Degenerate Gaussian factors for probabilistic inference

by

Johannes Cornelius Schoeman

Dissertation presented for the degree of Doctor of Philosophy in Electronic Engineering in the Faculty of Engineering at Stellenbosch University

Supervisors: Dr. Corné E. van Daalen
Prof. Johan A. du Preez

December 2021

Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: December 2021 .....

Abstract

Gaussian random variables and distributions are widely used for inference across many applications, where correlations within probabilistic models can be used to calculate accurate posterior distributions over unobserved variables. In the case of perfect correlation, however, the covariance matrix of a Gaussian distribution is only positive *semi*-definite and therefore singular. This effectively means that linear dependencies exist among the random variables and can either be a direct artefact of the constructed model or the result of machine precision limitations during ill-conditioned numerical calculations. Consequently, traditional Gaussian parametrisations and calculations involving the inverse of the covariance matrix cannot be used in these *degenerate* settings.

In this dissertation, we propose a parametrised factor that enables accurate and automatic inference on Gaussian networks in such degenerate settings at little additional computational cost. In contrast, a common practical solution is to employ ridge regularisation, which trades accuracy for numerical stability through approximations. Other, more principled solutions in turn do not provide all the capabilities of non-degenerate parametrisations. Our factor representation is effectively a generalisation of traditional Gaussian parametrisations where the positive-definite constraint (of the covariance matrix) has been relaxed. This is achieved by representing any possible degeneracies using Dirac delta functions.

To extend the capabilities of Gaussian factors to degenerate settings, we derive various statistical operations and results (such as marginalisation, multiplication and affine transformations of random variables) using our parametrised factors. The computational complexity of these operations is shown to be at most $\mathcal{O}(n^3)$. In addition, we present means for accommodating both linear and nonlinear models as well as for performing Bayesian model comparison. Finally, we apply our methodology to a representative example involving recursive state estimation of cooperative mobile robots. This illustrates the advantages of computing with explicit degenerate Gaussian factors when degeneracies arise inconsistently and unpredictably. Experimental results also reveal that using our factor definition leads to shorter computation times while requiring fewer parameters when compared to existing approaches.

Opsomming

Gaussiese toevalsveranderlikes en verspreidings word algemeen gebruik vir inferensie in verskeie toepassings, waar korrelasies in waarskynlikheidsmodelle gebruik kan word om akkurate posterieuse verspreidings oor onwaargenome veranderlikes te bereken. In die geval van perfekte korrelasie is die kovariansie matriks van 'n Gaussiese verspreiding egter slegs positief *semi*-definiet en dus singulier. Dit beteken effektief dat lineêre afhanklikhede tussen die toevalsveranderlikes bestaan en kan óf 'n direkte gevolg van die opgestelde model wees óf die resultaat van masjien-presisie beperkings tydens swak-gekondisioneerde numeriese bewerkings. Gevolglik kan tradisionele Gaussiese parametriserings en bewerkings wat die inverse van die kovariansie matriks bevat nie gebruik word in hierdie *ontaarde* kontekste nie.

In hierdie proefskrif stel ons 'n geparametriseerde faktor voor wat akkurate en outomatiese inferensie op Gaussiese netwerke in sulke ontaarde kontekste bewerkstellig teen min addisionele berekeningskoste. Daarinteen is 'n algemene praktiese oplossing om rif-regularisering te gebruik, wat akkuraatheid ruil vir numeriese stabiliteit deur middel van benaderings. Ander, meer beginselvaste oplossings verskaf weer nie al die vermoëns van nie-ontaarde parametriserings nie. Ons faktor voorstelling is effektief 'n veralgemening van tradisionele Gaussiese parametriserings waar die positief-definiete beperking (van die kovariansie matriks) verslap word. Dit word reggekry deur Dirac delta funksies te gebruik om enige moontlike ontaardhede voor te stel.

Om die vermoëns van Gaussiese faktore na ontaarde kontekste toe uit te brei lei ons verskeie statistiese operasies en resultate (soos marginalisering, vermenigvuldiging en lineêre transformasies van toevalsveranderlikes) af deur gebruik te maak van ons geparametriseerde faktore. Daar word gewys dat die berekeningskompleksiteit van hierdie operasies $\mathcal{O}(n^3)$ is op die meeste. Verder stel ons maniere voor om beide lineêre en nie-lineêre modelle te akkommodeer sowel as om Bayesiese model vergelyking te doen. Laastens pas ons ons metodologie toe op 'n verteenwoordigende voorbeeld wat rekursiewe toestandsafskatting van samewerkende mobiele robotte behels. Dit illustreer die voordele daarvan om met eksplisiete ontaarde Gaussiese faktore te bereken wanneer ontaardhede onkonsekwent en onvoorspelbaar opduik. Eksperimentele resultate wys ook dat die gebruik van ons faktor definisie na korter berekeningstye lei en minder parameters vereis in vergelyking met bestaande benaderings.

Acknowledgements

I would like to express my sincere gratitude to the following people and organisations:

- The Wilhelm Frank Bursary Fund who funded my studies from 2017 to 2019 and thereby enabled me to complete the research reported in this dissertation.
- The Department of Electrical and Electronic Engineering for appointing me as a lecturer in 2019 while still allowing me to complete my postgraduate studies.
- My two supervisors who provided invaluable insights and guidance during our countless meetings over the past five years.
- My fellow postgraduate students in the Electronic Systems Laboratory (ESL) for creating an enjoyable yet productive work environment.
- My family and friends who have always helped me to maintain a balanced lifestyle with many adventures along the way.
- My wife, Michelle, for her unconditional love, support, patience and kindness.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms and abbreviations

DBN	deep belief network
EM	expectation maximisation
FLOP	floating point operation
GMM	Gaussian mixture model
GP	Gaussian process
HMM	hidden Markov model
KFM	Kalman filter model
KL	Kullback-Leibler
LDPC	low-density parity check
MAP	maximum a posteriori
MAV	micro aerial vehicle
PGM	probabilistic graphical model
SVD	singular value decomposition

Notation

x	scalar
$\|x\|$	absolute value of a scalar
$\mathbf{x}$	vector
$\mathbf{x}_0$	observed value of a random vector
$\mathbb{E}[\mathbf{x}]$	expected value of a random vector
$\mathrm{Cov}[\mathbf{x}]$	covariance of a random vector
X	matrix
X^T	transpose of a matrix
X^{-1}	inverse of a matrix
X^+	pseudo-inverse of a matrix
$\|X\|$	determinant of a matrix
$\mathrm{tr}(X)$	trace of a matrix
$\sqrt{X}$	square root of a matrix
$C(X)$	column space of a matrix
$N(X)$	nullspace of a matrix
$C(X)^\perp$	orthogonal complement of a subspace
$\mathcal{X}$	matrix of sigma points

Chapter 1

Introduction

Using probability theory to solve problems in engineering and computer science often follows the familiar process of modelling, observation and inference. Under this methodology, the starting point is usually to construct a parametrised mathematical *model* of the given system based on equations describing its underlying behaviour. These equations often have some physical origin, for example differential equations describing the dynamics of a robotic vehicle. A subset of the random variables in the model are then *observed* and used to *infer* the posterior distribution over other unobserved (but correlated) random variables. A good example of this is the Kalman filter used in recursive state estimation, where noisy measurements are used to estimate time-dependent latent states [1]. When dealing with continuous random variables such as these, a popular choice for keeping the inference process tractable is to make the Gaussian assumption for all prior and conditional distributions [2]. Under this assumption, the representation is closed under typical statistical operations required for inference algorithms. In other words, the result of each operation is conveniently once again a Gaussian distribution [3].

1.1 Research motivation

An important (and necessary) constraint for using Gaussian models is that the covariance matrix of any Gaussian distribution must be positive definite [4]. If the covariance matrix were instead only positive *semi*-definite, the likelihood of the distribution would not be defined, nor would the precision matrix (i.e., the inverse of the covariance matrix) exist. The latter would be problematic even for basic operations such as multiplication and conditioning on evidence. Such positive semi-definite settings correspond to cases of perfect correlation and are commonly referred to as *degenerate*, where it is understood that linear dependencies exist among certain random variables [5]. This implies that the density in reality only has support (i.e., is not equal to zero) on a lower-dimensional manifold.

Although certain applications necessitate the explicit modelling of linear dependencies between random variables, another common manifestation thereof is the result of machine precision limitations [6]. A simple example of the latter is the repeated noisy observation of a constant latent state. Although the uncertainty in the posterior distribution will theoretically only be zero in the limit, this will happen after finite time in practice. Since such characteristics are typically a function of model parameters, it can easily lead to numerical errors in

certain cases if not handled appropriately.

A straightforward solution to guard against degeneracies is to simply redefine the problem to remove redundant variables as needed. For example, if a vehicle is moving on a constrained horizontal plane, it is unnecessary to estimate its position in the entire three-dimensional space. Similarly, should two random variables become perfectly correlated due to machine precision effects, one could manually reduce the dimensionality of the space until sufficient uncertainty once again allows explicit separation. Although this solution might work, it involves repeated human engineering, which can be especially tedious when dealing with time-dependent models where degeneracies arise inconsistently and unpredictably. An alternative approach that can automatically handle degenerate cases is therefore desirable.

1.2 Aims and objectives

The aim of this project is to develop a generalised representation for Gaussian distributions (or more generally for unnormalised Gaussian factors) that can handle statistical operations in degenerate cases. Ideally, the capabilities of traditional, non-degenerate parametrisations (such as model comparison and linearisation) should still be preserved and the additional computational cost should not be significant. Along with proposing an appropriate definition for such a degenerate Gaussian factor, it is therefore also necessary to derive typical inference operations from first principles. Concretely, we identify the following research objectives:

- To propose a representation for Gaussian factors that can express possible degeneracies without suffering from over-parametrisation.
- To derive the appropriate statistical operations such that the capabilities of non-degenerate factors are preserved.
- To implement the developed solution and showcase its advantages at the hand of a representative example.

If successful, this will ultimately enable numerically stable inference (due to well-conditioned matrices) in degenerate or near-degenerate settings without resorting to unnecessary approximations.

1.3 Existing approaches to inference in degenerate settings

With these objectives in mind, we review the existing literature on degenerate Gaussian factors (or distributions). In the majority of cases, degeneracies are correctly identified, but no subsequent mechanisms are developed which would enable inference. For example, in their work on Gaussian influence diagrams, Shachter and Kenley [2] recognise that zero variance corresponds to linear dependencies, but do not provide a means to handle such cases automatically. Lauritzen and Jensen [5] also acknowledge possible degeneracies in their definition of conditional Gaussian distributions – a hybrid parametrisation where the joint distribution of continuous random variables given discrete random variables is assumed to be multivariate Gaussian. However, since they employ singular covariance matrices for this purpose, representing more general factors (with singular precision matrices) is not possible.

Despite this shortcoming, there are a wide variety of applications where the need for degenerate Gaussian factors is apparent. Two examples include the work by Pawula et al. [7], where they determine formulas for the symbol error rate in digital frequency modulation, and that by Cao and Shen [8] on the fault detection of photovoltaic cells using Gaussian mixture models (GMMs). In both of these cases, deterministic formulas lead to degeneracies when combined with standard probabilistic models. Two further studies by Quinonero-Candela and Rasmussen [9] and Biernacki and Chrétien [10] show that degenerate cases arise even in general regression and estimation techniques such as Gaussian processes (GPs) and expectation maximisation (EM).

To support the methodology in the rest of this dissertation, there are three approaches (for handling degenerate Gaussian factors) that are worth discussing in more detail. The first is a means to avoid the need for computing with degeneracies altogether at the cost of additional approximations. This will additionally serve as a benchmark for evaluating the performance of our proposed solution in Chapter 9. Similar to most approaches, the second does not provide any means for performing inference with degenerate factors. However, the definition employing a Dirac delta function is similar to ours and therefore deserves a mention. The third approach is the most relevant. Although their factor definition is different from ours and consequently does not extend all the capabilities of non-degenerate factors, there are a number of parallels in the mathematical results.

1.3.1 Ridge regularisation

A popular practical solution to deal with degeneracies is to employ ridge regularisation, where a widely-used strategy is to add a small scalar value to the diagonal terms of the covariance matrix. This strategy was first introduced by Hoerl and Kennard [11], where it was used to reduce the mean square error in linear regression. This not only ensures positive definiteness, but also that the condition number of the matrix remains manageable. Specifically, for a singular covariance matrix Σ, Warton [12] writes the regularised covariance as

$$\Sigma_\lambda = \Sigma + \lambda I. \tag{1.1}$$

Although a subset of the eigenvalues of Σ are trivial, the addition of a scaled identity matrix results in an invertible approximation. Warton [12] further states that the *correlation* matrix R is often regularised instead, since it has diagonal elements equal to unity.

Unfortunately, this added robustness often comes at the cost of a decrease in accuracy [13]. This makes sense intuitively, since the effective covariance matrix used for inference is different from the true statistic. As a result, there is always a trade-off between more numerically stable computations for larger values of λ and less biased estimates otherwise. Although a number of studies have investigated optimal strategies for designing these regularisation values [14, 15], whether this approximation is tolerable or not ultimately depends on the particular application [16].

1.3.2 Dirac delta functions and pseudo-inverses

In contrast to ridge regularisation, Mikheev [17] proposes an explicit representation for degenerate Gaussian distributions. For the special case where the mean is zero, they express

the density function over an n-dimensional vector $\mathbf{x}$ as

$$p(\mathbf{x}) = (2\pi)^{-r/2} \left(\prod_{i=1}^{r} \sigma_i \right)^{-1/2} \exp\left(-\frac{1}{2} \mathbf{x}^T \Sigma^+ \mathbf{x} \right) \prod_{i=r+1}^{n} \delta(\mathbf{x}^T \mathbf{v}_i), \tag{1.2}$$

where the covariance matrix Σ has rank r, eigenvalues σ_i, eigenvectors $\mathbf{v}_i$ and pseudo-inverse Σ^+. This is then simplified to an expression for the density function that is only valid on the lower-dimensional manifold where the linear constraints are satisfied, given by

$$p(\mathbf{x}) = \eta \exp\left(-\frac{1}{2} \mathbf{x}^T \Sigma^+ \mathbf{x} \right), \tag{1.3}$$

where η is the normalisation constant. The final result therefore does not include a Dirac delta function and looks very similar to the standard Gaussian density where the matrix inverse is just replaced with a pseudo-inverse. By first making sure that the given vector satisfies the linear constraints imposed by the covariance matrix, Equation 1.3 can then be used to calculate the relative likelihood. Mikheev [17] does not, however, provide any means for performing inference with this representation.

1.3.3 Complementary subspaces and indicator functions

The approach by Raphael [18] is the only one that we are aware of that both represents parametrised degenerate factors explicitly and then also provides a means to use this representation to perform inference. In this case, the factors are represented as the product of an exponential factor and a multivariate indicator function. For this purpose, Raphael defines three mutually orthogonal subspaces that form a decomposition of $\mathbb{R}^n$ according to (a) components associated with finite variance, (b) those that need to satisfy the implied linear constraints, and (c) those that the factor does not depend on. Raphael then derives the necessary operations for inference algorithms and demonstrates their application to an example of automatic musical accompaniment.

In addition to the three subspaces, which are denoted by U^+, U^0 and U^∞ and represented by bases in the columns of matrices, Raphael's parametrisation includes an n-dimensional mean vector $\boldsymbol{\mu}$ and two non-negative definite $n \times n$ matrices Σ and S that are equivalent to the covariance and precision matrices in non-degenerate cases. This results in a hextuple $(U^+, U^0, U^\infty, \Sigma, S, \boldsymbol{\mu})$, although only three of these parameters appear in their factor definition

$$\phi(\mathbf{x}) = \mathbb{1}(P_{U^0}\mathbf{x} = P_{U^0}\boldsymbol{\mu}) \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T S (\mathbf{x} - \boldsymbol{\mu}) \right). \tag{1.4}$$

Firstly, the projection matrix P_{U^0} is used to check whether the given vector $\mathbf{x}$ lies on the lower-dimensional manifold satisfying the linear constraints. The exponential term in turn resembles a non-degenerate Gaussian factor with precision matrix S. The matrix Σ, which does not appear in the definition but is used in their subsequent statistical operations, is the pseudo-inverse of S and the column space of both of these is U^+. Finally, the mean vector $\boldsymbol{\mu}$ is chosen such that $P_{U^\infty}\boldsymbol{\mu} = 0$.

Although the representation in Equation 1.4 makes the notation and results of some statistical operations more concise, it is over-parametrised. More specifically, it should not be

necessary to store multiple parameters that can be obtained from one another by computing, for example, pseudo-inverses or column spaces. In addition, Raphael [18] mentions that this definition cannot express unnormalised factors, which prohibits model comparison. This limitation is ultimately due to their use of an indicator function, where the Dirac delta proposed by Mikheev [17] provides a less restrictive alternative. Finally, they do not provide a method for approximating nonlinear models. This is especially relevant when using the unscented transform, where deterministic samples cannot be drawn naively from a singular covariance matrix.

1.4 Solution overview and contributions

In light of the shortcomings of existing approaches, we present a generalised representation for Gaussian factors in this dissertation that can handle statistical operations in degenerate cases both automatically and accurately. This is in contrast to ridge regularisation [11] which introduces unnecessary approximations. Our parametric form makes use of a canonical factor with a diagonal precision matrix to express the uncertainty in the (possibly) lower-dimensional manifold and a Dirac delta function to represent any linear dependencies. Unlike the work by Mikheev [17], we keep the Dirac delta functions as part of the representation to enable the derivation of inference operations. This is also in contrast to Raphael's [18] use of the indicator function.

Unlike Raphael, we only keep track of two complementary subspaces. This is because canonical factors themselves can already express random variables with zero precision (or infinite variance). The bases for the appropriate decomposition into these two subspaces are then represented by the columns of two semi-orthogonal matrices. By using a *diagonal* precision matrix, our representation achieves the same expressibility as that by Raphael with only $n^2 + 2n$ scalar parameters, as opposed to their $3n^2 + n$ parameters. An added benefit is that deterministic samples can then also be drawn from a singular covariance matrix in a straightforward manner.

Our representation explicitly keeps track of the normalisation constant, thereby allowing model comparison and maximum a posteriori (MAP) estimation. This is not possible in degenerate settings with any existing approaches. We subsequently derive typical inference operations for our representation from first principles that are applicable to both linear and nonlinear models. Lastly, this generalised representation also provides a method to represent the distribution over a rank-deficient transformation of non-degenerate Gaussian random variables.

Collectively, these contributions have the implication that all the capabilities of Gaussian factors can be extended to degenerate settings, without requiring any manual checks nor a different strategy for appropriately handling each case. Since Gaussian random variables are utilised in such a wide variety of applications, this would be very useful in practice. As previously mentioned, traditional inference can also become unstable due to machine precision errors. By choosing an appropriate threshold based on the problem specifics, this can be improved by introducing artificial degeneracies in ill-conditioned cases.

1.5 Dissertation outline

To ensure that our development of degenerate factors is relevant, it is necessary to keep the context general enough. Throughout this dissertation, we therefore present the definition of degenerate Gaussian factors, as well as the resulting statistical operations, in the context of probabilistic graphical models (PGMs). (This does not, however, limit the application of the subsequent development to this context alone.) For this purpose, we introduce various PGM concepts in Chapter 2 and specifically discuss the typical operations required for inference on Gaussian networks. In Chapter 3 we review concepts from the field of linear algebra that are relevant for our definition of degenerate Gaussian factors and used throughout the derivations of the subsequent statistical operations. Utilising concepts such as orthogonality and projections is ultimately what results in a representation that is not over-parametrised. In Chapter 4 we present the definition of both the one-dimensional and multidimensional Dirac delta functions, since the latter subsequently forms part of our factor parametrisation.

In Chapter 5 we introduce our definition of the degenerate Gaussian factor. We also derive the first- and second-order moments of the generalised density function and cover the affine transformation of random variables. In Chapter 6 we derive the statistical operations (such as marginalisation, multiplication and reduction according to evidence) required for inference with degenerate factors and in Chapter 7 we discuss the computational complexity thereof. Chapter 8 presents further operations that are necessary for performing inference on Bayesian networks and Chapter 9 investigates a recursive state estimation example that illustrates the advantages of explicitly representing and performing inference with degenerate Gaussian factors. In Chapter 10 we provide concluding remarks and discuss possible avenues for future research.

Chapter 2

Gaussian networks

Gaussian random variables are widely used across many engineering and computer science applications, where a few examples include robotics, computer vision and natural language processing. We use the term "Gaussian network" to refer to any probabilistic model where the joint distribution over all the random variables is Gaussian. In order to choose an appropriate representation for degenerate Gaussian factors, we first need to know how these factors will typically be used. We therefore start with a review of inference with traditional, non-degenerate Gaussian factors. To keep the context general enough, we make use of probabilistic graphical models (PGMs) for this purpose.

2.1 Probabilistic graphical models

PGMs are a family of statistical techniques that represent the factorisation of a problem using directed or undirected graphs [19]. Since this is a rather general framework, many traditional machine learning techniques and algorithms can be considered special cases of the larger class of PGMs. Such examples include temporal classifiers like hidden Markov models (HMMs) and Kalman filter models (KFMs), error correction techniques such as low-density parity-check (LDPC) coding, certain graph-colouring algorithms and a class of neural network, called deep belief networks (DBNs).

PGMs take advantage of conditional independence between random variables within a model, thereby expressing the joint distribution over all the random variables as a product of factors. This results in a less computationally expensive, factored approach. The use of PGMs for solving inference problems typically amounts to (1) modelling the problem using either a Bayesian network or Markov random field, (2) converting to another, more convenient type of graphical model such as a factor or cluster graph, and (3) inferring the posterior distributions over certain unobserved variables that are of interest. In the remainder of this section we will provide an overview of these concepts at the hand of a simple example. For a more in-depth discussion, kindly refer to the work by Koller and Friedman [3] or Barber [19].

2.1.1 Bayesian networks and Markov random fields

For modelling an inference problem using a PGM, one generally has two options. The subclass of *Bayesian networks* are often used to model systems with clear causal relationships between their random variables, whereas *Markov random fields* (also called *Markov networks*) make use

of a different structure allowing the representation of non-causal relationships. In both cases, random variables are usually represented by round nodes, while the relationships between them are represented by directed or undirected edges in the case of Bayesian or Markov networks, respectively. In a Bayesian network, each of the variable nodes are associated with a conditional distribution and the (absence of) edges indicate conditional (in)dependencies among the random variables [20]. In a Markov random field, the edges represent a more general factorisation of the joint distribution [21]. Although we focus on Bayesian networks to introduce subsequent concepts and terminology in the remainder of this section, the majority of the development applies to Markov random fields as well.

In a Bayesian network, each node with an edge towards another node is called the *parent* of that node, and the other node is in turn called the *child* of the first node. It is possible for any node to have multiple parents or children. A node with no parents is called a *root* node and a node with no children is called a *leaf* node. To represent an inference problem, we require the *conditional* distribution of each variable given all of its parents, or the *prior* distribution in the case of a root node [19]. For a given Bayesian network, the implicit assumption is that the product of all these prior and conditional distributions is equal to the *joint* distribution over all the random variables. We usually indicate random variables that are *observed*, i.e., for which the value will be specified exactly, using shaded nodes. These definitions are illustrated in Figure 2.1 for a simple example.

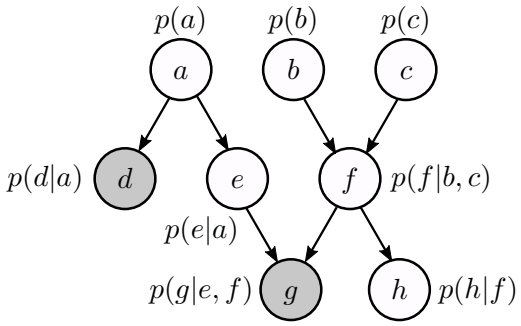


Figure 2.1: Example of a Bayesian network for a system with random variables a to h, where the variables d and g are observed. Nodes a, b and c are root nodes, and nodes d, g and h are leaf nodes. Node f is a child of nodes b and c, and a parent of nodes g and h. The three root nodes have no parents and are therefore associated with prior distributions, whereas nodes d, e and h each have one parent and nodes f and g each have two parents and are therefore associated with the appropriate conditional distributions. The joint distribution is $p(a,b,c,d,e,f,g,h) = p(a)\,p(b)\,p(c)\,p(d|a)\,p(e|a)\,p(f|b,c)\,p(g|e,f)\,p(h|f)$.

2.1.2 Factor and cluster graphs

Although a Bayesian network is useful for capturing the structure and causal relationships of a system, it provides relatively few techniques for performing inference on its own. A popular choice is therefore to convert the Bayesian network to another type of PGM, called a factor graph, which is the foundation for a large number of inference algorithms [3]. In a factor graph, each prior or conditional distribution of the corresponding Bayesian network is represented by a more general *factor*. This factor is a positive, real-valued function of both

the given random variable as well as its parents – collectively called the *scope* of the factor. Each factor is then connected to every random variable within its scope via undirected edges. The factor graph corresponding to the Bayesian network in Figure 2.1 is shown in Figure 2.2.

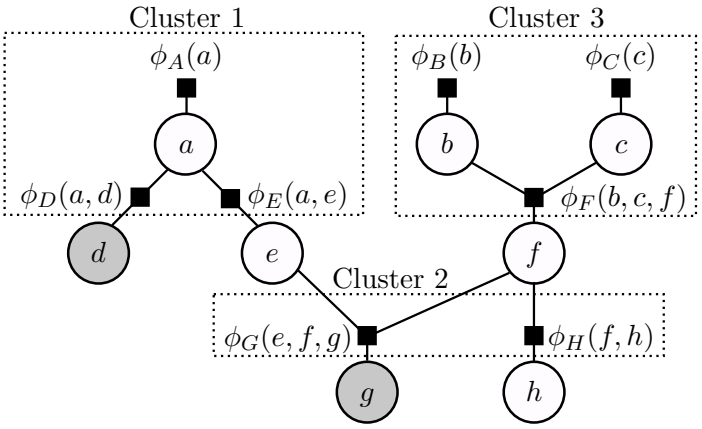


Figure 2.2: The factor graph corresponding to the Bayesian network shown in Figure 2.1. Each factor $\phi(\cdot)$ is indicated by a square node, with undirected edges to each random variable node within its scope. For example, the factor $\phi_G(e, f, g) = p(g|e, f)$ is connected to the random variable nodes e, f and g. The dotted lines indicate a possible choice of grouping factors together to form clusters (discussed next).

By further grouping subsets of these factors together and specifying the connections between these newly formed *clusters*, we can construct another type of PGM called a cluster graph. Although this factor grouping is typically not unique, Koller and Friedman [3] highlight two important constraints for a valid cluster graph:

- The family preservation property: Every factor needs to be assigned to some cluster, such that the scope of the factor is a subset of the scope of the cluster.
- The running intersection property: For every variable and any two clusters that contain that variable as part of their scopes, there should be exactly one path between the two clusters such that the given variable is in the scope of every cluster along that path.

For each cluster, the cluster *potential* is the product of all the factors that are assigned to that cluster and constitutes a node in the cluster graph. The *sepset* between any two clusters is then defined as a subset of the intersection of their potentials' scopes. These sepsets are the foundation for a very popular family of inference algorithms, known as *message passing*. Each sepset is also a node in the cluster graph, with undirected edges to the appropriate cluster potentials. The cluster graph resulting from the factor groupings in Figure 2.2 is shown in Figure 2.3.

2.1.3 Message passing algorithms

Message passing algorithms, such as belief propagation by Shenoy and Shafer [22] or belief update by Lauritzen and Spiegelhalter [23], are a popular choice for performing inference on

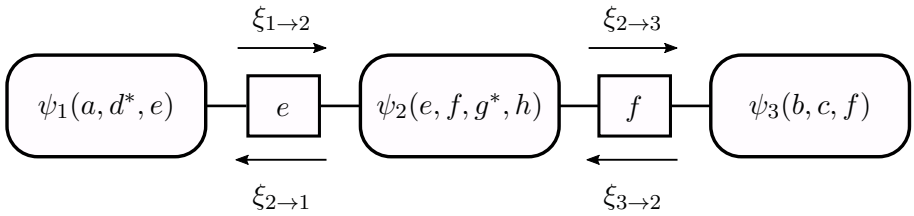


Figure 2.3: The cluster graph created by grouping the factors as shown in Figure 2.2. The cluster potentials and sepsets are represented by oval and square nodes, respectively. As an example, the intersection of the scopes of the two potentials $\psi_1(a, d, e) = \phi_A(a)\,\phi_D(a, d)\,\phi_E(a, e)$ and $\psi_2(e, f, g, h) = \phi_G(e, f, g)\,\phi_H(f, h)$ is the variable e. Observed variables are denoted with an asterisk and the directions of every message ξ (discussed next) is indicated by an arrow.

PGMs. According to these methodologies, the Bayesian or Markov network must first be converted to a factor or cluster graph as discussed previously. All observed variables are then treated as evidence and the posterior distribution, or *belief*, over any unobserved variable can be calculated.

For each sepset in a cluster graph, there exists a pair of *messages* in opposite directions between the two appropriate clusters, where each message is a factor with the sepset as its scope. These messages provide a means for the clusters to share their belief regarding common variables with one another. Koller and Friedman [3] define the messages by performing variable elimination in computing marginal distributions from the joint distribution over all the random variables. According to their sum-product algorithm, the process of computing a given outgoing message from a cluster starts by *multiplying* all the other incoming messages with the cluster potential. We then proceed by incorporating any *evidence* regarding variables within the cluster scope, and finally *marginalise* over all the variables not in the desired sepset. In general, for an outgoing message $\xi_{i\to j}(\mathbf{x}_{i,j})$ from a cluster with potential $\psi_i(\mathbf{x}_i)$, this is given by

$$\xi_{i\to j}(\mathbf{x}_{i,j}) = \int \psi_i(\mathbf{x}_i) \prod_{k\neq j} \xi_{k\to i}(\mathbf{x}_{i,k})\, \mathrm{d}\mathbf{x}_{i,-j}, \tag{2.1}$$

where $\mathbf{x}_{i,-j}$ indicates the variables that are in $\mathbf{x}_i$ but not in $\mathbf{x}_{i,j}$. Returning to the example in Figure 2.3, the four messages are defined as

$$\begin{aligned}
\xi_{1\to2}(e) &= \int \psi_1(a, d = d_0, e)\, da \\
\xi_{2\to1}(e) &= \int \psi_2(e, f, g = g_0, h)\, \xi_{3\to2}(f)\, df\, dh \\
\xi_{2\to3}(f) &= \int \psi_2(e, f, g = g_0, h)\, \xi_{1\to2}(e)\, de\, dh \\
\xi_{3\to2}(f) &= \int \psi_3(b, c, f)\, db\, dc,
\end{aligned} \tag{2.2}$$

where d_0 and g_0 are the observed values of variables d and g, respectively.

An important observation can be made from the message computations in Equation 2.2. Since the two outward messages $\xi_{2\to1}$ and $\xi_{2\to3}$ depend on the two inward messages $\xi_{3\to2}$ and $\xi_{1\to2}$, the latter should be computed first. This reveals that the order of message computations

is significant. Another important consideration when using these algorithms is the structure of the cluster graph. In a tree-structured graph (such as the one in Figure 2.3), inference is exact and messages only need to be computed once. In a graph with loops, however, inference is approximate and requires iteration for convergence [3].

Once all the messages in a cluster graph have been computed, the belief regarding any unobserved variable can be obtained in one of two ways. Firstly, we could multiply any cluster potential where the given variable forms part of that cluster's scope with all the incoming messages to that cluster, and then marginalise over all the other variables. Alternatively, we could combine any pair of messages with the given variable as part of the sepset, and once again marginalise over all the other variables. For the example in Figure 2.3, we can use the former method to calculate the belief over h as

$$p(h|d = d_0, g = g_0) \propto \int \psi_2(e, f, g = g_0, h)\, \xi_{1\to 2}(e)\, \xi_{3\to 2}(f)\, de\, df. \tag{2.3}$$

This then concludes the factored approach of using PGMs to infer the posterior distribution over an unobserved variable given the available evidence.

2.2 Canonical factors

Up to this point, the discussion of inference using PGMs was not specific to Gaussian random variables. Now that the general framework has been presented, we proceed to the specific details for a popular representation of Gaussian factors, called canonical factors. In this section, the definitions and results only hold for the non-degenerate case.

2.2.1 Definition and parametrisation

For representing parametrised factors in Gaussian networks, Koller and Friedman [3] define the canonical factor over an n-dimensional random vector $\mathbf{x}$ as

$$\mathcal{C}\left(\mathbf{x}; K, \mathbf{h}, g\right) \triangleq \exp\left(-\frac{1}{2}\mathbf{x}^T K \mathbf{x} + \mathbf{h}^T \mathbf{x} + g\right), \tag{2.4}$$

where K is a symmetric $n \times n$ precision matrix, $\mathbf{h}$ is an n-dimensional vector and g is a scalar normalisation constant. The factor in Equation 2.4 is a normalised probability density function if and only if the matrix K is positive definite and

$$g = -\frac{1}{2}\mathbf{h}^T K^{-1}\mathbf{h} - \frac{1}{2}\log\left|2\pi K^{-1}\right|. \tag{2.5}$$

Koller and Friedman [3] further show that any Gaussian density function with mean vector $\boldsymbol{\mu}$ and covariance matrix Σ can be expressed as such a canonical factor, according to

$$\begin{aligned}
\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \Sigma) &\triangleq \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \\
&= \exp\left(-\frac{1}{2}\mathbf{x}^T\Sigma^{-1}\mathbf{x} + \boldsymbol{\mu}^T\Sigma^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}^T\Sigma^{-1}\boldsymbol{\mu} - \frac{1}{2}\log|2\pi\Sigma|\right) \\
&= \mathcal{C}\left(\mathbf{x}; \Sigma^{-1}, \Sigma^{-1}\boldsymbol{\mu}, -\frac{1}{2}\boldsymbol{\mu}^T\Sigma^{-1}\boldsymbol{\mu} - \frac{1}{2}\log|2\pi\Sigma|\right).
\end{aligned} \tag{2.6}$$

Conversely, a normalised canonical factor can also be expressed as a Gaussian density with covariance parametrisation according to

$$\mathcal{C}(\mathbf{x}; K, \mathbf{h}, g) = \mathcal{N}(\mathbf{x}; K^{-1}\mathbf{h}, K^{-1}). \tag{2.7}$$

One of the main benefits of canonical factors is their ability to represent conditional Gaussian densities. (Recall that this is necessary for modelling a system using a Bayesian network.) To illustrate this, note that any canonical factor with scope $U\mathbf{x}+\mathbf{v}$ can be expressed over scope $\mathbf{x}$ according to

$$\begin{aligned}
\mathcal{C}(U\mathbf{x}+\mathbf{v}; K, \mathbf{h}, g) &= \exp\left(-\frac{1}{2}(U\mathbf{x}+\mathbf{v})^T K(U\mathbf{x}+\mathbf{v}) + \mathbf{h}^T(U\mathbf{x}+\mathbf{v}) + g\right) \\
&= \exp\left(-\frac{1}{2}\mathbf{x}^T U^T K U\mathbf{x} + (\mathbf{h}-K\mathbf{v})^T U\mathbf{x} + g + \left(\mathbf{h}-\frac{1}{2}K\mathbf{v}\right)^T \mathbf{v}\right) \\
&= \mathcal{C}\left(\mathbf{x}; U^T K U, U^T(\mathbf{h}-K\mathbf{v}), g + \left(\mathbf{h}-\frac{1}{2}K\mathbf{v}\right)^T \mathbf{v}\right).
\end{aligned} \tag{2.8}$$

Next, given the affine transformation

$$\mathbf{y} = A\mathbf{x} + \mathbf{b} + \mathbf{w} \tag{2.9}$$

subject to independent Gaussian noise

$$\mathbf{w} \sim \mathcal{N}(\mathbf{w}; \boldsymbol{\mu}, \Sigma), \tag{2.10}$$

we can represent the conditional density

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}; \boldsymbol{\mu} + A\mathbf{x} + \mathbf{b}, \Sigma) = \mathcal{N}(\mathbf{y} - A\mathbf{x}; \boldsymbol{\mu} + \mathbf{b}, \Sigma) \tag{2.11}$$

by adding the offset $A\mathbf{x} + \mathbf{b}$ to the mean. By then using the result in Equation 2.6 and expressing the scope of the resulting canonical factor as a matrix product, we obtain

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{C}\left(\begin{bmatrix}-A & I\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}; \Sigma^{-1}, \Sigma^{-1}(\boldsymbol{\mu}+\mathbf{b}), -\frac{1}{2}(\boldsymbol{\mu}+\mathbf{b})^T\Sigma^{-1}(\boldsymbol{\mu}+\mathbf{b}) - \frac{1}{2}\log|2\pi\Sigma|\right). \tag{2.12}$$

Finally, applying the result in 2.8 yields

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{C}\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}; K', \mathbf{h}', g'\right), \tag{2.13}$$

where

$$\begin{aligned}
K' &= \begin{bmatrix}-A & I\end{bmatrix}^T \Sigma^{-1} \begin{bmatrix}-A & I\end{bmatrix} \\
\mathbf{h}' &= \begin{bmatrix}-A & I\end{bmatrix}^T \Sigma^{-1}(\boldsymbol{\mu}+\mathbf{b}) \\
g' &= -\frac{1}{2}(\boldsymbol{\mu}+\mathbf{b})^T\Sigma^{-1}(\boldsymbol{\mu}+\mathbf{b}) - \frac{1}{2}\log|2\pi\Sigma|.
\end{aligned} \tag{2.14}$$

Since the precision matrix K' is singular, the conditional density cannot be expressed with scope $\{\mathbf{x}, \mathbf{y}\}$ using the covariance parametrisation.

2.2.2 Statistical operations

A very useful property of canonical factors is that they are closed under typical message passing operations. A closer inspection of the individual computations in Equation 2.2 reveals that message passing requires results for *marginalisation*, *multiplication* and *reduction* according to evidence. For belief propagation [22], also known as the Shenoy-Shafer algorithm, we only require these three operations. For belief update [23], also known as the Lauritzen-Spiegelhalter algorithm, a fourth operation is necessary, namely *division*.

Koller and Friedman [3] show that for the partitioned canonical factor

$$\phi(\mathbf{x},\mathbf{y}) = \mathcal{C}\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}; \begin{bmatrix}K_{\mathbf{xx}} & K_{\mathbf{xy}}\\ K_{\mathbf{yx}} & K_{\mathbf{yy}}\end{bmatrix}, \begin{bmatrix}\mathbf{h}_{\mathbf{x}}\\ \mathbf{h}_{\mathbf{y}}\end{bmatrix}, g\right), \tag{2.15}$$

marginalising over $\mathbf{y}$ results in the marginal canonical factor

$$\int \phi(\mathbf{x},\mathbf{y})\,\mathrm{d}\mathbf{y} = \mathcal{C}\left(\mathbf{x}; K_{\mathbf{xx}} - K_{\mathbf{xy}}K_{\mathbf{yy}}^{-1}K_{\mathbf{yx}}, \mathbf{h}_{\mathbf{x}} - K_{\mathbf{xy}}K_{\mathbf{yy}}^{-1}\mathbf{h}_{\mathbf{y}}, g + \frac{1}{2}\mathbf{h}_{\mathbf{y}}^T K_{\mathbf{yy}}^{-1}\mathbf{h}_{\mathbf{y}} + \frac{1}{2}\log\left|2\pi K_{\mathbf{yy}}^{-1}\right|\right). \tag{2.16}$$

Furthermore, the product of two canonical factors over the same scope $\mathbf{x}$ is given by

$$\mathcal{C}(\mathbf{x}; K_1, \mathbf{h}_1, g_1)\,\mathcal{C}(\mathbf{x}; K_2, \mathbf{h}_2, g_2) = \mathcal{C}(\mathbf{x}; K_1 + K_2, \mathbf{h}_1 + \mathbf{h}_2, g_1 + g_2). \tag{2.17}$$

Similarly, the quotient of two canonical factors is given by

$$\frac{\mathcal{C}(\mathbf{x}; K_1, \mathbf{h}_1, g_1)}{\mathcal{C}(\mathbf{x}; K_2, \mathbf{h}_2, g_2)} = \mathcal{C}(\mathbf{x}; K_1 - K_2, \mathbf{h}_1 - \mathbf{h}_2, g_1 - g_2). \tag{2.18}$$

Finally, the canonical factor in Equation 2.15 can be reduced according to available evidence by setting $\mathbf{y} = \mathbf{y}_0$, resulting in

$$\phi(\mathbf{x},\mathbf{y}_0) = \mathcal{C}(\mathbf{x}; K_{\mathbf{xx}}, \mathbf{h}_{\mathbf{x}} - K_{\mathbf{xy}}\mathbf{y}_0, g + \mathbf{h}_{\mathbf{y}}^T\mathbf{y}_0 - \frac{1}{2}\mathbf{y}_0^T K_{\mathbf{yy}}\mathbf{y}_0). \tag{2.19}$$

The results in Equations 2.16 to 2.19 form the building blocks for message passing algorithms in Gaussian networks. To perform inference using a more general factor representation that can accommodate degeneracies will therefore require similar results.

2.3 Approximation of nonlinear dependencies

Constructing a Bayesian network of a system with continuous random variables usually starts with a given system of equations. These relationships typically have some physical origin (for example differential equations describing the dynamics of a vehicle) and are often nonlinear. Since only *linear* transformations of Gaussian random variables are again Gaussian distributed, the latter warrants the use of linearisation techniques to keep the inference tractable. Consider, for example, the nonlinear transform

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) \tag{2.20}$$

and the prior distribution

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \Sigma). \tag{2.21}$$

Since the exact distribution $p(\mathbf{y})$ will not necessarily be Gaussian, a standard approach is to use the approximation

$$\mathbf{y} \approx A\mathbf{x} + \mathbf{b} \tag{2.22}$$

to obtain the marginal distribution

$$p(\mathbf{y}) \approx \mathcal{N}(\mathbf{y}; A\boldsymbol{\mu} + \mathbf{b}, A\Sigma A^T) = \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}', \Sigma'), \tag{2.23}$$

where the quantities A and $\mathbf{b}$ can be computed in various ways. Two of the most popular linearisation techniques that are widely-used are first-order Taylor series expansion and the unscented transform [24].

2.3.1 Taylor series expansion

For a univariate scalar function $g(x)$, recall the Taylor series

$$g(x) = g(a) + \frac{g'(a)}{1!}(x-a) + \frac{g''(a)}{2!}(x-a)^2 + \frac{g'''(a)}{3!}(x-a)^3 + \dots. \tag{2.24}$$

Such an expansion can be truncated to obtain a first-order approximation of the nonlinear function in Equation 2.20, namely

$$\mathbf{f}(\mathbf{x}) \approx \mathbf{f}(\boldsymbol{\mu}) + J(\boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu}), \tag{2.25}$$

where $J(\mathbf{x})$ is the Jacobian matrix of the function $\mathbf{f}(\mathbf{x})$ [25]. Since only the mean of the prior distribution in Equation 2.21 is used, this is considered as linearising around a single point. Combining Equations 2.20 and 2.25 and rearranging terms yields

$$\mathbf{y} \approx J(\boldsymbol{\mu})\,\mathbf{x} + (\mathbf{f}(\boldsymbol{\mu}) - J(\boldsymbol{\mu})\,\boldsymbol{\mu}). \tag{2.26}$$

Since Equation 2.26 is in the desired form of Equation 2.22, Equation 2.23 becomes

$$p(\mathbf{y}) \approx \mathcal{N}(\mathbf{y}; \mathbf{f}(\boldsymbol{\mu}), J(\boldsymbol{\mu})\Sigma J(\boldsymbol{\mu})^T) \tag{2.27}$$

in the case of the Taylor series approximation. Note, however, that this requires the tedious recalculation of problem-specific partial derivatives. We therefore also consider the unscented transform as an alternative, for which the performance has been proven to be at least as good as that of the Taylor approximation on average [26].

2.3.2 The unscented transform

In the unscented transform, multiple deterministic samples (known as sigma points) are instead drawn from the prior distribution (Equation 2.21) and then individually propagated through the nonlinear transform (Equation 2.20). For an n-dimensional Gaussian prior, Thrun et al. [25] arrange the set of $2n+1$ sigma points as the columns of the matrix

$$\mathcal{X} = \begin{bmatrix}\mathbf{x}^{[0]} & \dots & \mathbf{x}^{[2n]}\end{bmatrix} = \begin{bmatrix}\mathbf{0}, & \gamma\sqrt{\Sigma}, & -\gamma\sqrt{\Sigma}\end{bmatrix} + \boldsymbol{\mu}\mathbf{1}^T, \tag{2.28}$$

where the square root of a matrix $\sqrt{A}$ is typically determined using the Cholesky decomposition. In addition, the scaling parameter γ determines how far the sigma points are distributed

from the mean and consequently what the relative weight of each sample should be. Thrun et al. [25] define two weights

$$w_m^{[i]} = \begin{cases} 1 - \frac{n}{\gamma^2}, & \text{if } i = 0 \\ \frac{1}{2\gamma^2}, & \text{otherwise} \end{cases} \tag{2.29}$$

and

$$w_c^{[i]} = \begin{cases} 4 - \frac{n}{\gamma^2} - \frac{\gamma^2}{n}, & \text{if } i = 0 \\ \frac{1}{2\gamma^2}, & \text{otherwise} \end{cases} \tag{2.30}$$

for this purpose. The i'th sigma point is then propagated through the nonlinear transformation to yield

$$\mathbf{y}^{[i]} = \mathbf{f}\left(\mathbf{x}^{[i]}\right), \tag{2.31}$$

which is in turn used to calculate the resulting mean and covariance (of the distribution in Equation 2.23) according to

$$\boldsymbol{\mu}' = \sum_{i=0}^{2n} w_m^{[i]}\, \mathbf{y}^{[i]} \tag{2.32}$$

and

$$\Sigma' = \sum_{i=0}^{2n} w_c^{[i]} \left(\mathbf{y}^{[i]} - \boldsymbol{\mu}'\right)\left(\mathbf{y}^{[i]} - \boldsymbol{\mu}'\right)^T. \tag{2.33}$$

As we will show in Chapter 8, these results can also be used to calculate an equivalent affine transformation as in Equation 2.22, which can in turn be used to represent conditional Gaussian densities in nonlinear models.

In summary, we provided an overview of Gaussian networks in this chapter and specifically presented the necessary operations for performing inference using canonical factors. The canonical factor will form the first component of our definition for degenerate Gaussian factors in Chapter 5. The subsequent message passing operations outlined in this chapter not only serve as the foundation for Chapter 6, but are themselves used in various proofs of the main results. In addition, the techniques for approximating nonlinear dependencies are extended to the case of degenerate distributions in Chapter 8.

Chapter 3

Linear algebra

Vectors and matrices are prominent in countless branches of engineering and computer science, where a few examples include image processing, Fourier analysis, electrical circuit theory and machine learning. In its purest form, linear algebra is concerned with solving equations of the form $A\mathbf{x} = \mathbf{b}$ for a given matrix A and vector $\mathbf{b}$. In general, it is a proper understanding of matrix properties and the appropriate application of matrix operations that are imperative for successful application. Degenerate Gaussian factors are no exception. Although the definition of such a factor using a Dirac delta component is not unique, choosing matrices with appropriate properties avoids over-parametrisation and leads to simpler statistical results.

3.1 Vector spaces

Before discussing the matrix properties relevant to degenerate factors, an introduction to vector spaces is appropriate. Strang [27] defines a *vector space* as a set V together with rules for vector addition (i.e., $\mathbf{x} + \mathbf{y}$) and multiplication by real numbers (i.e., $c\mathbf{x}$) that satisfy the following eight axioms:

1. Commutativity: $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$
2. Associativity: $\mathbf{x} + (\mathbf{y} + \mathbf{z}) = (\mathbf{x} + \mathbf{y}) + \mathbf{z}$
3. Additive identity: There exists a unique $\mathbf{0}$ such that $\forall \mathbf{x}$, $\mathbf{x} + \mathbf{0} = \mathbf{x}$
4. Additive inverses: $\forall \mathbf{x}$, there exists $-\mathbf{x}$ such that $\mathbf{x} + (-\mathbf{x}) = \mathbf{0}$
5. Unitarity: $1\mathbf{x} = \mathbf{x}$
6. Compatibility: $(cd)\mathbf{x} = c(d\mathbf{x})$
7. Distributivity w.r.t. vectors: $c(\mathbf{x} + \mathbf{y}) = c\mathbf{x} + c\mathbf{y}$
8. Distributivity w.r.t. scalars: $(c + d)\mathbf{x} = c\mathbf{x} + d\mathbf{x}$

Arguably the most well-known example of a vector space is the n-dimensional space $\mathbb{R}^n$, which consists of all column vectors with n components. Although vector spaces are not limited to spaces of column vectors alone, this will be the context throughout this dissertation.

A *subspace* of a vector space is in turn defined as any set of vectors that satisfy two requirements:

- Closed under addition: If $\mathbf{v}$ and $\mathbf{w}$ are vectors that are both in the subspace, then so is their sum $\mathbf{v} + \mathbf{w}$.
- Closed under multiplication: If $\mathbf{v}$ is a vector in the subspace, then so is $c\mathbf{v}$ where c is any scalar.

These two requirements can be combined to reveal that any subspace containing $\mathbf{v}$ and $\mathbf{w}$ must also contain all linear combinations $c\mathbf{v} + d\mathbf{w}$. By setting $c = 0$ in the second, we also see that all subspaces must necessarily contain the zero vector $\mathbf{0}$. An example of a subspace of $\mathbb{R}^2$ is a straight line L that passes through the origin $(0, 0)$, where we write $L \subset \mathbb{R}^2$.

The *intersection* of two subspaces V and W is further defined as the set of all vectors that are elements of both V and W and is denoted by $V \cap W$. In turn, their *union* is the set of all vectors that are elements of either V or W and is denoted by $V \cup W$. Unlike its intersection, the union of two subspaces is not closed under addition (i.e., for two vectors $\mathbf{v} \in V$ and $\mathbf{w} \in W$, the sum $\mathbf{v} + \mathbf{w}$ is not necessarily an element of V or W) and is therefore not a subspace itself. An example of this is the union of the x-axis and the y-axis in $\mathbb{R}^2$.

3.1.1 The four fundamental subspaces of a matrix

A useful perspective on the properties of and the operations on a given matrix $A_{m\times n}$ (with m rows and n columns) is what Strang [27] calls its four *fundamental* subspaces. These are the column space, nullspace, row space and left nullspace. Even if two matrices appear to be different at first glance, if they share certain properties they can be manipulated in similar ways. Furthermore, such matrices can often be used interchangeably to convey equivalent ideas, as we will see in later chapters.

The *column space* (also called the *range*) of A refers to the set of all possible vectors that can be constructed from a linear combination of the columns of A and is denoted by

$$C(A) \subseteq \mathbb{R}^m. \tag{3.1}$$

Since this linear combination can be written as $A\mathbf{x}$, the equation $A\mathbf{x} = \mathbf{b}$ is solvable if and only if $\mathbf{b} \in C(A)$, i.e., if the vector $\mathbf{b}$ is in the column space of A. Since the columns of A have m components, the column space is a subspace of $\mathbb{R}^m$ (and not $\mathbb{R}^n$). The *row space* of A is simply the column space of A^T, i.e., $C\left(A^T\right) \subseteq \mathbb{R}^n$.

The *nullspace* (also called the *kernel*) of A is the set of all solutions $\mathbf{x}$ to the equation $A\mathbf{x} = \mathbf{0}$ and is denoted by

$$N(A) \subseteq \mathbb{R}^n. \tag{3.2}$$

For invertible matrices, the only solution to this equation is $\mathbf{x} = \mathbf{0}$, but in general there could be non-trivial solutions as well. Since the rows of A have n components, so does the vector $\mathbf{x}$ and consequently the nullspace is a subspace of $\mathbb{R}^n$ (and not $\mathbb{R}^m$). The *left nullspace* of A is simply the nullspace of A^T, i.e., $N\left(A^T\right) \subseteq \mathbb{R}^m$.

3.1.2 Matrix rank, independence and dimensionality

Another fundamental property of a matrix is its *rank*, denoted by $r = \text{rank}(A)$ and defined as the number of linearly independent columns of A [27]. A set of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ is *linearly independent* if the only solution to the equation

$$c_1\mathbf{v}_1 + \ldots + c_n\mathbf{v}_n = \mathbf{0} \tag{3.3}$$

is $\forall i, c_i = 0$. Note that any set of n vectors in $\mathbb{R}^m$ is necessarily linearly dependent if $n > m$. The number of independent rows of A is also equal to its rank and therefore

$$0 \leq r \leq \min[m, n]. \tag{3.4}$$

A matrix is said to have *full column rank* when $r = n$ (or full row rank if $r = m$). In this case, the nullspace (or left nullspace) is *trivial*, i.e., $N(A) = \{\mathbf{0}\}$.

A set of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ is further said to *span* the subspace comprising all possible linear combinations $c_1\mathbf{v}_1 + \ldots + c_n\mathbf{v}_n$. The columns of a matrix therefore span its column space, even though these columns do not need to be independent. Similarly, the rows of a matrix span its row space. If a set of vectors that span a given space are in fact independent, this is called a *basis* for that space. According to Strang [27], although the basis for a vector space is not unique, the number of vectors in every such basis must be the same. This is called the *dimension* of the space. From these definitions, the rank of A is equal to the dimension of the column space of A, i.e.,

$$r = \dim(C(A)). \tag{3.5}$$

Finally, the *sum* of two subspaces V and W is defined as the span of the sum of any of their elements and is denoted by $V + W$. For example, the sum of the x-axis and the y-axis is $\mathbb{R}^2$.

3.2 Orthogonality

Up to this point, our discussion has been applicable to general matrices. Although it is well established that two vectors $\mathbf{x}$ and $\mathbf{y}$ are *orthogonal* when $\mathbf{x}^T\mathbf{y} = \mathbf{0}$, this idea can be extended to orthogonal subspaces and orthogonal matrices as well [27].

3.2.1 Orthogonal subspaces and matrices

Two *subspaces* V and W are said to be orthogonal if every vector $\mathbf{v} \in V$ is orthogonal to every vector $\mathbf{w} \in W$. An important special case of this is that for any matrix A, the column space and left nullspace are orthogonal, i.e., $C(A) \perp N(A^T)$. Similarly, its row space and nullspace are orthogonal, i.e., $C(A^T) \perp N(A)$. In fact, these fundamental subspaces are not just orthogonal, but are orthogonal *complements*. Strang [27] defines the orthogonal complement of a subspace V (denoted by $V^\perp$) as the set of *all* vectors orthogonal to V. We can therefore write

$$C(A) = N(A^T)^\perp \qquad \text{and} \qquad C(A^T) = N(A)^\perp. \tag{3.6}$$

The latter follows directly from the definition of the nullspace, since every vector $\mathbf{x}$ that is orthogonal to the rows of A must satisfy $A\mathbf{x} = \mathbf{0}$.

A square *matrix* Q is orthogonal if and only if $Q^{-1} = Q^T$ and subsequently

$$Q^TQ = QQ^T = I. \tag{3.7}$$

If Q is not square, but its n columns are *orthonormal*, then $Q^TQ = I$ and Q is said to be *semi*-orthogonal. A set of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ is orthonormal if

$$\forall i, j, \quad \mathbf{v}_i^T\mathbf{v}_j = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{3.8}$$

where δ_{ij} is called the Kronecker delta.

3.2.2 Projections and least squares approximations

Oftentimes – both in this dissertation but also in general – we need to compute the orthogonal *projection* of a vector $\mathbf{b}$ onto a given subspace V. This projection $\mathbf{p} \in V$ is defined such that the norm of the error

$$\mathbf{e} = \mathbf{b} - \mathbf{p} \tag{3.9}$$

due to the projection is a minimum. If the subspace V is given as the column space of a matrix A, then we can write

$$\mathbf{p} = A\mathbf{x} \tag{3.10}$$

for some value of $\mathbf{x}$. Since the error $\mathbf{e}$ will be orthogonal to $C(A)$, by substituting Equations 3.9 and 3.10, Strang [27] writes

$$A^T\mathbf{e} = A^T(\mathbf{b} - A\mathbf{x}) = \mathbf{0} \qquad \Longrightarrow \qquad A^TA\mathbf{x} = A^T\mathbf{b}. \tag{3.11}$$

Solving for $\mathbf{x}$ and substituting back into Equation 3.10 then yields the solution

$$\mathbf{p} = A\left(A^TA\right)^{-1}A^T\mathbf{b}. \tag{3.12}$$

Note, however, that the matrix A^TA is invertible if and only if A has full column rank [27].

Furthermore, the matrix

$$P = A\left(A^TA\right)^{-1}A^T \tag{3.13}$$

is called the *projection matrix* onto $C(A)$ and the matrix

$$A^+ = (A^TA)^{-1}A^T \tag{3.14}$$

is called the *pseudo-inverse* of the matrix A. In an overdetermined system of equations $A\mathbf{x} = \mathbf{b}$ (where $m > n$), the *least squares* solution

$$\mathbf{x} = (A^TA)^{-1}A^T\mathbf{b} \tag{3.15}$$

makes use of the latter. Also note that for any projection matrix P,

$$P^2 = P = P^T. \tag{3.16}$$

Some special cases are worth mentioning here. Firstly, in the case where the vector $\mathbf{b}$ is already in the column space of the matrix A, we can write $\mathbf{b} = A\mathbf{x}$ and consequently the projection in Equation 3.12 simplifies to

$$\mathbf{p} = A\left(A^TA\right)^{-1}A^TA\mathbf{x} = A\mathbf{x} = \mathbf{b} \tag{3.17}$$

as expected. Secondly, for a semi-orthogonal matrix Q, the projection matrix in Equation 3.13 simplifies to

$$P = Q\left(Q^TQ\right)^{-1}Q^T = QQ^T \tag{3.18}$$

and for an orthogonal matrix to

$$P = I. \tag{3.19}$$

Finally, for two semi-orthogonal matrices Q and R such that $C(Q) = C(R)^\perp$, the matrix $\begin{bmatrix} Q & R \end{bmatrix}$ is orthogonal and therefore

$$\begin{bmatrix} Q & R \end{bmatrix}\begin{bmatrix} Q & R \end{bmatrix}^T = QQ^T + RR^T = I. \tag{3.20}$$

Collectively, all these properties make orthogonal matrices a convenient subclass to work with.

3.3 The singular value decomposition

A popular method for computing orthonormal bases for the fundamental subspaces of a matrix is using the singular value decomposition (SVD) [28]. The SVD of any real matrix

$$A = U\Sigma V^T \tag{3.21}$$

is a factorisation into an $m \times m$ orthogonal matrix U, an $m \times n$ rectangular diagonal matrix Σ and an $n \times n$ orthogonal matrix V. If A is real and symmetric, the SVD is equivalent to the eigendecomposition

$$A = Q\Lambda Q^T. \tag{3.22}$$

The scalar values on the diagonal of Σ are called the *singular values* of A and are usually arranged in decreasing order. For a positive-definite matrix (with only positive singular values), the ratio of its largest to its smallest singular value is known as its *condition number* and is denoted by $\kappa(A)$. On machines with finite precision, computations involving matrices with large condition numbers lead to a decrease in accuracy (due to the loss of significant digits) and results in numeric instability beyond a certain value.

The number of non-trivial singular values of a matrix A is equal to its rank r. Partitioning the columns of U and V and the diagonal of Σ in Equation 3.21 according to r results in the *compact* SVD

$$A = \begin{bmatrix} U_r & U_0 \end{bmatrix} \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_r & V_0 \end{bmatrix}^T = U_r \Sigma_r V_r^T. \tag{3.23}$$

Since the SVD (with complexity $\mathcal{O}(n^3)$) is numerically stable compared to other matrix decompositions and enjoys efficient implementations across many different platforms, Strang [27] shows that it provides a very effective method to determine orthonormal bases for the column space

$$C(A) = C(U_r) \tag{3.24}$$

and the nullspace

$$N(A) = C(V_0). \tag{3.25}$$

In addition, the SVD can be used to compute the pseudo-inverse

$$A^+ = V\Sigma^+ U^T = V\left(\Sigma^T\Sigma\right)^{-1}\Sigma^T U^T \tag{3.26}$$

as well as an orthonormal basis for the orthogonal complement

$$C(A)^\perp = C(U_0). \tag{3.27}$$

The linear algebra ideas presented in this chapter are used throughout the rest of this dissertation – both in our definition of degenerate factors (Chapter 5) as well as in the results and proofs for the statistical operations (Chapters 6 and 8). It is ultimately the convenient properties of orthogonal subspaces and projections that lead to the proposed factor parametrisation, where the SVD is used as a mechanism for the necessary computations. Having introduced this theory, we proceed to the second component of our degenerate factor representation, namely the Dirac delta function.

Chapter 4

Dirac delta functions

To represent degeneracies in a Gaussian factor, we need a multi-dimensional function that can describe linear constraints. The work by Raphael [18] made use of an indicator function for this purpose. However, the simplicity of this choice meant that unnormalised factors could no longer be represented and therefore Bayesian model comparison was no longer possible. A more principled alternative that preserves *all* the capabilities of non-degenerate Gaussian factors, is using Dirac delta functions.

4.1 Definition as the limit of a Gaussian

The Dirac delta function $\delta(x)$ is a distribution (or generalised function) that satisfies the two properties

$$\delta(x) = 0, \text{ for } x \neq 0 \tag{4.1}$$

and

$$\int_{-\infty}^{\infty} \delta(x)\, dx = 1. \tag{4.2}$$

It was originally proposed as part of the early development of quantum mechanics [29], but also plays an integral role in other applications of engineering and physics. In Fourier analysis, it is referred to as the unit impulse [30] and is the inverse Fourier transform of the spectrum $G(f) = 1$, i.e.,

$$\delta(t) = \int_{-\infty}^{\infty} e^{j2\pi ft}\, df. \tag{4.3}$$

In applied mathematics, it is often used to express the force due to an instantaneous exchange of momentum, i.e.,

$$F(t) = P\delta(t), \tag{4.4}$$

where the units of $\delta(t)$ are s^{-1}.

Three further properties of the Dirac delta function follow from those in Equations 4.1 and 4.2. Firstly, *multiplying* a function $f(x)$ that is continuous at $x = a$ with the Dirac delta $\delta(x - a)$ yields

$$f(x)\,\delta(x - a) = f(a)\,\delta(x - a). \tag{4.5}$$

Since $\delta(x-a) = 0$ where $x \neq a$, the product is independent of $f(x)$ where $x \neq a$. Furthermore, the *sampling* (or *sifting*) property follows directly from Equation 4.5 and states that

$$\int_{-\infty}^{\infty} f(x)\,\delta(x-a)\,dx = f(a)\int_{-\infty}^{\infty} \delta(x-a)\,dx = f(a). \tag{4.6}$$

Finally, the *scaling* property states that

$$\delta(kx) = \frac{1}{|k|}\delta(x). \tag{4.7}$$

To see that $|k|\delta(kx) = \delta(x)$, we check that the two properties in Equations 4.1 and 4.2 are satisfied. The first follows directly and for the second we have that

$$\begin{aligned}\int_{-\infty}^{\infty} |k|\delta(kx)\,dx &= \begin{cases} |k|\int_{-\infty}^{\infty}\delta(y)\,\frac{dy}{k} & k>0 \\ |k|\int_{\infty}^{-\infty}\delta(y)\,\frac{dy}{k} & k<0 \end{cases} \\ &= \frac{k}{k}\int_{-\infty}^{\infty}\delta(y)\,dy \\ &= 1. \end{aligned} \tag{4.8}$$

Due to the property in Equation 4.1, another useful interpretation of the Dirac delta is that it imposes a constraint on its argument, i.e., $x = 0$. The Dirac delta can therefore be regarded as the limit of a function that spikes at the origin and is consequently used in physics to model an idealised point mass. According to Shankar [31], a common definition for the one-dimensional Dirac delta that makes use of a Gaussian distribution (with zero mean and variance a) is

$$\delta(x) \triangleq \lim_{a\to 0}\mathcal{N}(x;0,a) = \lim_{a\to 0}\frac{1}{\sqrt{2\pi a}}\exp\left(-\frac{x^2}{2a}\right). \tag{4.9}$$

Since this definition of the Dirac delta as the limit of an exponential term conveniently resembles that of the canonical factor in Equation 2.4, it is the one we will use throughout this dissertation.

4.2 Extension to multiple dimensions

The multidimensional Dirac delta function over a k-dimensional vector $\mathbf{x}$ is defined as the product of the one-dimensional Dirac delta functions over every component x_i, i.e.,

$$\delta(\mathbf{x}) \triangleq \prod_{i=1}^{k}\delta(x_i). \tag{4.10}$$

After combining Equations 4.9 and 4.10, we can write the multidimensional Dirac delta as

$$\delta(\mathbf{x}) = \lim_{a\to 0}\mathcal{N}(\mathbf{x};\mathbf{0},aI) = \lim_{a\to 0}(2\pi a)^{-\frac{k}{2}}\exp\left(-\frac{1}{2a}\mathbf{x}^T\mathbf{x}\right). \tag{4.11}$$

Using the definition in Equation 4.10 we can further use element-wise arguments to show that similar properties to those in Equations 4.5 and 4.6 hold for the multidimensional Dirac delta function, i.e.,

$$f(\mathbf{x})\,\delta(\mathbf{x}-\mathbf{a}) = f(\mathbf{a})\,\delta(\mathbf{x}-\mathbf{a}) \tag{4.12}$$

and

$$\int_{-\infty}^{\infty} f(\mathbf{x})\,\delta(\mathbf{x}-\mathbf{a})\,\mathrm{d}\mathbf{x} = f(\mathbf{a}). \tag{4.13}$$

The more subtle generalisation of the scaling property in Equation 4.7 to multiple dimensions is given by Result 1.

Result 1. *For an $n \times n$ non-singular matrix A, we can write*

$$\delta\left(A\mathbf{x}+\mathbf{b}\right) = \frac{1}{\sqrt{|A|^2}}\delta\left(\mathbf{x}+A^{-1}\mathbf{b}\right). \tag{4.14}$$

If A is orthogonal and $\mathbf{b}=\mathbf{0}$, the result reduces to

$$\delta\left(A\mathbf{x}\right) = \delta\left(\mathbf{x}\right). \tag{4.15}$$

Proof. By using the definition of the multidimensional Dirac delta in Equation 4.11, the SVD $A = U\Sigma V^T$ in Equation 3.21 and the fact that $U^TU = I$, we can write

$$\begin{aligned}\delta\left(A\mathbf{x}\right) &= \lim_{a\to 0}(2\pi a)^{-\frac{n}{2}}\exp\left(-\frac{1}{2a}\mathbf{x}^TA^TA\mathbf{x}\right)\\ &= \lim_{a\to 0}(2\pi a)^{-\frac{n}{2}}\exp\left(-\frac{1}{2a}\mathbf{x}^TV\Sigma^T\Sigma V^T\mathbf{x}\right) = \delta\left(\Sigma V^T\mathbf{x}\right).\end{aligned} \tag{4.16}$$

Since Σ is diagonal, we can use Equation 4.10 to write

$$\delta\left(\Sigma V^T\mathbf{x}\right) = \prod_{i=1}^{n}\delta\left(\sigma_i\mathbf{v}_i^T\mathbf{x}\right), \tag{4.17}$$

where σ_i is the ith singular value of A and $\mathbf{v}_i$ is the ith column of V. Using the scaling property of the Dirac delta function in Equation 4.7, we can combine Equations 4.16 and 4.17 to write

$$\delta\left(A\mathbf{x}\right) = \prod_{i=1}^{n}\delta\left(\sigma_i\mathbf{v}_i^T\mathbf{x}\right) = \prod_{i=1}^{n}\frac{1}{|\sigma_i|}\delta\left(\mathbf{v}_i^T\mathbf{x}\right) = \left(\prod_{i=1}^{n}\frac{1}{|\sigma_i|}\right)\delta\left(V^T\mathbf{x}\right). \tag{4.18}$$

Using the definition in Equation 4.11 again, and since $VV^T = I$ and $|A| = |\Sigma| = \prod\sigma_i$, we can further expand the Dirac delta in Equation 4.18 to write

$$\delta\left(A\mathbf{x}\right) = \left(\prod_{i=1}^{n}\frac{1}{|\sigma_i|}\right)\lim_{a\to 0}(2\pi a)^{-\frac{n}{2}}\exp\left(-\frac{1}{2a}\mathbf{x}^T\mathbf{x}\right) = \frac{1}{\sqrt{|A|^2}}\delta\left(\mathbf{x}\right). \tag{4.19}$$

The more general result for a non-zero offset $\mathbf{b}$ is obtained by substituting $\mathbf{x} = \mathbf{y}+A^{-1}\mathbf{b}$ into Equation 4.19 and if A is orthogonal, $|A| = 1$. This concludes the proof for Result 1. □

In conclusion, the multidimensional Dirac delta function defined in Equation 4.11 can be regarded as the limit of a normalised canonical factor with infinite precision. This will therefore be used in the next chapter to model the degenerate component of our novel factor parametrisation. The three properties in Equations 4.12, 4.13 and 4.14 are then used throughout Chapters 6 and 8 to derive the relevant statistical operations for this factor representation.

Chapter 5

Degenerate Gaussian factors

A popular approach for performing inference with continuous random variables is to employ parametrised factors. In the special case of Gaussian networks, the canonical factor in Equation 2.4 is a good example. In this chapter, we combine a canonical factor with a Dirac delta function to form a novel parametrised factor that can express possible degeneracies. We also discuss the requirements for this factor to be a normalised density function and derive its moments. Since our definition is a generalisation of traditional parametrisations, non-degenerate Gaussian densities can still be represented. Finally, we show that explicitly representing degeneracies within a joint density also enables affine transformations of Gaussian random variables to higher-dimensional spaces.

5.1 Definition and parameterisation

Using the definitions of the canonical factor (Equation 2.4) and the Dirac delta function (Equation 4.11), we represent a degenerate Gaussian factor over an n-dimensional vector $\mathbf{x}$ (with k degrees of degeneracy) as the product

$$\mathcal{D}(\mathbf{x}; Q, R, \Lambda, \mathbf{h}, \mathbf{c}, g) \triangleq \mathcal{C}(Q^T\mathbf{x}; \Lambda, \mathbf{h}, g)\, \delta(R^T\mathbf{x} - \mathbf{c}). \tag{5.1}$$

In this parametrisation, Λ is a nonnegative[1], diagonal precision matrix describing the uncertainty in the $(n-k)$-dimensional affine space where the factor is non-zero. The vector $\mathbf{h}$ determines the location of the peak in this affine space and the normalisation constant g its height. The vector $\mathbf{c}$ in turn describes the minimum offset from the coordinate origin to the affine space. For mathematical convenience in later chapters, the column spaces of the semi-orthogonal matrices $Q_{n\times(n-k)}$ and $R_{n\times k}$ are constrained to be orthogonal complements. A useful interpretation of the degenerate factor in Equation 5.1 is as a lower-dimensional, non-degenerate factor (parametrised by Λ, $\mathbf{h}$ and g) expanded to a higher-dimensional space through an affine transformation (parametrised by Q, R and $\mathbf{c}$). Some of these properties are illustrated in Figure 5.1.

When comparing our representation in Equation 5.1 to that by Mikheev [17] in Equation 1.2, we can highlight a couple of key differences. In addition to the limitation that the

[1]We restrict our parametrisation to positive semi-definite precision matrices. Negative-definite precision matrices that may occur due to approximations (as for example in expectation propagation [32]) could be handled in a similar way as for non-degenerate representations.

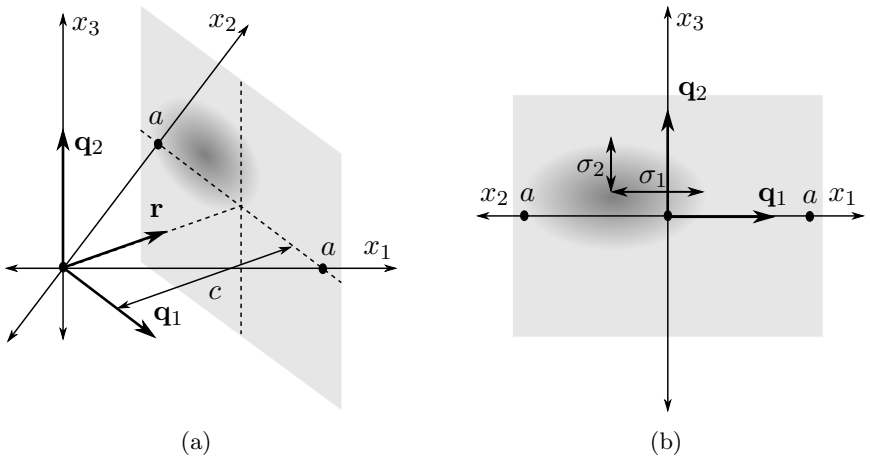


Figure 5.1: (a) A visualisation of a degenerate factor in $\mathbb{R}^3$ subject to the linear constraint $x_1+x_2 = a$. The variation in shading represents the scalar value of the canonical component in the affine space and the dashed lines indicate the principal axes. Away from this plane (along the vector $\mathbf{r}$) the degenerate factor evaluates to zero. The shortest (perpendicular) distance from the origin in $\mathbb{R}^3$ to this plane is c. (b) From a view perpendicular to the affine plane, we see that the vectors $\mathbf{q}_1$ and $\mathbf{q}_2$ have been chosen to align with the principal axes of the lower-dimensional distribution. The corresponding standard deviations are related to the precisions according to $\sigma_1 = 1/\sqrt{\lambda_1}$ and $\sigma_2 = 1/\sqrt{\lambda_2}$.

latter is only defined for normalised factors with zero mean, the use of a covariance matrix has the implication that variables with zero precision cannot be represented. Furthermore, since the eigenvalues and eigenvectors are needed in any case, the quadratic term in the exponent can instead be written in a factored form, which reduces the cost of subsequent computations. This is precisely what we achieve in Equation 5.1 by using a diagonal precision matrix, where the parameters are related according to $\Sigma^+ = Q\Lambda Q^T$. Lastly, our use of the singular value decomposition (SVD) for subsequent computations is more stable compared to the eigendecomposition, especially when identical singular values arise [27].

There are also some key differences to note when comparing our representation to that by Raphael [18] in Equation 1.4. Despite the fact that the latter cannot express any normalisation information, only three of their parameters are actually used in the over-parametrised factor definition. The relationships between the two sets of subspaces are that $C(R) = C(U^0)$ and $C(Q) = C(U^+) + C(U^\infty)$, although Q is chosen specifically to align with the principal axes. This means that the precision matrix can again be factorised as $S = Q\Lambda Q^T$. Lastly, the check whether the given vector $\mathbf{x}$ lies on the lower-dimensional manifold is simplified in our parametrisation. Instead of projecting to the column space of U^0 and comparing to the projection of the mean vector $\boldsymbol{\mu}$, which would be equivalent to the check $RR^T\mathbf{x} = R\mathbf{c}$, it is sufficient to perform a lower-dimensional check for the component perpendicular to the affine space, i.e., $R^T\mathbf{x} = \mathbf{c}$.

5.2 The degenerate density function

For the special case where the factor in Equation 5.1 is a valid density function $p(\mathbf{x})$, we require that

$$\int_{-\infty}^{\infty} p(\mathbf{x})\,\mathrm{d}\mathbf{x} = \int_{-\infty}^{\infty} \mathcal{C}(Q^T\mathbf{x};\Lambda,\mathbf{h},g)\,\delta(R^T\mathbf{x}-\mathbf{c})\,\mathrm{d}\mathbf{x} = 1. \tag{5.2}$$

In addition, we require that the density function is non-negative, but this is already true for the general factor in Equation 5.1. The integral in Equation 5.2 can be rewritten using the substitutions $\boldsymbol{\epsilon} = Q^T\mathbf{x}$ and $\boldsymbol{\eta} = R^T\mathbf{x}$. Since Q and R are orthogonal complements, this change of variables is orthogonal and (since the integral over a Dirac delta is unity) we therefore require that

$$\int_{-\infty}^{\infty} \mathcal{C}(\boldsymbol{\epsilon};\Lambda,\mathbf{h},g)\,\delta(\boldsymbol{\eta}-\mathbf{c})\,\mathrm{d}\boldsymbol{\epsilon}\,\mathrm{d}\boldsymbol{\eta} = \int_{-\infty}^{\infty} \mathcal{C}(\boldsymbol{\epsilon};\Lambda,\mathbf{h},g)\,\mathrm{d}\boldsymbol{\epsilon} = 1. \tag{5.3}$$

Consequently, we can use the result in Equation 2.5 to see that

$$g = -\frac{1}{2}\mathbf{h}^T\Lambda^{-1}\mathbf{h} - \frac{1}{2}\log\left|2\pi\Lambda^{-1}\right|. \tag{5.4}$$

This also requires that Λ is non-singular. Note that, since Λ is diagonal, the matrix inverse in Equation 5.4 is straightforward and can be computed in linear time. This computational advantage is applicable to many operations throughout the rest of this dissertation. A useful perspective on the degenerate Gaussian density function is given by Result 2.

Result 2. *A degenerate Gaussian density can be expressed as the limit of a non-degenerate density according to*

$$p(\mathbf{x}) = \mathcal{D}(\mathbf{x};Q,R,\Lambda,\mathbf{h},\mathbf{c},g) = \lim_{a\to 0}\mathcal{N}\left(\mathbf{x};Q\Lambda^{-1}\mathbf{h}+R\mathbf{c},Q\Lambda^{-1}Q^T+aRR^T\right). \tag{5.5}$$

Proof. By substituting the result in Equation 2.7 as well as the definition of a multidimensional Dirac delta in Equation 4.11 into the definition of a degenerate factor in Equation 5.1, we can write

$$p(\mathbf{x}) = \mathcal{D}(\mathbf{x};Q,R,\Lambda,\mathbf{h},\mathbf{c},g) = \mathcal{N}(Q^T\mathbf{x};\Lambda^{-1}\mathbf{h},\Lambda^{-1})\lim_{a\to 0}\mathcal{N}\left(R^T\mathbf{x}-\mathbf{c};\mathbf{0},aI\right). \tag{5.6}$$

By moving the first term into the limit and expressing the factored product as a joint density, Equation 5.6 becomes

$$p(\mathbf{x}) = \lim_{a\to 0}\mathcal{N}\left(\begin{bmatrix}Q^T\\R^T\end{bmatrix}\mathbf{x};\begin{bmatrix}\Lambda^{-1}\mathbf{h}\\\mathbf{c}\end{bmatrix},\begin{bmatrix}\Lambda^{-1} & 0\\0 & aI\end{bmatrix}\right). \tag{5.7}$$

Since $\mathcal{N}(U^T\mathbf{x};\boldsymbol{\mu},\Sigma) = \mathcal{N}(\mathbf{x};U\boldsymbol{\mu},U\Sigma U^T)$ for an orthogonal transformation U, Equation 5.7 can further be simplified according to

$$\begin{aligned} p(\mathbf{x}) &= \lim_{a\to 0}\mathcal{N}\left(\mathbf{x};\begin{bmatrix}Q & R\end{bmatrix}\begin{bmatrix}\Lambda^{-1}\mathbf{h}\\\mathbf{c}\end{bmatrix},\begin{bmatrix}Q & R\end{bmatrix}\begin{bmatrix}\Lambda^{-1} & 0\\0 & aI\end{bmatrix}\begin{bmatrix}Q & R\end{bmatrix}^T\right)\\ &= \lim_{a\to 0}\mathcal{N}\left(\mathbf{x};Q\Lambda^{-1}\mathbf{h}+R\mathbf{c},Q\Lambda^{-1}Q^T+aRR^T\right). \end{aligned} \tag{5.8}$$

This concludes the proof for Result 2. □

We can determine the moments (specifically the mean and the covariance) of a degenerate Gaussian density directly from Result 2. For the mean, we can use Equation 5.5 to calculate the expected value

$$\mathbb{E}[\mathbf{x}] = \lim_{a\to 0} \left(Q\Lambda^{-1}\mathbf{h} + R\mathbf{c}\right) = Q\Lambda^{-1}\mathbf{h} + R\mathbf{c}. \tag{5.9}$$

For the covariance, we obtain

$$\mathrm{Cov}\,[\mathbf{x}] = \lim_{a\to 0} \left(Q\Lambda^{-1}Q^T + aRR^T\right) = Q\Lambda^{-1}Q^T. \tag{5.10}$$

For k degrees of degeneracy, the covariance will therefore have rank $n-k$, as expected.

Conveniently, any non-degenerate Gaussian density function with mean vector $\boldsymbol{\mu}$ and covariance matrix Σ can still be expressed as a degenerate Gaussian factor such that

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x};\boldsymbol{\mu},\Sigma) = \mathcal{D}(\mathbf{x};Q,R,\Lambda,\mathbf{h},\mathbf{c},g). \tag{5.11}$$

In this non-degenerate case, $C(Q) = \mathbb{R}^n$ and subsequently $C(R) = \{\mathbf{0}\}$. The other extreme where $C(Q) = \{\mathbf{0}\}$ and $C(R) = \mathbb{R}^n$ in turn relates to zero uncertainty and n degrees of degeneracy. Although a subset of the parameters in Equation 5.1 will be empty in both of these cases, the degenerate factor is still well defined under the convention that canonical factors and Dirac delta functions with empty arguments equate to unity.

Returning to the unknown parameters in Equation 5.11, we can use Equation 5.10 to write

$$\Sigma = Q\Lambda^{-1}Q^T. \tag{5.12}$$

Since Σ is symmetric and positive definite, Q and Λ can be calculated directly using the SVD. With the values of Q and Λ in Equation 5.11 known, we proceed to calculate the quantities $\mathbf{h}$ and g. Note that in the non-degenerate case (i.e., when $k = 0$), the factor in Equation 5.1 as well as the expectation in Equation 5.9 no longer depend on the empty arguments R and $\mathbf{c}$. Since Q is orthogonal, the reduced form of the latter yields

$$\mathbf{h} = \left(Q\Lambda^{-1}\right)^{-1}\boldsymbol{\mu} = \Lambda Q^T\boldsymbol{\mu}. \tag{5.13}$$

Finally, substituting Equation 5.13 followed by Equation 5.12 into Equation 5.4 yields

$$g = -\frac{1}{2}\boldsymbol{\mu}^T Q\Lambda Q^T\boldsymbol{\mu} - \frac{1}{2}\log\left|2\pi\Lambda^{-1}\right| = -\frac{1}{2}\boldsymbol{\mu}^T\Sigma^{-1}\boldsymbol{\mu} - \log\sqrt{(2\pi)^n|\Sigma|}, \tag{5.14}$$

which is equivalent to the normalisation constant in traditional parametrisations. Equation 5.11 illustrates the idea that our representation for degenerate Gaussian factors is a generalisation of existing parametrisations that relaxes positive definiteness constraints.

5.3 Affine transformations of degenerate random variables

Among the many useful properties of Gaussian random variables is the fact that an affine transformation

$$\mathbf{y} = A\mathbf{x} + \mathbf{b} \tag{5.15}$$

of a Gaussian random vector $\mathbf{x}$ is still Gaussian-distributed. However, one important condition for the use of traditional parametrisations to express the resulting density function is, once again, that the resulting covariance matrix must be positive definite. Since

$$\mathrm{Cov}\left[\mathbf{y}\right] = A\,\mathrm{Cov}\left[\mathbf{x}\right]A^T, \tag{5.16}$$

this will only be the case when the matrix A has full row rank. By instead using the parametrisation in Equation 5.1 to express the prior density

$$p(\mathbf{x}) = \mathcal{D}(\mathbf{x}; Q, R, \Lambda, \mathbf{h}, \mathbf{c}, g), \tag{5.17}$$

we can relax this constraint and represent the density $p(\mathbf{y})$ for *any* matrix A.

Since an affine transformation only alters the respective subspaces associated with (a) jointly Gaussian-distributed random variables and (b) linear dependencies, $p(\mathbf{y})$ will also be a degenerate Gaussian density with parameters Q', R', Λ', $\mathbf{h}'$, $\mathbf{c}'$ and g'. The procedure to calculate these parameters – derived through moment matching using the results in Equations 5.9 and 5.10 – is summarised in Algorithm 1. The detailed derivation follows after a discussion of the algorithm. In this as well as further algorithms we use the tuple $\theta = (Q, R, \Lambda, \mathbf{h}, \mathbf{c}, g)$ as a shorthand for the parameters.

Algorithm 1 `AffineTransformation`

Input: θ, A, $\mathbf{b}$ such that $p(\mathbf{x}) = \mathcal{D}(\mathbf{x}; \theta)$ and $\mathbf{y} = A\mathbf{x} + \mathbf{b}$
Output: θ' such that $p(\mathbf{y}) = \mathcal{D}(\mathbf{y}; \theta')$

1: $Q', \Sigma', \_ \leftarrow$ `CompactSVD` $\left(AQ\Lambda^{-1}Q^TA^T\right)$
2: $\Lambda' \leftarrow \Sigma'^{-1}$
3: $R' \leftarrow$ `Complement`(Q')
4: $\boldsymbol{\mu}' \leftarrow A(Q\Lambda^{-1}\mathbf{h} + R\mathbf{c}) + \mathbf{b}$
5: $\mathbf{c}' \leftarrow R'^T\boldsymbol{\mu}'$
6: $\mathbf{h}' \leftarrow \Lambda'Q'^T\boldsymbol{\mu}'$
7: $g' \leftarrow -\frac{1}{2}\mathbf{h}'^T\Lambda'^{-1}\mathbf{h}' - \frac{1}{2}\log\left|2\pi\Lambda'^{-1}\right|$

Since the resulting covariance might only be positive semi-definite, the quantities Q' and Λ' are calculated (in lines 1 and 2 of Algorithm 1) using the compact SVD, i.e., by partitioning the SVD according to its non-trivial singular values. Note that the resulting degree of degeneracy in $p(\mathbf{y})$ depends on the mutual properties of A and Q. Next, the semi-orthogonal matrix R' is computed (in line 3) such that $C(R') = C(Q')^\perp$. The mean vector $\boldsymbol{\mu}' = \mathbb{E}[\mathbf{y}]$ is calculated (in line 4) by substituting the expectation in Equation 5.9 into Equation 5.15. As in Equation 5.13, $\mathbf{c}'$ and $\mathbf{h}'$ are then calculated (in lines 5 and 6) by exploiting the mutual orthogonality of R' and Q'. Since the resulting density $p(\mathbf{y})$ will be normalised, g' is calculated (in line 7) by direct application of Equation 5.4.

Proof. To determine the parameters of the degenerate factor representing the density $p(\mathbf{y})$ as calculated according to Algorithm 1, we can use Equations 5.9, 5.10 and 5.15 to calculate the mean

$$\mathbb{E}[\mathbf{y}] = A\,\mathbb{E}[\mathbf{x}] + \mathbf{b} \quad\Longrightarrow\quad Q'\Lambda'^{-1}\mathbf{h}' + R'\mathbf{c}' = A(Q\Lambda^{-1}\mathbf{h} + R\mathbf{c}) + \mathbf{b} \tag{5.18}$$

and covariance

$$\mathrm{Cov}\left[\mathbf{y}\right] = A\,\mathrm{Cov}\left[\mathbf{x}\right]A^T \quad \Longrightarrow \quad Q'\Lambda'^{-1}Q'^T = AQ\Lambda^{-1}Q^TA^T. \tag{5.19}$$

The quantities Q' and Λ' can then be calculated directly from Equation 5.19 using the compact SVD. By definition, $C(R') = C(Q')^{\perp}$, and R' can be determined as in Equation 3.27. With these quantities known, $\mathbf{h}'$ and $\mathbf{c}'$ can be determined from Equation 5.18 by exploiting the mutual orthogonality of Q' and R', where we can write

$$R'^T\left(Q'\Lambda'^{-1}\mathbf{h}' + R'\mathbf{c}'\right) = \mathbf{c}' = R'^T\left(A(Q\Lambda^{-1}\mathbf{h} + R\mathbf{c}) + \mathbf{b}\right) \tag{5.20}$$

and

$$\Lambda'Q'^T\left(Q'\Lambda'^{-1}\mathbf{h}' + R'\mathbf{c}'\right) = \mathbf{h}' = \Lambda'Q'^T\left(A(Q\Lambda^{-1}\mathbf{h} + R\mathbf{c}) + \mathbf{b}\right). \tag{5.21}$$

Finally, since the resulting density $p(\mathbf{y})$ will be normalised, g' is calculated by direct application of Equation 5.4. This concludes the derivation of Algorithm 1. □

In summary, our definition of degenerate Gaussian factors in Equation 5.1 can express $k \in [0, n]$ degeneracies. It comprises a canonical factor and Dirac delta component, as necessary. For the degenerate factor to represent a normalised density function, the canonical factor must be normalised as in Equation 5.4. The mean of a degenerate density in Equation 5.9 is then the sum of two orthogonal vectors in the respective subspaces and the rank of the covariance matrix in Equation 5.10 is equal to $n - k$. Any non-degenerate Gaussian density can be converted to the degenerate parametrisation as in Equation 5.11. Finally, affine transformations of degenerate (or non-degenerate) Gaussian random variables are also distributed according to a degenerate Gaussian density.

Chapter 6

Statistical operations on degenerate factors

To use degenerate Gaussian factors for inference, the appropriate statistical operations need to be derived. In the context of message passing algorithms, these include marginalisation, multiplication, division and reduction according to available evidence. Following the notion that the degenerate factor is a generalisation of the canonical factor, the results in this chapter mirror those in Section 2.2.2. We show a procedure to perform each operation in the form of an algorithm and provide visualisations of simple examples where appropriate. The detailed derivations of the results are also included at the end of each section. As always, it is desirable that the factor representation is closed under these operations, i.e., that the result of each operation is once again a degenerate Gaussian factor parametrised according to Equation 5.1.

6.1 Marginalisation

The first statistical operation that we present is *marginalisation*. In particular, for the partitioned degenerate factor

$$\phi(\mathbf{x}, \mathbf{y}) = \mathcal{D}\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}; \begin{bmatrix}Q_{\mathbf{x}}\\ Q_{\mathbf{y}}\end{bmatrix}, \begin{bmatrix}R_{\mathbf{x}}\\ R_{\mathbf{y}}\end{bmatrix}, \Lambda, \mathbf{h}, \mathbf{c}, g\right), \tag{6.1}$$

marginalising over $\mathbf{y}$ results in the marginal factor

$$\int \phi(\mathbf{x}, \mathbf{y})\, \mathrm{d}\mathbf{y} = \mathcal{D}\left(\mathbf{x}; Q', R', \Lambda', \mathbf{h}', \mathbf{c}', g'\right), \tag{6.2}$$

where the parameters Q', R', Λ', $\mathbf{h}'$, $\mathbf{c}'$ and g' are determined according to Algorithm 2. Since marginalisation amounts to projecting to a subspace, the idea is to use an appropriate orthogonal decomposition (according to the properties of the four block matrices $Q_{\mathbf{x}}$, $Q_{\mathbf{y}}$, $R_{\mathbf{x}}$ and $R_{\mathbf{y}}$) to handle possible degeneracies before marginalising the resulting canonical factor as in Equation 2.16.

The first step (in line 1 of Algorithm 2) is to compute appropriate bases U, V and W according to the block matrices $Q_{\mathbf{x}}$ and $R_{\mathbf{x}}$. As mentioned in Section 3.3, this can be achieved using the singular value decomposition (SVD). Depending on the degree of degeneracy k, the dimension n' of the vector $\mathbf{x}$ and the rank of each respective block matrix, some of these

Algorithm 2 `Marginalise`

Input: θ such that $\phi(\mathbf{x},\mathbf{y}) = \mathcal{D}\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix};\theta\right)$

Output: θ' such that $\int \phi(\mathbf{x},\mathbf{y})\,\mathrm{d}\mathbf{y} = \mathcal{D}(\mathbf{x};\theta')$

1: $U \leftarrow \texttt{Columnspace}\left(Q_{\mathbf{x}}\right)$, $V \leftarrow \texttt{Nullspace}\left(Q_{\mathbf{x}}\right)$, $W \leftarrow \texttt{Columnspace}\left(R_{\mathbf{x}}^T Q_{\mathbf{x}}\right)$

2: $R' \leftarrow \texttt{Complement}(U)$

3: $\mathbf{c}' \leftarrow R'^T R_{\mathbf{x}} \mathbf{c}$

4: $F \leftarrow \left(W\left(R_{\mathbf{y}} W\right)^+ Q_{\mathbf{y}}\right)^T$, $G \leftarrow \left(Q_{\mathbf{x}}^T - F R_{\mathbf{x}}^T\right) U$, $S \leftarrow V\left(V^T \Lambda V\right)^{-1} V^T$

5: $Z, \Lambda', \_ \leftarrow \texttt{SVD}\left(G^T(\Lambda - \Lambda S \Lambda)\, G\right)$

6: $Q' \leftarrow UZ$

7: $\mathbf{h}' \leftarrow Z^T G^T (I - \Lambda S)(\mathbf{h} - \Lambda F \mathbf{c})$

8: $g' \leftarrow g + \left(\mathbf{h} - \frac{1}{2}\Lambda F \mathbf{c}\right)^T F\mathbf{c} + \frac{1}{2}(\mathbf{h} - \Lambda F \mathbf{c})^T S (\mathbf{h} - \Lambda F \mathbf{c}) + \frac{1}{2}\log \frac{\left|2\pi (V^T \Lambda V)^{-1}\right|}{\left|W^T R_{\mathbf{y}}^T R_{\mathbf{y}} W\right|}$

bases could be trivial. The degree of degeneracy k' in the marginal factor is then equal to $n' - \mathrm{rank}(Q_{\mathbf{x}})$ and the basis for this marginal degeneracy in $\mathbb{R}^{n'}$ is $C(U)^{\perp}$ (in line 2). The corresponding offset (in line 3) is $R'^T R_{\mathbf{x}} \mathbf{c}$. The three auxiliary matrices F, G and S (in line 4) make the rest of the notation more concise. The resulting full-rank precision matrix is diagonalised using the SVD (in line 5). The corresponding orthogonal matrix Z is then used (in line 6) to determine the $(n'-k')$-dimensional basis for the marginal support in $\mathbb{R}^{n'}$. Finally, the vector $\mathbf{h}'$ and normalisation constant g' are also calculated (in lines 7 and 8, respectively). Figure 6.1 illustrates various aspects of Algorithm 2 for a simple example.

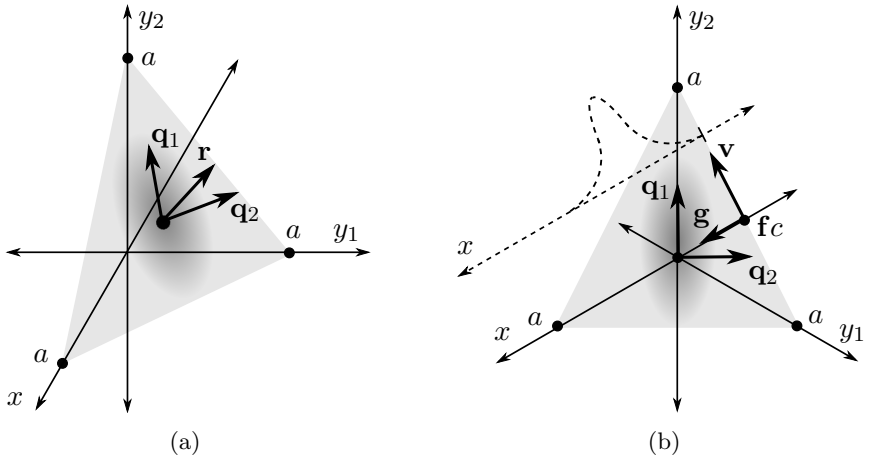


Figure 6.1: The marginalisation over $\{y_1, y_2\}$ of (a) a degenerate factor in $\mathbb{R}^3$ subject to the linear constraint $x + y_1 + y_2 = a$. The vector $\mathbf{r}$ is perpendicular to the affine plane and the two vectors $\mathbf{q}_1$ and $\mathbf{q}_2$ align with the principal axes. (b) From a view perpendicular to this plane, we see the 2-D point $\mathbf{f}c$. Furthermore, the orthogonal 2-D vectors $\mathbf{g}$ and $\mathbf{v}$ align with the x-axis and the edge of the linear constraint in the y_1-y_2 plane, respectively. Collectively, these three components describe an appropriate transformation within the affine plane based on the required marginalisation in the coordinate axes. The marginal factor over x is then indicated by the dashed 1-D curve, where the corresponding origin is the projection of the point $\mathbf{f}c$ (where $x = 0$ in $\mathbb{R}^3$).

Proof. To determine the parameters of the marginal factor $\phi(\mathbf{x})$ as calculated according to Algorithm 2, we need to evaluate the integral

$$\mathcal{J} = \int \mathcal{C}\left(\begin{bmatrix} Q_{\mathbf{x}}^T & Q_{\mathbf{y}}^T \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}; \Lambda, \mathbf{h}, g\right) \delta\left(\begin{bmatrix} R_{\mathbf{x}}^T & R_{\mathbf{y}}^T \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} - \mathbf{c}\right) \mathrm{d}\mathbf{y}. \tag{6.3}$$

Since the matrix $\begin{bmatrix} Q & R \end{bmatrix}$ is orthogonal, some important relationships that will be useful throughout this derivation are

$$\begin{aligned} Q_{\mathbf{x}}^T R_{\mathbf{x}} + Q_{\mathbf{y}}^T R_{\mathbf{y}} = 0 \quad &\Longrightarrow \quad Q_{\mathbf{x}}^T R_{\mathbf{x}} = -Q_{\mathbf{y}}^T R_{\mathbf{y}} \\ Q_{\mathbf{x}}^T Q_{\mathbf{x}} + Q_{\mathbf{y}}^T Q_{\mathbf{y}} = I \quad &\Longrightarrow \quad Q_{\mathbf{x}}^T Q_{\mathbf{x}} = I - Q_{\mathbf{y}}^T Q_{\mathbf{y}} \\ R_{\mathbf{x}}^T R_{\mathbf{x}} + R_{\mathbf{y}}^T R_{\mathbf{y}} = I \quad &\Longrightarrow \quad R_{\mathbf{x}}^T R_{\mathbf{x}} = I - R_{\mathbf{y}}^T R_{\mathbf{y}}. \end{aligned} \tag{6.4}$$

The first step for calculating the integral in Equation 6.3 is to define an appropriate substitution. This is achieved by determining orthonormal bases for the nullspaces of the four block matrices $Q_{\mathbf{x}}$, $Q_{\mathbf{y}}$, $R_{\mathbf{x}}$ and $R_{\mathbf{y}}$ such that

$$C(V_1) = N(Q_{\mathbf{x}}), \quad C(V_2) = N(Q_{\mathbf{y}}), \quad C(V_3) = N(R_{\mathbf{x}}) \quad \text{and} \quad C(V_4) = N(R_{\mathbf{y}}). \tag{6.5}$$

Since the matrix $\begin{bmatrix} Q & R \end{bmatrix}$ has full rank, $N(Q_{\mathbf{x}}) \perp N(Q_{\mathbf{y}})$ and $N(R_{\mathbf{x}}) \perp N(R_{\mathbf{y}})$, although the pairs are not necessarily orthogonal complements. This is because there could for example be a vector $\mathbf{w} \in \mathbb{R}^{n-k}$ such that $\mathbf{w} \notin N(Q_{\mathbf{x}})$ and $\mathbf{w} \notin N(Q_{\mathbf{y}})$. The implication is that the semi-orthogonal matrices $\begin{bmatrix} V_1 & V_2 \end{bmatrix}$ and $\begin{bmatrix} V_3 & V_4 \end{bmatrix}$ are not necessarily square. We therefore need additional bases to guarantee an orthogonal decomposition of $\mathbb{R}^n$. For this purpose, we define two ad-hoc matrices A and B such that $A_{\mathbf{y}} = 0$ and $B_{\mathbf{x}} = 0$. Using the definitions in Equation 6.5, the orthogonal matrix

$$\begin{bmatrix} QV_1 & QV_2 & RV_3 & RV_4 & A & B \end{bmatrix} = \begin{bmatrix} 0 & Q_{\mathbf{x}}V_2 & 0 & R_{\mathbf{x}}V_4 & A_{\mathbf{x}} & 0 \\ Q_{\mathbf{y}}V_1 & 0 & R_{\mathbf{y}}V_3 & 0 & 0 & B_{\mathbf{y}} \end{bmatrix} \tag{6.6}$$

reveals that

$$C(A_{\mathbf{x}}) = \left(C\left(Q_{\mathbf{x}}V_2\right) + C\left(R_{\mathbf{x}}V_4\right)\right)^{\perp} \quad \text{and} \quad C(B_{\mathbf{y}}) = \left(C\left(Q_{\mathbf{y}}V_1\right) + C\left(R_{\mathbf{y}}V_3\right)\right)^{\perp}. \tag{6.7}$$

From Equations 6.4 and 6.5, we further have that $Q_{\mathbf{x}}^T R_{\mathbf{x}} V_4 = 0$ and consequently

$$C(Q_{\mathbf{x}}) = C\left(R_{\mathbf{x}}V_4\right)^{\perp} = C\left(\begin{bmatrix} Q_{\mathbf{x}}V_2 & A_{\mathbf{x}} \end{bmatrix}\right). \tag{6.8}$$

Returning to the integral in Equation 6.3 and using the substitution

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} QV_1 & QV_2 & RV_3 & RV_4 & A & B \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \\ \boldsymbol{\gamma} \\ \boldsymbol{\theta} \\ \boldsymbol{\omega} \\ \boldsymbol{\rho} \end{bmatrix}, \tag{6.9}$$

as well as the fact that $Q^T R = 0$, we can write

$$\begin{bmatrix} Q_{\mathbf{x}}^T & Q_{\mathbf{y}}^T \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} V_1 & V_2 & Q_{\mathbf{x}}^T A_{\mathbf{x}} & Q_{\mathbf{y}}^T B_{\mathbf{y}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \\ \boldsymbol{\omega} \\ \boldsymbol{\rho} \end{bmatrix} \tag{6.10}$$

and

$$\begin{bmatrix} R_{\mathbf{x}}^T & R_{\mathbf{y}}^T \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} V_3 & V_4 & R_{\mathbf{x}}^T A_{\mathbf{x}} & R_{\mathbf{y}}^T B_{\mathbf{y}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\gamma} \\ \boldsymbol{\theta} \\ \boldsymbol{\omega} \\ \boldsymbol{\rho} \end{bmatrix}. \tag{6.11}$$

The next step is to decompose the Dirac delta in Equation 6.3 into three convenient components. This is achieved by multiplying its argument with the orthogonal matrix $\begin{bmatrix} V_3 & V_4 & W \end{bmatrix}^T$, where

$$C(W) = \left(N\left(R_{\mathbf{x}}\right) + N\left(R_{\mathbf{y}}\right)\right)^{\perp} = C\left(R_{\mathbf{x}}^T\right) \cap C\left(R_{\mathbf{y}}^T\right) = C\left(R_{\mathbf{x}}^T Q_{\mathbf{x}}\right) = C\left(R_{\mathbf{y}}^T Q_{\mathbf{y}}\right) \tag{6.12}$$

follows from Equations 6.4 and 6.5. Using the substitution in Equation 6.11 and the result in Equation 4.15, after this multiplication the Dirac delta becomes

$$\delta\left(\begin{bmatrix} R_{\mathbf{x}}^T & R_{\mathbf{y}}^T \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} - \mathbf{c}\right) = \delta\left(\begin{bmatrix} \boldsymbol{\gamma} - V_3^T \mathbf{c} \\ \boldsymbol{\theta} - V_4^T \mathbf{c} \\ W^T R_{\mathbf{x}}^T A_{\mathbf{x}} \boldsymbol{\omega} + W^T R_{\mathbf{y}}^T B_{\mathbf{y}} \boldsymbol{\rho} - W^T \mathbf{c} \end{bmatrix}\right). \tag{6.13}$$

Note that $V_3^T R_{\mathbf{x}}^T A_{\mathbf{x}} = 0$ and $V_4^T R_{\mathbf{y}}^T B_{\mathbf{y}} = 0$ from Equation 6.5 and $V_3^T R_{\mathbf{y}}^T B_{\mathbf{y}} = 0$ and $V_4^T R_{\mathbf{x}}^T A_{\mathbf{x}} = 0$ from Equation 6.7. By also using the substitution in Equation 6.10 and the definition in Equation 4.10, and since $\boldsymbol{\theta}$ is independent of the variables of integration and the integral over a Dirac delta is unity, the integral in Equation 6.3 becomes

$$\mathcal{J} = \delta(\boldsymbol{\theta} - V_4^T \mathbf{c}) \int_{\mathcal{C}} \left(\begin{bmatrix} V_1 & V_2 & Q_{\mathbf{x}}^T A_{\mathbf{x}} & Q_{\mathbf{y}}^T B_{\mathbf{y}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \\ \boldsymbol{\omega} \\ \boldsymbol{\rho} \end{bmatrix}; \Lambda, \mathbf{h}, g \right) \times \delta\left(W^T R_{\mathbf{x}}^T A_{\mathbf{x}} \boldsymbol{\omega} + W^T R_{\mathbf{y}}^T B_{\mathbf{y}} \boldsymbol{\rho} - W^T \mathbf{c}\right) \, \mathrm{d}\boldsymbol{\alpha} \, \mathrm{d}\boldsymbol{\rho}. \tag{6.14}$$

The next step is to use the sifting property in Equation 4.13. For this purpose, we first use Equation 4.14 to express the argument of the Dirac delta inside the integral in terms of $\boldsymbol{\rho}$, i.e.,

$$\frac{1}{\sqrt{\left|W^T R_{\mathbf{y}}^T B_{\mathbf{y}}\right|^2}} \delta\left(\boldsymbol{\rho} + \left(W^T R_{\mathbf{y}}^T B_{\mathbf{y}}\right)^{-1} W^T \left(R_{\mathbf{x}}^T A_{\mathbf{x}} \boldsymbol{\omega} - \mathbf{c}\right)\right). \tag{6.15}$$

By using Equations 6.4, 6.5 and 6.7 once again, we further obtain

$$\begin{aligned} B_{\mathbf{y}} B_{\mathbf{y}}^T R_{\mathbf{y}} W &= \left(I - Q_{\mathbf{y}} V_1 V_1^T Q_{\mathbf{y}}^T - R_{\mathbf{y}} V_3 V_3^T R_{\mathbf{y}}^T\right) R_{\mathbf{y}} W \\ &= R_{\mathbf{y}} W + Q_{\mathbf{y}} V_1 V_1^T Q_{\mathbf{x}}^T R_{\mathbf{x}} W - R_{\mathbf{y}} V_3 V_3^T (I - R_{\mathbf{x}}^T R_{\mathbf{x}}) W = R_{\mathbf{y}} W. \end{aligned} \tag{6.16}$$

Therefore, substituting the expression in Equation 6.15 into the integral in Equation 6.14 and using the sifting property in Equation 4.13 yields

$$\mathcal{J} = \frac{1}{\sqrt{\left|W^T R_{\mathbf{y}}^T R_{\mathbf{y}} W\right|}}\,\delta(\boldsymbol{\theta} - V_4^T\mathbf{c}) \int \mathcal{C}\left(\begin{bmatrix} V_1 & V_2 & M\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\\ \boldsymbol{\omega}\end{bmatrix} + F\mathbf{c}; \Lambda, \mathbf{h}, g\right) \mathrm{d}\boldsymbol{\alpha}, \tag{6.17}$$

where we made use of the matrix definitions

$$F = Q_{\mathbf{y}}^T B_{\mathbf{y}} (W^T R_{\mathbf{y}}^T B_{\mathbf{y}})^{-1} W^T = \left(W(R_{\mathbf{y}} W)^{+} Q_{\mathbf{y}}\right)^T \tag{6.18}$$

and

$$M = Q_{\mathbf{x}}^T A_{\mathbf{x}} - F R_{\mathbf{x}}^T A_{\mathbf{x}} = \left(Q_{\mathbf{x}}^T - F R_{\mathbf{x}}^T\right) A_{\mathbf{x}}. \tag{6.19}$$

Next, the canonical factor inside the integral in Equation 6.17 can be rewritten to have scope $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\omega}$ according to Equation 2.8, namely

$$\mathcal{C}\left(\begin{bmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\\ \boldsymbol{\omega}\end{bmatrix}; \begin{bmatrix}V_1^T\\ V_2^T\\ M^T\end{bmatrix} \Lambda \begin{bmatrix}V_1^T\\ V_2^T\\ M^T\end{bmatrix}^T, \begin{bmatrix}V_1^T\\ V_2^T\\ M^T\end{bmatrix}(\mathbf{h} - \Lambda F\mathbf{c}), g + \left(\mathbf{h} - \frac{1}{2}\Lambda F\mathbf{c}\right)^T F\mathbf{c}\right). \tag{6.20}$$

Using the result for marginalisation of a canonical factor in Equation 2.16, the integral in Equation 6.17 becomes

$$\mathcal{J} = \frac{1}{\sqrt{\left|W^T R_{\mathbf{y}}^T R_{\mathbf{y}} W\right|}}\,\delta(\boldsymbol{\theta} - V_4^T\mathbf{c})\,\mathcal{C}\left(\begin{bmatrix}\boldsymbol{\beta}\\ \boldsymbol{\omega}\end{bmatrix}; \hat{K}, \hat{\mathbf{h}}, \hat{g}\right), \tag{6.21}$$

where

$$\begin{aligned}
\hat{K} &= \begin{bmatrix}V_2 & M\end{bmatrix}^T (\Lambda - \Lambda S \Lambda)\begin{bmatrix}V_2 & M\end{bmatrix}\\
\hat{\mathbf{h}} &= \begin{bmatrix}V_2 & M\end{bmatrix}^T (I - \Lambda S)(\mathbf{h} - \Lambda F\mathbf{c})\\
\hat{g} &= g + \left(\mathbf{h} - \frac{1}{2}\Lambda F\mathbf{c}\right)^T F\mathbf{c} + \frac{1}{2}(\mathbf{h} - \Lambda F\mathbf{c})^T S(\mathbf{h} - \Lambda F\mathbf{c}) + \frac{1}{2}\log\left|2\pi(V_1^T \Lambda V_1)^{-1}\right|
\end{aligned} \tag{6.22}$$

and where we have made use of the matrix definition

$$S = V_1 \left(V_1^T \Lambda V_1\right)^{-1} V_1^T. \tag{6.23}$$

Reversing the change of variables according to Equation 6.9 then yields

$$\mathcal{J} = \frac{1}{\sqrt{\left|W^T R_{\mathbf{y}}^T R_{\mathbf{y}} W\right|}}\,\mathcal{C}\left(\begin{bmatrix}Q_{\mathbf{x}} V_2 & A_{\mathbf{x}}\end{bmatrix}^T \mathbf{x}; \hat{K}, \hat{\mathbf{h}}, \hat{g}\right)\,\delta(V_4^T R_{\mathbf{x}}^T \mathbf{x} - V_4^T \mathbf{c}). \tag{6.24}$$

Although the specific orthonormal bases $\begin{bmatrix}Q_{\mathbf{x}} V_2 & A_{\mathbf{x}}\end{bmatrix}$ and $R_{\mathbf{x}} V_4$ were convenient for deriving this result, recall from Equation 6.8 that the former is also a basis for $C(Q_{\mathbf{x}})$. Consequently, using an arbitrary basis U such that $C(U) = C(Q_{\mathbf{x}})$ results in a simpler computation of

the canonical factor in Equation 6.24 (without requiring the explicit calculation of V_2, V_3, V_4, $A_{\mathbf{x}}$ or $B_{\mathbf{y}}$). However, it is then also necessary to adapt $\hat{K}$ and $\hat{\mathbf{h}}$ to correspond to this new basis. Specifically, we require that

$$U\hat{K}U^T = \begin{bmatrix} Q_{\mathbf{x}}V_2 & A_{\mathbf{x}} \end{bmatrix} \begin{bmatrix} V_2 & M \end{bmatrix}^T (\Lambda - \Lambda S \Lambda) \begin{bmatrix} V_2 & M \end{bmatrix} \begin{bmatrix} Q_{\mathbf{x}}V_2 & A_{\mathbf{x}} \end{bmatrix}^T. \tag{6.25}$$

Expanding the first two factors and using Equations 6.4, 6.5 and 6.7 yields

$$\begin{aligned} \begin{bmatrix} Q_{\mathbf{x}}V_2 & A_{\mathbf{x}} \end{bmatrix} \begin{bmatrix} V_2 & M \end{bmatrix}^T &= Q_{\mathbf{x}}V_2V_2^T + A_{\mathbf{x}}A_{\mathbf{x}}^T(Q_{\mathbf{x}} - R_{\mathbf{x}}F) \\ &= Q_{\mathbf{x}}V_2V_2^T + (I - Q_{\mathbf{x}}V_2V_2^TQ_{\mathbf{x}}^T - R_{\mathbf{x}}V_4V_4^TR_{\mathbf{x}}^T)(Q_{\mathbf{x}} - R_{\mathbf{x}}F) \\ &= Q_{\mathbf{x}}(V_2V_2^T(I - Q_{\mathbf{x}}^TQ_{\mathbf{x}}) + I) - (I - R_{\mathbf{x}}V_4V_4^TR_{\mathbf{x}}^T)R_{\mathbf{x}}F \\ &= Q_{\mathbf{x}}(V_2V_2^TQ_{\mathbf{y}}^TQ_{\mathbf{y}} + I) - UU^TR_{\mathbf{x}}F \\ &= UU^T(Q_{\mathbf{x}} - R_{\mathbf{x}}F) \end{aligned} \tag{6.26}$$

and consequently the precision matrix

$$\hat{K} = U^T(Q_{\mathbf{x}} - R_{\mathbf{x}}F)(\Lambda - \Lambda S\Lambda)(Q_{\mathbf{x}} - R_{\mathbf{x}}F)^TU = Z\hat{\Lambda}Z^T \tag{6.27}$$

can be diagonalised using the SVD. Similarly,

$$U\hat{\mathbf{h}} = \begin{bmatrix} Q_{\mathbf{x}}V_2 & A_{\mathbf{x}} \end{bmatrix} \begin{bmatrix} V_2 & M \end{bmatrix}^T (I - \Lambda S)(\mathbf{h} - \Lambda F\mathbf{c}) \tag{6.28}$$

and therefore

$$\hat{\mathbf{h}} = U^T(Q_{\mathbf{x}} - R_{\mathbf{x}}F)(I - \Lambda S)(\mathbf{h} - \Lambda F\mathbf{c}). \tag{6.29}$$

Finally, for any R' such that $C(R') = C(U)^{\perp} = C(R_{\mathbf{x}}V_4)$, by multiplying its argument with $R'^TR_{\mathbf{x}}V_4$ and using the result in Equation 4.15, we can express the Dirac delta in Equation 6.24 as

$$\delta\left(V_4^TR_{\mathbf{x}}^T\mathbf{x} - V_4^T\mathbf{c}\right) = \delta\left(V_4^TR_{\mathbf{x}}^T\mathbf{x} - V_4^T(R_{\mathbf{x}}^TR_{\mathbf{x}} + R_{\mathbf{y}}^TR_{\mathbf{y}})\,\mathbf{c}\right) = \delta\left(R'^T\mathbf{x} - R'^TR_{\mathbf{x}}\mathbf{c}\right). \tag{6.30}$$

Substituting these new definitions into Equation 6.24 then yields

$$\begin{aligned} \mathcal{J} &= \frac{1}{\sqrt{\left|W^TR_{\mathbf{y}}^TR_{\mathbf{y}}W\right|}}\,\mathcal{C}\left(U^T\mathbf{x}; Z\hat{\Lambda}Z^T, \hat{\mathbf{h}}, \hat{g}\right)\,\delta(R'^T\mathbf{x} - R'^TR_{\mathbf{x}}\mathbf{c}) \\ &= \mathcal{D}\left(\mathbf{x}; UZ, R', \hat{\Lambda}, Z^T\hat{\mathbf{h}}, R'^TR_{\mathbf{x}}\mathbf{c}, \hat{g} - \frac{1}{2}\log\left|W^TR_{\mathbf{y}}^TR_{\mathbf{y}}W\right|\right). \end{aligned} \tag{6.31}$$

Note that since both U and Z are semi-orthogonal, so is their product (as required). This concludes the derivation of Algorithm 2. □

6.2 Multiplication

The next statistical operation that we present is *multiplication*. In particular, the product of two degenerate factors over the same scope $\mathbf{x}$ is given by

$$\mathcal{D}(\mathbf{x}; Q_1, R_1, \Lambda_1, \mathbf{h}_1, \mathbf{c}_1, g_1)\,\mathcal{D}(\mathbf{x}; Q_2, R_2, \Lambda_2, \mathbf{h}_2, \mathbf{c}_2, g_2) = \mathcal{D}(\mathbf{x}; Q', R', \Lambda', \mathbf{h}', \mathbf{c}', g'), \tag{6.32}$$

where the parameters Q', R', Λ', $\mathbf{h}'$, $\mathbf{c}'$ and g' are determined according to Algorithm 3. Since each degenerate factor only has support on a lower-dimensional manifold, the affine space of the product will be the intersection of those of the two original factors. The idea is therefore to reduce each factor to this intersection and then to multiply the resulting canonical factors as in Equation 2.17.

Algorithm 3 `Multiply`

Input: θ_1, θ_2 such that $\phi_1(\mathbf{x}) = \mathcal{D}(\mathbf{x};\theta_1)$ and $\phi_2(\mathbf{x}) = \mathcal{D}(\mathbf{x};\theta_2)$
Output: θ' such that $\phi_1(\mathbf{x}) \times \phi_2(\mathbf{x}) = \mathcal{D}(\mathbf{x};\theta')$
1: $V \leftarrow \texttt{Columnspace}\left(Q_1 Q_1^T R_2\right)$
2: $R' \leftarrow \begin{bmatrix} R_1 & V \end{bmatrix}$
3: $\mathbf{b} \leftarrow (R_2^T V)^{-1}(\mathbf{c}_2 - R_2^T R_1 \mathbf{c}_1)$
4: $\mathbf{c}' \leftarrow \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{b} \end{bmatrix}$
5: $U \leftarrow \texttt{Complement}\left(R'\right)$
6: $Z, \Lambda', \_ \leftarrow \texttt{SVD}\left(U^T(Q_1\Lambda_1 Q_1^T + Q_2\Lambda_2 Q_2^T)\,U\right)$
7: $Q' \leftarrow UZ$
8: $\mathbf{h}' \leftarrow Q'^T(Q_1(\mathbf{h}_1 - \Lambda_1 Q_1^T V\mathbf{b}) + Q_2(\mathbf{h}_2 - \Lambda_2 Q_2^T R'\mathbf{c}'))$
9: $g' \leftarrow g_1 + g_2 + \left(\mathbf{h}_1 - \frac{1}{2}\Lambda_1 Q_1^T V\mathbf{b}\right)^T Q_1^T V\mathbf{b} + \left(\mathbf{h}_2 - \frac{1}{2}\Lambda_2 Q_2^T R'\mathbf{c}'\right)^T Q_2^T R'\mathbf{c}' - \frac{1}{2}\log\left|R_2^T V\right|^2$

The first step (in line 1 of Algorithm 3) is to find an orthonormal basis V for the column space of the projection of R_2 onto $N(R_1^T) = C(Q_1)$. This represents the additional degeneracy due to the second factor and, together with R_1, forms the orthonormal basis (in line 2) for the resulting degenerate space in $\mathbb{R}^n$. The offset $\mathbf{b}$ (in line 3) is determined by the intersection of the affine spaces of the two degenerate factors and this forms the second component of the offset $\mathbf{c}'$ (in line 4). In the edge case where $C(R_1) \cap C(R_2) \neq \{\mathbf{0}\}$, the matrix $R_2^T V$ will not be square and consequently its pseudo-inverse $(V^T R_2 R_2^T V)^{-1} V^T R_2$ should be used instead[1]. The orthonormal basis U (in line 5) is then used to determine the resulting full-rank precision matrix, which is in turn diagonalised using the SVD (in line 6). As was the case in Algorithm 2, the corresponding orthogonal matrix Z is used (in line 7) to determine the basis for the support in $\mathbb{R}^n$. Finally, the vector $\mathbf{h}'$ and normalisation constant g' are also calculated (in lines 8 and 9, respectively). In the case where $R_2^T V$ is not square, the determinant $|V^T R_2 R_2^T V|$ should be used instead of $|R_2^T V|^2$. Figure 6.2 illustrates various aspects of Algorithm 3 for a simple example.

Proof. To determine the parameters of the resulting degenerate factor $\phi(\mathbf{x})$ as calculated according to Algorithm 3, we need to compute the product

$$\mathcal{P} = \mathcal{C}(Q_1^T\mathbf{x};\Lambda_1,\mathbf{h}_1,g_1)\,\delta(R_1^T\mathbf{x} - \mathbf{c}_1)\,\mathcal{C}(Q_2^T\mathbf{x};\Lambda_2,\mathbf{h}_2,g_2)\,\delta(R_2^T\mathbf{x} - \mathbf{c}_2). \tag{6.33}$$

We start by finding an orthonormal basis for the column space of R'. Since the product in Equation 6.33 will evaluate to zero if and only if either of the Dirac deltas are equal to

[1] This will only be necessary when the same degeneracy exists in both factors and simplifies to $\left(R_2^T V\right)^{-1}$ otherwise.

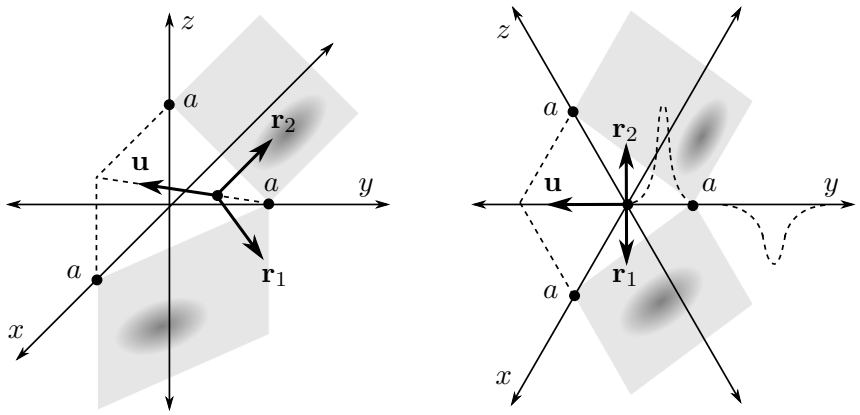


Figure 6.2: The multiplication of (a) two degenerate factors in $\mathbb{R}^3$ subject to the linear constraints $x + y = a$ and $y + z = a$, respectively. The intersection of the two affine planes is the line that goes through the points $(0, a, 0)$ and $(a, 0, a)$ – along the vector $\mathbf{u} = \mathbf{q}'$. The vectors $\mathbf{r}_1$ (which is also perpendicular to the z-axis) and $\mathbf{r}_2$ (which is also perpendicular to the x-axis) are perpendicular to the respective planes. These three vectors are anchored at the point on the intersection with the minimum distance to the origin in $\mathbb{R}^3$. (b) From a view where this anchor point aligns with the origin, we see the true length of the vector $\mathbf{u}$. Each reduced factor along $\mathbf{u}$ is indicated by a dashed 1-D curve, where the product of these is equal to the product of the two degenerate factors along the line of intersection.

zero, we can write that

$$C(R') = C(R_1) + C(R_2). \tag{6.34}$$

In general, this basis is not unique. However, a convenient choice is to have the $n - k_1$ columns of R_1 as part of the basis. This will always be possible since the columns of R_1 are orthonormal per definition and $\text{rank}(R') \geq \text{rank}(R_1)$. The rest of the basis vectors of $C(R')$ can be obtained by projecting the columns of R_2 onto $N(R_1^T) = C(Q_1)$ and computing its column space. Since Q_1 is semi-orthogonal, the necessary projection matrix is $Q_1 Q_1^T$. By defining the orthonormal basis V such that

$$C(V) = C(Q_1 Q_1^T R_2), \tag{6.35}$$

the columns of the matrix $\begin{bmatrix} R_1 & V \end{bmatrix}$ then form an orthonormal basis for $C(R')$. As with the marginalisation operation, we now define a convenient orthogonal change of variables

$$\mathbf{x} = \begin{bmatrix} U & V & R_1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \\ \boldsymbol{\gamma} \end{bmatrix}, \tag{6.36}$$

where

$$C(U) = (C(V) + C(R_1))^{\perp}. \tag{6.37}$$

Also note that $C(Q_1) = (C(U) + C(V))$. Returning to the product in Equation 6.33 and

using the substitution in Equation 6.36 yields

$$\begin{aligned}\mathcal{P} = &\mathcal{C}(Q_1^T(U\boldsymbol{\alpha} + V\boldsymbol{\beta}); \Lambda_1, \mathbf{h}_1, g_1)\, \delta(\boldsymbol{\gamma} - \mathbf{c}_1) \times \\ &\mathcal{C}(Q_2^T(U\boldsymbol{\alpha} + V\boldsymbol{\beta} + R_1\boldsymbol{\gamma}); \Lambda_2, \mathbf{h}_2, g_2)\, \delta\left(R_2^T(V\boldsymbol{\beta} + R_1\boldsymbol{\gamma}) - \mathbf{c}_2\right). \end{aligned} \tag{6.38}$$

Note that $R_2^T U = 0$ because, from Equation 6.34, $C(R_2) \subseteq C(R') = C(U)^{\perp}$. Using the property in Equation 4.12 followed by the result in Equation 4.14, we can express the product of the two Dirac delta functions in Equation 6.38 as

$$\begin{aligned}\delta(\boldsymbol{\gamma} - \mathbf{c}_1)\ \delta\left(R_2^T V\boldsymbol{\beta} + R_2^T R_1\boldsymbol{\gamma} - \mathbf{c}_2\right) &= \delta(\boldsymbol{\gamma} - \mathbf{c}_1)\ \delta\left(R_2^T V\boldsymbol{\beta} + R_2^T R_1\mathbf{c}_1 - \mathbf{c}_2\right) \\ &= \frac{1}{\sqrt{\left|R_2^T V\right|^2}} \delta(\boldsymbol{\gamma} - \mathbf{c}_1)\ \delta(\boldsymbol{\beta} - \mathbf{b}), \end{aligned} \tag{6.39}$$

where

$$\mathbf{b} = \left(R_2^T V\right)^{-1}(\mathbf{c}_2 - R_2^T R_1 \mathbf{c}_1). \tag{6.40}$$

Substituting Equation 6.39 into the product in Equation 6.38 and using the property in Equation 4.12 once again, we can write

$$\mathcal{P} = \frac{1}{\sqrt{\left|R_2^T V\right|^2}} \mathcal{C}(Q_1^T(U\boldsymbol{\alpha}+V\mathbf{b}); \Lambda_1, \mathbf{h}_1, g_1)\, \mathcal{C}(Q_2^T(U\boldsymbol{\alpha}+V\mathbf{b}+R_1\mathbf{c}_1); \Lambda_2, \mathbf{h}_2, g_2)\, \delta\left(\begin{bmatrix}\boldsymbol{\gamma}\\ \boldsymbol{\beta}\end{bmatrix} - \begin{bmatrix}\mathbf{c}_1\\ \mathbf{b}\end{bmatrix}\right). \tag{6.41}$$

Each of the two canonical factors can then be rewritten to have scope $\boldsymbol{\alpha}$ according to Equation 2.8 and can be combined using the result in Equation 2.17, yielding

$$\mathcal{P} = \frac{1}{\sqrt{\left|R_2^T V\right|^2}} \mathcal{C}(\boldsymbol{\alpha}; \hat{K}, \hat{\mathbf{h}}, \hat{g})\, \delta\left(\begin{bmatrix}\boldsymbol{\gamma}\\ \boldsymbol{\beta}\end{bmatrix} - \begin{bmatrix}\mathbf{c}_1\\ \mathbf{b}\end{bmatrix}\right), \tag{6.42}$$

where we have made use of the definitions

$$\begin{aligned}\hat{K} &= U^T(Q_1\Lambda_1 Q_1^T + Q_2\Lambda_2 Q_2^T)\, U \\ \hat{\mathbf{h}} &= U^T(Q_1(\mathbf{h}_1 - \Lambda_1 Q_1^T V\mathbf{b}) + Q_2(\mathbf{h}_2 - \Lambda_2 Q_2^T(V\mathbf{b} + R_1\mathbf{c}_1))) \\ \hat{g} &= g_1 + g_2 + \left(\mathbf{h}_1 - \frac{1}{2}\Lambda_1 Q_1^T V\mathbf{b}\right)^T Q_1^T V\mathbf{b} + \left(\mathbf{h}_2 - \frac{1}{2}\Lambda_2 Q_2^T(V\mathbf{b} + R_1\mathbf{c}_1)\right)^T Q_2^T(V\mathbf{b} + R_1\mathbf{c}_1). \end{aligned} \tag{6.43}$$

The precision matrix in Equation 6.43 can be diagonalised using the SVD

$$\hat{K} = Z\hat{\Lambda}Z^T. \tag{6.44}$$

Reversing the change of variables in Equation 6.36 and substituting Equation 6.44 into

Equation 6.42 yields

$$\begin{aligned}\mathcal{P} &= \frac{1}{\sqrt{\left|R_2^T V\right|^2}}\mathcal{C}(U^T\mathbf{x}; Z\hat{\Lambda}Z^T, \hat{\mathbf{h}}, \hat{g})\,\delta\left(\begin{bmatrix}R_1 & V\end{bmatrix}^T\mathbf{x} - \begin{bmatrix}\mathbf{c}_1\\ \mathbf{b}\end{bmatrix}\right)\\ &= \mathcal{C}\left(Z^TU^T\mathbf{x}; \hat{\Lambda}, Z^T\hat{\mathbf{h}}, \hat{g} - \frac{1}{2}\log\left|R_2^TV\right|^2\right)\,\delta\left(\begin{bmatrix}R_1 & V\end{bmatrix}^T\mathbf{x} - \begin{bmatrix}\mathbf{c}_1\\ \mathbf{b}\end{bmatrix}\right)\\ &= \mathcal{D}\left(\mathbf{x}; UZ, \begin{bmatrix}R_1 & V\end{bmatrix}, \hat{\Lambda}, Z^T\hat{\mathbf{h}}, \begin{bmatrix}\mathbf{c}_1\\ \mathbf{b}\end{bmatrix}, \hat{g} - \frac{1}{2}\log\left|R_2^TV\right|^2\right). \qquad (6.45)\end{aligned}$$

Note that since both U and Z are semi-orthogonal, so is their product (as required). This concludes the derivation of Algorithm 3. □

6.3 Division

The next statistical operation that we present is *division*. In particular, the quotient of two degenerate factors over the same scope $\mathbf{x}$ is given by

$$\frac{\mathcal{D}(\mathbf{x}; Q_1, R_1, \Lambda_1, \mathbf{h}_1, \mathbf{c}_1, g_1)}{\mathcal{D}(\mathbf{x}; Q_2, R_2, \Lambda_2, \mathbf{h}_2, \mathbf{c}_2, g_2)} = \mathcal{D}(\mathbf{x}; Q', R', \Lambda', \mathbf{h}', \mathbf{c}', g'), \qquad (6.46)$$

where the parameters Q', R', Λ', $\mathbf{h}'$, $\mathbf{c}'$ and g' are determined according to Algorithm 4. Due to the usual context of the division operation as part of the belief update algorithm [23] (where the numerator is a product of the denominator and other factors), we assume that $C(R_2) \subseteq C(R_1)$ and that

$$\mathbf{c}_2 = R_2^TR_1\mathbf{c}_1, \qquad (6.47)$$

i.e., that a trivial denominator implies a trivial numerator[2]. By choosing the affine space of the quotient so that it is (a) a subspace of that of the numerator and (b) orthogonal to that of the denominator, we consequently reduce the denominator to the affine space of the numerator before dividing the resulting canonical factors as in Equation 2.18. Except for the different computation of R' (in line 1 of Algorithm 4) and Q' (in line 4), Algorithm 4 follows a similar procedure to the multiplication operation as outlined in Algorithm 3.

Proof. To determine the parameters of the resulting degenerate factor $\phi(\mathbf{x})$ as calculated according to Algorithm 4, we need to compute the quotient

$$\mathcal{Q} = \frac{\mathcal{C}(Q_1^T\mathbf{x}; \Lambda_1, \mathbf{h}_1, g_1)\,\delta(R_1^T\mathbf{x} - \mathbf{c}_1)}{\mathcal{C}(Q_2^T\mathbf{x}; \Lambda_2, \mathbf{h}_2, g_2)\,\delta(R_2^T\mathbf{x} - \mathbf{c}_2)} = \mathcal{D}(\mathbf{x}; Q', R', \Lambda', \mathbf{h}', \mathbf{c}', g'). \qquad (6.48)$$

Since multiplication of Dirac delta functions is better defined than division, a more precise formulation is to calculate the quantities Q', R', Λ', $\mathbf{h}'$, $\mathbf{c}'$ and g' such that

$$\begin{aligned}\mathcal{P} &= \mathcal{C}(Q'^T\mathbf{x}; \Lambda', \mathbf{h}', g')\,\delta(R'^T\mathbf{x} - \mathbf{c}')\,\mathcal{C}(Q_2^T\mathbf{x}; \Lambda_2, \mathbf{h}_2, g_2)\,\delta(R_2^T\mathbf{x} - \mathbf{c}_2)\\ &= \mathcal{C}(Q_1^T\mathbf{x}; \Lambda_1, \mathbf{h}_1, g_1)\,\delta(R_1^T\mathbf{x} - \mathbf{c}_1). \qquad (6.49)\end{aligned}$$

[2] In addition, the results in Algorithm 4 are only valid if Λ' is non-negative as required by our definition of degenerate factors in Equation 5.1.

Algorithm 4 `Divide`

Input: θ_1, θ_2 such that $\phi_1(\mathbf{x}) = \mathcal{D}(\mathbf{x};\theta_1)$ and $\phi_2(\mathbf{x}) = \mathcal{D}(\mathbf{x};\theta_2)$
Output: θ' such that $\phi_1(\mathbf{x})/\phi_2(\mathbf{x}) = \mathcal{D}(\mathbf{x};\theta')$
1: $R' \leftarrow$ `Complement` $\left(\begin{bmatrix} Q_1 & R_2 \end{bmatrix}\right)$
2: $\mathbf{c}' \leftarrow R'^T R_1 \mathbf{c}_1$
3: $Z, \Lambda_+, \_ \leftarrow$ `SVD`$(\Lambda_1 - Q_1^T Q_2 \Lambda_2 Q_2^T Q_1)$
4: $Q' \leftarrow \begin{bmatrix} Q_1 Z & R_2 \end{bmatrix}$
5: $\Lambda' \leftarrow \begin{bmatrix} \Lambda_+ & 0 \\ 0 & 0 \end{bmatrix}$
6: $\mathbf{h}' \leftarrow Q'^T(Q_1\mathbf{h}_1 - Q_2(\mathbf{h}_2 - \Lambda_2 Q_2^T R_1 \mathbf{c}_1))$
7: $g' \leftarrow g_1 - g_2 - \left(\mathbf{h}_2 - \frac{1}{2}\Lambda_2 Q_2^T R_1 \mathbf{c}_1\right)^T Q_2^T R_1 \mathbf{c}_1$

This is similar to the product in Equation 6.33, where the θ'-superscripts and θ_1-subscripts have been reversed. Therefore, just as in Equation 6.34, Equation 6.49 requires that

$$C(R_1) = C(R') + C(R_2). \tag{6.50}$$

This satisfies the assumption in Section 6.3 that $C(R_2) \subseteq C(R_1)$ and consequently

$$Q_1^T R_2 = 0. \tag{6.51}$$

To prove that the proposed parameters

$$\begin{aligned} R' &= \texttt{Complement}\left(\begin{bmatrix} Q_1 & R_2 \end{bmatrix}\right) \\ \mathbf{c}' &= R'^T R_1 \mathbf{c}_1 \\ Q' &= \begin{bmatrix} Q_1 Z & R_2 \end{bmatrix} \\ \Lambda' &= \begin{bmatrix} \Lambda_+ & 0 \\ 0 & 0 \end{bmatrix} \\ \mathbf{h}' &= Q'^T(Q_1\mathbf{h}_1 - Q_2(\mathbf{h}_2 - \Lambda_2 Q_2^T R_1 \mathbf{c}_1)) \\ g' &= g_1 - g_2 - \left(\mathbf{h}_2 - \frac{1}{2}\Lambda_2 Q_2^T R_1 \mathbf{c}_1\right)^T Q_2^T R_1 \mathbf{c}_1, \end{aligned} \tag{6.52}$$

where

$$Z\Lambda_+ Z^T = \Lambda_1 - Q_1^T Q_2 \Lambda_2 Q_2^T Q_1, \tag{6.53}$$

satisfy Equation 6.49, we use the results in Algorithm 3, with the θ_1-subscripts replaced by θ'-superscripts and the latter in turn replaced by θ''-superscripts. From Equation 6.35, substituting Equation 6.51 yields

$$C(V) = C\left(\begin{bmatrix} Q_1 Z & R_2 \end{bmatrix}\begin{bmatrix} Q_1 Z & R_2 \end{bmatrix}^T R_2\right) = C\left((Q_1 Q_1^T + R_2 R_2^T) R_2\right) = C(R_2) \tag{6.54}$$

in this case. From Equation 6.37, we further have that

$$C(U) = \left(C(R_2) + (C(Q_1) + C(R_2))^\perp\right)^\perp = C(Q_1). \tag{6.55}$$

From Equation 6.45, this yields

$$\mathcal{P} = \mathcal{C}(Q''^T\mathbf{x}; \Lambda'', \mathbf{h}'', g'')\,\delta(R''^T\mathbf{x} - \mathbf{c}''), \tag{6.56}$$

where

$$\begin{aligned}
R'' &= \begin{bmatrix} R' & R_2 \end{bmatrix} \\
\mathbf{c}'' &= \begin{bmatrix} \mathbf{c}' \\ \mathbf{c}_2 - R_2^T R'\mathbf{c}' \end{bmatrix} = \begin{bmatrix} R'^T R_1\mathbf{c}_1 \\ R_2^T R_1\mathbf{c}_1 - \mathbf{0} \end{bmatrix} = \begin{bmatrix} R' & R_2 \end{bmatrix}^T R_1\mathbf{c}_1 \\
\Lambda'' &= Q_1^T \left(\begin{bmatrix} Q_1 Z & R_2 \end{bmatrix} \begin{bmatrix} \Lambda_+ & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} Q_1 Z & R_2 \end{bmatrix}^T + Q_2\Lambda_2 Q_2^T \right) Q_1 \\
&= Q_1^T \left(Q_1 \left(\Lambda_1 - Q_1^T Q_2 \Lambda_2 Q_2^T Q_1 \right) Q_1^T + Q_2\Lambda_2 Q_2^T \right) Q_1 = \Lambda_1 \\
Q'' &= Q_1 \\
\mathbf{h}'' &= Q_1^T \left(\begin{bmatrix} Q_1 Z & R_2 \end{bmatrix} \left(\begin{bmatrix} Q_1 Z & R_2 \end{bmatrix}^T (Q_1\mathbf{h}_1 - Q_2(\mathbf{h}_2 - \Lambda_2 Q_2^T R_1\mathbf{c}_1)) - \right. \right. \\
&\qquad \left. \left. \begin{bmatrix} \Lambda_+ & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} Q_1 Z & R_2 \end{bmatrix}^T R_2\mathbf{c}_2 \right) + Q_2(\mathbf{h}_2 - \Lambda_2 Q_2^T R''\mathbf{c}'') \right) = \mathbf{h}_1 \\
g'' &= g_1 - g_2 - \left(\mathbf{h}_2 - \frac{1}{2}\Lambda_2 Q_2^T R_1\mathbf{c}_1 \right)^T Q_2^T R_1\mathbf{c}_1 + g_2 + \\
&\qquad \left(\mathbf{h}' - \frac{1}{2}\Lambda' Q'^T R_2\mathbf{c}_2 \right)^T Q'^T R_2\mathbf{c}_2 + \left(\mathbf{h}_2 - \frac{1}{2}\Lambda_2 Q_2^T R''\mathbf{c}'' \right)^T Q_2^T R''\mathbf{c}'' \\
&= g_1 + \begin{bmatrix} Z^T Q_1^T (Q_1\mathbf{h}_1 - Q_2(\mathbf{h}_2 - \Lambda_2 Q_2^T R_1\mathbf{c}_1)) \\ \mathbf{0} \end{bmatrix}^T \begin{bmatrix} \mathbf{0} \\ \mathbf{c}_2 \end{bmatrix} = g_1.
\end{aligned} \tag{6.57}$$

This reveals that the product of the canonical factors in Equation 6.49 is correct. Since $C(R_1) = C\left(\begin{bmatrix} R' & R_2 \end{bmatrix}\right) = C(Q_1)^\perp$, by multiplying its argument with $R_1^T \begin{bmatrix} R' & R_2 \end{bmatrix}$ and using the result in Equation 4.15, we can further express the Dirac delta in Equation 6.56 as

$$\delta\left(\begin{bmatrix} R' & R_2 \end{bmatrix}^T \mathbf{x} - \begin{bmatrix} R' & R_2 \end{bmatrix}^T R_1\mathbf{c}_1 \right) = \delta\left(R_1^T\mathbf{x} - \mathbf{c}_1 \right). \tag{6.58}$$

This concludes the derivation of Algorithm 4. □

6.4 Reduction

The final statistical operation that we present in this chapter is *reduction*. In particular, the partitioned degenerate factor

$$\phi(\mathbf{x}, \mathbf{y}) = \mathcal{D}\left(\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}; \begin{bmatrix} Q_\mathbf{x} \\ Q_\mathbf{y} \end{bmatrix}, \begin{bmatrix} R_\mathbf{x} \\ R_\mathbf{y} \end{bmatrix}, \Lambda, \mathbf{h}, \mathbf{c}, g \right) \tag{6.59}$$

can be reduced according to available evidence $\mathbf{y}_0$. Setting $\mathbf{y} = \mathbf{y}_0$ in Equation 6.59 results in

$$\phi(\mathbf{x}, \mathbf{y}_0) = \mathcal{D}\left(\mathbf{x}; Q', R', \Lambda', \mathbf{h}', \mathbf{c}', g'\right), \tag{6.60}$$

where the parameters Q', R', Λ', $\mathbf{h}'$, $\mathbf{c}'$ and g' are determined according to Algorithm 5. Since evidence can be regarded as imposing an additional linear constraint in the original space, the

idea is to treat reduction as a special case of multiplication. Consequently, since the support of this product will be independent of the variables in $\mathbf{y}$, the lower-dimensional reduced factor is readily obtained through partitioning.

Algorithm 5 `Reduce`

Input: θ, $\mathbf{y}_0$ such that $\phi(\mathbf{x},\mathbf{y}) = \mathcal{D}\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix};\theta\right)$

Output: θ' such that $\phi(\mathbf{x},\mathbf{y}_0) = \mathcal{D}(\mathbf{x};\theta')$

1: $R' \leftarrow$ `Columnspace` $(R_{\mathbf{x}})$
2: $\mathbf{c}' \leftarrow \left(R_{\mathbf{x}}^T R'\right)^{-1}\left(\mathbf{c} - R_{\mathbf{y}}^T \mathbf{y}_0\right)$
3: $U_{\mathbf{x}} \leftarrow$ `Complement` (R')
4: $Z, \Lambda', \_ \leftarrow$ `SVD`$(U_{\mathbf{x}}^T Q_{\mathbf{x}} \Lambda Q_{\mathbf{x}}^T U_{\mathbf{x}})$
5: $Q' \leftarrow U_{\mathbf{x}} Z$
6: $\mathbf{h}' \leftarrow Q'^T Q_{\mathbf{x}}\left(\mathbf{h} - \Lambda Q^T \begin{bmatrix}R'\mathbf{c}'\\ \mathbf{y}_0\end{bmatrix}\right)$
7: $g' \leftarrow g + \left(\mathbf{h} - \frac{1}{2}\Lambda Q^T \begin{bmatrix}R'\mathbf{c}'\\ \mathbf{y}_0\end{bmatrix}\right)^T Q^T \begin{bmatrix}R'\mathbf{c}'\\ \mathbf{y}_0\end{bmatrix} - \frac{1}{2}\log\left|R_{\mathbf{x}}^T R'\right|^2$

The first step (in line 1 of Algorithm 5) is to find an orthonormal basis for the resulting degenerate space in $\mathbb{R}^{n'}$, where n' is the dimension of the vector $\mathbf{x}$. The corresponding offset $\mathbf{c}'$ (in line 2) is determined by the intersection of the original affine space and the additional constraint $\mathbf{y} = \mathbf{y}_0$. In the edge case where the block matrix $R_{\mathbf{x}}$ does not have full column rank, the matrix $R_{\mathbf{x}}^T R'$ will not be square and consequently its pseudo-inverse $(R'^T R_{\mathbf{x}} R_{\mathbf{x}}^T R')^{-1} R'^T R_{\mathbf{x}}$ should be used instead[3]. The orthonormal basis $U_{\mathbf{x}}$ (in line 3) is then used to determine the resulting full-rank precision matrix, which is in turn diagonalised using the SVD (in line 4). As was the case in Algorithms 2, 3 and 4, the corresponding orthogonal matrix Z is once again used (in line 5) to determine the basis for the support in $\mathbb{R}^{n'}$. Finally, the vector $\mathbf{h}'$ and normalisation constant g' are also calculated (in lines 6 and 7, respectively). In the case where $R_{\mathbf{x}}^T R'$ is not square, the determinant $|R'^T R_{\mathbf{x}} R_{\mathbf{x}}^T R'|$ should be used instead of $|R_{\mathbf{x}}^T R'|^2$. Figure 6.3 illustrates various aspects of Algorithm 5 for a simple example.

Proof. To determine the parameters of the resulting degenerate factor $\phi(\mathbf{x})$ as calculated according to Algorithm 5, we need to substitute the evidence $\mathbf{y}_0$ into Equation 6.59 to obtain $\phi(\mathbf{x},\mathbf{y}_0)$ and then express this factor over scope $\mathbf{x}$. Instead of doing this directly, we start by defining an auxiliary factor with scope $\{\mathbf{x},\mathbf{y}\}$, namely

$$\phi_0(\mathbf{x},\mathbf{y}) = \phi(\mathbf{x},\mathbf{y})\,\delta\left(\mathbf{y} - \mathbf{y}_0\right). \tag{6.61}$$

This multiplication of the joint factor with a Dirac delta can be interpreted as imposing an additional linear constraint due to the evidence $\mathbf{y} = \mathbf{y}_0$. If we further multiply by a carefully-chosen vacuous canonical factor (with $K = 0$, $\mathbf{h} = \mathbf{0}$ and $g = 0$) and rearrange

[3] This will only be necessary when an existing degeneracy is again included in the evidence and simplifies to $\left(R_{\mathbf{x}}^T R'\right)^{-1}$ otherwise.

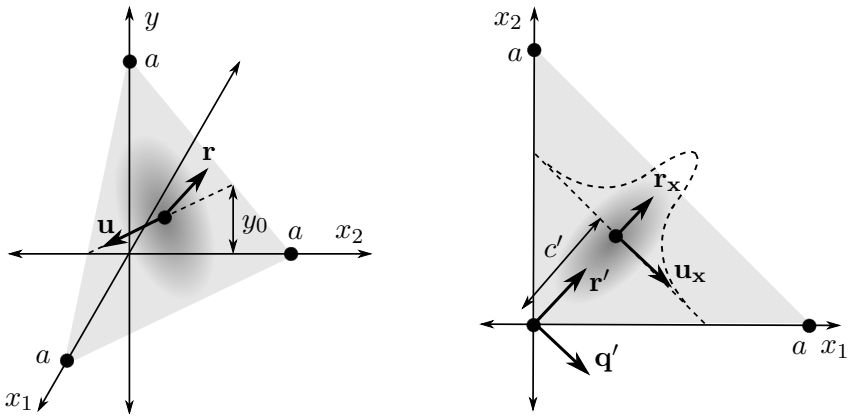


Figure 6.3: The reduction of (a) a degenerate factor in $\mathbb{R}^3$ subject to the linear constraint $x_1 + x_2 + y = a$. The evidence $y = y_0$ can be regarded as a horizontal plane and its intersection with the affine plane is indicated by the dashed line (along the vector $\mathbf{u}$). (b) From a view perpendicular to the x_1-x_2 plane, we see the orthogonal 2-D components $\mathbf{r_x}$ and $\mathbf{u_x} = \mathbf{q}'$ (where the latter is a unit vector). The reduced factor over $\{x_1, x_2\}$ then has a probability distribution along $\mathbf{q}'$ as indicated by the dashed 1-D curve and is equal to zero away from the dashed line (in the direction of $\mathbf{r}'$ with corresponding offset c').

the factors, Equation 6.61 becomes

$$\begin{aligned}\phi_0(\mathbf{x},\mathbf{y}) &= \mathcal{C}\left(\mathbf{x}; \mathit{0}, \mathbf{0}, 0\right)\, \delta\left(\mathbf{y} - \mathbf{y}_0\right)\, \phi(\mathbf{x},\mathbf{y}) \\ &= \mathcal{C}\left(\begin{bmatrix} I & \mathit{0}\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}; \mathit{0}, \mathbf{0}, 0\right)\, \delta\left(\begin{bmatrix}\mathit{0} & I\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} - \mathbf{y}_0\right) \\ &\quad \times \mathcal{C}\left(\begin{bmatrix} Q_{\mathbf{x}}^T & Q_{\mathbf{y}}^T\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}; \Lambda, \mathbf{h}, g\right)\, \delta\left(\begin{bmatrix} R_{\mathbf{x}}^T & R_{\mathbf{y}}^T\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} - \mathbf{c}\right). \end{aligned} \tag{6.62}$$

The motivation for this expansion becomes apparent when comparing Equations 6.33 and 6.62. In particular, note that the evidence operation for degenerate Gaussian factors becomes a special case of the multiplication operation, where $Q_1 = \begin{bmatrix} I & \mathit{0}\end{bmatrix}^T$, $R_1 = \begin{bmatrix}\mathit{0} & I\end{bmatrix}^T$ and $\mathbf{c}_1 = \mathbf{y}_0$. We can therefore use the results from Equation 6.45 to determine the product in Equation 6.62. Returning to the definition of the matrix V in Equation 6.35, however, we see that in this case

$$C(V) = C\left(\begin{bmatrix} I \\ \mathit{0}\end{bmatrix}\begin{bmatrix} I & \mathit{0}\end{bmatrix}\begin{bmatrix} R_{\mathbf{x}} \\ R_{\mathbf{y}}\end{bmatrix}\right) = C\left(\begin{bmatrix} R_{\mathbf{x}} \\ \mathit{0}\end{bmatrix}\right). \tag{6.63}$$

This means that $V_{\mathbf{y}} = \mathit{0}$. Therefore, the expression for the vector $\mathbf{b}$ in Equation 6.40 can be simplified to

$$\mathbf{b} = (R_{\mathbf{x}}^T V_{\mathbf{x}})^{-1}(\mathbf{c} - R_{\mathbf{y}}^T \mathbf{y}_0). \tag{6.64}$$

Furthermore, since $C(U) \perp C\left(\begin{bmatrix} 0 & I \end{bmatrix}^T\right)$, $U_{\mathbf{y}} = 0$. This means that

$$UZ = \begin{bmatrix} U_{\mathbf{x}} \\ 0 \end{bmatrix} Z = \begin{bmatrix} U_{\mathbf{x}} Z \\ 0 \end{bmatrix} \qquad \text{and} \qquad U^T Q = \begin{bmatrix} U_{\mathbf{x}} \\ 0 \end{bmatrix}^T \begin{bmatrix} Q_{\mathbf{x}} \\ Q_{\mathbf{y}} \end{bmatrix} = U_{\mathbf{x}}^T Q_{\mathbf{x}}. \tag{6.65}$$

Using Equation 6.45 (where $\Lambda_1 = 0$, $\mathbf{h}_1 = \mathbf{0}$ and $g_1 = 0$) and substituting Equations 6.63 to 6.65, Equation 6.62 becomes

$$\phi_0(\mathbf{x}, \mathbf{y}) = \mathcal{D}\left(\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}; \begin{bmatrix} U_{\mathbf{x}} Z \\ 0 \end{bmatrix}, \begin{bmatrix} 0 & V_{\mathbf{x}} \\ I & 0 \end{bmatrix}, \hat{\Lambda}, \hat{\mathbf{h}}, \begin{bmatrix} \mathbf{y}_0 \\ \mathbf{b} \end{bmatrix}, \hat{g}\right), \tag{6.66}$$

where

$$\begin{aligned} Z\hat{\Lambda}Z^T &= U_{\mathbf{x}}^T Q_{\mathbf{x}} \Lambda Q_{\mathbf{x}}^T U_{\mathbf{x}} \\ \hat{\mathbf{h}} &= Z^T U_{\mathbf{x}}^T Q_{\mathbf{x}} \left(\mathbf{h} - \Lambda Q^T \begin{bmatrix} V_{\mathbf{x}} \mathbf{b} \\ \mathbf{y}_0 \end{bmatrix}\right) \\ \hat{g} &= g + \left(\mathbf{h} - \frac{1}{2}\Lambda Q^T \begin{bmatrix} V_{\mathbf{x}} \mathbf{b} \\ \mathbf{y}_0 \end{bmatrix}\right)^T Q^T \begin{bmatrix} V_{\mathbf{x}} \mathbf{b} \\ \mathbf{y}_0 \end{bmatrix} - \frac{1}{2} \log \left| R_{\mathbf{x}}^T V_{\mathbf{x}} \right|^2. \end{aligned} \tag{6.67}$$

Using the definition of the degenerate factor in Equation 5.1 and that of the multidimensional Dirac delta function in Equation 4.10, Equation 6.66 then becomes

$$\begin{aligned} \phi_0(\mathbf{x}, \mathbf{y}) &= \mathcal{C}\left(\begin{bmatrix} Z^T U_{\mathbf{x}}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}; \hat{\Lambda}, \hat{\mathbf{h}}, \hat{g}\right) \delta\left(\begin{bmatrix} 0 & I \\ V_{\mathbf{x}}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} - \begin{bmatrix} \mathbf{y}_0 \\ \mathbf{b} \end{bmatrix}\right) \\ &= \mathcal{C}\left(Z^T U_{\mathbf{x}}^T \mathbf{x}; \hat{\Lambda}, \hat{\mathbf{h}}, \hat{g}\right) \delta\left(V_{\mathbf{x}}^T \mathbf{x} - \mathbf{b}\right) \delta\left(\mathbf{y} - \mathbf{y}_0\right). \end{aligned} \tag{6.68}$$

However, returning to Equation 6.61 and using the multiplicative property for a Dirac delta function in Equation 4.12, we also obtain

$$\phi_0(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}, \mathbf{y}_0)\, \delta(\mathbf{y} - \mathbf{y}_0). \tag{6.69}$$

By comparing Equations 6.68 and 6.69, we can therefore conclude that

$$\phi(\mathbf{x}, \mathbf{y}_0) = \mathcal{C}\left(Z^T U_{\mathbf{x}}^T \mathbf{x}; \hat{\Lambda}, \hat{\mathbf{h}}, \hat{g}\right) \delta\left(V_{\mathbf{x}}^T \mathbf{x} - \mathbf{b}\right) = \mathcal{D}(\mathbf{x}; U_{\mathbf{x}} Z, V_{\mathbf{x}}, \hat{\Lambda}, \hat{\mathbf{h}}, \mathbf{b}, \hat{g}). \tag{6.70}$$

This concludes the derivation of Algorithm 5. □

In summary, the aim of this chapter was to derive results for marginalisation, multiplication, division and reduction for our definition of degenerate Gaussian factors as proposed in Equation 5.1. The procedures summarised in Algorithms 2 to 5 reveal that this parametrisation is closed under these statistical operations. Consequently, these general procedures enable the use of degenerate factors for solving typical inference problems. Importantly, we keep track of normalisation constants throughout, which allows model comparison and maximum a posteriori (MAP) estimation in degenerate settings. In the next chapter, we derive the asymptotic time complexity of these procedures and evaluate their execution times in an experimental setup.

Chapter 7

Computational complexity

In computer science, asymptotic complexity is an important characteristic of any algorithm or mathematical procedure. Also referred to as "big O" notation, it provides an indication of the processing time or memory requirements as the dimensionality of the input grows. When deriving the theoretical complexity of an algorithm, the modus operandi is to assume the worst-case scenario where applicable and to only retain the highest-order term without any coefficients in the final answer. For example, in the case of the degenerate Gaussian factor in Equation 5.1, the number of parameters $n^2+2n-k+1 = \mathcal{O}(n^2)$ increases quadratically as the total dimension of the vector $\mathbf{x}$ tends to infinity. The purpose of this chapter is to calculate the time complexity, measured in floating point operations (FLOPs), of the marginalisation, multiplication, division and reduction operations as derived in the previous chapter. Since many of these computations involve common operations such as matrix multiplication and inversion, computing subspaces and calculating the singular value decomposition (SVD), we start by discussing the complexity of these matrix operations. We then derive the complexity of the statistical operations themselves. Finally, we provide an additional perspective on the computational cost of inference using degenerate factors by evaluating actual execution times in an experimental setup.

7.1 Complexity of matrix operations

The time complexity of the common matrix operations used in Algorithms 2 to 5 are summarised in Table 7.1, where some results are straightforward and others require a slightly longer motivation. Importantly, for square, $n \times n$ matrices, all of these operations require at most $\mathcal{O}(n^3)$ FLOPs. In particular:

- The *sum* of two $m \times n$ matrices consists of mn scalar additions.
- The *product* of an $m \times n$ matrix and an $n \times p$ matrix amounts to mp additions of n scalar products each. In the special case where the second matrix is $n \times n$ and diagonal, this reduces to mn scalar multiplications.
- Using Gauss-Jordan elimination [27], the *inverse* of an $n \times n$ matrix is computed by performing $\mathcal{O}(n^2)$ row reductions on rows with $\mathcal{O}(n)$ elements each and then rescaling each row. Matrix inversion can be done in $\mathcal{O}(n^{2.807})$ [33] or even $\mathcal{O}(n^{2.376})$ [34], although the large constant coefficient in the latter renders it impractical for typical matrices.

- According to Equation 3.14, the *pseudo-inverse* of an $m \times n$ matrix with full column rank amounts to a matrix product, an inverse and another product with complexity $\mathcal{O}(mn^2)$, $\mathcal{O}(n^3)$ and $\mathcal{O}(mn^2)$, respectively.
- Using Gaussian elimination with complexity $\mathcal{O}(n^3)$, the *determinant* of an $n \times n$ matrix is the product of the n pivots [27].
- According to Golub and van Loan [35], the *SVD* of an $m \times n$ matrix where $m \geq n$ can be computed in $\mathcal{O}(m^2n)$ time. In the case where $m < n$, the SVD $A^T = V\Sigma^T U^T$ with complexity $\mathcal{O}(mn^2)$ is used instead.
- Equation 3.24 reveals that an orthonormal basis for the *column space* of an $m \times n$ matrix can be obtained from its compact SVD. If $m \geq n$, the complexity of the compact SVD is $\mathcal{O}(mn^2)$ [35]. In the case where $m < n$, the SVD $A^T = V_r\Sigma_r U_r^T$ with complexity $\mathcal{O}(m^2n)$ is used instead.
- In turn, Equation 3.25 reveals that an orthonormal basis for the *nullspace* of an $m \times n$ matrix can be obtained from its SVD. However, in the case where $m > n$, calculating the matrix V alone only requires $\mathcal{O}(mn^2)$ time [35].
- Finally, Equation 3.27 reveals that an orthonormal basis for the orthogonal *complement* of the column space of an $m \times n$ matrix can also be obtained from its SVD. Since such matrices are always semi-orthogonal in Algorithms 2 to 5, the (slightly faster) QR decomposition can also be used.

Depending on the details, note that it could be possible to reduce the complexity of an algorithm by combining multiple of these matrix operations. For example, when using Gauss-Jordan elimination to compute an inverse, the determinant can be calculated as a by-product. Similarly, computing both the column space and nullspace of the same matrix only requires a single SVD. Such insights are imperative when implementing an algorithm in practice, as well as when deriving the overall complexity thereof.

Operation	**Input**	**Output**	**Complexity**
Sum	$m \times n$ and $m \times n$	$m \times n$	$\mathcal{O}(mn)$
Product	$m \times n$ and $n \times p$	$m \times p$	$\mathcal{O}(mnp)$
Inverse	$n \times n$	$n \times n$	$\mathcal{O}(n^3)$
Pseudo-inverse	$m \times n$, $m \geq n$	$n \times m$	$\mathcal{O}(mn^2)$
Determinant	$n \times n$	scalar	$\mathcal{O}(n^3)$
SVD	$m \times n$	$m \times m$, $m \times n$ and $n \times n$	$\mathcal{O}(\max[m^2n, mn^2])$
Column space	$m \times n$, rank r	$m \times r$	$\mathcal{O}(\min[m^2n, mn^2])$
Nullspace	$m \times n$, rank r	$n \times (n-r)$	$\mathcal{O}(mn^2)$
Complement	$m \times n$, $\perp$	$m \times (m-n)$	$\mathcal{O}(m^2n)$

Table 7.1: Asymptotic time complexity of common matrix operations in terms of the input and output dimensions of the applicable matrices.

7.2 Complexity of operations on degenerate factors

By using the complexity of the matrix operations in Table 7.1, we can derive the complexity of the marginalisation, multiplication, division and reduction of degenerate Gaussian factors. For each statistical operation, this requires identifying the complexity corresponding to every step of the specific algorithm and then selecting the terms that could be the most expensive overall (which are indicated throughout this section by underlining them). In addition to the total dimension of the input, each of the operations could have additional dimensions of interest, for example the number of variables in the marginal factor in the case of marginalisation. We therefore include all the relevant quantities explicitly in order to derive more detailed results, except where the expressions can be simplified significantly by assuming the worst-case scenario.

7.2.1 Marginalisation

For the *marginalisation* operation as outlined in Algorithm 2, the matrix R has shape $n \times k$ and R' has shape $n' \times k'$. By also specifying that $\text{rank}(R_{\mathbf{x}}^T Q_{\mathbf{x}}) = l$, the shapes of all the relevant matrices can then be determined. Since $k' \leq \min[n', k]$ and $l \leq \min[n', k, n - n', n - k]$, the complexity of each line of the algorithm is as follows:

1. U involves a column space requiring $\mathcal{O}(\min[n'^2(n-k), n'(n-k)^2])$. V involves a nullspace requiring $\underline{\mathcal{O}(n'(n-k)^2)}$. W involves a multiplication requiring $\underline{\mathcal{O}(n'k(n-k))}$ and a column space requiring $\underline{\mathcal{O}(\min[k^2(n-k), k(n-k)^2])}$.

2. R' involves a complement requiring $\underline{\mathcal{O}(n'^2(n-k))}$.

3. $\mathbf{c}'$ involves two matrix-vector products (starting from the right) requiring $\mathcal{O}(n'k)$ and $\mathcal{O}(n'k')$.

4. F involves a product requiring $\underline{\mathcal{O}(kl(n-n'))}$, a pseudo-inverse requiring $\mathcal{O}(l^2(n-n'))$ and two further products (starting from the right) requiring $\underline{\mathcal{O}(l(n-n')(n-k))}$ and $\mathcal{O}(kl(n-k))$. G involves a product requiring $\underline{\mathcal{O}(n'k(n-k))}$, a sum requiring $\mathcal{O}(n'(n-k))$ and another product requiring $\mathcal{O}(n'(n-k)(n'-k'))$. S involves two products requiring $\mathcal{O}((n-k)(n-k-(n'-k')))$ and $\mathcal{O}((n-k)(n-k-(n'-k'))^2)$, an inverse requiring $\mathcal{O}((n-k-(n'-k'))^3)$ and two further products requiring $\mathcal{O}((n-k)(n-k-(n'-k'))^2)$ and $\underline{\mathcal{O}((n-k)^2(n-k-(n'-k')))}$.

5. Z and Λ' involve two products and a sum requiring $\mathcal{O}((n-k)^2)$ each, two further products requiring $\mathcal{O}((n-k)^2(n'-k'))$ and $\mathcal{O}((n-k)(n'-k')^2)$ and an SVD requiring $\mathcal{O}((n'-k')^3)$.

6. Q' involves a product requiring $\mathcal{O}(n'(n'-k')^2)$.

7. $\mathbf{h}'$ involves various sums and products, but all of these include either vectors or diagonal matrices.

8. g' also involves various such sums and products and the two determinants can be calculated as by-products of the two inverses.

The overall complexity of the algorithm can therefore best be summarised as follows: If the largest block matrix has shape $d_1 \times d_2$, the marginalisation of a degenerate factor has complexity $\mathcal{O}(x)$, where

$$x = \begin{cases} d_1^2(n-k) & d_1 \geq d_2, n' \geq n - n' \\ \max[d_1 d_2(n-d_1), d_2(n-k)^2] & d_1 \geq d_2, n' < n - n' \\ d_1 d_2(n-k) & d_1 < d_2. \end{cases} \tag{7.1}$$

Since d_1, d_2 and k are all bound by n, the complexity is $\mathcal{O}(n^3)$ at most. Furthermore, for constant n, the complexity decreases as the degree of degeneracy $k \to n$.

7.2.2 Multiplication

For the *multiplication* operation as outlined in Algorithm 3, the matrix R_1 has shape $n \times k_1$ and R_2 has shape $n \times k_2$. By assuming that $k_1 \geq k_2$ without loss of generality, the complexity of each line of the algorithm is as follows:

1. V involves two products (from right to left) requiring $\mathcal{O}(nk_2(n-k_1))$ each and a column space requiring $\mathcal{O}(nk_2^2)$.
2. R' simply involves a concatenation.
3. $\mathbf{b}$ involves a product requiring $\mathcal{O}(nk_2^2)$, an inverse requiring $\mathcal{O}(k_2^3)$ and various sums and products involving vectors.
4. $\mathbf{c}'$ simply involves a concatenation.
5. U involves a complement requiring $\underline{\mathcal{O}(n^2(k_1+k_2))}$.
6. Z and Λ' involve six products (starting from the outside) requiring $\mathcal{O}(n(n-k_1)(n-k_1-k_2))$, $\mathcal{O}((n-k_1)(n-k_1-k_2))$, $\mathcal{O}((n-k_1)(n-k_1-k_2)^2)$, $\underline{\mathcal{O}(n(n-k_2)(n-k_1-k_2))}$, $\mathcal{O}((n-k_2)(n-k_1-k_2))$ and $\mathcal{O}((n-k_2)(n-k_1-k_2)^2)$, a sum requiring $\mathcal{O}((n-k_1-k_2)^2)$ and an SVD requiring $\mathcal{O}((n-k_1-k_2)^3)$.
7. Q' involves a product requiring $\mathcal{O}(n(n-k_1-k_2)^2)$.
8. $\mathbf{h}'$ involves various sums and products, but all of these include either vectors or diagonal matrices.
9. g' also involves various such sums and products and the determinant can be calculated as a by-product of the inverse.

The multiplication of two degenerate factors therefore has complexity $\mathcal{O}(x)$, where

$$x = \max[n^2(k_1+k_2), n(n-k_2)(n-k_1-k_2)]. \tag{7.2}$$

Since k_1 and k_2 are both bound by n, the complexity is also $\mathcal{O}(n^3)$ at most. Furthermore, for constant n and small k_2, the complexity decreases as the degree of degeneracy $k_1 \to \frac{n}{2}$.

7.2.3 Division

For the *division* operation as outlined in Algorithm 4, the matrix R_1 has shape $n \times k_1$ and R_2 has shape $n \times k_2$. Since $k_1 \geq k_2$, the complexity of each line of the algorithm is as follows:

1. R' involves a concatenation and a complement requiring $\underline{\mathcal{O}(n^2(n-k_1+k_2))}$.
2. $\mathbf{c}'$ involves two matrix-vector products (starting from the right) requiring $\mathcal{O}(nk_1)$ and $\mathcal{O}(n(k_1-k_2))$.
3. Z and Λ_+ involve three products (starting from the outside) requiring $\mathcal{O}(n(n-k_1)(n-k_2))$, $\mathcal{O}((n-k_1)(n-k_2))$ and $\mathcal{O}((n-k_1)^2(n-k_2))$, a sum requiring $\mathcal{O}((n-k_1)^2)$ and an SVD requiring $\mathcal{O}((n-k_1)^3)$.
4. Q' involves a product requiring $\mathcal{O}(n(n-k_1)^2)$ and a concatenation.
5. Λ' simply involves a concatenation.
6. $\mathbf{h}'$ involves various sums and products, but all of these include either vectors or diagonal matrices.
7. g' also involves various such sums and products.

The division of two degenerate factors therefore has complexity $\mathcal{O}(x)$, where

$$x = n^2(n-k_1+k_2). \tag{7.3}$$

Since k_1 and k_2 are both bound by n, the complexity is also $\mathcal{O}(n^3)$ at most. Furthermore, for constant n and small k_2, the complexity decreases as the degree of degeneracy $k_1 \to n$.

7.2.4 Reduction

For the *reduction* operation as outlined in Algorithm 5, the matrix R has shape $n \times k$ and the vector $\mathbf{y}_0$ has $n - n'$ components. Since $k \leq n'$, the complexity of each line of the algorithm is as follows:

1. R' involves a column space requiring $\mathcal{O}(\min[n'^2k, n'k^2])$.
2. $\mathbf{c}'$ involves a product requiring $\mathcal{O}(n'k^2)$, an inverse requiring $\mathcal{O}(k^3)$ and various sums and products involving vectors.
3. $U_{\mathbf{x}}$ involves a complement requiring $\underline{\mathcal{O}(n'^2k)}$.
4. Z and Λ' involve three products (starting from the outside) requiring $\underline{\mathcal{O}(n'(n'-k)(n-k))}$, $\mathcal{O}((n'-k)(n-k))$ and $\mathcal{O}((n'-k)^2(n-k))$, and an SVD requiring $\underline{\mathcal{O}((n'-k)^3)}$.
5. Q' involves a product requiring $\mathcal{O}(n'(n'-k)^2)$.
6. $\mathbf{h}'$ involves various sums and products, but all of these include either vectors or diagonal matrices.
7. g' also involves various such sums and products and the determinant can be calculated as a by-product of the inverse.

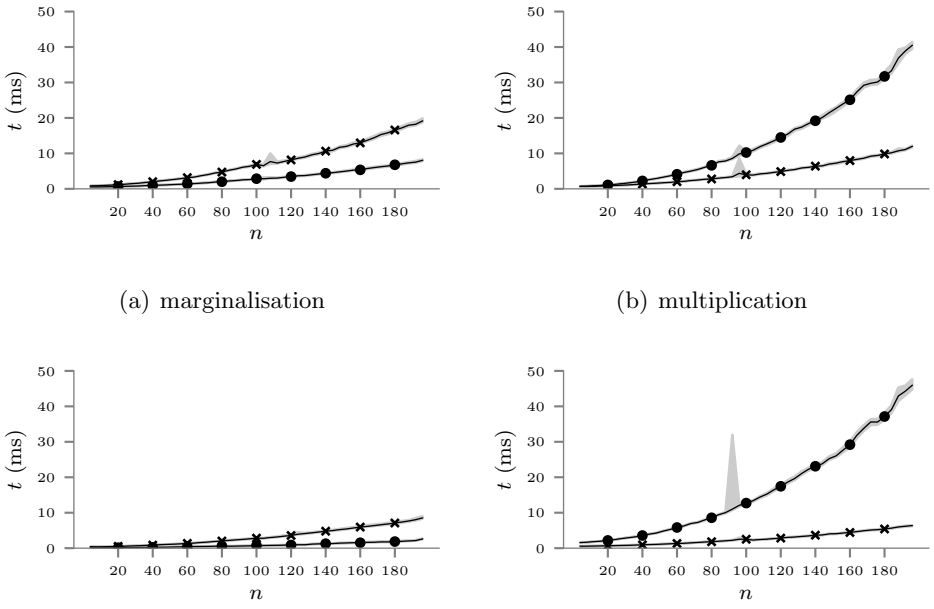
The reduction of a degenerate factor therefore has complexity $\mathcal{O}(x)$, where

$$x = \max[n'^2 k, n'(n' - k)(n - k)]. \tag{7.4}$$

Since n' and k are both bound by n, the complexity is also $\mathcal{O}(n^3)$ at most.

7.3 Measured execution times

Although the asymptotic complexity provides useful insights into the computational cost of an algorithm, it has a number of shortcomings when viewed in isolation. Firstly, since the analysis is only concerned with the limit as the input dimension tends toward infinity, any coefficients and lower-order terms are omitted. For finite (and especially small) input sizes, both of these could in fact be very significant. For example, although $n^2 + 100n = \mathcal{O}(n^2)$, $100n > n^2$ for $n < 100$. Secondly, these results only provide a *theoretical* indication of the computational cost of an algorithm, where another important perspective is the *measured* execution times for an implementation thereof. For each of the four operations in Algorithms 2 to 5, this is indicated in Figure 7.1, where we used the parametrisation for degenerate factors proposed by Raphael [18] as a baseline.



(a) marginalisation

(b) multiplication

(c) division

(d) reduction

Figure 7.1: Measured execution times (in milliseconds) of the four statistical operations using our degenerate Gaussian factors (indicated by the lines with x-shaped markers) and those proposed by Raphael [18] (indicated by the lines with circular markers) as a function of the input dimension n. The 25'th and 75'th percentile (based on multiple experiments) are indicated by the shaded regions.

For these experiments, we varied the dimension of the input matrices from 1 to 200 and repeated the process 500 times. For both parametrisations, we used unoptimised implemen-

tations using standard Python libraries such as `numpy` and `scipy`. The computations were performed on a single thread of an Intel® Core™ i7 CPU. To conform to the methodologies of belief propagation [22] and belief update [23], we adapted the reduction operation of Raphael [18] to marginalise over the observed variables after the additional constraint has been incorporated. Despite the fact that this is also necessary for calculating normalisation constants, it reduces the dimension of the output which leads to less computational time downstream.

As indicated by Figure 7.1, the execution times are longer using our parametrisation in the case of marginalisation and division, but shorter in the case of multiplication and reduction. Importantly, both the difference and the absolute times are larger for the latter two operations. This indicates that these are the more expensive operations and therefore the advantage of our parametrisation is significant. In addition, for typical inference problems, multiplication is performed more frequently compared to marginalisation and reduction, and division is usually performed for factors with smaller scopes. Ultimately, one needs to consider the overall execution time for a specific inference problem to obtain an additional comparison. This is included at the end of Chapter 9 for a representative example.

In summary, due to our use of the SVD, the asymptotic time complexity of the operations in Algorithms 2 to 5 is $\mathcal{O}(n^3)$. This is similar to that for Raphael's [18] representation. In comparison, the multiplication, division and reduction operations for Koller and Friedman's [3] (non-degenerate) canonical factors require only $\mathcal{O}(n^2)$. However, due to the matrix inversion in Equation 2.16, their marginalisation operation also requires $\mathcal{O}(n^3)$ and therefore often dominates the computational time in practice. The expressiveness of degenerate factors therefore comes at little additional computational cost.

Chapter 8

Additional operations necessary for inference

In addition to the four operations on degenerate factors discussed in Chapter 6, we present further operations that are necessary for performing inference on Bayesian networks in this chapter. These include correctly manipulating degenerate factors with different scopes, modelling stochastic systems by representing conditional densities (associated with both linear and nonlinear dependencies) and checking for convergence of message passing algorithms. The detailed derivations are once again included at the end of each section where applicable.

8.1 Extending and rearranging factor scopes

Performing inference on PGMs often amounts to multiplying (or dividing) two factors with different scopes. This is true even for the simple case of computing the joint density $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y}|\mathbf{x})\, p(\mathbf{x})$, where the scope of the second factor is a subset of that of the first. In such cases, we need to extend the scope of each factor in the product to be the union of all the scopes. For this purpose, we can extend the scope of a given degenerate factor with scope $\mathbf{x}$ to include the vector $\mathbf{y}$, such that the condition

$$\mathcal{D}\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}; Q', R', \Lambda', \mathbf{h}', \mathbf{c}, g\right) = \mathcal{D}\left(\mathbf{x}; Q, R, \Lambda, \mathbf{h}, \mathbf{c}, g\right) \tag{8.1}$$

is satisfied. A possible choice for the parameters in Equation 8.1 that conforms to the definition of the degenerate factor in Equation 5.1 is

$$Q' = \begin{bmatrix} Q & 0 \\ 0 & I \end{bmatrix}, \quad R' = \begin{bmatrix} R \\ 0 \end{bmatrix}, \quad \Lambda' = \begin{bmatrix} \Lambda & 0 \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{h}' = \begin{bmatrix} \mathbf{h} \\ \mathbf{0} \end{bmatrix}. \tag{8.2}$$

Note that Λ' is diagonal and that $C\left(Q'\right) = C\left(R'\right)^{\perp}$ as required.

Another operation that is necessary for a message passing implementation is rearranging the scope of a given factor. For instance, consider a degenerate Gaussian factor with scope $\{\mathbf{x}, \mathbf{y}\}$ where the matrices Q and R are partitioned accordingly. We can determine another factor with scope $\{\mathbf{y}, \mathbf{x}\}$, such that

$$\mathcal{D}\left(\begin{bmatrix}\mathbf{y}\\ \mathbf{x}\end{bmatrix}; Q', R', \Lambda, \mathbf{h}, \mathbf{c}, g\right) = \mathcal{D}\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}; \begin{bmatrix}Q_{\mathbf{x}}\\ Q_{\mathbf{y}}\end{bmatrix}, \begin{bmatrix}R_{\mathbf{x}}\\ R_{\mathbf{y}}\end{bmatrix}, \Lambda, \mathbf{h}, \mathbf{c}, g\right), \tag{8.3}$$

by simply rearranging the rows of the two matrices

$$Q' = \begin{bmatrix} Q_{\mathbf{y}} \\ Q_{\mathbf{x}} \end{bmatrix} \quad \text{and} \quad R' = \begin{bmatrix} R_{\mathbf{y}} \\ R_{\mathbf{x}} \end{bmatrix}. \tag{8.4}$$

This operation is necessary to insure that, prior to applying any statistical operations, the scopes of multiple factors not only have the same dimension but are also aligned. By extending the result in Equation 8.3 for more partitions of Q and R, any scope rearrangement can be performed.

8.2 Representing conditional density functions

To model a stochastic system using a Bayesian network, it is necessary to represent conditional density functions – in our case as parametrised degenerate factors. Similar to Equation 2.13 for canonical factors, we start with the special case of linear transformations, where the expression for the conditional density is exact. However, since it is usually also necessary to represent more general nonlinear relationships, we propose a method for approximating these as in Section 2.3.

8.2.1 Linear dependencies

Given the affine transformation

$$\mathbf{y} = A\mathbf{x} + \mathbf{b} + \mathbf{w} \tag{8.5}$$

subject to (possibly degenerate) independent noise

$$\mathbf{w} \sim \mathcal{D}(\mathbf{w}; Q, R, \Lambda, \mathbf{h}, \mathbf{c}, g), \tag{8.6}$$

we can represent the conditional density

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{D}\left(\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}; Q', R', \Lambda', \mathbf{h}', \mathbf{c}', g'\right), \tag{8.7}$$

where the parameters Q', R', Λ', $\mathbf{h}'$, $\mathbf{c}'$ and g' are determined according to Algorithm 6. By treating the vector $\mathbf{x}$ as constant, the idea is to use an affine transformation of the random vector $\mathbf{w}$ (according to Algorithm 1) to express $p(\mathbf{y}|\mathbf{x})$ as a factor over $\mathbf{y}$. We then rewrite this factor as a parametrised degenerate factor with scope $\{\mathbf{x}, \mathbf{y}\}$.

The first step (in line 2 of Algorithm 6) is to use the auxiliary matrix F (in line 1) to find an orthonormal basis R' for the resulting degenerate space in $\mathbb{R}^{n_{\mathbf{x}}+n_{\mathbf{y}}}$, where the matrix A has shape $n_{\mathbf{y}} \times n_{\mathbf{x}}$. The corresponding k-dimensional offset $\mathbf{c}'$ (in line 4) is then determined using the auxiliary matrix Z (in line 3). To ensure that $C(Q') \perp C(R')$, the projection matrix (in line 5) is applied before the compact SVD (in line 6). This yields the basis Q_+ corresponding to the $n_{\mathbf{y}} - k$ non-trivial singular values Λ_+, where we use the same θ_+-subscripts as Raphael [18]. The remainder of the orthogonal decomposition of $\mathbb{R}^{n_{\mathbf{x}}+n_{\mathbf{y}}}$ corresponding to the $n_{\mathbf{x}}$ trivial singular values is then determined (in line 7) and used to augment Q' and Λ' (in lines 8 and 9, respectively). Finally, the vector $\mathbf{h}'$ and normalisation constant g' are also calculated (in lines 10 and 11, respectively). Figure 8.1 illustrates various aspects of Algorithm 6 for a simple example.

Algorithm 6 `RepresentConditional`

Input: θ, A, $\mathbf{b}$ such that $p(\mathbf{w}) = \mathcal{D}(\mathbf{w};\theta)$ and $\mathbf{y} = A\mathbf{x} + \mathbf{b} + \mathbf{w}$

Output: θ' such that $p(\mathbf{y}|\mathbf{x}) = \mathcal{D}\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix};\theta'\right)$

$F \leftarrow \begin{bmatrix}-A & I\end{bmatrix}$
$R' \leftarrow$ `Columnspace` $\left(F^T R\right)$
$Z \leftarrow R'^T F^T R \left(R^T(I + AA^T)R\right)^{-1}$
$\mathbf{c}' \leftarrow Z(\mathbf{c} + R^T\mathbf{b})$
$P \leftarrow \left(I - R'R'^T\right)$
$Q_+, \Lambda_+, \_ \leftarrow$ `CompactSVD` $\left(PF^TQ\Lambda Q^TFP\right)$
$Q_\infty \leftarrow$ `Complement` $\left(\begin{bmatrix}Q_+ & R'\end{bmatrix}\right)$
$Q' \leftarrow \begin{bmatrix}Q_+ & Q_\infty\end{bmatrix}$
$\Lambda' = \begin{bmatrix}\Lambda_+ & 0\\ 0 & 0\end{bmatrix}$
$\mathbf{h}' \leftarrow Q'^T F^T Q(\mathbf{h} + \Lambda Q^T(\mathbf{b} - FR'\mathbf{c}'))$
$g' \leftarrow g - (\mathbf{h} + \frac{1}{2}\Lambda Q^T\mathbf{b})^T Q^T\mathbf{b} + (\mathbf{h} + \Lambda Q^T(\mathbf{b} - \frac{1}{2}FR'\mathbf{c}'))^T Q^T FR'\mathbf{c}' + \log|Z|$

Proof. To determine the parameters of the degenerate factor representing the conditional density $p(\mathbf{y}|\mathbf{x})$ as calculated according to Algorithm 6, we rewrite Equation 8.5 as

$$\mathbf{y} = I\mathbf{w} + (A\mathbf{x} + \mathbf{b}). \tag{8.8}$$

Using the results in Algorithm 1 for this special case with the identity matrix, we can then express the conditional density as

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{D}(\mathbf{y}; Q, R, \Lambda, \hat{\mathbf{h}}, \hat{\mathbf{c}}, \hat{g}), \tag{8.9}$$

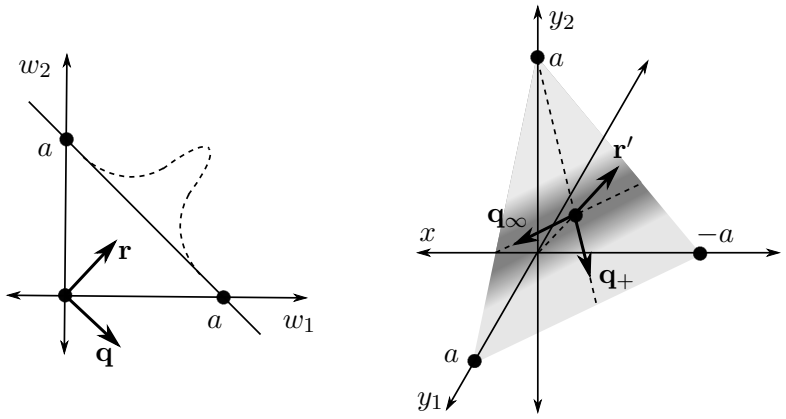
where

$$\begin{aligned}
\hat{\mathbf{c}} &= R^T\left((Q\Lambda^{-1}\mathbf{h} + R\mathbf{c}) + (A\mathbf{x} + \mathbf{b})\right) = \mathbf{c} + R^T(A\mathbf{x} + \mathbf{b})\\
\hat{\mathbf{h}} &= \Lambda Q^T\left((Q\Lambda^{-1}\mathbf{h} + R\mathbf{c}) + (A\mathbf{x} + \mathbf{b})\right) = \mathbf{h} + \Lambda Q^T(A\mathbf{x} + \mathbf{b})\\
\hat{g} &= -\frac{1}{2}(\mathbf{h} + \Lambda Q^T(A\mathbf{x} + \mathbf{b}))^T\Lambda^{-1}(\mathbf{h} + \Lambda Q^T(A\mathbf{x} + \mathbf{b})) - \frac{1}{2}\log\left|2\pi\Lambda^{-1}\right|\\
&= g - \frac{1}{2}(A\mathbf{x} + \mathbf{b})^T Q\Lambda Q^T(A\mathbf{x} + \mathbf{b}) - \mathbf{h}^T Q^T(A\mathbf{x} + \mathbf{b})
\end{aligned} \tag{8.10}$$

and where we have made use of the result in Equation 5.4 in the last line. We can then use the definition of the degenerate factor in Equation 5.1 to expand Equation 8.9 according to

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{C}\left(Q^T\mathbf{y}; \Lambda, \mathbf{h} + \Lambda Q^T(A\mathbf{x} + \mathbf{b}), \hat{g}\right)\ \delta\left(R^T\mathbf{y} - \left(\mathbf{c} + R^T\left(A\mathbf{x} + \mathbf{b}\right)\right)\right). \tag{8.11}$$

By using the definition in Equation 2.4 and rearranging terms, the canonical factor in



(a) (b)

Figure 8.1: (a) A visualisation of a degenerate factor in $\mathbb{R}^2$ subject to the linear noise constraint $w_1 + w_2 = a$. The dashed 1-D curve indicates the probability distribution along the constraint (and in the direction of the vector $\mathbf{q}$). Away from this line (in the direction of $\mathbf{r}$) the factor is equal to zero. (b) The constructed degenerate factor $p(y_1, y_2|x)$ in $\mathbb{R}^3$, where $y_1 = w_1 + x$ and $y_2 = w_2$. This factor is consequently subject to the linear constraint $y_1 + y_2 - x = a$ and the vector $\mathbf{r}'$ is perpendicular to the resulting affine plane. In contrast, the vectors $\mathbf{q}_+$ and $\mathbf{q}_\infty$ lie in the plane, where the factor is invariant to any change along the latter.

Equation 8.11 can further be rewritten as

$$\begin{aligned}
&\exp\left(-\frac{1}{2}\mathbf{y}^T Q\Lambda Q^T\mathbf{y} + (\mathbf{h} + \Lambda Q^T(A\mathbf{x} + \mathbf{b}))^T Q^T\mathbf{y} + \hat{g}\right) \\
&= \exp\left(-\frac{1}{2}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}^T \begin{bmatrix}-A & I\end{bmatrix}^T Q\Lambda Q^T \begin{bmatrix}-A & I\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} + \right. \\
&\left. (\mathbf{h} + \Lambda Q^T\mathbf{b})^T Q^T \begin{bmatrix}-A & I\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} + g - (\mathbf{h} + \frac{1}{2}\Lambda Q^T\mathbf{b})^T Q^T\mathbf{b}\right). \qquad (8.12)
\end{aligned}$$

Similarly, the Dirac delta in Equation 8.11 can be rewritten as

$$\delta\left(R^T(\mathbf{y} - A\mathbf{x}) - (\mathbf{c} + R^T\mathbf{b})\right) = \delta\left(R^T\begin{bmatrix}-A & I\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} - (\mathbf{c} + R^T\mathbf{b})\right). \qquad (8.13)$$

To guarantee that the matrix R' will be semi-orthogonal, we define the orthonormal basis U such that

$$C(U) = C\left(\begin{bmatrix}-A & I\end{bmatrix}^T R\right). \qquad (8.14)$$

Since the two corresponding projection matrices are also equal, we can multiply the argument of the Dirac delta in Equation 8.13 by the carefully-chosen matrix

$$Z = U^T\begin{bmatrix}-A & I\end{bmatrix}^T R\left(R^T\left(I + AA^T\right)R\right)^{-1} \qquad (8.15)$$

and using Equation 4.14, we can write

$$\delta\left(R^T\begin{bmatrix}-A & I\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}-(\mathbf{c}+R^T\mathbf{b})\right)=|Z|\,\delta\left(R'^T\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}-\mathbf{c}'\right), \tag{8.16}$$

where

$$R'=U \qquad \text{and} \qquad \mathbf{c}'=Z(\mathbf{c}+R^T\mathbf{b}). \tag{8.17}$$

To find the remainder of the parameters Q', Λ', $\mathbf{h}'$ and g', the quadratic term in the exponent in Equation 8.12 should be expanded according to

$$\begin{aligned}-\frac{1}{2}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}^T S\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}=&-\frac{1}{2}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}^T(I-R'R'^T)S(I-R'R'^T)\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}+\\ &\frac{1}{2}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}^T R'R'^T SR'R'^T\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}-\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}^T R'R'^T S\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix},\end{aligned} \tag{8.18}$$

where we made use of the matrix definitions

$$S=F^TQ\Lambda Q^TF \qquad \text{and} \qquad F=\begin{bmatrix}-A & I\end{bmatrix}. \tag{8.19}$$

Substituting Equations 8.12, 8.13, 8.16 and 8.18 into Equation 8.11 and using the multiplicative property in Equation 4.12 yields

$$\begin{aligned}p(\mathbf{y}|\mathbf{x})=\exp\Bigg(&-\frac{1}{2}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}^T(I-R'R'^T)S(I-R'R'^T)\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}+\frac{1}{2}\mathbf{c}'^TR'^TSR'\mathbf{c}'-\mathbf{c}'^TR'^TS\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}+\\ &(\mathbf{h}+\Lambda Q^T\mathbf{b})^TQ^TF\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}+g-(\mathbf{h}+\frac{1}{2}\Lambda Q^T\mathbf{b})^TQ^T\mathbf{b}+\log|Z|\Bigg)\,\delta\left(R'^T\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}-\mathbf{c}'\right).\end{aligned} \tag{8.20}$$

This reveals the motivation behind the expansion in Equation 8.18, namely that, for the compact SVD of the quadratic coefficient in Equation 8.20

$$Q_+\Lambda_+Q_+^T=(I-R'R'^T)S(I-R'R'^T), \tag{8.21}$$

we ensure that $C(Q_+)\perp C(R')$. However, if the density function over the vector $\mathbf{w}\in\mathbb{R}^{n_\mathbf{y}}$ in Equation 8.6 has k degrees of degeneracy, then

$$\text{rank}(R')=\text{rank}(R)=k \qquad \text{and} \qquad \text{rank}(Q_+)=\text{rank}(Q)=n_\mathbf{y}-k \tag{8.22}$$

and consequently $C(Q_+)+C(R')\subset\mathbb{R}^{n_\mathbf{x}+n_\mathbf{y}}$, where $\mathbf{x}\in\mathbb{R}^{n_\mathbf{x}}$. We therefore define a third orthonormal basis Q_∞ (with dimension $n_\mathbf{x}$) to complete the decomposition such that

$$C(Q_\infty)=(C(Q_+)+C(R'))^\perp. \tag{8.23}$$

For such an orthogonal decomposition, we can relate the respective projection matrices according to

$$I - R'R'^T = Q_+Q_+^T + Q_\infty Q_\infty^T = \begin{bmatrix} Q_+ & Q_\infty \end{bmatrix} \begin{bmatrix} Q_+ & Q_\infty \end{bmatrix}^T. \tag{8.24}$$

Similar to Equation 8.18, we now expand the linear term in the exponent in Equation 8.20 according to

$$\mathbf{w}^T \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \mathbf{w}^T (I - R'R'^T) \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} + \mathbf{w}^T R'R'^T \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}, \tag{8.25}$$

where we have made use of the definition

$$\mathbf{w} = F^T Q(\mathbf{h} + \Lambda Q^T \mathbf{b}) - SR'\mathbf{c}' = F^T Q(\mathbf{h} + \Lambda Q^T(\mathbf{b} - FR'\mathbf{c}')). \tag{8.26}$$

Finally, substituting Equations 8.21 and 8.25 followed by Equation 8.24 into Equation 8.20 and once again using the multiplicative property in Equation 4.12 yields

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{D}\left(\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}; \begin{bmatrix} Q_+ & Q_\infty \end{bmatrix}, R', \begin{bmatrix} \Lambda_+ & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} Q_+ & Q_\infty \end{bmatrix}^T \mathbf{w}, \mathbf{c}', g' \right), \tag{8.27}$$

where

$$g' = g - (\mathbf{h} + \frac{1}{2}\Lambda Q^T \mathbf{b})^T Q^T \mathbf{b} + (\mathbf{h} + \Lambda Q^T(\mathbf{b} - \frac{1}{2}FR'\mathbf{c}'))^T Q^T FR'\mathbf{c}' + \log|Z| \tag{8.28}$$

since

$$\begin{aligned} \frac{1}{2}SR'\mathbf{c}' + \mathbf{w} &= \frac{1}{2}F^T Q\Lambda Q^T FR'\mathbf{c}' + F^T Q(\mathbf{h} + \Lambda Q^T(\mathbf{b} - FR'\mathbf{c}')) \\ &= F^T Q\left(\mathbf{h} + \Lambda Q^T\left(\mathbf{b} - \frac{1}{2}FR'\mathbf{c}'\right)\right). \end{aligned} \tag{8.29}$$

This concludes the derivation of Algorithm 6. □

8.2.2 Nonlinear dependencies

Now consider the more general nonlinear transformation

$$\mathbf{y} = \mathbf{f}(\mathbf{x}, \mathbf{w}) \tag{8.30}$$

and suppose that we once again need to represent the conditional density

$$p(\mathbf{y}|\mathbf{x}) \approx \mathcal{D}\left(\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}; Q', R', \Lambda', \mathbf{h}', \mathbf{c}', g' \right). \tag{8.31}$$

In the nonlinear case, just as in Section 2.3, we need to resort to approximation through linearisation. When using the Taylor series expansion, the approach is very similar to the non-degenerate case, where Jacobian matrices can be used to obtain an affine approximation of Equation 8.30 before applying the results in Algorithm 6. However, to use the unscented

transform and sigma points drawn from the prior distribution[1]

$$p(\mathbf{x}, \mathbf{w}) = p(\mathbf{x})\, p(\mathbf{w}) = \mathcal{D}\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{w}\end{bmatrix}; Q, R, \Lambda, \mathbf{h}, \mathbf{c}, g\right) \tag{8.32}$$

requires a more subtle approach.

The idea is to use samples from the canonical component of the prior in Equation 8.32 to obtain an equivalent affine transformation $\mathbf{x} \mapsto \mathbf{y}$, given by

$$\mathbf{y} \approx \widetilde{A}\mathbf{x} + \widetilde{\mathbf{b}} + \widetilde{\mathbf{w}} \tag{8.33}$$

and where the affine transform noise is

$$\widetilde{\mathbf{w}} \sim \mathcal{D}(\widetilde{\mathbf{w}}; \widetilde{Q}, \widetilde{R}, \widetilde{\Lambda}, \widetilde{\mathbf{h}}, \widetilde{\mathbf{c}}, \widetilde{g}). \tag{8.34}$$

This is achieved in Algorithm 7 through moment-matching of the joint density $p(\mathbf{x}, \mathbf{y})$ resulting from each transformation in Equations 8.30 and 8.33. Once the parameters in Equations 8.33 and 8.34 have been calculated, we can approximate the conditional density in Equation 8.31 using Algorithm 6.

Algorithm 7 `EquivalentTransformation`

Input: θ, $\mathbf{f}(\cdot)$ such that $p(\mathbf{x}, \mathbf{w}) = \mathcal{D}\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{w}\end{bmatrix}; \theta\right)$ and $\mathbf{y} = \mathbf{f}(\mathbf{x}, \mathbf{w})$

Output: $\widetilde{\theta}$, $\widetilde{A}$, $\widetilde{\mathbf{b}}$ such that $p(\widetilde{\mathbf{w}}) = \mathcal{D}(\widetilde{\mathbf{w}}; \widetilde{\theta})$ and $\mathbf{y} \approx \widetilde{A}\mathbf{x} + \widetilde{\mathbf{b}} + \widetilde{\mathbf{w}}$

1: $\begin{bmatrix}\mathcal{X}\\ \mathcal{W}\end{bmatrix} \leftarrow \left[\mathbf{0}, \quad \gamma Q\sqrt{\Lambda^{-1}}, \quad -\gamma Q\sqrt{\Lambda^{-1}}\right] + \left(Q\Lambda^{-1}\mathbf{h} + R\mathbf{c}\right)\mathbf{1}^T$

2: **for** $i = 0, 2(n-k)$ **do**

3: $\quad \mathbf{y}^{[i]} \leftarrow \mathbf{f}\left(\mathbf{x}^{[i]}, \mathbf{w}^{[i]}\right)$

4: **end for**

5: $\boldsymbol{\mu} \leftarrow \sum_{i=0}^{2(n-k)} w_m^{[i]} \begin{bmatrix}\mathbf{x}^{[i]}\\ \mathbf{y}^{[i]}\end{bmatrix}$

6: $\Sigma \leftarrow \sum_{i=0}^{2(n-k)} w_c^{[i]} \left(\begin{bmatrix}\mathbf{x}^{[i]}\\ \mathbf{y}^{[i]}\end{bmatrix} - \boldsymbol{\mu}'\right)\left(\begin{bmatrix}\mathbf{x}^{[i]}\\ \mathbf{y}^{[i]}\end{bmatrix} - \boldsymbol{\mu}'\right)^T$

7: $\widetilde{A} \leftarrow \Sigma_{\mathbf{xy}}^T \Sigma_{\mathbf{xx}}^+$

8: $\widetilde{\mathbf{b}} \leftarrow \boldsymbol{\mu}_{\mathbf{y}} - \widetilde{A}\boldsymbol{\mu}_{\mathbf{x}}$

9: $\widetilde{Q}, \widetilde{\Sigma}, \_ \leftarrow$ `CompactSVD` $\left(\Sigma_{\mathbf{yy}} - \widetilde{A}\Sigma_{\mathbf{xy}}\right)$

10: $\widetilde{\Lambda} \leftarrow \widetilde{\Sigma}^{-1}$

11: $\widetilde{R} \leftarrow$ `Complement`$(\widetilde{Q})$

12: $\widetilde{\mathbf{h}} \leftarrow \mathbf{0}$, $\widetilde{\mathbf{c}} \leftarrow \mathbf{0}$

13: $\widetilde{g} \leftarrow -\frac{1}{2}\log\left|2\pi\widetilde{\Lambda}^{-1}\right|$

The first step (in line 1 of Algorithm 7) is to draw $2(n-k)$ sigma points from the prior distribution $p(\mathbf{x}, \mathbf{w})$, where k is the degree of degeneracy and γ is a scaling parameter. Each

[1] For statistically independent random vectors $\mathbf{x}$ and $\mathbf{w}$, this joint factor can easily be computed using the extension operation in Equation 8.2 on each marginal factor followed by (a special case of) the multiplication operation in Algorithm 3.

sigma point is then propagated through the nonlinear transform (in line 3). Similar to Equations 2.32 and 2.33, the mean (in line 5) and covariance (in line 6) of the joint density $p(\mathbf{x}, \mathbf{y})$ are approximated using the sigma points and corresponding weights. By partitioning the moments according to $\mathbf{x}$ and $\mathbf{y}$ and making use of the pseudo-inverse, the matrix $\widetilde{A}$ is determined (in line 7). This value of $\widetilde{A}$ is then used to determine that of $\widetilde{\mathbf{b}}$ (in line 8) as well as in the compact SVD of the noise covariance (in line 9). The quantities $\widetilde{\Lambda}$ (in line 10) and $\widetilde{R}$ (in line 11) are determined in the usual manner. Without loss of generality, the noise $\widetilde{\mathbf{w}}$ can be chosen as zero-mean, which implies that $\widetilde{\mathbf{h}} = \mathbf{0}$ and $\widetilde{\mathbf{c}} = \mathbf{0}$ (in line 12). Finally, the normalisation constant (in line 13) is calculated according to Equation 5.4.

Proof. To determine the parameters of the equivalent affine transformation in Equation 8.33 as well as that of the degenerate factor representing the transform noise in Equation 8.34 as calculated according to Algorithm 7, we start by approximating the moments of the joint density $p(\mathbf{x}, \mathbf{y})$ using the unscented transform. However, since the prior distribution in Equation 8.32 is degenerate and the Cholesky decomposition for a positive semi-definite matrix is not defined, the $2n + 1$ sigma points in Equation 2.28 cannot be used directly. Instead, we propose adapting the result by Thrun et al. [25] to draw $2(n - k) + 1$ sigma points

$$\mathcal{E} = \begin{bmatrix} \mathbf{0}, & \gamma\sqrt{\Lambda^{-1}}, & -\gamma\sqrt{\Lambda^{-1}} \end{bmatrix} + \Lambda^{-1}\mathbf{h}\mathbf{1}^T \tag{8.35}$$

from only the canonical component of the degenerate factor in Equation 5.1, where k is the degree of degeneracy and where we have made use of the substitution

$$Q^T \begin{bmatrix} \mathbf{x} \\ \mathbf{w} \end{bmatrix} = \boldsymbol{\epsilon}. \tag{8.36}$$

Since all samples (with non-zero likelihood) from the prior distribution in Equation 8.32 must satisfy

$$R^T \begin{bmatrix} \mathbf{x} \\ \mathbf{w} \end{bmatrix} = \mathbf{c}, \tag{8.37}$$

the $2k$ sigma points corresponding to the Dirac delta component of the degenerate factor and the basis R are not necessary. By combining Equation 8.36 and Equation 8.37, and since $C(Q) = C(R)^{\perp}$, we can then write

$$QQ^T \begin{bmatrix} \mathbf{x} \\ \mathbf{w} \end{bmatrix} + RR^T \begin{bmatrix} \mathbf{x} \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} \mathbf{x} \\ \mathbf{w} \end{bmatrix} = Q\boldsymbol{\epsilon} + R\mathbf{c}. \tag{8.38}$$

This can in turn be used to transform the sigma points in Equation 8.35 to correspond to the prior $p(\mathbf{x}, \mathbf{w})$, i.e.,

$$\begin{bmatrix} \mathcal{X} \\ \mathcal{W} \end{bmatrix} = \begin{bmatrix} \mathbf{0}, & \gamma Q\sqrt{\Lambda^{-1}}, & -\gamma Q\sqrt{\Lambda^{-1}} \end{bmatrix} + (Q\Lambda^{-1}\mathbf{h} + R\mathbf{c})\mathbf{1}^T. \tag{8.39}$$

Note that the approximated moments of the prior using these sigma points as in Equa-

tions 2.32 and 2.33, namely

$$\begin{aligned}\mathbb{E}\left[\begin{bmatrix}\mathbf{x}\\ \mathbf{w}\end{bmatrix}\right] &\approx \sum_{i=0}^{2(n-k)} w_m^{[i]} \begin{bmatrix}\mathbf{x}^{[i]}\\ \mathbf{w}^{[i]}\end{bmatrix} \\ &= \left(1 - \frac{n-k}{\gamma^2} + \sum_{i=1}^{2(n-k)} \frac{1}{2\gamma^2}\right)(Q\Lambda^{-1}\mathbf{h} + R\mathbf{c}) + \sum_{i=1}^{n-k} \frac{1}{2\gamma^2}\left(\frac{\gamma\mathbf{q}_i}{\sqrt{\lambda_i}} - \frac{\gamma\mathbf{q}_i}{\sqrt{\lambda_i}}\right) \\ &= Q\Lambda^{-1}\mathbf{h} + R\mathbf{c} \end{aligned} \tag{8.40}$$

and

$$\begin{aligned}\mathrm{Cov}\left[\begin{bmatrix}\mathbf{x}\\ \mathbf{w}\end{bmatrix}\right] &\approx \sum_{i=0}^{2(n-k)} w_c^{[i]} \left(\begin{bmatrix}\mathbf{x}^{[i]}\\ \mathbf{w}^{[i]}\end{bmatrix} - (Q\Lambda^{-1}\mathbf{h} + R\mathbf{c})\right)\left(\begin{bmatrix}\mathbf{x}^{[i]}\\ \mathbf{w}^{[i]}\end{bmatrix} - (Q\Lambda^{-1}\mathbf{h} + R\mathbf{c})\right)^T \\ &= \sum_{i=1}^{n-k} \frac{1}{2\gamma^2}\left(\frac{\gamma\mathbf{q}_i}{\sqrt{\lambda_i}}\right)\left(\frac{\gamma\mathbf{q}_i}{\sqrt{\lambda_i}}\right)^T + \sum_{i=1}^{n-k} \frac{1}{2\gamma^2}\left(-\frac{\gamma\mathbf{q}_i}{\sqrt{\lambda_i}}\right)\left(-\frac{\gamma\mathbf{q}_i}{\sqrt{\lambda_i}}\right)^T \\ &= \sum_{i=1}^{n-k} \frac{\mathbf{q}_i\mathbf{q}_i^T}{\lambda_i} = Q\Lambda^{-1}Q^T, \end{aligned} \tag{8.41}$$

are equal to those calculated in Equations 5.9 and 5.10. Next, each of the sigma point pairs $\mathbf{x}^{[i]}$ and $\mathbf{w}^{[i]}$ are propagated through the nonlinear transformation in Equation 8.30 to yield $\mathbf{y}^{[i]} = \mathbf{f}\left(\mathbf{x}^{[i]}, \mathbf{w}^{[i]}\right)$. Using the same weights, the moments of the joint density $p(\mathbf{x}, \mathbf{y})$ can then be approximated as

$$\mathbb{E}\left[\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}\right] \approx \sum_{i=0}^{2(n-k)} w_m^{[i]} \begin{bmatrix}\mathbf{x}^{[i]}\\ \mathbf{y}^{[i]}\end{bmatrix} \tag{8.42}$$

and

$$\mathrm{Cov}\left[\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}\right] \approx \sum_{i=0}^{2(n-k)} w_c^{[i]} \left(\begin{bmatrix}\mathbf{x}^{[i]}\\ \mathbf{y}^{[i]}\end{bmatrix} - \mathbb{E}\left[\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}\right]\right)\left(\begin{bmatrix}\mathbf{x}^{[i]}\\ \mathbf{y}^{[i]}\end{bmatrix} - \mathbb{E}\left[\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}\right]\right)^T. \tag{8.43}$$

For the equivalent affine transformation in Equation 8.33, the moments involving $\mathbf{y}$ are given by

$$\begin{aligned}\mathrm{Cov}[\mathbf{x}, \mathbf{y}] &= \mathrm{Cov}\left[\mathbf{x}\right]\widetilde{A}^T \\ \mathrm{Cov}[\mathbf{y}] &= \widetilde{A}\,\mathrm{Cov}\left[\mathbf{x}\right]\widetilde{A}^T + \mathrm{Cov}\left[\widetilde{\mathbf{w}}\right] \\ \mathbb{E}[\mathbf{y}] &= \widetilde{A}\,\mathbb{E}[\mathbf{x}] + \widetilde{\mathbf{b}} + \mathbb{E}[\widetilde{\mathbf{w}}], \end{aligned} \tag{8.44}$$

where $\mathrm{Cov}\left[\mathbf{x}\right]$ and $\mathbb{E}[\mathbf{x}]$ are already specified according to Equation 8.32. Therefore, given the moments in Equations 8.42 and 8.43, the three equations in Equation 8.44 can be used to determine $\widetilde{A}$, $\mathrm{Cov}[\widetilde{\mathbf{w}}]$ and $\widetilde{\mathbf{b}}$, respectively. Specifically, if we let the marginal density $p(\mathbf{x}) = \mathcal{D}\left(\mathbf{x}; Q_\mathbf{x}, R_\mathbf{x}, \Lambda_\mathbf{x}, \mathbf{h}_\mathbf{x}, \mathbf{c}_\mathbf{x}, g_\mathbf{x}\right)$, we can use the first part of Equation 8.44 to write

$$\mathrm{Cov}[\mathbf{x}, \mathbf{y}] = Q_\mathbf{x}\Lambda_\mathbf{x}^{-1}Q_\mathbf{x}^T\widetilde{A}^T \quad \Longrightarrow \quad Q_\mathbf{x}^T\widetilde{A}^T = \Lambda_\mathbf{x}Q_\mathbf{x}^T\mathrm{Cov}[\mathbf{x}, \mathbf{y}]. \tag{8.45}$$

By defining the decomposition $\widetilde{A} = \widetilde{A}_Q + \widetilde{A}_R$ such that

$$C\left(\widetilde{A}_Q^T\right) \subseteq C\left(Q_{\mathbf{x}}\right) \qquad \text{and} \qquad C\left(\widetilde{A}_R^T\right) \subseteq C\left(R_{\mathbf{x}}\right), \tag{8.46}$$

multiplying on both sides of Equation 8.45 by $Q_{\mathbf{x}}$ yields

$$Q_{\mathbf{x}} Q_{\mathbf{x}}^T \left(\widetilde{A}_Q^T + \widetilde{A}_R^T\right) = Q_{\mathbf{x}} \Lambda_{\mathbf{x}} Q_{\mathbf{x}}^T \mathrm{Cov}[\mathbf{x}, \mathbf{y}] \quad \Longrightarrow \quad \widetilde{A}_Q = \mathrm{Cov}\left[\mathbf{x}, \mathbf{y}\right]^T \mathrm{Cov}\left[\mathbf{x}\right]^+, \tag{8.47}$$

where we made use of the pseudo-inverse of the SVD for short. With this value of $\widetilde{A}_Q$ known, the second part of Equation 8.44 can be used to write

$$\begin{aligned} \mathrm{Cov}\left[\widetilde{\mathbf{w}}\right] &= \mathrm{Cov}\left[\mathbf{y}\right] - \left(\widetilde{A}_Q^T + \widetilde{A}_R^T\right) Q_{\mathbf{x}} \Lambda_{\mathbf{x}}^{-1} Q_{\mathbf{x}}^T \left(\widetilde{A}_Q^T + \widetilde{A}_R^T\right)^T \\ &= \mathrm{Cov}\left[\mathbf{y}\right] - \widetilde{A}_Q \,\mathrm{Cov}\left[\mathbf{x}\right] \widetilde{A}_Q^T, \end{aligned} \tag{8.48}$$

which does not depend on $\widetilde{A}_R$. Finally, the third part of Equation 8.44 can be used to write

$$\widetilde{\mathbf{b}} + \widetilde{A}_R \,\mathbb{E}[\mathbf{x}] + \mathbb{E}[\widetilde{\mathbf{w}}] = \mathbb{E}[\mathbf{y}] - \widetilde{A}_Q \,\mathbb{E}[\mathbf{x}]. \tag{8.49}$$

Since this is an underdetermined system of equations, we are free to choose $\widetilde{A}_R = 0$ and $\mathbb{E}[\widetilde{\mathbf{w}}] = \mathbf{0}$, without loss of generality. The former implies that $\widetilde{A} = \widetilde{A}_Q$ and the latter that $\widetilde{\mathbf{h}} = \mathbf{0}$ and $\widetilde{\mathbf{c}} = \mathbf{0}$. The remainder of the parameters $\widetilde{Q}$, $\widetilde{R}$, $\widetilde{\Lambda}$ and $\widetilde{g}$ can then be obtained by computing the compact SVD of Equation 8.48 and calculating the normalisation constant according to Equation 5.4. This concludes the derivation of Algorithm 7. □

8.3 Kullback-Leibler divergence

As mentioned in Section 2.1.3, inference on graphs containing loops is approximate. In addition, even for tree-structured graphs, approximations could be introduced due to linearisation. For message passing algorithms, this means that messages typically need to be computed iteratively until convergence. A popular method to check for convergence of such an algorithm is to use the Kullback-Leibler (KL) divergence

$$D_{\mathrm{KL}}(P||Q) \triangleq \int_{-\infty}^{\infty} p(\mathbf{x}) \log\left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right) \mathrm{d}\mathbf{x} = \mathbb{E}_{p(\mathbf{x})}\left[\log p(\mathbf{x}) - \log q(\mathbf{x})\right]. \tag{8.50}$$

This provides a measure of the relative entropy from density Q to density P and the condition

$$D_{\mathrm{KL}}(P||Q) = 0 \tag{8.51}$$

only holds when $p(\mathbf{x}) = q(\mathbf{x})$.

In the case of two degenerate Gaussian densities

$$p(\mathbf{x}) = \mathcal{D}(\mathbf{x}; Q_1, R_1, \Lambda_1, \mathbf{h}_1, \mathbf{c}_1, g_1) \quad \text{and} \quad q(\mathbf{x}) = \mathcal{D}(\mathbf{x}; Q_2, R_2, \Lambda_2, \mathbf{h}_2, \mathbf{c}_2, g_2), \tag{8.52}$$

the KL divergence can be calculated according to

$$\begin{aligned} D_{\mathrm{KL}}(P||Q) = \frac{1}{2}\Big(&\mathrm{tr}\left(Q_2 \Lambda_2 Q_2^T Q_1 \Lambda_1^{-1} Q_1^T\right) + \mathbf{h}_1^T \Lambda_1^{-1} Q_1^T Q_2 \Lambda_2 Q_2^T Q_1 \Lambda_1^{-1} \mathbf{h}_1 + \\ &\mathbf{h}_1^T \Lambda_1^{-1} \mathbf{h}_1 - n + k\Big) - \mathbf{h}_1^T \Lambda_1^{-1} Q_1^T Q_2 \mathbf{h}_2 + g_1 - g_2. \end{aligned} \tag{8.53}$$

Note, however, that this result is only valid when the two degenerate densities have support on the same lower-dimensional manifold, i.e., if $C(R_1) = C(R_2)$ and $R_1\mathbf{c}_1 = R_2\mathbf{c}_2$. Conversely, if these conditions are not satisfied, the KL divergence will instead be infinite. In the context of message passing algorithms, however, this should not be the case near convergence.

Proof. To calculate the KL divergence in Equation 8.50 for the two degenerate densities in Equation 8.52, we need to calculate the expectation

$$D_{\mathrm{KL}}(P||Q) = \mathbb{E}_{p(\mathbf{x})}\left[\log\frac{p(\mathbf{x})}{q(\mathbf{x})}\right]. \tag{8.54}$$

According to Algorithm 4, for the quotient of two degenerate densities

$$\frac{p(\mathbf{x})}{q(\mathbf{x})} = \mathcal{D}(\mathbf{x}; Q', R', \Lambda', \mathbf{h}', \mathbf{c}', g') \tag{8.55}$$

where $C(R_1) = C(R_2)$, $C(R') = \{\mathbf{0}\}$. The expectation in Equation 8.54 (with respect to the density P) therefore becomes

$$D_{\mathrm{KL}}(P||Q) = \mathbb{E}\left[\log\mathcal{C}(Q'^T\mathbf{x}; \Lambda', \mathbf{h}', g')\right] = -\frac{1}{2}\mathbb{E}\left[\mathbf{x}^T Q'\Lambda' Q'^T\mathbf{x}\right] + \mathbb{E}[\mathbf{x}]^T Q'\mathbf{h}' + g'. \tag{8.56}$$

Since the Dirac delta components of the two densities are equivalent, we are effectively only comparing their canonical components. The next step is to calculate the three terms in Equation 8.56. By using the results in Algorithm 4, and since the matrix Z is orthogonal and $C(Q_1) = C(Q_2)$, we can write

$$Q'\Lambda' Q'^T = Q_1(\Lambda_1 - Q_1^T Q_2\Lambda_2 Q_2^T Q_1)\,Q_1^T \quad \text{and} \quad Q'\mathbf{h}' = Q_1\mathbf{h}_1 - Q_2\mathbf{h}_2. \tag{8.57}$$

Since

$$\mathbb{E}\left[\mathbf{x}^T A\mathbf{x}\right] = \mathbb{E}[\mathbf{x}]^T A\,\mathbb{E}[\mathbf{x}] + \mathrm{tr}(A\,\mathrm{Cov}[\mathbf{x}]), \tag{8.58}$$

we use the first part of Equation 8.57 and the results in Equations 5.9 and 5.10 to calculate

$$\mathbb{E}[\mathbf{x}]^T Q'\Lambda' Q'^T\mathbb{E}[\mathbf{x}] = \mathbf{h}_1^T\Lambda_1^{-1}(\Lambda_1 - Q_1^T Q_2\Lambda_2 Q_2^T Q_1)\,\Lambda_1^{-1}\mathbf{h}_1 \tag{8.59}$$

and

$$\begin{aligned}
\mathrm{tr}(Q'\Lambda' Q'^T\mathrm{Cov}[\mathbf{x}]) &= \mathrm{tr}\left(Q_1(\Lambda_1 - Q_1^T Q_2\Lambda_2 Q_2^T Q_1)\,Q_1^T Q_1\Lambda_1^{-1}Q_1^T\right)\\
&= \mathrm{tr}\left(Q_1 Q_1^T - Q_1 Q_1^T Q_2\Lambda_2 Q_2^T Q_1\Lambda_1^{-1}Q_1^T\right)\\
&= \mathrm{tr}\left(Q_1 Q_1^T\right) - \mathrm{tr}\left(Q_2\Lambda_2 Q_2^T Q_1\Lambda_1^{-1}Q_1^T Q_1 Q_1^T\right)\\
&= n - k - \mathrm{tr}\left(Q_2\Lambda_2 Q_2^T Q_1\Lambda_1^{-1}Q_1^T\right).
\end{aligned} \tag{8.60}$$

In the second-to-last line of Equation 8.60 we used the fact that $\mathrm{tr}(AB) = \mathrm{tr}(BA)$ and in the last line that the trace of a projection matrix is equal to its rank. Next, by using the second part of Equation 8.57, we can write

$$\mathbb{E}[\mathbf{x}]^T Q'\mathbf{h}' = \left(Q_1\Lambda_1^{-1}\mathbf{h}_1 + R_1\mathbf{c}_1\right)^T(Q_1\mathbf{h}_1 - Q_2\mathbf{h}_2) = \mathbf{h}_1^T\Lambda_1^{-1}(\mathbf{h}_1 - Q_1^T Q_2\mathbf{h}_2). \tag{8.61}$$

Finally, since $g' = g_1 - g_2$, we can substitute Equations 8.59, 8.60 and 8.61 into Equation 8.56 to yield

$$D_{\mathrm{KL}}(P||Q) = \frac{1}{2}\Big(\mathrm{tr}\left(Q_2\Lambda_2 Q_2^T Q_1 \Lambda_1^{-1} Q_1^T\right) + \mathbf{h}_1^T \Lambda_1^{-1} Q_1^T Q_2 \Lambda_2 Q_2^T Q_1 \Lambda_1^{-1} \mathbf{h}_1 + \mathbf{h}_1^T \Lambda_1^{-1} \mathbf{h}_1 - n + k\Big) - \mathbf{h}_1^T \Lambda_1^{-1} Q_1^T Q_2 \mathbf{h}_2 + g_1 - g_2. \tag{8.62}$$

This concludes the derivation of the KL divergence in Equation 8.53. □

Together with the results from Chapter 6, this chapter provided the means to perform inference on Bayesian networks using degenerate Gaussian factors. This includes representing both linear and nonlinear models (as outlined in Algorithms 6 and 7) as well as practical aspects such as aligning factor scopes and checking for convergence of message passing algorithms. Note, however, that this does not limit the application of degenerate factors to Bayesian networks alone, where many of the results derived thus far can readily form part of more general inference processes. In the next chapter, we support our theoretical development by applying these results to a recursive state estimation problem.

Chapter 9

An example: State estimation for autonomous robots

To illustrate the advantages of performing inference with degenerate factors, we consider a representative example. More specifically, we demonstrate a scenario where it is essential to account for degeneracies (a) during the modelling process as well as (b) when performing subsequent computations. For this purpose, we first provide an overview of recursive state estimation – specifically in the context of mobile robotics. In order to keep the discussion pertinent yet not overly complex, we choose to model the estimation problem with a tree-structured graph. Since we perform inference on a nonlinear model using the belief propagation algorithm, all the operations outlined in Chapters 6 and 8 (except for division) are relevant. We also comment on the shortcomings of existing approaches using appropriate qualitative and quantitative results.

9.1 Recursive state estimation

Recursive state estimation (or Bayesian filtering) refers to a setting where the posterior distribution over a time-dependent latent state must be inferred using noisy measurements thereof [36]. Common applications are found in mobile robotics [25], where accurate control of a vehicle heavily relies on a good estimate of its state. For practical robotic systems, the estimation usually occurs online and in real time. Since regularisation techniques are often used to improve numerical robustness in certain applications [37, 38], this is an appropriate context for showcasing the advantages of degenerate Gaussian factors.

The standard Markovian formulation of the estimation problem consists of a motion and a measurement model. The *motion* model is a discrete-time, nonlinear function

$$\mathbf{x}_k = \mathbf{g}(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{w}_k) \tag{9.1}$$

relating the current state $\mathbf{x}_k$ to the previous state $\mathbf{x}_{k-1}$, the control inputs $\mathbf{u}_k$ and the process noise $\mathbf{w}_k$. In turn, the *measurement* model is a function

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{v}_k) \tag{9.2}$$

relating the measurement $\mathbf{z}_k$ to the state $\mathbf{x}_k$ and the measurement noise $\mathbf{v}_k$. Additionally, the prior distributions over the process noise $\mathbf{w}_k$ and measurement noise $\mathbf{v}_k$ are generally assumed

to be known. The aim of the estimator is then to calculate what Thrun et al. [25] call the *belief* for every time step $k \in \{1, 2, \dots K\}$. The belief is the posterior distribution

$$\text{bel}(\mathbf{x}_k) \triangleq p(\mathbf{x}_k | \mathbf{u}_{1:k}, \mathbf{z}_{1:k}) \tag{9.3}$$

over the state $\mathbf{x}_k$ given all the control inputs $\mathbf{u}_{1:k}$ and measurements $\mathbf{z}_{1:k}$ up to time k. Under the Markov assumption, the estimate in Equation 9.3 can be calculated recursively. The Bayesian network and subsequent factor graph representing this inference problem are shown in Figure 9.1, where we distinguish between motion factors

$$\psi_k(\mathbf{x}_{k-1}, \mathbf{x}_k, \mathbf{u}_k) \triangleq p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{u}_k) \tag{9.4}$$

and measurement factors

$$\rho_k(\mathbf{x}_k, \mathbf{z}_k) \triangleq p(\mathbf{z}_k | \mathbf{x}_k) \tag{9.5}$$

in the latter. Due to the nonlinear models in Equations 9.1 and 9.2, it is generally necessary to approximate the factors in Equations 9.4 and 9.5 using a linearisation technique such as the unscented transform.

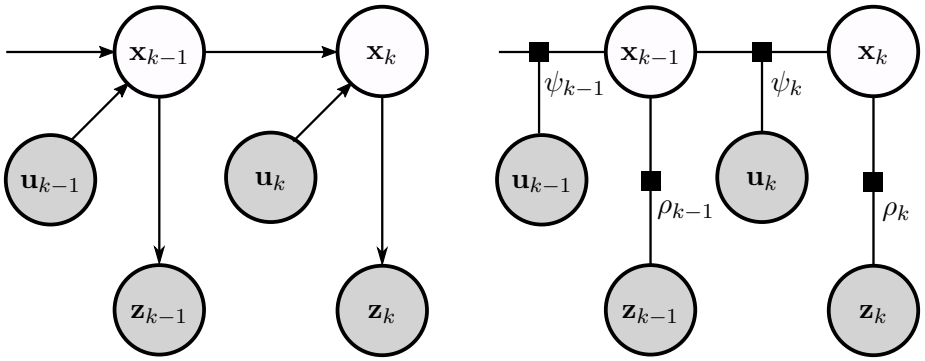


Figure 9.1: (a) A Bayesian network that models the recursive state estimation problem (as adapted from Thrun et al. [25]). Round nodes represent random variables (where a shaded node indicates that the variable is observed) and directed edges indicate causal dependencies. For each discrete time step (as indicated by the subscript k), the state $\mathbf{x}_k$ is latent while the control inputs $\mathbf{u}_k$ and measurement $\mathbf{z}_k$ are observed. (b) The corresponding factor graph. Black squares indicate factors, where each factor is connected to all the random variables in its scope via undirected edges. We distinguish between motion factors ψ_k and measurement factors ρ_k.

To solve the estimation problem (i.e., to calculate the belief over every state $\mathbf{x}_k$), we subsequently construct the cluster graph shown in Figure 9.2. Note, however, that this choice of cluster graph is not unique. According to the general definition of an outgoing message in

Equation 2.1, the rightward, leftward, downward and upward messages are

$$\begin{aligned}
\xi_k^{\rightarrow}(\mathbf{x}_k) &= \int \psi_k(\mathbf{x}_{k-1}, \mathbf{x}_k, \mathbf{u}_k)\, \xi_{k-1}^{\rightarrow}(\mathbf{x}_{k-1})\, \xi_k^{\uparrow}(\mathbf{x}_k)\, \mathrm{d}\mathbf{x}_{k-1} \\
\xi_k^{\leftarrow}(\mathbf{x}_k) &= \int \psi_{k+1}(\mathbf{x}_k, \mathbf{x}_{k+1}, \mathbf{u}_{k+1})\, \xi_{k+1}^{\leftarrow}(\mathbf{x}_{k+1})\, \xi_{k+1}^{\uparrow}(\mathbf{x}_{k+1})\, \mathrm{d}\mathbf{x}_{k+1} \\
\xi_k^{\downarrow}(\mathbf{x}_k) &= \int \psi_k(\mathbf{x}_{k-1}, \mathbf{x}_k, \mathbf{u}_k)\, \xi_{k-1}^{\rightarrow}(\mathbf{x}_{k-1})\, \xi_k^{\leftarrow}(\mathbf{x}_k)\, \mathrm{d}\mathbf{x}_{k-1} \\
\xi_k^{\uparrow}(\mathbf{x}_k) &= \rho_k(\mathbf{x}_k, \mathbf{z}_k),
\end{aligned} \tag{9.6}$$

respectively. The computations in the first three cases each make use of two multiplication operations followed by a single marginalisation operation. In addition, all four computations make use of a reduction operation based on either the known control inputs $\mathbf{u}_k$ or the observed measurement $\mathbf{z}_k$. Once all of the messages have been computed (usually through iteration until convergence), the posterior distribution

$$p(\mathbf{x}_k|\mathbf{u}_{1:K}, \mathbf{z}_{1:K}) = \xi_k^{\rightarrow}(\mathbf{x}_k)\, \xi_k^{\leftarrow}(\mathbf{x}_k) \tag{9.7}$$

can be determined for each time step. Note the slight difference in conditioning between Equation 9.3 (known as filtering) and Equation 9.7 (known as smoothing). Although the former inference problem can also be solved using Equations 9.6 and 9.7 by simply omitting all leftward messages $\xi_k^{\leftarrow}$, we will only consider the latter going forward as it is (a) usually more accurate and (b) a more natural context for PGMs.

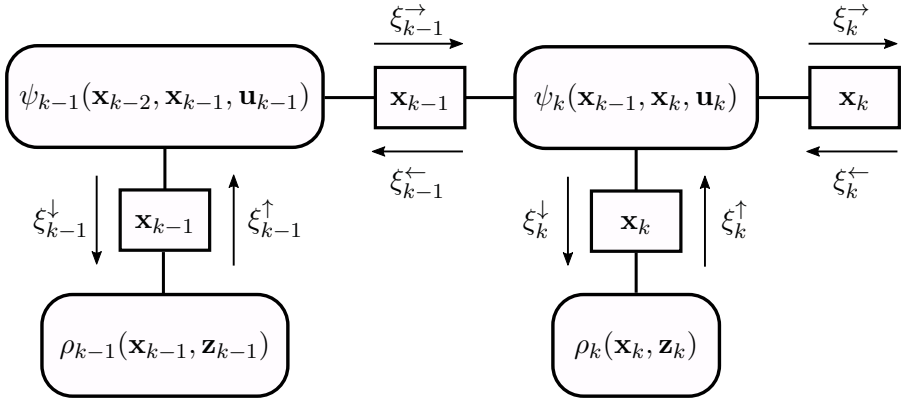


Figure 9.2: A possible cluster graph corresponding to the model of the recursive state estimation problem in Figure 9.1, where each factor has been placed in its own cluster. Each cluster potential is therefore equal to the appropriate motion or measurement factor. The sepsets are indicated by rectangular nodes and are connected to the clusters via undirected edges. Messages are indicated by arrows and are differentiated (by their labels) according to both their scopes and their directions.

Even though the computations in Equations 9.6 and 9.7 are valid for any factor representation, the advantages of Gaussian factors (for example the fact that they are closed under the required operations) make them a popular approximation in robotics applications. The most natural context that further warrants the use of *degenerate* Gaussian factors is one where the

models in Equations 9.1 and 9.2 include deterministic relationships between a subset of the variables. Note, however, that if a given degeneracy is present for all time steps, it could be sufficient to construct an equivalent, lower-dimensional model of the system without requiring degenerate factors. An appropriate example to illustrate the advantages of degenerate factors is therefore one with time-dependent models, where degeneracies arise inconsistently and unpredictably.

9.2 Cooperative transportation robots

For our illustrative example, we draw inspiration from the practical system by Loianno and Kumar [39], where they develop a fleet of micro aerial vehicles (MAVs) that can transport a rigid body using permanent electromagnets. If the geometry of the transported object is assumed to be known a priori, they show that this additional information can aid the localisation of the vehicles by formulating the estimation problem as an optimisation problem. In this section, we show how such additional information can be incorporated automatically into the well-established state estimation formulation (outlined in the previous section) using degenerate Gaussian factors. To keep the dimensionality manageable, however, we limit our discussion to ground vehicles with three degrees of freedom.

In particular, consider a scenario where a fleet of mobile robots need to work together to transport objects on a warehouse floor. Suppose that these objects can vary in shape and size, and consequently multiple and varying subsets of robots cooperate at any given time. Since each robot operates independently for the majority of the time, we use the nonlinear odometry motion model

$$\mathbf{x}_k^i = \begin{bmatrix} x_k^i \\ y_k^i \\ \theta_k^i \end{bmatrix} = \begin{bmatrix} x_{k-1}^i + r_k^i \cos\left(\theta_{k-1}^i + \alpha_k^i\right) \\ y_{k-1}^i + r_k^i \sin\left(\theta_{k-1}^i + \alpha_k^i\right) \\ \theta_{k-1}^i + \alpha_k^i + \beta_k^i \end{bmatrix} + \mathbf{w}_k^i \tag{9.8}$$

as proposed by Thrun et al. [25], where the i'th robot has position (x_k^i, y_k^i) and orientation θ_k^i and where the corresponding control inputs comprise a rotation α_k^i followed by a translation r_k^i and another rotation β_k^i. Suppose that each robot also receives a noisy measurement of its position

$$\mathbf{z}_k^i = \begin{bmatrix} x_k^i \\ y_k^i \end{bmatrix} + \mathbf{v}_k^i \tag{9.9}$$

at every time step and that the noise distributions

$$p(\mathbf{w}_k^i) = \mathcal{N}(\mathbf{w}_k^i; \mathbf{0}, \Sigma_{\mathbf{w}}) \qquad \text{and} \qquad p(\mathbf{v}_k^i) = \mathcal{N}(\mathbf{v}_k^i; \mathbf{0}, \Sigma_{\mathbf{v}}) \tag{9.10}$$

are Gaussian. To keep this model consistent with the one in Figure 9.1, we then combine the states (and similarly the measurements and controls) of all the individual robots into a single state vector $\mathbf{x}_k$ (as well as measurement vector $\mathbf{z}_k$ and control vector $\mathbf{u}_k$).

Now consider a moment in time k' when a group of N robots $\{j_1, j_2, \ldots j_N\}$ are transporting an object. If the shape of the object is known, this provides additional information that would be useful for estimating the states of the robots. In this example, we assume that the distances between these robots as well as their relative orientations are specified exactly[1].

[1] For the sake of simplicity, we only consider translations of the object (as opposed to rotations as well).

This additional information can therefore be included in a noiseless *auxiliary* measurement $\hat{\mathbf{z}}_{k'}$, where its n'th component is given by

$$\hat{\mathbf{z}}_{k'}^{n} = \begin{bmatrix} \sqrt{\left(x_{k'}^{j_n} - x_{k'}^{j_{n-1}}\right)^2 + \left(y_{k'}^{j_n} - y_{k'}^{j_{n-1}}\right)^2} \\ \theta_{k'}^{j_n} - \theta_{k'}^{j_{n-1}} \end{bmatrix}. \tag{9.11}$$

The *augmented* measurement model for time step k' then appends all of the nonlinear auxiliary measurements in Equation 9.11 to the default measurements in Equation 9.9 to produce a single measurement vector $\mathbf{z}_{k'}$.

During physical operation, each robot will receive its odometry information and sensor measurements as normal. If this were naively used to perform state estimation separately for each robot, the correlation between cooperating robots could not be utilised to improve the accuracy of the estimated beliefs. Instead, all the robot poses should be combined into a single state vector and auxiliary measurements of the form in Equation 9.11 included where applicable. Note that, although such a measurement is expressed in terms of the robot states, this is not a causal relationship. This equation is merely used to construct the necessary conditional densities as required by the model in Figure 9.1. The actual observations are instead determined by the geometry of the particular object and will be the same at every time step for the duration of the transportation task. Omitting the auxiliary measurements in Equation 9.11 altogether would be equivalent to ignoring the additional information provided by the known shape of the object.

A closer look at the necessary inference operations (as outlined in Section 9.1) reveals the need for computing with degenerate factors in this example. Since the augmented measurement model contains noiseless components, the covariance of the measurement noise will be rank deficient and consequently the measurement factor $\rho_{k'}$ in Equation 9.5 cannot be represented using non-degenerate parametrisations. We therefore need to represent the noise distribution using a degenerate factor as in Equation 8.6. Furthermore, the upward message $\xi_{k'}^{\uparrow}$ as computed in Equation 9.6 will also be degenerate and will in turn be used to compute other messages (for example $\xi_{k'}^{\rightarrow}$ and $\xi_{k'-1}^{\leftarrow}$). This requires the reduction, multiplication and marginalisation operations as outlined in Chapter 6. Even the context for linearising the motion factor $\psi_{k'+1}$ – typically the message $\xi_{k'}^{\rightarrow}$ – will be degenerate. This reveals that, once degeneracies arise during inference, it is necessary that *all* of the downstream operations handle such cases appropriately.

9.3 Experiments and results

To support this argument, the transportation task in Section 9.2 was simulated, where generated control inputs and sampled noise values were used to calculate the trajectories and measurements of the robots. The control inputs and measurements were then used to calculate the messages in Figure 9.2 until convergence. As an example, the beliefs for a group of three robots and a single object are shown in Figure 9.3. By computing with degenerate factors, we were able to handle cases where the motion of the three robots are independent (first and last 10 time steps) as well as highly-correlated (middle 20 time steps) – both automatically and without encountering numerical errors. In comparison, using ridge regularisation (i.e., adding a small scalar value λ to the diagonal terms of the singular covariance matrix)

as an approximate solution resulted in more uncertain beliefs. This was also the case when ignoring the additional information provided by the known shape of the object entirely.

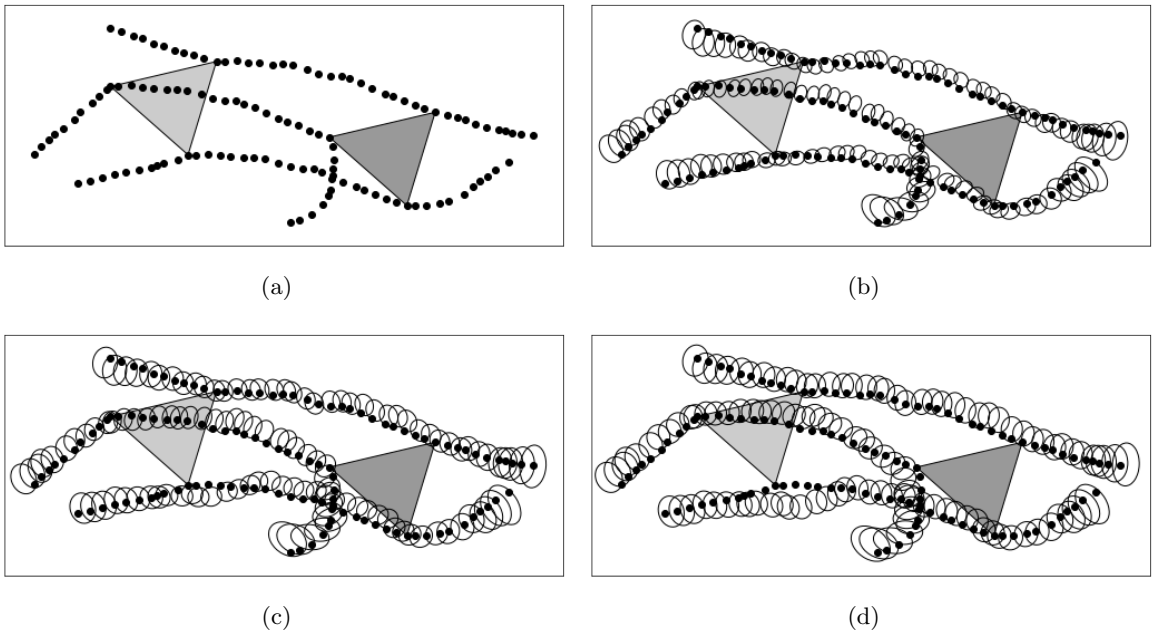


Figure 9.3: (a) Three robots transporting a triangular object (from left to right) on a 2-D warehouse floor, where the object's initial and final positions are indicated by the lighter and darker triangles, respectively. At each time step, the robots' actual positions are represented with solid dots. The control inputs and measurements are then used to perform state estimation for this transportation task when (b) using degenerate factors, (c) using canonical factors with ridge regularisation and (d) ignoring the known shape of the object. In each case, the inferred beliefs are indicated using 67% confidence ellipses. Note that these ellipses are smaller in (b) compared to (c) and (d), as seen by the separation between the trajectories (specifically near the top right vertex of the lighter triangle and the left vertex of the darker triangle).

For repeated experiments, Figure 9.4 shows the trade-off between high accuracy for small regularisation values and well-conditioned matrices for larger values. Although the critical point when too large a condition number results in numerical errors depends on the machine precision and other application-specific details, it is worth noting that even for this example there is a limited range of regularisation values that achieve acceptable accuracy without affecting the conditioning of the problem adversely. This is in contrast to our principled solution using degenerate factors, where the condition numbers are determined by the problem definition alone and not increased by unnecessary approximations.

Since the degenerate parametrisation proposed by Raphael [18] does not include a procedure for approximating nonlinear models, we need to convert to (and from) our parametrisation to enable a comparison. The more significant drawback of the former, however, is the absence of normalisation constants due to their use of indicator functions (as opposed to our use of Dirac delta functions) for representing degeneracies. The implication is that,

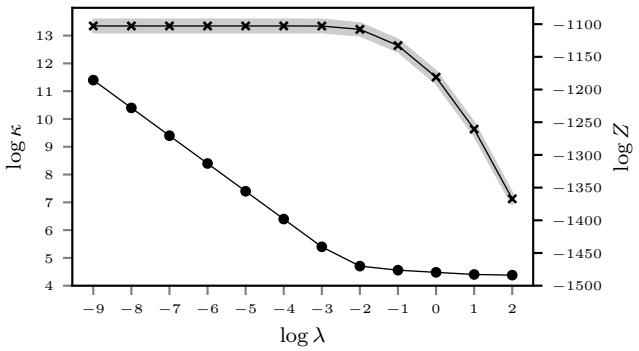


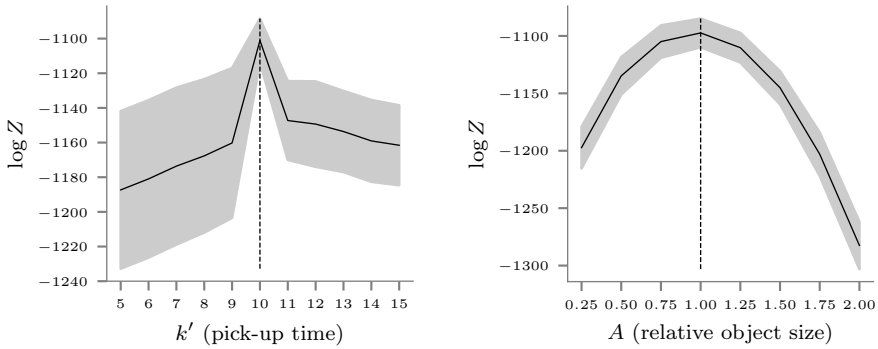
Figure 9.4: The effect of ridge regularisation (for varying values of λ) on the maximum condition number κ (indicated by the line with circular markers) as well as on the model likelihood Z (indicated by the line with x-shaped markers). The standard deviation for the latter (based on multiple experiments with different control inputs and noise values) are indicated by the shaded region.

although the first- and second-order moments of the computed beliefs are equivalent in both cases, model comparison is only possible with our parametrisation (where the messages remain unnormalised).

To illustrate this advantage, suppose that the time when the object was picked up is unknown but of interest. By considering multiple hypotheses (i.e., one for each time step), performing inference for each model and then extracting the model likelihood using the normalisation information, we are able to identify the correct model. This is shown in Figure 9.5(a), where the peak at $k' = 10$ corresponds to the true time step when the object was picked up in Figure 9.3. Similarly, Figure 9.5(b) illustrates an alternative context where the transported object could be any one out of a finite set of candidates and the relative size of the object is correctly identified (as $A = 1$) using the model likelihood.

As a final comparison of our methodology to both ridge regularisation and Raphael's factor representation, we investigate the overall execution times for inferring the posterior belief over the robots' trajectories. Recall from Section 7.3 that this provides an important practical perspective on the computational cost of such algorithms or solutions. In the case of ridge regularisation, the use of Koller and Friedman's [3] canonical factors resulted in the shortest execution times, as expected. This approach required an average of 0.67 seconds to solve the entire inference problem using the same implementation and hardware as described in Chapter 7. Next, using the parametrisation for degenerate factors by Raphael [18] took 1.58 seconds on average. In some cases, however, this resulted in numerical errors when using the Cholesky decomposition to draw the necessary sigma points. This reveals that, although these factors can be used to approximate nonlinear models, the use of a *diagonal* precision matrix as in Equation 5.1 improves the numerical stability significantly, since its inverse and square root can be computed in an element-wise manner.

Finally, using our degenerate Gaussian factors to solve this recursive state estimation problem required 1.51 seconds on average. Although this is only slightly faster than Raphael's representation, recall that the normalisation constants are explicitly kept track of throughout the former, but not computed at all in the latter. For a fair comparison, we can therefore adapt our solution by omitting the final line in each of Algorithms 2 to 7. Consequently, if we are only



(a) (b)

Figure 9.5: Model comparison for the state estimation problem in Figure 9.3 to determine (a) the time step k' when the object was picked up and (b) the size A of the object (relative to the true object's size). In both cases, the mean and standard deviation of the model likelihood Z are indicated by the solid line and shaded region, respectively. The peak where the model likelihood is a maximum (and therefore corresponding to the most likely model) is indicated by the dashed line.

interested in the first- and second-order moments of the computed beliefs, the execution time drops to 1.34 seconds on average. In summary, ridge regularisation therefore had the shortest execution time, but requires approximations in degenerate cases. Raphael's representation can accommodate degeneracies exactly, but cannot be used for model comparison and required the longest execution time. As illustrated, using our parametrised factors instead enjoys the advantages of both while requiring more time than the former, but less than the latter.

Chapter 10

Conclusion

Probabilistic inference involving Gaussian factors is only valid for positive-definite covariance matrices. In positive *semi*-definite settings, linear dependencies among the random variables instead warrant the explicit representation of generalised *degenerate* Gaussian factors. In this chapter, we evaluate the development in this dissertation according to the research aims and objectives as outlined in Chapter 1. We also highlight the most significant properties and capabilities of our original solution. Finally, we discuss possible avenues for future research.

10.1 Evaluation of degenerate Gaussian factors

In brief, the aim of this research project was to extend the capabilities of Gaussian factors to degenerate settings, without resorting to approximations or suffering from over-parametrisation. Consequently, it was necessary to propose an appropriate representation for such degenerate factors, derive the typical statistical operations under this methodology and illustrate its advantages using a representative example. These are the three research objectives as outlined in Section 1.2.

To address the problem of performing inference in degenerate settings, we introduced a parametrisation comprising a lower-dimensional, non-degenerate component as well as a Dirac delta function describing possible degeneracies in Chapter 5. The definition of the Dirac delta as the limit of a Gaussian distribution appropriately captures the idea that the variance tends to zero in certain dimensions. We also derived the conditions for the general factor to be a valid density function as well as the first- and second-order moments in such cases. In addition, our definition can still express non-degenerate Gaussian distributions as a special case and furthermore provides a principled way of handling rank-deficient transformations of Gaussian random variables.

Next, we derived algorithms for the marginalisation, multiplication, division and reduction of degenerate Gaussian factors from first principles in Chapter 6. This mirrors the results for non-degenerate, canonical factors by Koller and Friedman [3] which are necessary for inference algorithms such as belief propagation [22] and belief update [23]. Importantly, all of these results can once again be expressed as parametrised degenerate Gaussian factors and, through the use of Dirac delta functions, the normalisation constant can be computed throughout. In Chapter 7, we showed that the computational complexity for performing inference using these factors is at most $\mathcal{O}(n^3)$.

To enable inference on Bayesian networks as outlined in Section 2.1, we also derived additional results for practical operations such as extending and rearranging factor scopes in Chapter 8. Together with modelling stochastic systems by representing conditional densities and checking for convergence of message passing algorithms, these results were necessary to illustrate the advantages of computing with degenerate factors as demonstrated by the example in Chapter 9.

10.2 Original contributions

Concretely, we highlight the following original contributions made in this dissertation:

1. The definition of the degenerate Gaussian factor in Equation 5.1: Different from the work by Raphael [18], our use of a diagonal precision matrix (a) reduces the number of parameters required to represent and compute with degenerate factors and (b) improves the numerical stability of inference in degenerate settings, without sacrificing any of the capabilities of Raphael's representation. Furthermore, our use of a Dirac delta function allows us to keep track of normalisation constants explicitly, which in turn enables model comparison in degenerate cases.

2. The statistical operations on degenerate factors as outlined in Algorithms 2 to 5: Since our definition of degenerate factors is different from any existing literature, these needed to be derived from first principles. Unlike ridge regularisation, these operations enable inference in degenerate setting without resorting to approximations. This also goes beyond the work by Mikheev [17], where the Dirac delta component was eventually omitted and the statistical operations could not be derived as a result. As demonstrated by the numerical results in Sections 7.3 and 9.3, using our methodology also resulted in faster overall execution times compared to the solution by Raphael [18].

3. The procedure for affine transformations of degenerate random variables as outlined in Algorithm 1: This not only extends another capability of non-degenerate Gaussian parametrisations to degenerate settings, but provides a principled way of handling rank-deficient transformations of non-degenerate random variables as well. In contrast, the resulting singular covariance matrix (due to linear dependencies within the transformation) cannot be expressed using traditional parametrisations.

4. The linearisation of nonlinear models via the unscented transform when the prior distribution is degenerate as outlined in Algorithm 7: Specifically, drawing sigma points from the lower-dimensional distribution avoids positive-definite constraints due to degeneracies. This would not be possible if the Cholesky decomposition was instead used on the full covariance matrix.

In summary, these contributions enable accurate and automatic inference in (possibly) degenerate settings at little additional computational cost. In contrast, approximate solutions sacrifice either accuracy or numerical stability as demonstrated by the recursive state estimation example.

10.3 Future work

In this dissertation, we used the context of probabilistic graphical models to motivate and illustrate our methodology for solving inference problems with degenerate components. However, most of the derived operations (such as marginalisation) and properties (such as the statistical moments) apply to other domains as well. Future work should therefore investigate other applications of Gaussian random variables – for example techniques such as Gaussian mixture models, Gaussian processes or Kalman filters – where it would be advantageous to compute with explicit degenerate factors. This will typically be the case when the inference problem and covariance matrices are ill-conditioned.

Although this was not the only motivation for computing with degenerate factors, we only considered an example in this dissertation that included deterministic constraints as part of its problem definition. As mentioned in Chapter 1, another potential situation where this could be beneficial is where inference problems become near-degenerate due to machine precision limitations. By developing a principled strategy or heuristic for introducing artificial degeneracies based on an appropriate threshold, the numerical stability of inference with ill-conditioned matrices could be improved significantly.

Finally, by utilising the lower-dimensional, diagonal precision matrix of our representation as part of an optimised implementation, future work could reduce the computational cost of performing inference on models with a significant degree of degeneracy. For example, although mathematically equivalent, there exist multiple algorithms for computing the singular value decomposition, where some could be able to exploit certain matrix properties that are present in Algorithms 2 to 5. With such an optimised implementation, degenerate Gaussian factors can be applied to large-scale problems – both to provide insight into the scalability of this methodology and to ultimately aid practical, real-world applications.

Bibliography

[1] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, pp. 35–45, 03 1960.

[2] R. D. Shachter and C. R. Kenley, "Gaussian influence diagrams," *Management Science*, vol. 35, no. 5, pp. 527–550, 1989.

[3] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques (Adaptive Computation and Machine Learning series)*. The MIT Press, 2009.

[4] P. Z. Peebles, *Probability, random variables, and random signal principles*. McGraw Hill, 1987.

[5] S. L. Lauritzen and F. Jensen, "Stable local computation with conditional Gaussian distributions," *Statistics and Computing*, vol. 11, no. 2, pp. 191–203, 2001.

[6] R. Fitzgerald, "Divergence of the Kalman filter," *IEEE Transactions on Automatic Control*, vol. 16, no. 6, pp. 736–747, 1971.

[7] R. Pawula, S. Rice, and J. Roberts, "Distribution of the phase angle between two vectors perturbed by Gaussian noise," *IEEE Transactions on Communications*, vol. 30, no. 8, pp. 1828–1841, 1982.

[8] Q. Cao, C. Shen, and M. Jia, "A fault detection scheme for PV panels in large scale PV stations with complex installation conditions," *arXiv preprint arXiv:2105.08943*, 2021.

[9] J. Quinonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate Gaussian process regression," *The Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.

[10] C. Biernacki and S. Chrétien, "Degeneracy in the maximum likelihood estimation of univariate Gaussian mixtures with EM," *Statistics & probability letters*, vol. 61, no. 4, pp. 373–382, 2003.

[11] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[12] D. I. Warton, "Penalized normal likelihood and ridge regularization of correlation and covariance matrices," *Journal of the American Statistical Association*, vol. 103, no. 481, pp. 340–349, 2008.

[13] C. Cortes, M. Mohri, and A. Talwalkar, "On the impact of kernel approximation on learning accuracy," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 113–120, JMLR Workshop and Conference Proceedings, 2010.

[14] G. H. Golub, M. Heath, and G. Wahba, "Generalized cross-validation as a method for choosing a good ridge parameter," *Technometrics*, vol. 21, no. 2, pp. 215–223, 1979.

[15] N. E. Heckman and J. O. Ramsay, "Penalized regression with model-based penalties," *Canadian Journal of Statistics*, vol. 28, no. 2, pp. 241–258, 2000.

[16] D. W. Marquardt and R. D. Snee, "Ridge regression in practice," *The American Statistician*, vol. 29, no. 1, pp. 3–20, 1975.

[17] P. Mikheev, "Multidimensional Gaussian probability density and its applications in the degenerate case," *Radiophysics and Quantum Electronics*, vol. 49, no. 7, pp. 564–571, 2006.

[18] C. Raphael, "Bayesian networks with degenerate Gaussian distributions," *Methodology and Computing in Applied Probability*, vol. 5, no. 2, pp. 235–263, 2003.

[19] D. Barber, *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[20] J. Pearl, "Fusion, propagation, and structuring in belief networks," *Artificial Intelligence*, vol. 29, no. 3, pp. 241–288, 1986.

[21] R. Kindermann, *Markov random fields and their applications*. American Mathematical Society, 1980.

[22] P. P. Shenoy and G. Shafer, "Propagating belief functions with local computations," *IEEE Expert*, vol. 1, no. 3, pp. 43–52, 1986.

[23] S. L. Lauritzen and D. J. Spiegelhalter, "Local computations with probabilities on graphical structures and their application to expert systems," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 50, no. 2, pp. 157–194, 1988.

[24] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," in *Signal Processing, Sensor Fusion, and Target Recognition VI*, vol. 3068, pp. 182 – 193, International Society for Optics and Photonics, 1997.

[25] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. Cambridge, Mass.: MIT Press, 2005.

[26] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," 2000.

[27] G. Strang, *Introduction to Linear Algebra*. Wellesley, MA: Wellesley-Cambridge Press, fourth ed., 2009.

[28] G. W. Stewart, "On the early history of the singular value decomposition," *SIAM Review*, vol. 35, no. 4, pp. 551–566, 1993.

[29] P. A. M. Dirac, *The principles of quantum mechanics*. No. 27, Oxford University Press, 1981.

[30] B. P. Lathi, *Modern digital and analog communication systems*. Oxford University Press, 1998.

[31] R. Shankar, *Principles of quantum mechanics*. Springer Science & Business Media, 2012.

[32] T. P. Minka, "Expectation propagation for approximate Bayesian inference," in *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pp. 362–369, 2001.

[33] V. Strassen, "Gaussian elimination is not optimal," *Numerische Mathematik*, vol. 13, no. 4, pp. 354–356, 1969.

[34] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," in *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, p. 1–6, 1987.

[35] G. H. Golub and C. F. Van Loan, *Matrix computations*, vol. 3. JHU Press, 2013.

[36] F. Schweppe, "Recursive state estimation: Unknown but bounded errors and system inputs," *IEEE Transactions on Automatic Control*, vol. 13, no. 1, pp. 22–28, 1968.

[37] T. Koshizen, P. Bartlett, and A. Zelinsky, "Sensor fusion of odometry and sonar sensors by the Gaussian mixture Bayes' technique in mobile robot position estimation," in *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 99CH37028)*, vol. 4, pp. 742–747, IEEE, 1999.

[38] R. S. Inoue, M. H. Terra, and J. P. Cerri, "Extended robust Kalman filter for attitude estimation," *IET Control Theory & Applications*, vol. 10, no. 2, pp. 162–172, 2016.

[39] G. Loianno and V. Kumar, "Cooperative transportation using small quadrotors using monocular vision and inertial sensing," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 680–687, 2017.