Lab 6: Laplace filter

MCHA4400

Semester 2 2025

Introduction

In this lab, you will do the following:

1. Complete the implementation of C++ classes for working with Gaussian distributions in square-root moment form.
2. Implement the process dynamics and measurement model for a ballistic state estimation problem.
3. Run a square-root Laplace filter.
4. Plot the results using VTK.

The lab should be completed within 4 hours. The assessment will be done in the lab at the end of your enrolled lab session(s). Once you complete the tasks, call the lab demonstrator to start your assessment.

The lab is worth 5% of your course grade and is graded from 0–5 marks.

GenAI Tip

You are strongly encouraged to explore the use of Large Language Models (LLMs), such as GPT, Gemini or Claude, to assist in completing this activity. If you do not already have access to an LLM, bots are available to use via the UoN Mechatronics Slack team, which you can find a link to from Canvas. If you are are unsure how to make the best use of these tools, or are not getting good results, please ask your lab demonstrator for advice.

1 Gaussian filtering (3 marks)

A Gaussian filter is an online state estimator that assumes a Gaussian state distribution, which is updated according to a stochastic differential equation describing the state dynamics (process model) and a sensor models (measurement likelihood functions).

If we parameterise the state Gaussian distribution by its first two moments—mean and covariance—we obtain a family of filters that includes the Laplace filter. With the additional assumption of a Gaussian measurement likelihood, we also obtain the nonlinear Kalman filters (e.g., EKF and UKF). Filters that propagate the moments of Gaussian distributions should be implemented in square-root form.

Let $\mathbf{x} \in \mathbb{R}^n$ and let

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

be a Gaussian distribution in moment form, where $\boldsymbol{\mu} \in \mathbb{R}^n$ is the mean vector and $\mathbf{P} \in \mathbb{R}^{n\times n}$ is the symmetric positive-definite covariance matrix. This Gaussian distribution can also be parameterised in square-root moment form as follows:

$$\mathcal{N}^{\frac{1}{2}}(\mathbf{x}; \boldsymbol{\mu}, \mathbf{S}) = \frac{1}{\left|\det \sqrt{2\pi}\mathbf{S}\right|} \exp\left(-\frac{1}{2}\left\|\mathbf{S}^{-\mathsf{T}}(\mathbf{x}-\boldsymbol{\mu})\right\|^2\right), \tag{2}$$

where $\mathbf{S} \in \mathbb{R}^{n \times n}$ is an upper-triangular matrix that satisfies the following relation with the covariance matrix:

$$\mathbf{S}^\mathsf{T}\mathbf{S} = \mathbf{P}. \tag{3}$$

In the Laplace filter, the k^{th} measurement correction seeks the state value that maximises the posterior density $p(\mathbf{x}_k|\mathbf{y}_{1:k})$, given the measurement likelihood $p(\mathbf{y}_k|\mathbf{x}_k)$ and the prior density $p(\mathbf{x}_k|\mathbf{y}_{1:k-1})$,

$$\begin{aligned}
\boldsymbol{\mu}_{k|k} &= \arg\max_{\mathbf{x}_k} p(\mathbf{x}_k|\mathbf{y}_{1:k}) \\
&= \arg\max_{\mathbf{x}_k} \frac{p(\mathbf{y}_k|\mathbf{x}_k)\, p(\mathbf{x}_k|\mathbf{y}_{1:k-1})}{p(\mathbf{y}_k|\mathbf{y}_{1:k-1})} \\
&= \arg\max_{\mathbf{x}_k} p(\mathbf{y}_k|\mathbf{x}_k)\, p(\mathbf{x}_k|\mathbf{y}_{1:k-1}) \\
&= \arg\max_{\mathbf{x}_k} p(\mathbf{y}_k|\mathbf{x}_k)\, \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}, \mathbf{P}_{k|k-1}).
\end{aligned}$$

In practice, it is more convenient to minimise the negative log of the above cost, i.e., let

$$\mathcal{V}(\mathbf{x}_k) = -\log p(\mathbf{y}_k|\mathbf{x}_k) - \log \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}, \mathbf{P}_{k|k-1}). \tag{4}$$

Then, under the Laplace approxmation, the posterior mean and covariance are given by

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

The optimisation to find $\boldsymbol{\mu}_{k|k}$ and $\mathbf{P}_{k|k}$ requires the gradient and Hessian,

$$\mathbf{g}_k = \frac{\partial \mathcal{V}}{\partial \mathbf{x}_k} = -\frac{\partial}{\partial \mathbf{x}_k} \log p(\mathbf{y}_k|\mathbf{x}_k) - \frac{\partial}{\partial \mathbf{x}_k} \log \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}, \mathbf{P}_{k|k-1}), \tag{5a}$$

$$\mathbf{H}_k = \frac{\partial^2 \mathcal{V}}{\partial \mathbf{x}_k^2} = -\frac{\partial^2}{\partial \mathbf{x}_k^2} \log p(\mathbf{y}_k|\mathbf{x}_k) - \frac{\partial^2}{\partial \mathbf{x}_k^2} \log \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}, \mathbf{P}_{k|k-1}), \tag{5b}$$

respectively.

Info

The cost (4), gradient (5a) and Hessian (5b) are implemented in `Measurement::costJointDensity`, which rely on a descendant class to implement `Measurement::logLikelihood`, which evaluates the log likelihood and its derivatives for a particular sensor type. The optimisation problem itself is solved in `Measurement::update`.

Since the prior is Gaussian, we therefore need a function to evaluate the log of a Gaussian distribution and its first two derivatives. In addition, if the measurement likelihood is also Gaussian, i.e.,

$$p(\mathbf{y}_k|\mathbf{x}_k) = \mathcal{N}(\mathbf{y}_k; \mathbf{h}(\mathbf{x}_k), \mathbf{R}),$$

for some mean function $\mathbf{h}(\cdot)$ and covariance $\mathbf{R}$, then we also need to be able to compute the derivatives, $\frac{\partial}{\partial \mathbf{x}_k} \log \mathcal{N}(\mathbf{y}_k; \mathbf{h}(\mathbf{x}_k), \mathbf{R})$ and $\frac{\partial^2}{\partial \mathbf{x}_k^2} \log \mathcal{N}(\mathbf{y}_k; \mathbf{h}(\mathbf{x}_k), \mathbf{R})$.

1.1 Log likelihood

The log of a Gaussian distribution in square-root moment form (2) is given by

$$\begin{aligned}
\log \mathcal{N}^{\frac{1}{2}}(\mathbf{x};\boldsymbol{\mu},\mathbf{S}) &= \log\left((\det 2\pi\mathbf{S}^{\mathsf{T}}\mathbf{S})^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}(\mathbf{S}^{\mathsf{T}}\mathbf{S})^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)\right) \\
&= -\frac{1}{2}\log(\det 2\pi\mathbf{S}^{\mathsf{T}}\mathbf{S}) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\mathbf{S}^{-1}\mathbf{S}^{-\mathsf{T}}(\mathbf{x}-\boldsymbol{\mu}) \\
&= -\frac{1}{2}\log\left((2\pi)^n(\det\mathbf{S})^2\right) - \frac{1}{2}\left(\mathbf{S}^{-\mathsf{T}}(\mathbf{x}-\boldsymbol{\mu})\right)^{\mathsf{T}}\mathbf{S}^{-\mathsf{T}}(\mathbf{x}-\boldsymbol{\mu}) \\
&= -\frac{n}{2}\log 2\pi - \log|\det\mathbf{S}| - \frac{1}{2}\left\|\mathbf{S}^{-\mathsf{T}}(\mathbf{x}-\boldsymbol{\mu})\right\|^2 \\
&= -\frac{n}{2}\log 2\pi - \log\left|\prod_{i=1}^{n}\mathbf{S}_{ii}\right| - \frac{1}{2}\|\mathbf{w}\|^2 \\
&= -\frac{n}{2}\log 2\pi - \sum_{i=1}^{n}\log|\mathbf{S}|_{ii} - \frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w}, \qquad (6)
\end{aligned}$$

where $|\cdot|_{ii}$ denotes the absolute value of the i^{th} diagonal element of its argument and $\mathbf{w}$ is the solution to the triangular system of equations $\mathbf{S}^{\mathsf{T}}\mathbf{w} = \mathbf{x} - \boldsymbol{\mu}$.

Tasks

a) Configure the build directory for a debug build with documentation generation enabled and build the application. This can be done directly in VS Code or via the terminal as shown below:

Terminal

```
nerd@basement:~/MCHA4400/lab6$ cmake -G Ninja -B build -DBUILD_DOCUMENTATION=ON -DCMAKE_BUILD_TYPE=Debug
    ↪ && cd build
nerd@basement:~/MCHA4400/lab6/build$ ninja
[Compiling intensifies]
[doctest] test cases:  13 |  11 passed | 0 failed | 18 skipped
[doctest] assertions: 614 | 602 passed | 0 failed |
[doctest] Status: SUCCESS!
```

Tip

To configure the project in VS Code, the `BUILD_DOCUMENTATION` flag can be set in `.vscode/settings.json`. It is already set to `ON` for this project.

The project is now configured to automatically generate documentation from the source code using `doxygen` as part of the build process. This can be accessed by opening `lab6/docs/html/index.html` in your web browser, which may be helpful for navigating the codebase. The style should look familiar, as `doxygen` is also used to generate the `Eigen` and `OpenCV` documentation.

b) Remove the `doctest::skip` decorator from each `SCENARIO` in `test/src/GaussianLog.cpp` to enable these unit tests and run `ninja` to verify that they *fail* due to the naïve implementation of (6) provided.

c) Complete a numerically robust implementation of (6) and its first two derivatives as the three `Gaussian::log` functions within `src/Gaussian.hpp`[1].

> **Tip**
>
> Since the members of `Gaussian` are dependent on the `Scalar` type, the compiler may need some extra hints by way of the `typename` and `template` keywords to help deduce if particular tokens are types or templates, respectively. For example, if you need an upper triangular view of `S_`, which is a dependent type `Eigen::MatrixX<Scalar>`, the `template` keyword may be required, i.e.,
>
> ```
> S_.template triangularView<Eigen::Upper>().solve(/*...*/)
> ```
>
> This is needed because type of `S_` is not known until the template is instantiated for a particular `Scalar` template argument, and prior to instantiation, the compiler cannot otherwise parse the syntax of the template.

d) Run `ninja` and ensure the unit tests in `test/src/GaussianLog.cpp` pass.

1.2 Affine transform

Let $p(\mathbf{x}) = \mathcal{N}^{\frac{1}{2}}(\mathbf{x}; \boldsymbol{\mu}, \mathbf{S})$ be a Gaussian distribution in square-root moment form, where $\mathbf{x}, \boldsymbol{\mu} \in \mathbb{R}^n$ and $\mathbf{S} \in \mathbb{R}^{n \times n}$ is an upper-triangular matrix. Then, let $\mathbf{h}\colon \mathbb{R}^n \to \mathbb{R}^m$ be a map and define $\mathbf{y} = \mathbf{h}(\mathbf{x})$, where $\mathbf{y} \in \mathbb{R}^m$. Then, we can approximate $p(\mathbf{y})$ as

$$\begin{aligned} p(\mathbf{y}) &= \mathcal{N}(\mathbf{y}; \mathbf{h}(\boldsymbol{\mu}), \mathbf{C}\mathbf{S}^\mathsf{T}\mathbf{S}\mathbf{C}^\mathsf{T}) \\ &= \frac{1}{\sqrt{|2\pi \mathbf{C}\mathbf{S}^\mathsf{T}\mathbf{S}\mathbf{C}^\mathsf{T}|}} \exp\left(-\frac{1}{2}\big(\mathbf{y} - \mathbf{h}(\boldsymbol{\mu})\big)^\mathsf{T}\big(\mathbf{C}\mathbf{S}^\mathsf{T}\mathbf{S}\mathbf{C}^\mathsf{T}\big)^{-1}\big(\mathbf{y} - \mathbf{h}(\boldsymbol{\mu})\big)\right), \end{aligned} \tag{7}$$

where $\mathbf{C} = \left.\frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\boldsymbol{\mu}}$ is the Jacobian of $\mathbf{h}$ evaluated at $\boldsymbol{\mu}$.

The distribution (7) is almost in square-root moment form; however, $p(\mathbf{y}) \neq \mathcal{N}^{\frac{1}{2}}(\mathbf{y}; \mathbf{h}(\boldsymbol{\mu}), \mathbf{S}\mathbf{C}^\mathsf{T})$ since $\mathbf{S}\mathbf{C}^\mathsf{T}$ is not necessarily an upper-triangular matrix. To resolve this, let

$$\mathbf{S}\mathbf{C}^\mathsf{T} = \underbrace{\begin{bmatrix}\mathbf{Y} & \mathbf{Z}\end{bmatrix}}_{\mathbf{Q}} \underbrace{\begin{bmatrix}\mathbf{R}_1 \\ \mathbf{0}\end{bmatrix}}_{\mathbf{R}}, \tag{8}$$

where $\mathbf{Q} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix and $\mathbf{R} \in \mathbb{R}^{n \times m}$ is an upper-triangular matrix. Then,

$$\begin{aligned} \big(\mathbf{S}\mathbf{C}^\mathsf{T}\big)^\mathsf{T}\mathbf{S}\mathbf{C}^\mathsf{T} &= \mathbf{R}^\mathsf{T}\mathbf{Q}^\mathsf{T}\mathbf{Q}\mathbf{R} = \mathbf{R}^\mathsf{T}\mathbf{R} && \text{(since } \mathbf{Q} \text{ is orthogonal)} \\ &= \begin{bmatrix}\mathbf{R}_1 \\ \mathbf{0}\end{bmatrix}^\mathsf{T}\begin{bmatrix}\mathbf{R}_1 \\ \mathbf{0}\end{bmatrix} \\ &= \mathbf{R}_1^\mathsf{T}\mathbf{R}_1. \end{aligned}$$

Therefore, the distribution of $\mathbf{y}$ in square-root moment form is given by

$$p(\mathbf{y}) = \mathcal{N}^{\frac{1}{2}}(\mathbf{y}; \mathbf{h}(\boldsymbol{\mu}), \mathbf{R}_1). \tag{9}$$

[1] Template functions are implemented in a header, since they are recipes for building functions that are needed by all compilation units that include this header.

Tasks

a) Remove the `doctest::skip` decorator from each `SCENARIO` in `test/src/GaussianTransform.cpp` to enable these unit tests and run `ninja` to verify that they *fail* due to the incomplete implementation of (9) provided.

b) Complete the implementation of the `Gaussian::affineTransform` template function in `src/Gaussian.hpp`.

c) Ensure that the unit tests provided in `test/src/GaussianTransform.cpp` pass.

1.3 Joint distribution

Let $\mathbf{x} \in \mathbb{R}^n$ and let

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \mathbf{P}) \tag{10}$$

be a multivariate Gaussian distribution, where $\boldsymbol{\mu} \in \mathbb{R}^n$ is the mean vector and $\mathbf{P} \in \mathbb{R}^{n \times n}$ is the symmetric positive-definite covariance matrix. If we partition

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{bmatrix}, \tag{11}$$

where the head $\mathbf{x}_a \in \mathbb{R}^{n_a}$ and the tail $\mathbf{x}_b \in \mathbb{R}^{n_b}$ with $n_a + n_b = n$, then we can write (10) as

$$p\left(\begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{bmatrix}; \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix}, \begin{bmatrix} \mathbf{P}_{aa} & \mathbf{P}_{ab} \\ \mathbf{P}_{ba} & \mathbf{P}_{bb} \end{bmatrix}\right), \tag{12}$$

where $\boldsymbol{\mu}_a \in \mathbb{R}^{n_a}$, $\boldsymbol{\mu}_b \in \mathbb{R}^{n_b}$, $\mathbf{P}_{aa} \in \mathbb{R}^{n_a \times n_a}$, $\mathbf{P}_{ab} \in \mathbb{R}^{n_b \times n_a}$, $\mathbf{P}_{ba} \in \mathbb{R}^{n_b \times n_a}$ and $\mathbf{P}_{bb} \in \mathbb{R}^{n_b \times n_b}$.

Let us also parameterise (12) in square-root moment form as follows:

$$p\left(\begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{bmatrix}\right) = \mathcal{N}^{\frac{1}{2}}\left(\begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{bmatrix}; \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix}, \begin{bmatrix} \mathbf{S}_1 & \mathbf{S}_2 \\ \mathbf{0} & \mathbf{S}_3 \end{bmatrix}\right), \tag{13}$$

where $\boldsymbol{\mu}_1 \in \mathbb{R}^{n_a}$, $\boldsymbol{\mu}_2 \in \mathbb{R}^{n_b}$, $\mathbf{S}_1 \in \mathbb{R}^{n_a \times n_a}$, $\mathbf{S}_2 \in \mathbb{R}^{n_a \times n_b}$, $\mathbf{S}_3 \in \mathbb{R}^{n_b \times n_b}$ and $\mathbf{S}_1$ and $\mathbf{S}_3$ are upper triangular.

Then, from (3), we have

$$\begin{aligned} \begin{bmatrix} \mathbf{P}_{aa} & \mathbf{P}_{ab} \\ \mathbf{P}_{ba} & \mathbf{P}_{bb} \end{bmatrix} &= \begin{bmatrix} \mathbf{S}_1 & \mathbf{S}_2 \\ \mathbf{0} & \mathbf{S}_3 \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \mathbf{S}_1 & \mathbf{S}_2 \\ \mathbf{0} & \mathbf{S}_3 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{S}_1^{\mathsf{T}} & \mathbf{0} \\ \mathbf{S}_2^{\mathsf{T}} & \mathbf{S}_3^{\mathsf{T}} \end{bmatrix} \begin{bmatrix} \mathbf{S}_1 & \mathbf{S}_2 \\ \mathbf{0} & \mathbf{S}_3 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{S}_1^{\mathsf{T}}\mathbf{S}_1 & \mathbf{S}_1^{\mathsf{T}}\mathbf{S}_2 \\ \mathbf{S}_2^{\mathsf{T}}\mathbf{S}_1 & \mathbf{S}_2^{\mathsf{T}}\mathbf{S}_2 + \mathbf{S}_3^{\mathsf{T}}\mathbf{S}_3 \end{bmatrix}. \end{aligned} \tag{14}$$

We will use (14) to obtain results for the marginal and conditional distributions of a Gaussian distribution in square-root moment form in the following subsections.

1.3.1 Marginal distribution

The marginal distributions of the head $\mathbf{x}_a$ and the tail $\mathbf{x}_b$ are then given as follows:

$$p(\mathbf{x}_a) = \mathcal{N}(\mathbf{x}_a; \boldsymbol{\mu}_a, \mathbf{P}_{aa}),$$
$$p(\mathbf{x}_b) = \mathcal{N}(\mathbf{x}_b; \boldsymbol{\mu}_b, \mathbf{P}_{bb}).$$

If we examine the marginal distribution of the head partition and compare the moment parameters to the square-root moment parameters, we find from (14) that

$$\mathbf{S}_1^\mathsf{T}\mathbf{S}_1 = \mathbf{P}_{aa}. \tag{15}$$

Therefore, the marginal distribution of the head partition of a Gaussian distribution in square-root moment form (13) is found simply from the partitions of its joint parameterisation as follows:

$$p(\mathbf{x}_a) = \mathcal{N}^{\frac{1}{2}}(\mathbf{x}_a; \boldsymbol{\mu}_a, \mathbf{S}_1). \tag{16}$$

If we want the marginal distribution of the tail partition, or any other subset of $\mathbf{x}$, we must permute that subset into the tail partition before employing (16).

Foreshadowing

In landmark SLAM, extracting the marginal distributions of a non-contiguous set of states is a common operation. At each time step, the marginal distribution of all landmarks that are predicted to be visible in the current frame is needed for correct data association with the measured image features in that frame. Since the order of the landmarks in the state is arbitrary, the states corresponding to visible landmarks do not form simple partitions of the state.

To do this, let us introduce an index vector $\mathcal{I} \in \mathbb{N}^{n_\mathcal{I}}$ to select the desired elements and denote this as $\mathbf{x}_\mathcal{I}$. Each element of $\mathbf{x}_\mathcal{I}$ is given by

$$(\mathbf{x}_\mathcal{I})_i = x_{\mathcal{I}_i}, \quad \text{for } i = 1, 2, \ldots, n_\mathcal{I}.$$

Example

Let $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix}^\mathsf{T}$ and let $\mathcal{I} = \begin{bmatrix} 3 & 5 & 1 \end{bmatrix}$. Then $\mathbf{x}_\mathcal{I} = \begin{bmatrix} x_3 & x_5 & x_1 \end{bmatrix}^\mathsf{T}$.

The marginal distribution of $\mathbf{x}_\mathcal{I}$ in moment form is given by

$$p(\mathbf{x}_\mathcal{I}) = \mathcal{N}(\mathbf{x}_\mathcal{I}; \boldsymbol{\mu}_\mathcal{I}, \mathbf{P}_{\mathcal{I}\mathcal{I}}), \tag{17}$$

where $\boldsymbol{\mu}_\mathcal{I} \in \mathbb{R}^{n_\mathcal{I}}$ with elements given by $(\boldsymbol{\mu}_\mathcal{I})_i = \mu_{\mathcal{I}_i}$ for $i = 1, 2, \ldots, n_\mathcal{I}$ and $\mathbf{P}_{\mathcal{I}\mathcal{I}} \in \mathbb{R}^{n_\mathcal{I} \times n_\mathcal{I}}$ with elements given by $(\mathbf{P}_{\mathcal{I}\mathcal{I}})_{i,j} = P_{\mathcal{I}_i, \mathcal{I}_j}$ for $i, j = 1, 2, \ldots, n_\mathcal{I}$.

Example

Let

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \\ \mu_5 \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} & P_{15} \\ P_{21} & P_{22} & P_{23} & P_{24} & P_{25} \\ P_{31} & P_{32} & P_{33} & P_{34} & P_{35} \\ P_{41} & P_{42} & P_{43} & P_{44} & P_{45} \\ P_{51} & P_{52} & P_{53} & P_{54} & P_{55} \end{bmatrix},$$

and let $\mathcal{I} = \begin{bmatrix} 3 & 5 & 1 \end{bmatrix}$. Then,

$$\boldsymbol{\mu}_{\mathcal{I}} = \begin{bmatrix} \mu_3 \\ \mu_5 \\ \mu_1 \end{bmatrix}, \quad \mathbf{P}_{\mathcal{I}\mathcal{I}} = \begin{bmatrix} P_{33} & P_{35} & P_{31} \\ P_{53} & P_{55} & P_{51} \\ P_{13} & P_{15} & P_{11} \end{bmatrix}.$$

Selecting $\mathcal{I}$-rows and $\mathcal{I}$-columns of $\mathbf{P}$ to form $\mathbf{P}_{\mathcal{I}\mathcal{I}}$ is equivalent to selecting *all*-rows and $\mathcal{I}$-columns of $\mathbf{S}$, i.e.,

$$\mathbf{P}_{\mathcal{I}\mathcal{I}} = \mathbf{S}_{:,\mathcal{I}}^{\mathsf{T}} \mathbf{S}_{:,\mathcal{I}},$$

where $\mathbf{S}_{:,\mathcal{I}} \in \mathbb{R}^{n \times n_{\mathcal{I}}}$.

Since $\mathbf{S}_{:,\mathcal{I}}$ is a tall matrix that is not necessarily upper triangular, we can let

$$\mathbf{S}_{:,\mathcal{I}} = \underbrace{\begin{bmatrix} \mathbf{Y} & \mathbf{Z} \end{bmatrix}}_{\mathbf{Q}} \underbrace{\begin{bmatrix} \mathbf{R}_1 \\ \mathbf{0} \end{bmatrix}}_{\mathbf{R}}$$

be a QR factorisation, where $\mathbf{Q} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix and $\mathbf{R} \in \mathbb{R}^{n \times n_{\mathcal{I}}}$ is an upper triangular matrix. Then,

$$\begin{aligned} \mathbf{S}_{:,\mathcal{I}}^{\mathsf{T}} \mathbf{S}_{:,\mathcal{I}} &= \mathbf{R}^{\mathsf{T}} \mathbf{Q}^{\mathsf{T}} \mathbf{Q} \mathbf{R} = \mathbf{R}^{\mathsf{T}} \mathbf{R} && \text{(since } \mathbf{Q} \text{ is orthogonal)} \\ &= \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{0} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{0} \end{bmatrix} \\ &= \mathbf{R}_1^{\mathsf{T}} \mathbf{R}_1. \end{aligned}$$

Therefore, the marginal distribution of $\mathbf{x}_{\mathcal{I}}$ in square-root moment form is given by

$$p(\mathbf{x}_{\mathcal{I}}) = \mathcal{N}^{\frac{1}{2}}(\mathbf{x}_{\mathcal{I}}; \boldsymbol{\mu}_{\mathcal{I}}, \mathbf{R}_1). \tag{18}$$

Info

Note that the closer $\mathcal{I}$ is to indexing a head partition of $\mathbf{x}$, the cheaper the QR decomposition becomes, becoming essentially free in the case where $\mathcal{I} = \begin{bmatrix} 1 & 2 & 3 & \cdots & n_{\mathcal{I}} \end{bmatrix}$ exactly indexes a head partition, since $\mathbf{S}_{:,\mathcal{I}}$ is already upper triangular in this case.

Tasks

a) Remove the `doctest::skip` decorator from each `SCENARIO` in `test/src/GaussianMarginal.cpp` to enable these unit tests and run `ninja` to verify that they *fail* due to the incomplete implementation of (18) provided.

b) Complete the implementation of the `Gaussian::marginal` template function in `src/Gaussian.hpp`.

c) Ensure that the unit tests provided in `test/src/GaussianMarginal.cpp` pass.

1.3.1 Conditional distribution

Consider the joint distribution (12) partitioned into a head partition $\mathbf{x}_a$ and a tail partition $\mathbf{x}_b$. Then, the conditional distributions of the head given the tail, and the tail given the head, respectively, are given by

$$p(\mathbf{x}_a|\mathbf{x}_b) = \mathcal{N}(\mathbf{x}_a; \underbrace{\boldsymbol{\mu}_a + \mathbf{P}_{ab}\mathbf{P}_{bb}^{-1}(\mathbf{x}_b - \boldsymbol{\mu}_b)}_{\boldsymbol{\mu}_{a|b}}, \underbrace{\mathbf{P}_{aa} - \mathbf{P}_{ab}\mathbf{P}_{bb}^{-1}\mathbf{P}_{ba}}_{\mathbf{P}_{a|b}}), \tag{19a}$$

$$p(\mathbf{x}_b|\mathbf{x}_a) = \mathcal{N}(\mathbf{x}_b; \underbrace{\boldsymbol{\mu}_b + \mathbf{P}_{ba}\mathbf{P}_{aa}^{-1}(\mathbf{x}_a - \boldsymbol{\mu}_a)}_{\boldsymbol{\mu}_{b|a}}, \underbrace{\mathbf{P}_{bb} - \mathbf{P}_{ba}\mathbf{P}_{aa}^{-1}\mathbf{P}_{ab}}_{\mathbf{P}_{b|a}}). \tag{19b}$$

If we examine the conditional distribution $p(\mathbf{x}_b|\mathbf{x}_a)$ and compare the moment parameters to the square-root moment parameters, we find from (14) that

$$\begin{aligned}
\mathbf{P}_{b|a} &= \mathbf{P}_{bb} - \mathbf{P}_{ba}\mathbf{P}_{aa}^{-1}\mathbf{P}_{ab} \\
&= \underbrace{\mathbf{S}_2^{\mathsf{T}}\mathbf{S}_2 + \mathbf{S}_3^{\mathsf{T}}\mathbf{S}_3}_{\mathbf{P}_{bb}} - \underbrace{\mathbf{S}_2^{\mathsf{T}}\mathbf{S}_1}_{\mathbf{P}_{ba}}\underbrace{(\mathbf{S}_1^{\mathsf{T}}\mathbf{S}_1)^{-1}}_{\mathbf{P}_{aa}^{-1}}\underbrace{\mathbf{S}_1^{\mathsf{T}}\mathbf{S}_2}_{\mathbf{P}_{ab}} \\
&= \mathbf{S}_2^{\mathsf{T}}\mathbf{S}_2 + \mathbf{S}_3^{\mathsf{T}}\mathbf{S}_3 - \mathbf{S}_2^{\mathsf{T}}\mathbf{S}_1\mathbf{S}_1^{-1}\mathbf{S}_1^{-\mathsf{T}}\mathbf{S}_1^{\mathsf{T}}\mathbf{S}_2 \\
&= \mathbf{S}_2^{\mathsf{T}}\mathbf{S}_2 + \mathbf{S}_3^{\mathsf{T}}\mathbf{S}_3 - \mathbf{S}_2^{\mathsf{T}}\mathbf{S}_2 \\
&= \mathbf{S}_3^{\mathsf{T}}\mathbf{S}_3,
\end{aligned}$$

and

$$\begin{aligned}
\boldsymbol{\mu}_{b|a} &= \boldsymbol{\mu}_b + \mathbf{P}_{ba}\mathbf{P}_{aa}^{-1}(\mathbf{x}_a - \boldsymbol{\mu}_a) \\
&= \boldsymbol{\mu}_b + \underbrace{\mathbf{S}_2^{\mathsf{T}}\mathbf{S}_1}_{\mathbf{P}_{ba}}\underbrace{(\mathbf{S}_1^{\mathsf{T}}\mathbf{S}_1)^{-1}}_{\mathbf{P}_{aa}^{-1}}(\mathbf{x}_a - \boldsymbol{\mu}_a) \\
&= \boldsymbol{\mu}_b + \mathbf{S}_2^{\mathsf{T}}\mathbf{S}_1\mathbf{S}_1^{-1}\mathbf{S}_1^{-\mathsf{T}}(\mathbf{x}_a - \boldsymbol{\mu}_a) \\
&= \boldsymbol{\mu}_b + \mathbf{S}_2^{\mathsf{T}}\mathbf{S}_1^{-\mathsf{T}}(\mathbf{x}_a - \boldsymbol{\mu}_a).
\end{aligned}$$

Therefore, the conditional distribution $p(\mathbf{x}_a|\mathbf{x}_b)$ in square-root moment form is given as follows:

$$p(\mathbf{x}_b|\mathbf{x}_a) = \mathcal{N}^{\frac{1}{2}}(\mathbf{x}_b; \boldsymbol{\mu}_b + \mathbf{S}_2^{\mathsf{T}}\mathbf{S}_1^{-\mathsf{T}}(\mathbf{x}_a - \boldsymbol{\mu}_a), \mathbf{S}_3). \tag{20}$$

If we want the conditional distribution of the head partition, or any other subset of $\mathbf{x}$, we must permute that subset into the tail partition before employing (20). To do this, let us introduce the index vectors

$\mathcal{A} \in \mathbb{N}^{n_\mathcal{A}}$ and $\mathcal{B} \in \mathbb{N}^{n_\mathcal{B}}$ to select the desired elements of each set and denote these as $\mathbf{x}_\mathcal{A}$ and $\mathbf{x}_\mathcal{B}$, respectively.

The conditional distribution of $\mathbf{x}_\mathcal{A}$ given $\mathbf{x}_\mathcal{B}$ is given by

$$p(\mathbf{x}_\mathcal{A}|\mathbf{x}_\mathcal{B}) = \mathcal{N}(\mathbf{x}_\mathcal{A}; \boldsymbol{\mu}_{\mathcal{A}|\mathcal{B}}, \mathbf{P}_{\mathcal{A}|\mathcal{B}}),$$

where

$$\begin{aligned}\boldsymbol{\mu}_{\mathcal{A}|\mathcal{B}} &= \boldsymbol{\mu}_\mathcal{A} + \mathbf{P}_{\mathcal{A}\mathcal{B}}\mathbf{P}_{\mathcal{B}\mathcal{B}}^{-1}(\mathbf{x}_\mathcal{B} - \boldsymbol{\mu}_\mathcal{B}),\\ \mathbf{P}_{\mathcal{A}|\mathcal{B}} &= \mathbf{P}_{\mathcal{A}\mathcal{A}} - \mathbf{P}_{\mathcal{A}\mathcal{B}}\mathbf{P}_{\mathcal{B}\mathcal{B}}^{-1}\mathbf{P}_{\mathcal{B}\mathcal{A}}.\end{aligned}$$

To obtain this conditional distribution in square-root form, we permute the elements as follows:

$$p\left(\begin{bmatrix}\mathbf{x}_\mathcal{B}\\ \mathbf{x}_\mathcal{A}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{x}_\mathcal{B}\\ \mathbf{x}_\mathcal{A}\end{bmatrix}; \begin{bmatrix}\boldsymbol{\mu}_\mathcal{B}\\ \boldsymbol{\mu}_\mathcal{A}\end{bmatrix}, \begin{bmatrix}\mathbf{P}_{\mathcal{B}\mathcal{B}} & \mathbf{P}_{\mathcal{B}\mathcal{A}}\\ \mathbf{P}_{\mathcal{A}\mathcal{B}} & \mathbf{P}_{\mathcal{A}\mathcal{A}}\end{bmatrix}\right). \tag{21}$$

Selecting $\mathcal{B}$ and $\mathcal{A}$ rows and columns of $\mathbf{P}$ is equivalent to selecting the $\mathcal{B}$ and $\mathcal{A}$ *columns* of $\mathbf{S}$, i.e.,

$$\begin{bmatrix}\mathbf{P}_{\mathcal{B}\mathcal{B}} & \mathbf{P}_{\mathcal{B}\mathcal{A}}\\ \mathbf{P}_{\mathcal{A}\mathcal{B}} & \mathbf{P}_{\mathcal{A}\mathcal{A}}\end{bmatrix} = \begin{bmatrix}\mathbf{S}_{:,\mathcal{B}} & \mathbf{S}_{:,\mathcal{A}}\end{bmatrix}^\mathsf{T}\begin{bmatrix}\mathbf{S}_{:,\mathcal{B}} & \mathbf{S}_{:,\mathcal{A}}\end{bmatrix},$$

where $\mathbf{S}_{:,\mathcal{A}} \in \mathbb{R}^{n\times n_\mathcal{A}}$ and $\mathbf{S}_{:,\mathcal{B}} \in \mathbb{R}^{n\times n_\mathcal{B}}$.

Since $\begin{bmatrix}\mathbf{S}_{:,\mathcal{B}} & \mathbf{S}_{:,\mathcal{A}}\end{bmatrix}$ is not necessarily upper triangular, we can let

$$\begin{bmatrix}\mathbf{S}_{:,\mathcal{B}} & \mathbf{S}_{:,\mathcal{A}}\end{bmatrix} = \mathbf{Q}\underbrace{\begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2\\ \mathbf{0} & \mathbf{R}_3\end{bmatrix}}_{\mathbf{R}}$$

be a QR factorisation, where $\mathbf{Q} \in \mathbb{R}^{n\times n}$ is an orthogonal matrix and $\mathbf{R} \in \mathbb{R}^{n\times n}$ is an upper triangular matrix with blocks $\mathbf{R}_1 \in \mathbb{R}^{n_\mathcal{B}\times n_\mathcal{B}}$, $\mathbf{R}_2 \in \mathbb{R}^{n_\mathcal{B}\times n_\mathcal{A}}$ and $\mathbf{R}_3 \in \mathbb{R}^{n_\mathcal{A}\times n_\mathcal{A}}$, where $\mathbf{R}_1$ and $\mathbf{R}_3$ are upper triangular. Then,

$$\begin{aligned}\begin{bmatrix}\mathbf{S}_{:,\mathcal{B}} & \mathbf{S}_{:,\mathcal{A}}\end{bmatrix}^\mathsf{T}\begin{bmatrix}\mathbf{S}_{:,\mathcal{B}} & \mathbf{S}_{:,\mathcal{A}}\end{bmatrix} &= \mathbf{R}^\mathsf{T}\mathbf{Q}^\mathsf{T}\mathbf{Q}\mathbf{R} = \mathbf{R}^\mathsf{T}\mathbf{R} && \text{(since } \mathbf{Q} \text{ is orthogonal)}\\ &= \begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2\\ \mathbf{0} & \mathbf{R}_3\end{bmatrix}^\mathsf{T}\begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2\\ \mathbf{0} & \mathbf{R}_3\end{bmatrix}.\end{aligned}$$

$$p\left(\begin{bmatrix}\mathbf{x}_\mathcal{B}\\ \mathbf{x}_\mathcal{A}\end{bmatrix}\right) = \mathcal{N}^{\frac{1}{2}}\left(\begin{bmatrix}\mathbf{x}_\mathcal{B}\\ \mathbf{x}_\mathcal{A}\end{bmatrix}; \begin{bmatrix}\boldsymbol{\mu}_\mathcal{B}\\ \boldsymbol{\mu}_\mathcal{A}\end{bmatrix}, \begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2\\ \mathbf{0} & \mathbf{R}_3\end{bmatrix}\right). \tag{22}$$

Finally, the conditional distribution of $\mathbf{x}_\mathcal{A}$ given $\mathbf{x}_\mathcal{B}$ is given by the following:

$$p(\mathbf{x}_\mathcal{A}|\mathbf{x}_\mathcal{B}) = \mathcal{N}^{\frac{1}{2}}(\mathbf{x}_\mathcal{A}; \underbrace{\boldsymbol{\mu}_\mathcal{A} + \mathbf{R}_2^\mathsf{T}\mathbf{R}_1^{-\mathsf{T}}(\mathbf{x}_\mathcal{B} - \boldsymbol{\mu}_\mathcal{B})}_{\boldsymbol{\mu}_{\mathcal{A}|\mathcal{B}}}, \underbrace{\mathbf{R}_3}_{\mathbf{S}_{\mathcal{A}|\mathcal{B}}}). \tag{23}$$

Info

Note that the closer $\mathcal{A}$ is to indexing a tail partition of $\mathbf{x}$ and the closer $\mathcal{B}$ is to indexing a head partition, the cheaper the QR decomposition becomes, becoming essentially free in the case where $\mathcal{A}$ and $\mathcal{B}$ exactly index tail and head partitions, respectively.

Tasks

a) Remove the `doctest::skip` decorator from each `SCENARIO` in `test/src/GaussianConditional.cpp` to enable these unit tests and run `ninja` to verify that they *fail* due to the faulty implementation of (23) provided.

b) Complete the implementation of the `Gaussian::conditional` template function in `src/Gaussian.hpp`.

c) Ensure that the unit tests provided in `test/src/GaussianConditional.cpp` pass.

1.4 Confidence region

The Bayesian credible region (BCR) or *confidence region* of a pdf $p(\cdot)$ is given by the set

$$\mathcal{R} = \{\mathbf{x} \in \mathcal{X} \mid p(\mathbf{x}) \geq t\}, \tag{24a}$$

where the density threshold, t, is the unique solution to

$$\int_{p(\mathbf{x})\geq t} p(\mathbf{x})\,\mathrm{d}\mathbf{x} = c, \tag{24b}$$

where $0 < c < 1$ is the desired probability mass to be contained within the BCR. Common choices include $c = 0.95$ or $c = 0.997$, which correspond to the $\mu \pm 2\sigma$ and $\mu \pm 3\sigma$ regions of a Gaussian distribution, respectively.

If $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \mathbf{P})$ where $\mathbf{x} \in \mathbb{R}^n$, then the BCR (24) is an n-dimensional hyperellipsoid that defined by the points $\mathbf{x}$ that satisfy the following inequality:

$$(\mathbf{x} - \boldsymbol{\mu})^\mathsf{T} \mathbf{P}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \leq \chi^2_n(c), \tag{25}$$

where $\chi^2_n(c)$ is the inverse cumulative distribution function for probability c of the chi-squared distribution with n degrees of freedom[2].

The points $\mathbf{x}$ that lie on the boundary of (25) satisfy the following equation:

$$\underbrace{\begin{bmatrix}\mathbf{x}\\1\end{bmatrix}^\mathsf{T}}_{\mathbf{p}^\mathsf{T}} \underbrace{\begin{bmatrix}\mathbf{P}^{-1} & -\mathbf{P}^{-1}\boldsymbol{\mu}\\ -\boldsymbol{\mu}^\mathsf{T}\mathbf{P}^{-1} & \boldsymbol{\mu}^\mathsf{T}\mathbf{P}^{-1}\boldsymbol{\mu} - \chi^2_n(c)\end{bmatrix}}_{\mathbf{Q}} \underbrace{\begin{bmatrix}\mathbf{x}\\1\end{bmatrix}}_{\mathbf{p}} = 0, \tag{26}$$

which admits the following compact form in homogeneous coordinates:

$$\mathbf{p}^\mathsf{T}\mathbf{Q}\mathbf{p} = 0. \tag{27}$$

[2] In MATLAB, this is equivalent to $\chi^2_n(c) =$ `chi2inv`(c, n).

If we parameterise the covariance with an upper triangular matrix $\mathbf{S}$ such that $\mathbf{S}^\mathsf{T}\mathbf{S} = \mathbf{P}$, then (25) can be written as

$$\begin{aligned}(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}(\mathbf{S}^\mathsf{T}\mathbf{S})^{-1}(\mathbf{x}-\boldsymbol{\mu}) &\leq \chi^2_n(c)\\ (\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\mathbf{S}^{-1}\mathbf{S}^{-\mathsf{T}}(\mathbf{x}-\boldsymbol{\mu}) &\leq \chi^2_n(c)\\ \left(\mathbf{S}^{-\mathsf{T}}(\mathbf{x}-\boldsymbol{\mu})\right)^\mathsf{T}\mathbf{S}^{-\mathsf{T}}(\mathbf{x}-\boldsymbol{\mu}) &\leq \chi^2_n(c)\\ \mathbf{w}^\mathsf{T}\mathbf{w} &\leq \chi^2_n(c),\end{aligned} \tag{28}$$

where $\mathbf{w}$ is the solution to the following triangular system of equations:

$$\mathbf{S}^\mathsf{T}\mathbf{w} = \mathbf{x}-\boldsymbol{\mu}. \tag{29}$$

The matrix that appears in the homogeneous equation (27) is given by the following:

$$\mathbf{Q} = \begin{bmatrix}\mathbf{S}^{-1}\mathbf{S}^{-\mathsf{T}} & -\mathbf{y}\\ -\mathbf{y}^\mathsf{T} & \mathbf{z}^\mathsf{T}\mathbf{z}-\chi^2_n(c)\end{bmatrix}, \tag{30}$$

where $\mathbf{z}$ and $\mathbf{y}$ are the solutions to the following triangular systems of equations:

$$\begin{aligned}\mathbf{S}^\mathsf{T}\mathbf{z} &= \boldsymbol{\mu},\\ \mathbf{S}\mathbf{y} &= \mathbf{z}.\end{aligned}$$

Tasks

a) Remove the `doctest::skip` decorator from each `SCENARIO` in `test/src/GaussianConfidence.cpp` to enable these unit tests and run `ninja` to verify that they *fail* due to the incomplete implementations provided.

b) Complete the implementation of the `GaussianBase::normcdf` static function in `src/GaussianBase.hpp`, which evaluates the cumulative density function for a standard normal distribution[3]. To do this, you can use `std::erfc`, noting that

$$\texttt{normcdf}(w) \equiv \frac{1}{2}\cdot\texttt{erfc}\left(\frac{-w}{\sqrt{2}}\right). \tag{31}$$

c) Complete the implementation of $\chi^2_\nu(p)$ in the `GaussianBase::chi2inv` static function in `src/GaussianBase.hpp`, which is the inverse cumulative distribution function for probability p of the chi-squared distribution with ν degrees of freedom. To do this, you can use `boost::math::gamma_p_inv`, noting that

$$\texttt{chi2inv}(p,\nu) \equiv 2\cdot\texttt{gamma\_p\_inv}\left(\frac{\nu}{2},p\right). \tag{32}$$

Include the appropriate `boost` header file(s) within `src/GaussianBase.hpp` as needed.

d) Remove the `doctest::skip` decorator from each `SCENARIO` in `test/src/GaussianLogIntegral.cpp` to enable these unit tests. These tests should pass with the provided implementation of `GaussianBase::logIntegral` in `GaussianBase.hpp`.

e) Implement (28) in the `Gaussian::isWithinConfidenceRegion` member function in `src/Gaussian.hpp`.

[3] zero mean and unit variance

Hint

The probability mass, c, contained within n_σ standard deviations of the mean of a Gaussian pdf is given by

$$c = \int_{\mu - n_\sigma \sigma}^{\mu + n_\sigma \sigma} \mathcal{N}(x; \mu, \sigma^2)\, \mathrm{d}x = \int_{-n_\sigma}^{n_\sigma} \mathcal{N}(x; 0, 1)\, \mathrm{d}x, \tag{33}$$

which can be integrated numerically or computed using the normal cumulative distribution function.

f) Complete the implementation of the `GaussianBase::confidenceEllipse` member function in `src/GaussianBase.hpp`, which returns a set of points on the elliptical boundary of a given confidence region for a bivariate ($n = 2$) Gaussian distribution.

Hint

For $n = 2$, $\mathbf{w} \in \mathbb{R}^2$ and the boundary of (28) becomes the equation of a circle with radius $\sqrt{\chi^2_n(c)}$. Points on this circle can be found in $\mathbf{w}$-coordinates and then transformed back to $\mathbf{x}$-coordinates using (29) to obtain points on the boundary of the confidence ellipse.

g) Complete the implementation of (30) in the `Gaussian::quadricSurface` member function in `src/Gaussian.hpp`, which calculates the homogeneous matrix representation for the quadric surface $\mathbf{Q}$, for a trivariate ($n = 3$) Gaussian distribution.

h) Build the application and ensure the unit tests in `test/src/GaussianConfidence.cpp` pass.

2 Ballistic state estimation (2 marks)

A SpaceY Starship is re-entering earth's atmosphere and a Gaussian filter is needed to estimate its motion. SpaceY engineers were not expecting the Starship to fall in the orientation depicted in Figure 1, as such, the drag coefficient needs to be estimated online along with the altitude and velocity of the rocket.

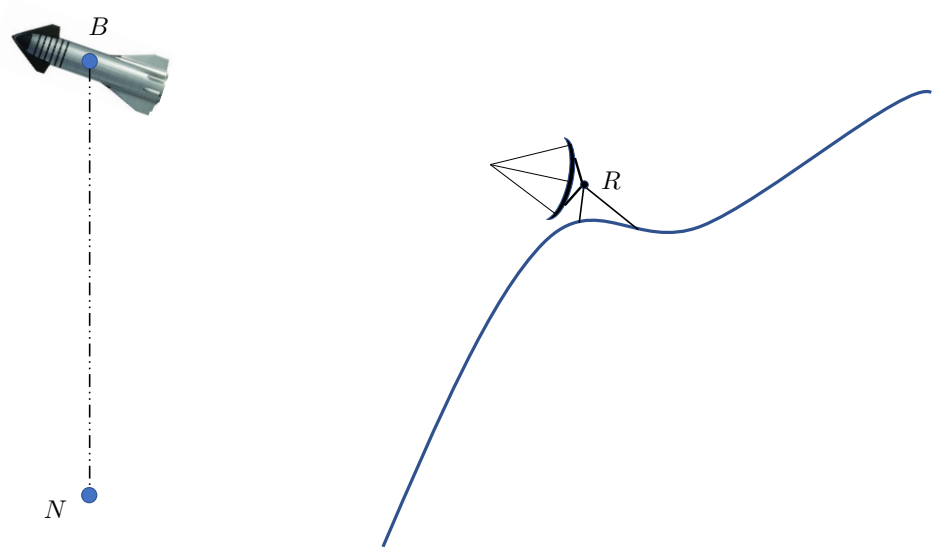

Figure 1: Vertical re-entry configuration of the SpaceY Starship

2.1 Ballistic process model

The following state vector has been proposed:

$$\mathbf{x} = \begin{bmatrix} h \\ v \\ c \end{bmatrix}, \tag{34}$$

where h is the altitude of the rocket above the reference point N, v is the velocity of the rocket and c is the ballistic drag coefficient to be estimated online.

The mean dynamics of the state vector can be expressed in the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \tag{35}$$

where the dynamics of the free-falling Starship are given by the following expression:

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ f_3(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} v \\ d - g \\ 0 \end{bmatrix}. \tag{36}$$

The acceleration d due to drag is given by

$$d = \frac{1}{2}\rho v^2 c \tag{37}$$

and the air density is given by

$$\rho = \frac{pM}{RT}, \tag{38}$$

where M is the molar mass of air, R is the ideal gas constant and T is the air temperature.

If the air is in hydrostatic equilibrium, then an infinitesimal change in pressure $\mathrm{d}p$ can be related to a infinitesimal change in altitude $\mathrm{d}h$ as follows:

$$\mathrm{d}p = -\rho g \, \mathrm{d}h. \tag{39}$$

Dividing both sides by p and substituting (38) yields

$$\frac{\mathrm{d}p}{p} = -\frac{\rho g}{p} \, \mathrm{d}h = -\frac{gM}{RT} \, \mathrm{d}h. \tag{40}$$

In the lower atmosphere, the temperature can be assumed to vary with altitude according to

$$T = T_0 - Lh, \tag{41}$$

where T_0 is the standard temperature and L is the temperature lapse rate. Substituting (41) into (40) and integrating yields

$$\begin{aligned}
\frac{\mathrm{d}p}{p} &= -\frac{gM}{R(T_0 - Lh)} \, \mathrm{d}h \\
\int_{p_0}^{p} \frac{1}{p'} \, \mathrm{d}p' &= -\frac{gM}{R} \int_0^h \frac{1}{T_0 - Lh'} \, \mathrm{d}h' \\
\left[\log p' \right]_{p'=p_0}^{p'=p} &= -\frac{gM}{R} \left[-\frac{1}{L} \log(T_0 - Lh') \right]_{h'=h_0}^{h'=h} \\
\log p - \log p_0 &= \frac{gM}{RL} (\log(T_0 - Lh) - \log T_0) \\
p &= p_0 \exp\left(\frac{gM}{RL} \log\left(1 - \frac{Lh}{T_0} \right) \right) \\
p &= p_0 \left(1 - \frac{Lh}{T_0} \right)^{\frac{gM}{RL}},
\end{aligned} \tag{42}$$

which is known as the barometric formula.

Substituting (42), (41) and (38) into (37) yields the drag acceleration as a function of states,

$$\begin{aligned}
d &= \frac{1}{2} \rho v^2 c \\
d &= \frac{1}{2} \frac{pM}{RT} v^2 c \\
d &= \frac{1}{2} \frac{M p_0}{R} \left(\frac{1}{T_0 - Lh} \right) \left(1 - \frac{Lh}{T_0} \right)^{\frac{gM}{RL}} v^2 c.
\end{aligned} \tag{43}$$

where $p_0 = 101\,325\,\mathrm{Pa}$ is the standard pressure at sea level, $T_0 = 288.15\,\mathrm{K}$ is the standard temperature at sea level, $g = 9.81\,\mathrm{m/s^2}$ is the acceleration due to gravity, $M = 0.028\,964\,4\,\mathrm{kg/mol}$ is the molar mass of dry air, $R = 8.314\,47\,\mathrm{J/(mol\,K)}$ is the ideal gas constant and $L = 0.0065\,\mathrm{K/m}$ is the temperature lapse rate in the lower atmosphere.

The process model for the rocket dynamics is then given by

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) + \dot{\mathbf{w}}(t), \qquad \dot{\mathbf{w}}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}\,\delta(t - t')), \tag{44}$$

where the power spectral density of the process noise is

$$\mathbf{Q} = \operatorname{diag}\left(\begin{bmatrix} 0 & 1 \times 10^{-20} & 25 \times 10^{-12} \end{bmatrix}\right). \tag{45}$$

Tasks

a) Remove the `doctest::skip` decorator from each `SCENARIO` in `test/src/SystemBallistic.cpp` to enable these unit tests and run `ninja` to verify that they *fail* due to the incomplete implementation provided.

b) Complete the implementation of the three `SystemBallistic::dynamics` functions in `src/SystemBallistic.cpp`.

c) Complete the implementations of the `SystemBallistic::processNoiseDensity` and `SystemBallistic::processNoiseIndex` functions in `src/SystemBallistic.cpp`

d) Ensure that the unit tests provided in `test/src/SystemBallistic.cpp` pass.

2.2 Range measurement model (1 mark)

A range-only RADAR (ROR) unit provides range measurements of the rocket from position $(r_1, r_2) = (5000\,\text{m}, 5000\,\text{m})$. Therefore, the measurement model is given by

$$h_{\text{rng}}(\mathbf{x}) = \sqrt{r_1^2 + (h - r_2)^2}, \tag{46}$$

with Gaussian measurement noise of $\sigma_{\text{rng}} = 50\,\text{m}$.

Tasks

a) Remove the `doctest::skip` decorator from each `SCENARIO` in `test/src/MeasurementRADAR.cpp` to enable these unit tests and run `ninja` to verify that they *fail* due to the incomplete implementation provided.

b) Complete the implementation of the `MeasurementRADAR::predict` functions in `src/MeasurementRADAR.cpp`.

c) Complete the implementation of `MeasurementRADAR::noise` in `src/MeasurementRADAR.cpp`.

d) Ensure that the unit tests provided in `test/src/MeasurementRADAR.cpp` pass.

2.3 State estimation and visualisation

The framework for an event-based square-root Gaussian filter has been implemented in the `DensityBase`, `GaussianBase`, `Gaussian`, `SystemBase`, `SystemEstimator Event` and `Measurement` classes (see the `.h/.cpp` or `.hpp` files for each). The `System*` and `Measurement*` classes are extended to `SystemBallistic` and `MeasurementRADAR`, respectively, to provide the problem-specific implementations of the process model dynamics and measurement model. More formally, the `SystemBallistic` class inherits from the

`SystemEstimator` class, while the `MeasurementRADAR` class inherits from the `MeasurementGaussianLikelihood` class, which in turn inherits from the `Measurement` class, which in turn inherits from `Event` class.

The main loop for each time step of the state estimator is implemented in `src/main.cpp`, which creates each measurement event and then processes it. Measurement events are processed by performing a time update on the state to the time stamp of the measurement followed by a measurement update. This can be seen in the `Event::process`, `SystemBallistic::predict` and `Measurement::update` functions.

The results are plotted in the `plot_simulation` function in `src/ballistic_plot.cpp` using Visualisation Toolkit (VTK), which is a powerful 2D and 3D visualisation library.

Tasks

a) Run the `lab6` application and ensure that you get a similar output to the following:

Terminal

```
nerd@basement:~/MCHA4400/lab6/build$ ninja && ./lab6
[Ninja-ing intensifies]
Reading data from ../data/estimationdata.csv
Found 501 rows within ../data/estimationdata.csv

Initial state estimate
mu[0] =
 14000
  -450
0.0005
P[0] =
4.84e+06        0        0
       0    10000        0
       0        0    1e-06
[t=000.000s] RADAR measurement update:........ done
[t=000.100s] RADAR measurement update:...... done
[t=000.200s] RADAR measurement update:..... done
[Filtering intensifies]
[t=049.800s] RADAR measurement update:.... done
[t=049.900s] RADAR measurement update:.... done
[t=050.000s] RADAR measurement update:.... done

Final state estimate
mu[end] =
    2185.02
   -152.732
0.000982088
P[end] =
    216.724     10.7621 0.000163735
    10.7621     1.57176 2.55059e-05
0.000163735 2.55059e-05 4.79729e-10
[A VTK plot window will display estimation results with some missing subplots]
```

b) Complete the implementation of the `plot_simulation` function, to produce a figure similar to Figure 2. The plot should include:

i) the true value of each state,

ii) the estimated mean for each state,

iii) the estimated 99.7% confidence region for each state,

iv) The marginal standard deviations of each state on a separate subfigures with a vertical log axis.

Tip

The VTK C++ examples are a useful resource for getting started with VTK. Some of the following API documentation may also be useful:

- vtkChartXY, for drawing 2-dimensional charts
- vtkPlot, for plotting lines within a chart
- vtkPlotArea, for plotting areas between two lines
- vtkChartLegend, for modifying legends of chart objects
- setLabel, for modifying labels of vtkPlot and vtkPlotArea objects.

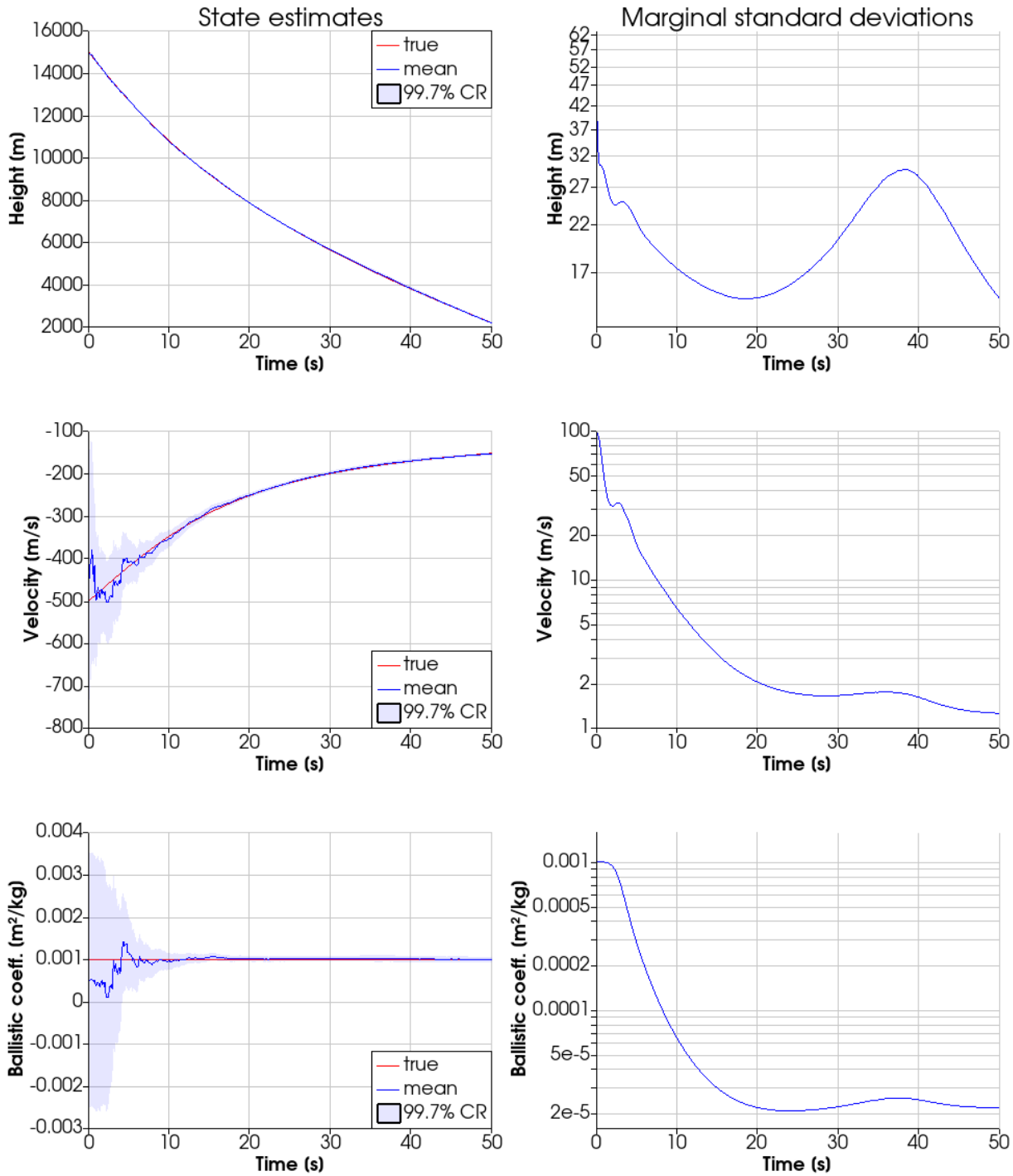

Figure 2: Expected plot.