Lab 7: Laplace information filter

MCHA4400

Semester 2 2025

Introduction

In this lab, you will derive gradients and Hessians of cost functions for use in optimisation using analytical derivatives and compare the performance against forward-mode and reverse-mode automatic differentiation. Then, you will consider the Gaussian distribution in square-root information form and use it to complete the implementation of a Laplace information filter (LIF) and use it on the same example problem considered in Lab 6.

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 Automatic differentiation and optimisation (1 mark)

In this task, you will implement the Rosenbrock function (see Figure 1) and its derivatives and use Newton and BFGS trust-region methods to find the minimum. The Rosenbrock function is given by

$$f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2. \tag{1}$$

The Rosenbrock function, also known as the banana function due to its shape, is a common benchmark for optimisation algorithms. The function's characteristic narrow, parabolic valley makes it challenging for many optimisation algorithms. While finding the valley is straightforward, converging to the global minimum is difficult due to the valley's sharp curvature and gentle slope towards the minimum.

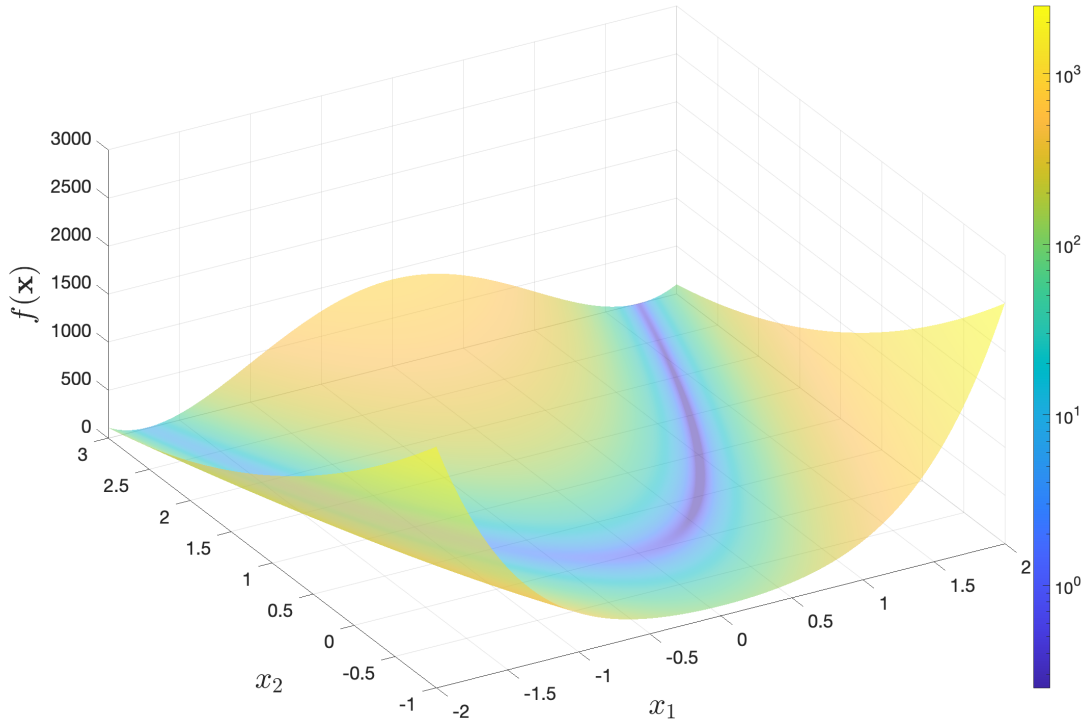

Figure 1: Rosenbrock banana function

Info

To enable the possibility of computing the derivatives of the Rosenbrock function using automatic differentiation, the `rosenbrock` function in `src/rosenbrock.cpp` has been templated to take an arbitrary `Scalar` type, such as `double` (default), `autodiff::real`, `autodiff::dual`, `autodiff::dual2nd` or `autodiff::val`.

Tasks

a) Remove the `doctest::skip` decorator from each `SCENARIO` (don't enable the `TEST_CASE`s just yet) in `test/src/rosenbrock.cpp` to enable these unit tests and run `ninja` to verify that they *fail* due to the incomplete implementation of the `RosenbrockAnalytical` functor provided.

Info

A functor (function object) is a C++ class that acts like a function. An instance of a functor can be called using the same syntax as a function, due to the implementation of `operator()`. In our case, it is convenient to bundle the Rosenbrock function and its derivatives into closely related functions within the same functor.

b) Implement the analytical gradient and Hessian of (1) in the `RosenbrockAnalytical` functor in `src/rosenbrock.cpp`.

c) Build the application and ensure the unit tests within each `SCENARIO` block in `test/src/rosenbrock.cpp` pass.

d) Remove the `doctest::skip` decorator from each `TEST_CASE` in `test/src/rosenbrock.cpp` to enable these additional unit tests and run `ninja` to verify that they *fail* due to the incomplete implementations of the `RosenbrockFwdAutoDiff` and `RosenbrockRevAutoDiff` functors provided.

e) Complete the implementation of the `RosenbrockFwdAutoDiff` functor using forward-mode autodiff in `src/rosenbrock.cpp`.

> **Info**
> The forward-mode `gradient` function propagates a `autodiff::dual` variable through the function of interest to compute its gradient, and the forward-mode `hessian` function propagates a `autodiff::dual2nd` variable through the function of interest to compute its Hessian.

> **GenAI Tip**
> The `autodiff` tutorial does not cover all the types and functions we will use and current LLMs may struggle to produce correct results until you include enough context with working examples. See the `VectorNormFwdAutoDiff` functor given in Lab 1 as an example of using forward-mode automatic differentiation to compute the gradient and Hessian of a scalar function.

f) Complete the implementation of the `RosenbrockRevAutoDiff` functor using reverse-mode autodiff in `src/rosenbrock.cpp`.

> **Info**
> The reverse-mode `gradient` and `hessian` functions require an `autodiff::var` variable that contains the expression tree built from a forward pass through the function.

> **GenAI Tip**
> The `autodiff` tutorial does not cover all the types and functions we will use and current LLMs may struggle to produce correct results until you include enough context with working examples. See the `VectorNormRevAutoDiff` functor given in Lab 1 as an example of using reverse-mode automatic differentiation to compute the gradient and Hessian of a scalar function.

g) Ensure that all unit tests provided in `test/src/rosenbrock.cpp` pass.

h) Switch to a release build, run `ninja` and review the benchmark results for the `Rosenbrock gradient only` and `Rosenbrock gradient and Hessian` cases. Compare the performance of forward-mode autodiff, reverse-mode autodiff, and the analytical expressions for evaluating the cost function gradient and Hessian.

Info

The performance benchmark is only meaningful in a release build, with assertions and bounds checking disabled and full compiler optimisations enabled. You can switch to a release build in VS Code directly via the CMake Tools status bar, or from the terminal as follows:

Terminal

```
nerd@basement:~/MCHA4400/lab7/build$ cd ..
nerd@basement:~/MCHA4400/lab7$ cmake -G Ninja -B build -DCMAKE_BUILD_TYPE=Release && cd build
nerd@basement:~/MCHA4400/lab7/build$ ninja
[Rosenbrock benchmark results]
```

You can switch back to a debug build from the VS Code CMake Tools status bar, or by running the above commands again with `Release` replaced with `Debug`.

i) In the `main` function in `src/main.cpp`, call the Newton trust region method `funcmin::NewtonTrust`, implemented in `src/funcmin.hpp`, to obtain the minimum of the Rosenbrock function.

j) Print the final minimiser, function value and its derivatives to the console as follows, and set the `verbosity` parameter of `funcmin::NewtonTrust` set to 3 to enable the iteration diagostic display:

Terminal

```
nerd@basement:~/MCHA4400/lab7/build$ ninja && ./lab7
[Ninja-ing intensifies]
[Output from tests and benchmarks]
[Output from optimisation]
Iter =    36, Cost =   4.92e-07, Newton decr^2 =   9.81e-07, Delta =   3.65e-01
Iter =    37, Cost =   2.74e-11, Newton decr^2 =   5.48e-11, Delta =   3.65e-01
Iter =    38, Cost =   6.49e-20, Newton decr^2 =   1.30e-19, Delta =   3.65e-01
CONVERGED: Newton decrement below threshold in 38 iterations

Final x =
[1.00000000024369;
 1.00000000047995]

f = 6.490384888509463e-20

g =
[ 3.45928752309987e-09;
 -1.48596691695602e-09]

H (actual) =
[ 802.000000392872, -400.000000097475;
 -400.000000097475,               200]

H (approx) =
[ 802.000000392872, -400.000000097475;
 -400.000000097475,               200]
```

k) Replace the call to `funcmin::NewtonTrust` with `funcmin::BFGSTrust` and re-run the optimisation. Note the number of iterations needed to converge and the quality of the Hessian approximation at the solution compared to the Newton trust-region method.

l) Experiment with the different Rosenbrock functors in `src/main.cpp` to ensure the implementations with analytical derivatives and forward-mode and reverse-mode automatic differentiation all yield the same results.

2 Gaussian filtering in square-root information form (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). If we parameterise the state Gaussian distribution by the information vector and information matrix, we obtain the family of information filters. This includes the extended information filter (EIF) and its second-order iterated variant, the Laplace information filter (LIF), which we will implement in square-root form in this lab.

The EIF is implemented simply using the marginal (see Section 2.2.1) and conditional (see Section 2.2.2) distributions and affine transform (see Section 2.3). The LIF differs from the EIF in that the measurement update is implemented by minimising a cost function which is the sum of the negative log-prior and negative log-likelihood functions. If we assume a Gaussian measurement likelihood function, then both terms in the cost function are simply logarithms of a Gaussian distribution (Sec-

tion 2.1). The minimiser of this cost yields the *maximum a posteriori* (MAP) estimate, which is taken as the posterior mean, and the Hessian matrix at the solution is taken as the posterior information matrix. Since we are using a square-root information parameterisation, it is convenient to use a square-root BFGS quasi-Newton optimisation method to implement the measurement update. This guarantees that the information matrix is always positive semi-definite and provides an estimate of the Hessian matrix without requiring second derivatives of the cost function.

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

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

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 information form as follows:

$$p(\mathbf{x}) = \mathcal{N}^{-\frac{1}{2}}(\mathbf{x}; \boldsymbol{\nu}, \boldsymbol{\Xi}) = \left|\det \frac{\boldsymbol{\Xi}}{\sqrt{2\pi}}\right| \exp\left(-\frac{1}{2}\left\|\boldsymbol{\Xi}\mathbf{x} - \boldsymbol{\nu}\right\|^2\right), \tag{3}$$

where $\boldsymbol{\nu} \in \mathbb{R}^n$ is the square-root information vector and $\boldsymbol{\Xi} \in \mathbb{R}^{n\times n}$ is the upper-triangular square-root information matrix, which satisfy the following relations with the moments:

$$\boldsymbol{\nu} = \boldsymbol{\Xi}\boldsymbol{\mu}, \tag{4a}$$

$$\boldsymbol{\Xi}^{\mathsf{T}}\boldsymbol{\Xi} = \mathbf{P}^{-1}. \tag{4b}$$

In the Laplace information 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 $p(\mathbf{x}_k|\mathbf{y}_{1:k-1})$,

$$\begin{aligned}
\boldsymbol{\mu}_{k|k} &= \arg\max_{\mathbf{x}_k} p(\mathbf{x}_k|\mathbf{y}_{1:k}) \\
&= \arg\max_{\mathbf{x}_k} \frac{p(\mathbf{y}_k|\mathbf{x}_k)\, p(\mathbf{x}_k|\mathbf{y}_{1:k-1})}{p(\mathbf{y}_k|\mathbf{y}_{1:k-1})} \\
&= \arg\max_{\mathbf{x}_k} p(\mathbf{y}_k|\mathbf{x}_k)\, p(\mathbf{x}_k|\mathbf{y}_{1:k-1}) \\
&= \arg\max_{\mathbf{x}_k} p(\mathbf{y}_k|\mathbf{x}_k)\, \mathcal{N}^{-\frac{1}{2}}(\mathbf{x}_k; \boldsymbol{\nu}_{k|k-1}, \boldsymbol{\Xi}_{k|k-1}).
\end{aligned}$$

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

$$\mathcal{V}(\mathbf{x}_k) = -\log p(\mathbf{y}_k|\mathbf{x}_k) - \log \mathcal{N}^{-\frac{1}{2}}(\mathbf{x}_k; \boldsymbol{\nu}_{k|k-1}, \boldsymbol{\Xi}_{k|k-1}). \tag{5}$$

Then, under the Laplace approxmation, the posterior square-root information vector and information matrix are given by

$$\begin{aligned}
\boldsymbol{\mu}_{k|k} &= \arg\min_{\mathbf{x}_k} \mathcal{V}(\mathbf{x}_k), \\
\boldsymbol{\Xi}_{k|k}^{\mathsf{T}}\boldsymbol{\Xi}_{k|k} &= \left.\frac{\partial^2 \mathcal{V}}{\partial \mathbf{x}_k^2}\right|_{\mathbf{x}_k = \boldsymbol{\mu}_{k|k}}, \\
\boldsymbol{\nu}_{k|k} &= \boldsymbol{\Xi}_{k|k}\boldsymbol{\mu}_{k|k},
\end{aligned}$$

where $\boldsymbol{\Xi}_{k|k}$ is upper triangular. The optimisation to find $\boldsymbol{\nu}_{k|k}$ and $\boldsymbol{\Xi}_{k|k}$ requires the gradient and Hessian,

$$\mathbf{g}_k = \frac{\partial \mathcal{V}}{\partial \mathbf{x}_k} = -\frac{\partial}{\partial \mathbf{x}_k} \log p(\mathbf{y}_k|\mathbf{x}_k) - \frac{\partial}{\partial \mathbf{x}_k} \log \mathcal{N}^{-\frac{1}{2}}(\mathbf{x}_k; \boldsymbol{\nu}_{k|k-1}, \boldsymbol{\Xi}_{k|k-1}), \tag{6a}$$

$$\mathbf{H}_k = \frac{\partial^2 \mathcal{V}}{\partial \mathbf{x}_k^2} = -\frac{\partial^2}{\partial \mathbf{x}_k^2} \log p(\mathbf{y}_k|\mathbf{x}_k) - \frac{\partial^2}{\partial \mathbf{x}_k^2} \log \mathcal{N}^{-\frac{1}{2}}(\mathbf{x}_k; \boldsymbol{\nu}_{k|k-1}, \boldsymbol{\Xi}_{k|k-1}), \tag{6b}$$

respectively.

Info

The cost (5), gradient (6a) and Hessian (6b) 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.,

$$p(\mathbf{y}_k|\mathbf{x}_k) = \mathcal{N}^{-\frac{1}{2}}(\mathbf{y}_k; \boldsymbol{\Xi}_{\mathbf{R}}\mathbf{h}(\mathbf{x}_k), \boldsymbol{\Xi}_{\mathbf{R}}),$$

for some mean function $\mathbf{h}(\cdot)$ and square-root information matrix $\boldsymbol{\Xi}_{\mathbf{R}}$, then we also need to be able to compute the derivatives, $\frac{\partial}{\partial \mathbf{x}_k} \log \mathcal{N}^{-\frac{1}{2}}(\mathbf{y}_k; \boldsymbol{\Xi}_{\mathbf{R}}\mathbf{h}(\mathbf{x}_k), \boldsymbol{\Xi}_{\mathbf{R}})$ and $\frac{\partial^2}{\partial \mathbf{x}_k^2} \log \mathcal{N}^{-\frac{1}{2}}(\mathbf{y}_k; \boldsymbol{\Xi}_{\mathbf{R}}\mathbf{h}(\mathbf{x}_k), \boldsymbol{\Xi}_{\mathbf{R}})$.

2.1 Log likelihood

The log of a Gaussian distribution in square-root information form (3) is given by

$$\begin{aligned}
\log p(\mathbf{x}) &= \log\left(\left|\det \frac{\boldsymbol{\Xi}}{\sqrt{2\pi}}\right| \exp\left(-\frac{1}{2}\left\|\boldsymbol{\Xi}\mathbf{x} - \boldsymbol{\nu}\right\|^2\right)\right) \\
&= \log\left|\det \frac{\boldsymbol{\Xi}}{\sqrt{2\pi}}\right| - \frac{1}{2}\left\|\boldsymbol{\Xi}\mathbf{x} - \boldsymbol{\nu}\right\|^2 \\
&= \log\left((2\pi)^{-\frac{n}{2}}\left|\det \boldsymbol{\Xi}\right|\right) - \frac{1}{2}\left\|\boldsymbol{\Xi}\mathbf{x} - \boldsymbol{\nu}\right\|^2 \\
&= -\frac{n}{2}\log 2\pi + \log\left|\det \boldsymbol{\Xi}\right| - \frac{1}{2}\left\|\boldsymbol{\Xi}\mathbf{x} - \boldsymbol{\nu}\right\|^2 \\
&= -\frac{n}{2}\log 2\pi + \log\left|\prod_{i=1}^{n} \boldsymbol{\Xi}_{ii}\right| - \frac{1}{2}\left\|\boldsymbol{\Xi}\mathbf{x} - \boldsymbol{\nu}\right\|^2 \\
&= -\frac{n}{2}\log 2\pi + \sum_{i=1}^{n} \log|\boldsymbol{\Xi}|_{ii} - \frac{1}{2}\left\|\boldsymbol{\Xi}\mathbf{x} - \boldsymbol{\nu}\right\|^2,
\end{aligned} \tag{7}$$

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

Tasks

a) Remove the `doctest::skip` decorator from each `SCENARIO` in `test/src/GaussianInfoLog.cpp` to enable these unit tests and run `ninja` to verify that they *fail* due to the incomplete implementation of (7) provided.

b) Complete a numerically robust implementation of (7) and its first two derivatives as the three `GaussianInfo::log` functions within `src/GaussianInfo.hpp`.

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

2.2 Joint distribution

Let $\mathbf{x} \in \mathbb{R}^n$ and consider the Gaussian distribution given in moment form (2). If we partition

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{bmatrix}, \tag{8}$$

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 (2) as

$$p\left(\begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{bmatrix}; \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix}, \begin{bmatrix} \mathbf{P}_{aa} & \mathbf{P}_{ab} \\ \mathbf{P}_{ba} & \mathbf{P}_{bb} \end{bmatrix}\right), \tag{9}$$

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 (9) in square-root information form as follows:

$$p\left(\begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{bmatrix}\right) = \mathcal{N}^{-\frac{1}{2}}\left(\begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{bmatrix}; \begin{bmatrix} \boldsymbol{\nu}_1 \\ \boldsymbol{\nu}_2 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Xi}_1 & \boldsymbol{\Xi}_2 \\ \mathbf{0} & \boldsymbol{\Xi}_3 \end{bmatrix}\right), \tag{10}$$

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

Then, from (4a), we have

$$\begin{aligned} \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix} &= \begin{bmatrix} \boldsymbol{\Xi}_1 & \boldsymbol{\Xi}_2 \\ \mathbf{0} & \boldsymbol{\Xi}_3 \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\nu}_1 \\ \boldsymbol{\nu}_2 \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{\Xi}_1^{-1} & -\boldsymbol{\Xi}_1^{-1}\boldsymbol{\Xi}_2\boldsymbol{\Xi}_3^{-1} \\ \mathbf{0} & \boldsymbol{\Xi}_3^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\nu}_1 \\ \boldsymbol{\nu}_2 \end{bmatrix}, \end{aligned} \tag{11a}$$

and similarly, from (4b), we have

$$\begin{aligned} \begin{bmatrix} \mathbf{P}_{aa} & \mathbf{P}_{ab} \\ \mathbf{P}_{ba} & \mathbf{P}_{bb} \end{bmatrix} &= \left(\begin{bmatrix} \boldsymbol{\Xi}_1 & \boldsymbol{\Xi}_2 \\ \mathbf{0} & \boldsymbol{\Xi}_3 \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \boldsymbol{\Xi}_1 & \boldsymbol{\Xi}_2 \\ \mathbf{0} & \boldsymbol{\Xi}_3 \end{bmatrix}\right)^{-1} \\ &= \begin{bmatrix} \boldsymbol{\Xi}_1 & \boldsymbol{\Xi}_2 \\ \mathbf{0} & \boldsymbol{\Xi}_3 \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\Xi}_1 & \boldsymbol{\Xi}_2 \\ \mathbf{0} & \boldsymbol{\Xi}_3 \end{bmatrix}^{-\mathsf{T}} \\ &= \begin{bmatrix} \boldsymbol{\Xi}_1^{-1} & -\boldsymbol{\Xi}_1^{-1}\boldsymbol{\Xi}_2\boldsymbol{\Xi}_3^{-1} \\ \mathbf{0} & \boldsymbol{\Xi}_3^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Xi}_1^{-\mathsf{T}} & \mathbf{0} \\ -\boldsymbol{\Xi}_3^{-\mathsf{T}}\boldsymbol{\Xi}_2^{\mathsf{T}}\boldsymbol{\Xi}_1^{-\mathsf{T}} & \boldsymbol{\Xi}_3^{-\mathsf{T}} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{\Xi}_1^{-1}\boldsymbol{\Xi}_1^{-\mathsf{T}} + \boldsymbol{\Xi}_1^{-1}\boldsymbol{\Xi}_2\boldsymbol{\Xi}_3^{-1}\boldsymbol{\Xi}_3^{-\mathsf{T}}\boldsymbol{\Xi}_2^{\mathsf{T}}\boldsymbol{\Xi}_1^{-\mathsf{T}} & -\boldsymbol{\Xi}_1^{-1}\boldsymbol{\Xi}_2\boldsymbol{\Xi}_3^{-1}\boldsymbol{\Xi}_3^{-\mathsf{T}} \\ -\boldsymbol{\Xi}_3^{-1}\boldsymbol{\Xi}_3^{-\mathsf{T}}\boldsymbol{\Xi}_2^{\mathsf{T}}\boldsymbol{\Xi}_1^{-\mathsf{T}} & \boldsymbol{\Xi}_3^{-1}\boldsymbol{\Xi}_3^{-\mathsf{T}} \end{bmatrix}. \end{aligned} \tag{11b}$$

We will use (11) to obtain results for the marginal and conditional distributions of a Gaussian distribution in square-root information form in the following subsections.

2.2.1 Marginal distribution

The marginal distributions of the head partition $\mathbf{x}_a$ and the tail partition $\mathbf{x}_b$ of (9) are 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 tail partition and compare the moment parameters to the square-root information parameters, we find from (11a) that

$$\begin{aligned} \boldsymbol{\mu}_b &= \boldsymbol{\Xi}_3^{-1}\boldsymbol{\nu}_2 \\ \boldsymbol{\Xi}_3\boldsymbol{\mu}_b &= \boldsymbol{\nu}_2, \end{aligned} \tag{12a}$$

and from (11b) that

$$\begin{aligned} \mathbf{P}_{bb} &= \boldsymbol{\Xi}_3^{-1}\boldsymbol{\Xi}_3^{-\mathsf{T}} \\ \boldsymbol{\Xi}_3^{\mathsf{T}}\boldsymbol{\Xi}_3 &= \mathbf{P}_{bb}^{-1}. \end{aligned} \tag{12b}$$

Therefore, the marginal distribution of the tail partition of a Gaussian distribution in square-root information form (10) is found simply from the partitions of its joint parameterisation as follows:

$$p(\mathbf{x}_b) = \mathcal{N}^{-\frac{1}{2}}(\mathbf{x}_b; \boldsymbol{\nu}_2, \boldsymbol{\Xi}_3). \tag{13}$$

If we want the marginal distribution of the head partition, or any other subset of $\mathbf{x}$, we must permute that subset into the tail partition before employing (13). 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

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

Furthermore, let $\bar{\mathcal{I}} = \mathbb{N}^n \setminus \mathcal{I} \in \mathbb{N}^{n-n_\mathcal{I}}$ denote a complementary index vector, which contains all the indices not contained within $\mathcal{I}$ (in any order). Each element of $\mathbf{x}_{\bar{\mathcal{I}}}$ is given by

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

Example

Let $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix}^\mathsf{T}$ and let the index vector $\mathcal{I} = \begin{bmatrix} 3 & 5 & 1 \end{bmatrix}$, which yields $\mathbf{x}_\mathcal{I} = \begin{bmatrix} x_3 & x_5 & x_1 \end{bmatrix}^\mathsf{T}$. A valid complementary index vector is $\bar{\mathcal{I}} = \begin{bmatrix} 2 & 4 \end{bmatrix}$, which yields $\mathbf{x}_{\bar{\mathcal{I}}} = \begin{bmatrix} x_2 & x_4 \end{bmatrix}^\mathsf{T}$.

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

$$p(\mathbf{x}_\mathcal{I}) = \mathcal{N}(\mathbf{x}_\mathcal{I}; \boldsymbol{\mu}_\mathcal{I}, \mathbf{P}_{\mathcal{I}\mathcal{I}}), \tag{14}$$

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 $\mathcal{I} = \begin{bmatrix} 3 & 5 & 1 \end{bmatrix}$, which yields

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

To find the marginal distribution of $\mathbf{x}_{\mathcal{I}}$ in square-root information form, we must permute $\mathbf{x}$ so that $\mathbf{x}_{\mathcal{I}}$ appears in the tail. To do this, let $\boldsymbol{\Pi} \in \{0,1\}^n$ be a permutation matrix[1] such that

$$\begin{bmatrix} \mathbf{x}_{\bar{\mathcal{I}}} \\ \mathbf{x}_{\mathcal{I}} \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{\Pi}_{\bar{\mathcal{I}}} \\ \boldsymbol{\Pi}_{\mathcal{I}} \end{bmatrix}}_{\boldsymbol{\Pi}} \mathbf{x}, \tag{15}$$

where $\boldsymbol{\Pi}_{\mathcal{I}} \in \{0,1\}^{n_{\mathcal{I}} \times n}$ and $\boldsymbol{\Pi}_{\bar{\mathcal{I}}} \in \{0,1\}^{n_{\bar{\mathcal{I}}} \times n}$ are row partitions of $\boldsymbol{\Pi}$. Since $\boldsymbol{\Pi}$ is also an orthogonal matrix, then $\boldsymbol{\Pi}^{-1} = \boldsymbol{\Pi}^{\mathsf{T}}$ and we can write

$$\mathbf{x} = \underbrace{\begin{bmatrix} \boldsymbol{\Pi}_{\bar{\mathcal{I}}}^{\mathsf{T}} & \boldsymbol{\Pi}_{\mathcal{I}}^{\mathsf{T}} \end{bmatrix}}_{\boldsymbol{\Pi}^{\mathsf{T}}} \begin{bmatrix} \mathbf{x}_{\bar{\mathcal{I}}} \\ \mathbf{x}_{\mathcal{I}} \end{bmatrix}. \tag{16}$$

Substituting (16) into (3) yields

$$\begin{aligned} p\left(\begin{bmatrix} \mathbf{x}_{\bar{\mathcal{I}}} \\ \mathbf{x}_{\mathcal{I}} \end{bmatrix}\right) &= \mathcal{N}^{-\frac{1}{2}}\left(\boldsymbol{\Pi}^{\mathsf{T}} \begin{bmatrix} \mathbf{x}_{\bar{\mathcal{I}}} \\ \mathbf{x}_{\mathcal{I}} \end{bmatrix}; \boldsymbol{\nu}, \boldsymbol{\Xi}\right) \\ &\propto \exp\left(-\frac{1}{2}\left\|\boldsymbol{\Xi}\boldsymbol{\Pi}^{\mathsf{T}} \begin{bmatrix} \mathbf{x}_{\bar{\mathcal{I}}} \\ \mathbf{x}_{\mathcal{I}} \end{bmatrix} - \boldsymbol{\nu}\right\|^2\right) \\ &\propto \exp\left(-\frac{1}{2}\left\|\begin{bmatrix} \boldsymbol{\Xi}_{:,\bar{\mathcal{I}}} & \boldsymbol{\Xi}_{:,\mathcal{I}} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\bar{\mathcal{I}}} \\ \mathbf{x}_{\mathcal{I}} \end{bmatrix} - \boldsymbol{\nu}\right\|^2\right), \end{aligned} \tag{17}$$

where we note that the term $\boldsymbol{\Xi}\boldsymbol{\Pi}^{\mathsf{T}}$ amounts to permuting the columns of $\boldsymbol{\Xi}$ as follows:

$$\begin{aligned} \boldsymbol{\Xi}\boldsymbol{\Pi}^{\mathsf{T}} &= \boldsymbol{\Xi}\begin{bmatrix} \boldsymbol{\Pi}_{\bar{\mathcal{I}}}^{\mathsf{T}} & \boldsymbol{\Pi}_{\mathcal{I}}^{\mathsf{T}} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{\Xi}\boldsymbol{\Pi}_{\bar{\mathcal{I}}}^{\mathsf{T}} & \boldsymbol{\Xi}\boldsymbol{\Pi}_{\mathcal{I}}^{\mathsf{T}} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{\Xi}_{:,\bar{\mathcal{I}}} & \boldsymbol{\Xi}_{:,\mathcal{I}} \end{bmatrix}. \end{aligned} \tag{18}$$

[1] A permutation matrix is an orthogonal matrix containing only 0 and 1 elements.

Since (18) is not necessarily upper triangular, let

$$\mathbf{Q}\underbrace{\begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2 & \mathbf{r}_1\\ \mathbf{0} & \mathbf{R}_3 & \mathbf{r}_2\end{bmatrix}}_{\mathbf{R}} = \begin{bmatrix}\boldsymbol{\Xi}_{:,\bar{\mathcal{I}}} & \boldsymbol{\Xi}_{:,\mathcal{I}} & \boldsymbol{\nu}\end{bmatrix}, \tag{19}$$

where $\mathbf{Q} \in \mathbb{R}^{n\times n}$ is an orthogonal matrix and $\mathbf{R} \in \mathbb{R}^{n\times(n+1)}$ is an upper-triangular matrix partitioned into $\mathbf{R}_1 \in \mathbb{R}^{n_{\bar{\mathcal{I}}}\times n_{\bar{\mathcal{I}}}}$, $\mathbf{R}_2 \in \mathbb{R}^{n_{\bar{\mathcal{I}}}\times n_{\mathcal{I}}}$, $\mathbf{R}_3 \in \mathbb{R}^{n_{\mathcal{I}}\times n_{\mathcal{I}}}$, $\mathbf{r}_1 \in \mathbb{R}^{n_{\bar{\mathcal{I}}}}$ and $\mathbf{r}_2 \in \mathbb{R}^{n_{\mathcal{I}}}$. Substituting the relevant terms from (19) into (17) yields

$$\begin{aligned}
p\left(\begin{bmatrix}\mathbf{x}_{\bar{\mathcal{I}}}\\ \mathbf{x}_{\mathcal{I}}\end{bmatrix}\right) &\propto \exp\left(-\frac{1}{2}\left\|\begin{bmatrix}\boldsymbol{\Xi}_{:,\bar{\mathcal{I}}} & \boldsymbol{\Xi}_{:,\mathcal{I}}\end{bmatrix}\begin{bmatrix}\mathbf{x}_{\bar{\mathcal{I}}}\\ \mathbf{x}_{\mathcal{I}}\end{bmatrix} - \boldsymbol{\nu}\right\|^2\right)\\
&\propto \exp\left(-\frac{1}{2}\left\|\mathbf{Q}\begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2\\ \mathbf{0} & \mathbf{R}_3\end{bmatrix}\begin{bmatrix}\mathbf{x}_{\bar{\mathcal{I}}}\\ \mathbf{x}_{\mathcal{I}}\end{bmatrix} - \mathbf{Q}\begin{bmatrix}\mathbf{r}_1\\ \mathbf{r}_2\end{bmatrix}\right\|^2\right)\\
&\propto \exp\left(-\frac{1}{2}\left\|\begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2\\ \mathbf{0} & \mathbf{R}_3\end{bmatrix}\begin{bmatrix}\mathbf{x}_{\bar{\mathcal{I}}}\\ \mathbf{x}_{\mathcal{I}}\end{bmatrix} - \begin{bmatrix}\mathbf{r}_1\\ \mathbf{r}_2\end{bmatrix}\right\|^2\right).
\end{aligned}$$

Therefore,

$$p\left(\begin{bmatrix}\mathbf{x}_{\bar{\mathcal{I}}}\\ \mathbf{x}_{\mathcal{I}}\end{bmatrix}\right) = \mathcal{N}^{-\frac{1}{2}}\left(\begin{bmatrix}\mathbf{x}_{\bar{\mathcal{I}}}\\ \mathbf{x}_{\mathcal{I}}\end{bmatrix}; \begin{bmatrix}\mathbf{r}_1\\ \mathbf{r}_2\end{bmatrix}, \begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2\\ \mathbf{0} & \mathbf{R}_3\end{bmatrix}\right). \tag{20}$$

Since $\mathbf{x}_{\mathcal{I}}$ is now the tail partition of a Gaussian distribution in square-root information form, its marginal distribution is given by

$$p(\mathbf{x}_{\mathcal{I}}) = \mathcal{N}^{-\frac{1}{2}}(\mathbf{x}_{\mathcal{I}}; \mathbf{r}_2, \mathbf{R}_3). \tag{21}$$

Tasks

a) Remove the `doctest::skip` decorator from each `SCENARIO` in `test/src/GaussianInfoMarginal.cpp` to enable these unit tests and run `ninja` to verify that they *fail* due to the incomplete implementation of (21) provided.

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

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

2.2.2 Conditional distribution

Consider the joint distribution (9) 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

$$p(\mathbf{x}_a|\mathbf{x}_b) = \mathcal{N}(\mathbf{x}_a; \underbrace{\boldsymbol{\mu}_a + \mathbf{P}_{ab}\mathbf{P}_{bb}^{-1}(\mathbf{x}_b - \boldsymbol{\mu}_b)}_{\boldsymbol{\mu}_{a|b}}, \underbrace{\mathbf{P}_{aa} - \mathbf{P}_{ab}\mathbf{P}_{bb}^{-1}\mathbf{P}_{ba}}_{\mathbf{P}_{a|b}}), \tag{22a}$$

$$p(\mathbf{x}_b|\mathbf{x}_a) = \mathcal{N}(\mathbf{x}_b; \underbrace{\boldsymbol{\mu}_b + \mathbf{P}_{ba}\mathbf{P}_{aa}^{-1}(\mathbf{x}_a - \boldsymbol{\mu}_a)}_{\boldsymbol{\mu}_{b|a}}, \underbrace{\mathbf{P}_{bb} - \mathbf{P}_{ba}\mathbf{P}_{aa}^{-1}\mathbf{P}_{ab}}_{\mathbf{P}_{b|a}}). \tag{22b}$$

If we examine the conditional distribution $p(\mathbf{x}_a|\mathbf{x}_b)$ and compare the moment parameters to the square-root information parameters, we find from (11b) that

$$\begin{aligned}
\mathbf{P}_{a|b} &= \mathbf{P}_{aa} - \mathbf{P}_{ab}\mathbf{P}_{bb}^{-1}\mathbf{P}_{ba} \\
&= \underbrace{\boldsymbol{\Xi}_1^{-1}\boldsymbol{\Xi}_1^{-\mathsf{T}} + \boldsymbol{\Xi}_1^{-1}\boldsymbol{\Xi}_2\boldsymbol{\Xi}_3^{-1}\boldsymbol{\Xi}_3^{-\mathsf{T}}\boldsymbol{\Xi}_2^{\mathsf{T}}\boldsymbol{\Xi}_1^{-\mathsf{T}}}_{\mathbf{P}_{aa}} - \underbrace{(-\boldsymbol{\Xi}_1^{-1}\boldsymbol{\Xi}_2\boldsymbol{\Xi}_3^{-1}\boldsymbol{\Xi}_3^{-\mathsf{T}})}_{\mathbf{P}_{ab}}\underbrace{(\boldsymbol{\Xi}_3^{\mathsf{T}}\boldsymbol{\Xi}_3)}_{\mathbf{P}_{bb}^{-1}}\underbrace{(-\boldsymbol{\Xi}_3^{-1}\boldsymbol{\Xi}_3^{-\mathsf{T}}\boldsymbol{\Xi}_2^{\mathsf{T}}\boldsymbol{\Xi}_1^{-\mathsf{T}})}_{\mathbf{P}_{ba}} \\
&= \boldsymbol{\Xi}_1^{-1}\boldsymbol{\Xi}_1^{-\mathsf{T}} + \boldsymbol{\Xi}_1^{-1}\boldsymbol{\Xi}_2\boldsymbol{\Xi}_3^{-1}\boldsymbol{\Xi}_3^{-\mathsf{T}}\boldsymbol{\Xi}_2^{\mathsf{T}}\boldsymbol{\Xi}_1^{-\mathsf{T}} - \boldsymbol{\Xi}_1^{-1}\boldsymbol{\Xi}_2\boldsymbol{\Xi}_3^{-1}\boldsymbol{\Xi}_3^{-\mathsf{T}}\boldsymbol{\Xi}_3^{\mathsf{T}}\boldsymbol{\Xi}_3\boldsymbol{\Xi}_3^{-1}\boldsymbol{\Xi}_3^{-\mathsf{T}}\boldsymbol{\Xi}_2^{\mathsf{T}}\boldsymbol{\Xi}_1^{-\mathsf{T}} \\
&= \boldsymbol{\Xi}_1^{-1}\boldsymbol{\Xi}_1^{-\mathsf{T}} + \boldsymbol{\Xi}_1^{-1}\boldsymbol{\Xi}_2\boldsymbol{\Xi}_3^{-1}\boldsymbol{\Xi}_3^{-\mathsf{T}}\boldsymbol{\Xi}_2^{\mathsf{T}}\boldsymbol{\Xi}_1^{-\mathsf{T}} - \boldsymbol{\Xi}_1^{-1}\boldsymbol{\Xi}_2\boldsymbol{\Xi}_3^{-1}\boldsymbol{\Xi}_3^{-\mathsf{T}}\boldsymbol{\Xi}_2^{\mathsf{T}}\boldsymbol{\Xi}_1^{-\mathsf{T}} \\
&= \boldsymbol{\Xi}_1^{-1}\boldsymbol{\Xi}_1^{-\mathsf{T}} \\
\boldsymbol{\Xi}_1^{\mathsf{T}}\boldsymbol{\Xi}_1 &= \mathbf{P}_{a|b}^{-1},
\end{aligned} \tag{23a}$$

and from (11a) that

$$\begin{aligned}
\boldsymbol{\mu}_{a|b} &= \boldsymbol{\mu}_a + \mathbf{P}_{ab}\mathbf{P}_{bb}^{-1}(\mathbf{x}_b - \boldsymbol{\mu}_b) \\
&= \boldsymbol{\mu}_a + \underbrace{-\boldsymbol{\Xi}_1^{-1}\boldsymbol{\Xi}_2\boldsymbol{\Xi}_3^{-1}\boldsymbol{\Xi}_3^{-\mathsf{T}}}_{\mathbf{P}_{ab}}\underbrace{\boldsymbol{\Xi}_3^{\mathsf{T}}\boldsymbol{\Xi}_3}_{\mathbf{P}_{bb}^{-1}}(\mathbf{x}_b - \boldsymbol{\mu}_b) \\
\boldsymbol{\mu}_{a|b} &= \boldsymbol{\mu}_a - \boldsymbol{\Xi}_1^{-1}\boldsymbol{\Xi}_2(\mathbf{x}_b - \boldsymbol{\mu}_b) \\
\boldsymbol{\Xi}_1\boldsymbol{\mu}_{a|b} &= \boldsymbol{\Xi}_1\boldsymbol{\mu}_a - \boldsymbol{\Xi}_2(\mathbf{x}_b - \boldsymbol{\mu}_b) \\
&= \boldsymbol{\Xi}_1\boldsymbol{\mu}_a + \boldsymbol{\Xi}_2\boldsymbol{\mu}_b - \boldsymbol{\Xi}_2\mathbf{x}_b \\
\boldsymbol{\nu}_{a|b} &= \boldsymbol{\nu}_1 - \boldsymbol{\Xi}_2\mathbf{x}_b,
\end{aligned} \tag{23b}$$

where we have used the property $\boldsymbol{\nu}_1 = \boldsymbol{\Xi}_1\boldsymbol{\mu}_a + \boldsymbol{\Xi}_2\boldsymbol{\mu}_b$ from (11a) and $\boldsymbol{\Xi}_1\boldsymbol{\mu}_{a|b} = \boldsymbol{\Xi}_{a|b}\boldsymbol{\mu}_{a|b} = \boldsymbol{\nu}_{a|b}$ from (23a) and (4a).

Therefore, the conditional distribution $p(\mathbf{x}_a|\mathbf{x}_b)$ in square-root information form (10) is given as follows:

$$p(\mathbf{x}_a|\mathbf{x}_b) = \mathcal{N}^{-\frac{1}{2}}(\mathbf{x}_a; \boldsymbol{\nu}_1 - \boldsymbol{\Xi}_2\mathbf{x}_b, \boldsymbol{\Xi}_1). \tag{24}$$

If we want the conditional distribution of the tail partition, or any other subset of $\mathbf{x}$, we must permute that subset into the head partition before employing (24). 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.

Then, let $\boldsymbol{\Pi} \in \{0,1\}^n$ be a permutation matrix such that

$$\begin{bmatrix}\mathbf{x}_\mathcal{A} \\ \mathbf{x}_\mathcal{B}\end{bmatrix} = \underbrace{\begin{bmatrix}\boldsymbol{\Pi}_\mathcal{A} \\ \boldsymbol{\Pi}_\mathcal{B}\end{bmatrix}}_{\boldsymbol{\Pi}}\mathbf{x}, \tag{25}$$

where $\boldsymbol{\Pi}_\mathcal{A} \in \{0,1\}^{n_\mathcal{A}\times n}$ and $\boldsymbol{\Pi}_\mathcal{B} \in \{0,1\}^{n_\mathcal{B}\times n}$ are row partitions of $\boldsymbol{\Pi}$. Since $\boldsymbol{\Pi}$ is also an orthogonal matrix, then $\boldsymbol{\Pi}^{-1} = \boldsymbol{\Pi}^{\mathsf{T}}$ and we can write

$$\mathbf{x} = \underbrace{\begin{bmatrix}\boldsymbol{\Pi}_\mathcal{A}^{\mathsf{T}} & \boldsymbol{\Pi}_\mathcal{B}^{\mathsf{T}}\end{bmatrix}}_{\boldsymbol{\Pi}^{\mathsf{T}}}\begin{bmatrix}\mathbf{x}_\mathcal{A} \\ \mathbf{x}_\mathcal{B}\end{bmatrix}. \tag{26}$$

Substituting (26) into (3) yields

$$\begin{aligned} p\left(\begin{bmatrix}\mathbf{x}_{\mathcal{A}}\\ \mathbf{x}_{\mathcal{B}}\end{bmatrix}\right) &= \mathcal{N}^{-\frac{1}{2}}\left(\boldsymbol{\Pi}^{\mathsf{T}}\begin{bmatrix}\mathbf{x}_{\mathcal{A}}\\ \mathbf{x}_{\mathcal{B}}\end{bmatrix};\boldsymbol{\nu},\boldsymbol{\Xi}\right) \\ &\propto \exp\left(-\frac{1}{2}\left\|\boldsymbol{\Xi}\boldsymbol{\Pi}^{\mathsf{T}}\begin{bmatrix}\mathbf{x}_{\mathcal{A}}\\ \mathbf{x}_{\mathcal{B}}\end{bmatrix}-\boldsymbol{\nu}\right\|^2\right) \\ &\propto \exp\left(-\frac{1}{2}\left\|\begin{bmatrix}\boldsymbol{\Xi}_{:,\mathcal{A}} & \boldsymbol{\Xi}_{:,\mathcal{B}}\end{bmatrix}\begin{bmatrix}\mathbf{x}_{\mathcal{A}}\\ \mathbf{x}_{\mathcal{B}}\end{bmatrix}-\boldsymbol{\nu}\right\|^2\right). \end{aligned} \tag{27}$$

Since $\begin{bmatrix}\boldsymbol{\Xi}_{:,\mathcal{A}} & \boldsymbol{\Xi}_{:,\mathcal{B}}\end{bmatrix}$ is not necessarily upper triangular, let

$$\mathbf{Q}\underbrace{\begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2 & \mathbf{r}_1\\ \mathbf{0} & \mathbf{R}_3 & \mathbf{r}_2\end{bmatrix}}_{\mathbf{R}} = \begin{bmatrix}\boldsymbol{\Xi}_{:,\mathcal{A}} & \boldsymbol{\Xi}_{:,\mathcal{B}} & \boldsymbol{\nu}\end{bmatrix}, \tag{28}$$

where $\mathbf{Q}\in\mathbb{R}^{n\times n}$ is an orthogonal matrix and $\mathbf{R}\in\mathbb{R}^{n\times(n+1)}$ is an upper-triangular matrix partitioned into $\mathbf{R}_1\in\mathbb{R}^{n_{\mathcal{A}}\times n_{\mathcal{A}}}$, $\mathbf{R}_2\in\mathbb{R}^{n_{\mathcal{A}}\times n_{\mathcal{B}}}$, $\mathbf{R}_3\in\mathbb{R}^{n_{\mathcal{B}}\times n_{\mathcal{B}}}$, $\mathbf{r}_1\in\mathbb{R}^{n_{\mathcal{A}}}$ and $\mathbf{r}_2\in\mathbb{R}^{n_{\mathcal{B}}}$. Substituting the relevant terms from (28) into (27) yields

$$\begin{aligned} p\left(\begin{bmatrix}\mathbf{x}_{\mathcal{A}}\\ \mathbf{x}_{\mathcal{B}}\end{bmatrix}\right) &\propto \exp\left(-\frac{1}{2}\left\|\begin{bmatrix}\boldsymbol{\Xi}_{:,\mathcal{A}} & \boldsymbol{\Xi}_{:,\mathcal{B}}\end{bmatrix}\begin{bmatrix}\mathbf{x}_{\mathcal{A}}\\ \mathbf{x}_{\mathcal{B}}\end{bmatrix}-\boldsymbol{\nu}\right\|^2\right) \\ &\propto \exp\left(-\frac{1}{2}\left\|\mathbf{Q}\begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2\\ \mathbf{0} & \mathbf{R}_3\end{bmatrix}\begin{bmatrix}\mathbf{x}_{\mathcal{A}}\\ \mathbf{x}_{\mathcal{B}}\end{bmatrix}-\mathbf{Q}\begin{bmatrix}\mathbf{r}_1\\ \mathbf{r}_2\end{bmatrix}\right\|^2\right) \\ &\propto \exp\left(-\frac{1}{2}\left\|\begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2\\ \mathbf{0} & \mathbf{R}_3\end{bmatrix}\begin{bmatrix}\mathbf{x}_{\mathcal{A}}\\ \mathbf{x}_{\mathcal{B}}\end{bmatrix}-\begin{bmatrix}\mathbf{r}_1\\ \mathbf{r}_2\end{bmatrix}\right\|^2\right). \end{aligned}$$

Therefore,

$$p\left(\begin{bmatrix}\mathbf{x}_{\mathcal{A}}\\ \mathbf{x}_{\mathcal{B}}\end{bmatrix}\right) = \mathcal{N}^{-\frac{1}{2}}\left(\begin{bmatrix}\mathbf{x}_{\mathcal{A}}\\ \mathbf{x}_{\mathcal{B}}\end{bmatrix};\begin{bmatrix}\mathbf{r}_1\\ \mathbf{r}_2\end{bmatrix},\begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2\\ \mathbf{0} & \mathbf{R}_3\end{bmatrix}\right). \tag{29}$$

Since $\mathbf{x}_{\mathcal{A}}$ is now the head partition of a Gaussian distribution in square-root information form, its conditional distribution given the tail $\mathbf{x}_{\mathcal{B}}$ is given by

$$p(\mathbf{x}_{\mathcal{A}}|\mathbf{x}_{\mathcal{B}}) = \mathcal{N}^{-\frac{1}{2}}(\mathbf{x}_{\mathcal{A}};\mathbf{r}_1-\mathbf{R}_2\mathbf{x}_{\mathcal{B}},\mathbf{R}_1). \tag{30}$$

Tasks

a) Remove the `doctest::skip` decorator from each `SCENARIO` in `test/src/GaussianInfoConditional.cpp` to enable these unit tests and run `ninja` to verify that they *fail* due to the faulty implementation of (30) provided.

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

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

2.3 Affine transform

Let $p(\mathbf{x}) = \mathcal{N}^{-\frac{1}{2}}(\mathbf{x}; \boldsymbol{\nu}, \boldsymbol{\Xi})$ be a Gaussian distribution in square-root information form, where $\mathbf{x}, \boldsymbol{\nu} \in \mathbb{R}^n$ and $\boldsymbol{\Xi} \in \mathbb{R}^{n \times n}$ 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, consider the first-order Taylor series approximation of $\mathbf{h}(\cdot)$ about the mean $\boldsymbol{\mu} = \boldsymbol{\Xi}^{-1}\boldsymbol{\nu}$,

$$\begin{aligned}\mathbf{h}(\mathbf{x}) &\approx \mathbf{h}(\boldsymbol{\mu}) + \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\boldsymbol{\mu}} (\mathbf{x} - \boldsymbol{\mu}) \\ &= \underbrace{\left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\boldsymbol{\mu}}}_{\mathbf{A}} \mathbf{x} + \underbrace{\mathbf{h}(\boldsymbol{\mu}) - \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\boldsymbol{\mu}} \boldsymbol{\mu}}_{\mathbf{b}},\end{aligned}$$

which defines the Jacobian matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and the vector $\mathbf{b} \in \mathbb{R}^m$.

In the case where $m = n = \operatorname{rank}(\mathbf{A})$, then $\mathbf{A}$ is invertible, and the inverse affine map $\mathbf{h}^{-1} \colon \mathbf{y} \mapsto \mathbf{x} = \mathbf{A}^{-1}(\mathbf{y} - \mathbf{b})$. Then, $p(\mathbf{y})$ is given by

$$\begin{aligned}p(\mathbf{y}) &= \left|\frac{\partial \mathbf{h}^{-1}(\mathbf{y})}{\partial \mathbf{y}}\right| p(\mathbf{h}^{-1}(\mathbf{y})) \\ &= \left|\mathbf{A}^{-1}\right| \mathcal{N}^{-\frac{1}{2}}(\mathbf{A}^{-1}(\mathbf{y} - \mathbf{b}); \boldsymbol{\nu}, \boldsymbol{\Xi}) \\ &= \left|\mathbf{A}^{-1}\right| \left|\frac{\boldsymbol{\Xi}}{\sqrt{2\pi}}\right| \exp\left(-\frac{1}{2}\left\|\boldsymbol{\Xi}(\mathbf{A}^{-1}(\mathbf{y} - \mathbf{b})) - \boldsymbol{\nu}\right\|^2\right) \\ &= \left|\frac{\boldsymbol{\Xi}\mathbf{A}^{-1}}{\sqrt{2\pi}}\right| \exp\left(-\frac{1}{2}\left\|\boldsymbol{\Xi}\mathbf{A}^{-1}\mathbf{y} - (\boldsymbol{\nu} + \boldsymbol{\Xi}\mathbf{A}^{-1}\mathbf{b})\right\|^2\right),\end{aligned} \tag{31}$$

where $|\cdot| = \operatorname{abs}\det(\cdot)$.

The distribution (31) is almost in square-root information form; however, $p(\mathbf{y}) \neq \mathcal{N}^{-\frac{1}{2}}(\mathbf{y}; \boldsymbol{\nu} + \boldsymbol{\Xi}\mathbf{A}^{-1}\mathbf{b}, \boldsymbol{\Xi}\mathbf{A}^{-1})$ since $\boldsymbol{\Xi}\mathbf{A}^{-1}$ is not necessarily an upper triangular matrix. To resolve this, let

$$\mathbf{Q}\mathbf{R} = \boldsymbol{\Xi}\mathbf{A}^{-1}, \tag{32}$$

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. Substituting (32) into (31) yields

$$\begin{aligned}p(\mathbf{y}) &= \left|\frac{\mathbf{Q}\mathbf{R}}{\sqrt{2\pi}}\right| \exp\left(-\frac{1}{2}\left\|\mathbf{Q}\mathbf{R}\mathbf{y} - (\boldsymbol{\nu} + \mathbf{Q}\mathbf{R}\mathbf{b})\right\|^2\right) \\ &= \left|\frac{\mathbf{R}}{\sqrt{2\pi}}\right| \exp\left(-\frac{1}{2}\left\|\mathbf{R}\mathbf{y} - (\mathbf{Q}^{\mathsf{T}}\boldsymbol{\nu} + \mathbf{R}\mathbf{b})\right\|^2\right) \\ &= \mathcal{N}^{-\frac{1}{2}}\left(\mathbf{y}; \mathbf{Q}^{\mathsf{T}}\boldsymbol{\nu} + \mathbf{R}\mathbf{b}, \mathbf{R}\right),\end{aligned} \tag{33}$$

where we have used the properties $\det \mathbf{Q} = \pm 1$ and $\|\mathbf{Q}\mathbf{u}\| = \|\mathbf{u}\|, \forall \mathbf{u} \in \mathbb{R}^n$.

To avoid the expense of computing $\mathbf{Q}$ to evaluate the term $\mathbf{Q}^{\mathsf{T}}\boldsymbol{\nu}$, let

$$\mathbf{Q}\begin{bmatrix}\mathbf{R} & \mathbf{r}\end{bmatrix} = \begin{bmatrix}\boldsymbol{\Xi}\mathbf{A}^{-1} & \boldsymbol{\nu}\end{bmatrix}, \tag{34}$$

where $\mathbf{Q} \in \mathbb{R}^{n\times n}$ is an orthogonal matrix and $\begin{bmatrix}\mathbf{R} & \mathbf{r}\end{bmatrix} \in \mathbb{R}^{n\times(n+1)}$ is an upper-triangular matrix with column partitions $\mathbf{R} \in \mathbb{R}^{n\times n}$ and $\mathbf{r} \in \mathbb{R}^n$. Then, from (34), we recover $\mathbf{QR} = \boldsymbol{\Xi}\mathbf{A}^{-1}$, which reproduces (32), and $\mathbf{r} = \mathbf{Q}^\mathsf{T}\boldsymbol{\nu}$. Therefore,

$$p(\mathbf{y}) = \mathcal{N}^{-\frac{1}{2}}(\mathbf{y}; \mathbf{r} + \mathbf{R}\mathbf{b}, \mathbf{R}), \tag{35}$$

and we can summarise this special case of the affine transform for a Gaussian distribution in square-root information form as follows:

$$\mathcal{T}^{-\frac{1}{2}}\{\mathbf{h}\}\colon (\boldsymbol{\nu}, \boldsymbol{\Xi}) \mapsto (\mathbf{r} + \mathbf{R}\mathbf{b}, \mathbf{R}), \tag{36}$$

where $\mathbf{r}$ and $\mathbf{R}$ are obtained from the Q-less QR decomposition (34).

The above covers the case where $m = n = \operatorname{rank}(\mathbf{A})$ so that we can directly use $\mathbf{A}^{-1}$ in (32). To extend this result to a more general case where $m \neq n$ or $\operatorname{rank}(\mathbf{A}) < \min(m, n)$, we can proceed by letting

$$\mathbf{A} = \underbrace{\begin{bmatrix}\mathbf{U}_1 & \mathbf{U}_2\end{bmatrix}}_{\mathbf{U}} \underbrace{\begin{bmatrix}\boldsymbol{\Sigma}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}\end{bmatrix}}_{\boldsymbol{\Sigma}} \underbrace{\begin{bmatrix}\mathbf{V}_1 & \mathbf{V}_2\end{bmatrix}^\mathsf{T}}_{\mathbf{V}^\mathsf{T}}$$

be a singular value decomposition (SVD), where $\mathbf{U} \in \mathbb{R}^{m\times m}$ is an orthogonal matrix with columns containing the left singular vectors of $\mathbf{A}$, $\mathbf{V} \in \mathbb{R}^{n\times n}$ is an orthogonal matrix with columns containing the right singular vectors of $\mathbf{A}$, and $\boldsymbol{\Sigma} \in \mathbb{R}^{m\times n}$ is a diagonal matrix containing the singular values of $\mathbf{A}$. The partitions, $\mathbf{U}_1 \in \mathbb{R}^{m\times r}$, $\mathbf{U}_2 \in \mathbb{R}^{m\times(m-r)}$, $\mathbf{V}_1 \in \mathbb{R}^{n\times r}$, $\mathbf{V}_2 \in \mathbb{R}^{n\times(n-r)}$, and $\boldsymbol{\Sigma}_{11} \in \mathbb{R}^{r\times r}$ are defined such that $r = \operatorname{rank}(\mathbf{A}) \leq \min(m, n)$. Then, we can write the affine map as follows:

$$\begin{aligned}\mathbf{y} &= \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\mathsf{T}\mathbf{x} + \mathbf{b} \\ &= \mathbf{U}_1\boldsymbol{\Sigma}_{11}\mathbf{V}_1^\mathsf{T}\mathbf{x} + \mathbf{b}.\end{aligned} \tag{37}$$

Multiplying (37) from the left by $\mathbf{U}_1^\mathsf{T}$ yields

$$\mathbf{U}_1^\mathsf{T}\mathbf{y} = \mathbf{U}_1^\mathsf{T}\mathbf{U}_1\boldsymbol{\Sigma}_{11}\mathbf{V}_1^\mathsf{T}\mathbf{x} + \mathbf{U}_1^\mathsf{T}\mathbf{b}.$$

Since $\mathbf{U}_1^\mathsf{T}\mathbf{U}_1 = \mathbf{I}$, this can be rearranged as follows:

$$\mathbf{V}_1^\mathsf{T}\mathbf{x} = \boldsymbol{\Sigma}_{11}^{-1}\mathbf{U}_1^\mathsf{T}(\mathbf{y} - \mathbf{b}). \tag{38}$$

Since $\mathbf{V}_1^\mathsf{T}$ is a wide matrix, this system of equations is underdetermined in $\mathbf{x}$ for a given $\mathbf{y}$. To proceed further, let us define the new variable $\mathbf{z} = \mathbf{V}_2^\mathsf{T}\mathbf{x} \in \mathbb{R}^{n-r}$ and write

$$\begin{aligned}\begin{bmatrix}\mathbf{V}_1^\mathsf{T} \\ \mathbf{V}_2^\mathsf{T}\end{bmatrix}\mathbf{x} &= \begin{bmatrix}\boldsymbol{\Sigma}_{11}^{-1}\mathbf{U}_1^\mathsf{T}(\mathbf{y} - \mathbf{b}) \\ \mathbf{z}\end{bmatrix} \\ \mathbf{x} &= \begin{bmatrix}\mathbf{V}_1 & \mathbf{V}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{\Sigma}_{11}^{-1}\mathbf{U}_1^\mathsf{T}(\mathbf{y} - \mathbf{b}) \\ \mathbf{z}\end{bmatrix} \\ \mathbf{x} &= \mathbf{V}_1\boldsymbol{\Sigma}_{11}^{-1}\mathbf{U}_1^\mathsf{T}(\mathbf{y} - \mathbf{b}) + \mathbf{V}_2\mathbf{z} \\ \mathbf{x} &= \begin{bmatrix}\mathbf{V}_2 & \mathbf{V}_1\boldsymbol{\Sigma}_{11}^{-1}\mathbf{U}_1^\mathsf{T}\end{bmatrix}\begin{bmatrix}\mathbf{z} \\ \mathbf{y}\end{bmatrix} - \mathbf{V}_1\boldsymbol{\Sigma}_{11}^{-1}\mathbf{U}_1^\mathsf{T}\mathbf{b}.\end{aligned} \tag{39}$$

This can be substituted into $p(\mathbf{x}) = \mathcal{N}^{-\frac{1}{2}}(\mathbf{x}; \boldsymbol{\nu}, \boldsymbol{\Xi}) \propto \exp\left(-\frac{1}{2}\|\boldsymbol{\Xi}\mathbf{x} - \boldsymbol{\nu}\|^2\right)$ to yield

$$p(\mathbf{z}, \mathbf{y}) \propto \exp\left(-\frac{1}{2}\left\|\boldsymbol{\Xi}\begin{bmatrix}\mathbf{V}_2 & \mathbf{V}_1\boldsymbol{\Sigma}_{11}^{-1}\mathbf{U}_1^{\mathsf{T}}\end{bmatrix}\begin{bmatrix}\mathbf{z}\\ \mathbf{y}\end{bmatrix} - (\boldsymbol{\nu} + \boldsymbol{\Xi}\mathbf{V}_1\boldsymbol{\Sigma}_{11}^{-1}\mathbf{U}_1^{\mathsf{T}}\mathbf{b})\right\|^2\right),$$

which we could then marginalise out $\mathbf{z}$ to obtain $p(\mathbf{y})$; however, this distribution has support for values of $\mathbf{y}$ that do not satisfy (37). For example, any $\mathbf{y}$ of the form $\mathbf{U}_1\boldsymbol{\Sigma}_{11}\mathbf{V}_1^{\mathsf{T}}\mathbf{x} + \mathbf{b} + \mathbf{U}_2\mathbf{w}$, where $\mathbb{R}^{m-r} \ni \mathbf{w} \neq \mathbf{0}$ satisfies (38) since $\mathbf{U}_1^{\mathsf{T}}\mathbf{U}_2 = \mathbf{0}$, but this is not equal to $\mathbf{y}$ from (37).

Multiplying (37) from the left by $\mathbf{U}_2^{\mathsf{T}}$ yields

$$\mathbf{U}_2^{\mathsf{T}}\mathbf{y} = \mathbf{U}_2^{\mathsf{T}}\mathbf{U}_1\boldsymbol{\Sigma}_{11}\mathbf{V}_1^{\mathsf{T}}\mathbf{x} + \mathbf{U}_2^{\mathsf{T}}\mathbf{b}.$$

Since the columns of $\mathbf{U}_1$ and $\mathbf{U}_2$ are orthogonal, then $\mathbf{U}_2^{\mathsf{T}}\mathbf{U}_1 = \mathbf{0}$ and we obtain the following constraint:

$$\mathbf{U}_2^{\mathsf{T}}(\mathbf{y} - \mathbf{b}) = \mathbf{0}. \tag{40}$$

To enforce this constraint, and remove support for values of $\mathbf{y}$ that do not satisfy (37), we can let

$$\begin{aligned} p(\mathbf{z}, \mathbf{y}) &\propto \exp\left(-\frac{1}{2}\left\|\boldsymbol{\Xi}\begin{bmatrix}\mathbf{V}_2 & \mathbf{V}_1\boldsymbol{\Sigma}_{11}^{-1}\mathbf{U}_1^{\mathsf{T}}\end{bmatrix}\begin{bmatrix}\mathbf{z}\\ \mathbf{y}\end{bmatrix} - (\boldsymbol{\nu} + \boldsymbol{\Xi}\mathbf{V}_1\boldsymbol{\Sigma}_{11}^{-1}\mathbf{U}_1^{\mathsf{T}}\mathbf{b})\right\|^2 - \frac{1}{2}\left\|\kappa\mathbf{U}_2^{\mathsf{T}}(\mathbf{y} - \mathbf{b})\right\|^2\right) \\ &\propto \exp\left(-\frac{1}{2}\left\|\begin{bmatrix}\boldsymbol{\Xi}\mathbf{V}_2 & \boldsymbol{\Xi}\mathbf{V}_1\boldsymbol{\Sigma}_{11}^{-1}\mathbf{U}_1^{\mathsf{T}}\\ \mathbf{0} & \kappa\mathbf{U}_2^{\mathsf{T}}\end{bmatrix}\begin{bmatrix}\mathbf{z}\\ \mathbf{y}\end{bmatrix} - \begin{bmatrix}\boldsymbol{\nu} + \boldsymbol{\Xi}\mathbf{V}_1\boldsymbol{\Sigma}_{11}^{-1}\mathbf{U}_1^{\mathsf{T}}\mathbf{b}\\ \kappa\mathbf{U}_2^{\mathsf{T}}\mathbf{b}\end{bmatrix}\right\|^2\right), \end{aligned}$$

where $\kappa \gg \left\|\boldsymbol{\Xi}\begin{bmatrix}\mathbf{V}_2 & \mathbf{V}_1\boldsymbol{\Sigma}_{11}^{-1}\mathbf{U}_1^{\mathsf{T}}\end{bmatrix}\right\|_2$ is a large positive constant used to penalise the constraint violation and $\|\cdot\|_2$ denotes the 2-norm of a matrix[2].

Since the matrix $\begin{bmatrix}\boldsymbol{\Xi}\mathbf{V}_2 & \boldsymbol{\Xi}\mathbf{V}_1\boldsymbol{\Sigma}_{11}^{-1}\mathbf{U}_1^{\mathsf{T}}\\ \mathbf{0} & \kappa\mathbf{U}_2^{\mathsf{T}}\end{bmatrix}$ is not upper triangular, let

$$\mathbf{Q}\begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2 & \boldsymbol{\nu}_1\\ \mathbf{0} & \mathbf{R}_3 & \boldsymbol{\nu}_2\end{bmatrix} = \begin{bmatrix}\boldsymbol{\Xi}\mathbf{V}_2 & \boldsymbol{\Xi}\mathbf{V}_1\boldsymbol{\Sigma}_{11}^{-1}\mathbf{U}_1^{\mathsf{T}} & \boldsymbol{\nu} + \boldsymbol{\Xi}\mathbf{V}_1\boldsymbol{\Sigma}_{11}^{-1}\mathbf{U}_1^{\mathsf{T}}\mathbf{b}\\ \mathbf{0} & \kappa\mathbf{U}_2^{\mathsf{T}} & \kappa\mathbf{U}_2^{\mathsf{T}}\mathbf{b}\end{bmatrix}, \tag{41}$$

where $\mathbf{Q} \in \mathbb{R}^{(m+n-r)\times(m+n-r)}$ is an orthogonal matrix, $\mathbf{R}_1 \in \mathbb{R}^{(n-r)\times(n-r)}$ is upper triangular, $\mathbf{R}_2 \in \mathbb{R}^{(n-r)\times m}$, $\mathbf{R}_3 \in \mathbb{R}^{m\times m}$ is upper triangular, $\boldsymbol{\nu}_1 \in \mathbb{R}^{n-r}$ and $\boldsymbol{\nu}_2 \in \mathbb{R}^m$.

Since $\mathbf{y}$ forms the tail partition of $\left[\begin{smallmatrix}\mathbf{z}\\ \mathbf{y}\end{smallmatrix}\right]$, the marginal distribution $p(\mathbf{y})$ is obtained directly from (13), which yields

$$p(\mathbf{y}) = \mathcal{N}^{-\frac{1}{2}}(\mathbf{y}; \boldsymbol{\nu}_2, \mathbf{R}_3). \tag{42}$$

Therefore we can summarise the general case of an affine transform for a Gaussian distribution in square-root information form as follows:

$$\mathcal{T}^{-\frac{1}{2}}\{\mathbf{h}\}\colon (\boldsymbol{\nu}, \boldsymbol{\Xi}) \mapsto (\boldsymbol{\nu}_2, \mathbf{R}_3), \tag{43}$$

where $\boldsymbol{\nu}_2$ and $\mathbf{R}_3$ are obtained from the Q-less QR decomposition (41).

[2]The 2-norm of a matrix is equal to the largest singular value of that matrix.

Tasks

a) Remove the `doctest::skip` decorator from each `SCENARIO` in `test/src/GaussianInfoTransform.cpp` to enable these unit tests and run `ninja` to verify that they *fail* due to the incomplete implementation of (42) provided.

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

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

2.4 Confidence region

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

$$\mathcal{R} = \{\mathbf{x} \in \mathcal{X} \mid p(\mathbf{x}) \geq t\}, \tag{44a}$$

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

$$\int_{p(\mathbf{x})\geq t} p(\mathbf{x})\,\mathrm{d}\mathbf{x} = c, \tag{44b}$$

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 $p(\mathbf{x}) = \mathcal{N}^{-\frac{1}{2}}(\mathbf{x}; \boldsymbol{\nu}, \boldsymbol{\Xi})$ where $\mathbf{x} \in \mathbb{R}^n$, then the BCR (44) is an n-dimensional hyperellipsoid that defined by the points $\mathbf{x}$ that satisfy the following inequality:

$$(\boldsymbol{\Xi}\mathbf{x} - \boldsymbol{\nu})^\mathsf{T}(\boldsymbol{\Xi}\mathbf{x} - \boldsymbol{\nu}) \leq \chi^2_n(c), \tag{45}$$

where $\chi^2_n(c)$ is the inverse cumulative distribution function for probability c of the chi-squared distribution with n degrees of freedom[3].

For $n = 2$, let $\mathbf{w} = \boldsymbol{\Xi}\mathbf{x} - \boldsymbol{\nu}$ so that (45) can be written as

$$\mathbf{w}^\mathsf{T}\mathbf{w} \leq \chi^2_2(c). \tag{46}$$

Points on the boundary of this region form a circle in $\mathbf{w}$-coordinates with radius $\sqrt{\chi^2_2(c)}$, which can be transformed back into $\mathbf{x}$-coordinates by solving the following triangular system of equations:

$$\boldsymbol{\Xi}\mathbf{x} = \mathbf{w} + \boldsymbol{\nu}. \tag{47}$$

For $n = 3$, the points $\mathbf{x}$ that lie on the boundary of (45) describe a quadric surface, which satisfies the following equation:

$$\underbrace{\begin{bmatrix}\mathbf{x}\\1\end{bmatrix}^\mathsf{T}}_{\mathbf{p}^\mathsf{T}} \underbrace{\begin{bmatrix}\boldsymbol{\Xi}^\mathsf{T}\boldsymbol{\Xi} & -\boldsymbol{\Xi}^\mathsf{T}\boldsymbol{\nu}\\ -\boldsymbol{\nu}^\mathsf{T}\boldsymbol{\Xi} & \boldsymbol{\nu}^\mathsf{T}\boldsymbol{\nu} - \chi^2_3(c)\end{bmatrix}}_{\mathbf{Q}} \underbrace{\begin{bmatrix}\mathbf{x}\\1\end{bmatrix}}_{\mathbf{p}} = 0, \tag{48}$$

which admits the following compact form in homogeneous coordinates:

$$\mathbf{p}^\mathsf{T}\mathbf{Q}\mathbf{p} = 0. \tag{49}$$

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

Tasks

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

b) Merge your implementation of `src/GaussianBase.hpp` from Lab 6 into your working directory for this lab.

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

d) Implement (46) in `GaussianInfo::isWithinConfidenceRegion`.

e) Complete the implementation of `GaussianInfo::quadricSurface`, which calculates the homogeneous matrix representation for the quadric surface $\mathbf{Q}$ from (48), for a trivariate ($n = 3$) Gaussian distribution.

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

3 Ballistic state estimation (1 mark)

For this problem, we consider the same ballistic state estimation problem as described in Lab 6, but instead use a Gaussian filter implemented in square-root information form.

Tasks

a) In `src/ballistic_plot.cpp`, merge the `plot_simulation` function from Lab 6.

b) 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.

c) In `src/SystemBallistic.cpp`, merge the `SystemBallistic::dynamics`, `SystemBallistic::processNoiseDensity` and `SystemBallistic::processNoiseIndex` functions from Lab 6, noting that instances of `Gaussian` are replaced with `GaussianInfo`.

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

e) 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.

f) In `src/MeasurementRADAR.cpp`, merge the `MeasurementRADAR::predict` and `MeasurementRADAR::noiseDensity` functions from Lab 6, noting that instances of `Gaussian` are replaced with `GaussianInfo`.

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

h) In `src/main.cpp`, comment out the early `return` statement to enable the application to run the state estimator. Compare with the estimation results obtained from Lab 6.