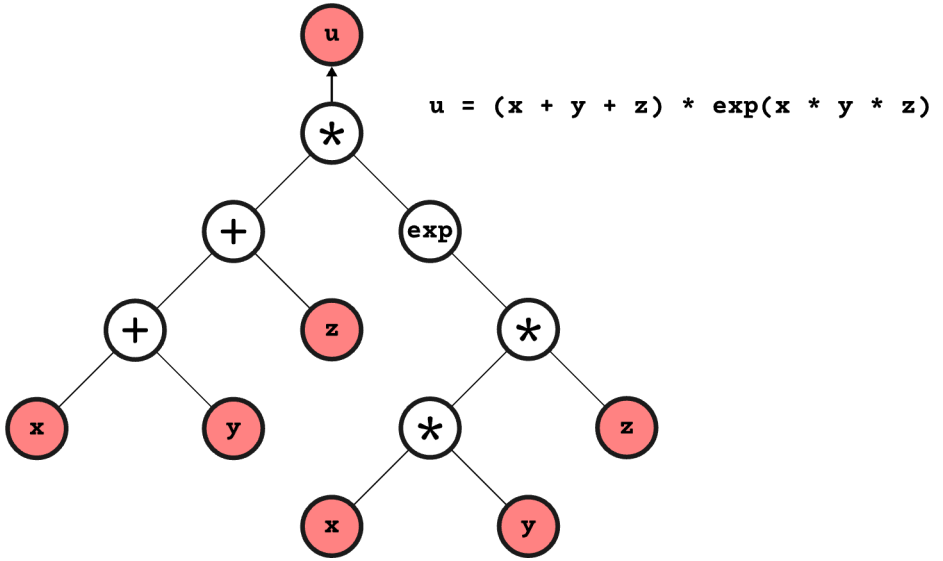
The rules for the other operators and functions can be similarly derived,

$$
\begin{aligned}
(x, x') + (y, y') &= (x + y, x' + y') \\
(x, x') - (y, y') &= (x - y, x' - y') \\
(x, x') \times (y, y') &= (uv, x'y + uv') \\
(x, x') \div (y, y') &= \left(\frac{x}{y}, \frac{x'y - uv'}{y^2}\right) \quad (y \neq 0) \\
\sin(x, x') &= (\sin(x), x' \cos(x)) \\
\cos(x, x') &= (\cos(x), -x' \sin(x)) \\
\exp(x, x') &= (\exp x, x' \exp x) \\
\log(x, x') &= \left(\log(x), \frac{x'}{x}\right) \quad (x > 0) \\
(x, x')^k &= (x^k, ku^{k-1}x') \quad (x \neq 0) \\
|(x, x')| &= (|x|, x' \mathrm{sign}(x)) \quad (x \neq 0)
\end{aligned}
$$

The derivative of a function f at x_0 is found by calculating $f($ $(x_0,1)$ $)$, which yields $(f(x_0), f'(x_0))$.

Reverse-mode autodifferentiation

Replace the scalar type with an object that can record the expression tree as the function is evaluated (forward pass).



The expression tree can be traversed from the output to all the inputs to compute all the derivatives in one backwards pass.

Forward mode vs. reverse mode

Forward-mode autodifferentiation requires one forward pass per input variable.

Reverse-mode autodifferentiation requires one reverse pass no matter how many input variables, but there is overhead is creating the expression tree in the forward pass.

If the number of input variables is low, forward mode may be faster.

If the number of input variables is high or higher derivatives are required, reverse mode may be faster.

Beware of numerical decompositions and solves—the expression tree for reverse mode may be large!

Benchmark both methods for your problem to find out.

Forward mode vs. reverse mode

$$f(\mathbf{x}) = \|\mathbf{x}\|$$
$$\frac{\partial f}{\partial \mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|}$$
$$\frac{\partial^2 f}{\partial \mathbf{x}^2} = \frac{1}{\|\mathbf{x}\|}\mathbf{I} - \frac{1}{\|\mathbf{x}\|^3}\mathbf{x}\mathbf{x}^\mathsf{T}$$

Time to compute gradient and Hessian of $f(\mathbf{x}) = \|\mathbf{x}\|$

Vector length	Analytical result	Forward mode	Reverse mode
5	0.35us	1.7us	12.6us
10	0.5us	3.5us	19.8us
20	1.0us	15.7us	49.9us
50	5.6us	160us	171us
60	7.1us	262us	248us
70	10.0us	404us	309us
100	19.0us	1092us	604us
200	85.6us	7876us	2262us
500	0.5ms	115ms	17.3ms
1000	3.2ms	903ms	75.1ms

References

- G.H. Golub and C.F. Van Loan, 2012. *Matrix Computations*, 4th Ed., JHU Press.
- Moré, J.J. and Sorensen, D.C., 1983. *Computing a trust region step*. SIAM Journal on scientific and statistical computing, 4(3), pp.553–572.
- Conn, A.R., Gould, N.I. and Toint, P.L., 2000. *Trust region methods*. Society for Industrial and Applied Mathematics.
- Nocedal, J. and Wright, S., 2006. *Numerical optimization*. Springer Science & Business Media.
- Farnworth, T., Renton, C., Strydom, R., Wills, A. and Perez, T., 2021. *A Heteroscedastic Likelihood Model for Two-Frame Optical Flow*. IEEE Robotics and Automation Letters, 6(2), pp.1200–1207.
- https://autodiff.github.io/