



UNIVERSITY OF  
CAMBRIDGE

Department of Engineering

## A theoretical bound for noise-robust speech recognition

R. C. van Dalen  
rcv25@cam.ac.uk

M. J. F. Gales  
mjfg@eng.cam.ac.uk

Technical Report CUED/F-INFENG/TR.648

18th September 2010

---

### Abstract

Model compensation techniques for noise-robust speech recognition approximate the corrupted speech distribution. This work introduces a sampling method that, given speech and noise distributions and a mismatch function, in the limit calculates the corrupted speech likelihood exactly. For this, it transforms the integral in the likelihood expression, and then applies sequential importance resampling. Though it is too slow to compensate a speech recognition system, it enables a more fine-grained assessment of compensation techniques, based on the KL divergences to the ideal compensation for individual components. The KL divergence appears to predict the word error rate well.

This technique also makes it possible to evaluate the impact of approximations that compensation schemes make. For example, this work examines the influence of the assumption that the corrupted speech distribution is Gaussian and diagonalising that Gaussian's covariance. It also assesses the impact of a common approximation to the mismatch function for VTS compensation, namely setting the *phase factor* to a fixed value.

---



---

## Contents

1	Introduction . . . . .	3
2	The effect of the noise . . . . .	5
2.1	Assumptions . . . . .	5
2.2	The mismatch function . . . . .	6
2.2.1	Derivation of the mismatch function . . . . .	7
2.2.1.1	The properties of $\alpha_i$ . . . . .	9
2.2.1.2	The distribution of $\alpha_i$ . . . . .	10
2.3	Corrupted speech distribution . . . . .	12
2.3.1	Sampling from the corrupted speech distribution . . . . .	12
3	Parametric representations for the corrupted speech distribution . . . . .	12
3.1	Data-driven parallel model combination . . . . .	13
3.2	Vector Taylor series approximation . . . . .	14
3.3	Algonquin . . . . .	16
3.4	Piecewise linear approximation . . . . .	21
4	Multi-dimensional integration with importance sampling . . . . .	22
4.1	Monte Carlo . . . . .	23
4.1.1	Importance sampling . . . . .	23
4.2	Sequential importance sampling . . . . .	25
4.3	Re-sampling . . . . .	28
4.4	Sampling from the target distribution . . . . .	29
5	Asymptotically exact likelihood evaluation . . . . .	30
5.1	Importance sampling from a Gaussian . . . . .	31
5.2	Transformed-space sampling . . . . .	34
5.2.1	Single-dimensional . . . . .	34
5.2.1.1	The shape of the integrand . . . . .	37
5.2.1.2	Importance sampling from $\gamma(u \alpha)$ . . . . .	41
5.2.2	Multi-dimensional . . . . .	43
5.2.2.1	Per-dimension sampling . . . . .	46
5.2.2.2	Postponed factorisation . . . . .	47
5.2.2.3	Quasi-conditional factorisation . . . . .	48
5.2.2.4	Applying sequential importance re-sampling . . . . .	49
6	Distance to the actual distribution . . . . .	50
7	Experiments . . . . .	52
7.1	Set-up . . . . .	52
7.2	Results . . . . .	54
7.2.1	Compensation methods . . . . .	55

CONTENTS

---

	7.2.2	Diagonal-covariance compensation . . . . .	57
	7.2.3	Influence of the phase factor . . . . .	59
8		Conclusion . . . . .	61
A		Well-known equalities . . . . .	62
	A.1	Transforming variables of probability distributions . . . . .	62
	A.2	Matrix identities . . . . .	62
	A.3	Multi-variate Gaussian factorisation . . . . .	62
B		Domain-specific mismatch function . . . . .	64
C		Transforming the likelihood for piecewise linear approximation . . . . .	66
	C.1	Single-dimensional . . . . .	66
	C.2	Multi-dimensional . . . . .	67
D		Transforming the likelihood for transformed-space sampling . . . . .	69
	D.1	Terms of the integrand for transformed-space sampling . . . . .	69
	D.2	Transforming the space of the multi-dimensional integral . . . . .	71
	D.3	Postponed factorisation of the integrand . . . . .	74
	D.4	Terms of the proposal distribution . . . . .	76

## 1 Introduction

Changes in background noise conditions can severely impact the performance of speech recognition systems. This is caused by the mismatch between training and testing conditions. There are two categories of approaches for dealing with this mismatch. Feature enhancement reconstructs the clean speech before decoding. Model compensation, on the other hand, replaces a speech recogniser's distributions over clean speech by ones over noise-corrupted speech. However, given standard speech and noise distributions, the corrupted speech distribution has no closed form. Model compensation normally approximates it with a parametric distribution, such as a Gaussian. In contrast, this work will endeavour to approximate the real corrupted speech distribution as exactly as possible. The interest in this can be motivated from a theoretical and a practical perspective.

Conceptually, a speech recogniser is a classifier that takes an observation sequence as input and classifies it as belonging to one of a set of word sequences. The Bayes decision rule for classification says that the best choice for labelling observations is the one with the highest probability. The probability is factorised into the prior probability of the label, and the likelihood, the probability of the observations given the label. If the prior and the likelihood are the true ones, then the Bayes decision rule produces the optimal classification. Speech recognisers essentially implement this rule.

Assuming that the speech and noise distributions are the true ones, decoding with the exact distributions for the corrupted speech would therefore yield the best recogniser performance. The objective of feature enhancement, reconstructing the clean speech, may be useful where the clean speech is required, for example, as a preprocessing step before passing the signal to humans. However, as a preprocessing step for speech recognition, it gives no guarantees about optimality.<sup>1</sup> It therefore makes more sense to find accurate model compensation than feature enhancement.

A practical perspective comes from considering methods for noise-robustness as speech recogniser adaptation. Given a labelling, adaptation estimates a transformation to the function that assigns a likelihood to an observation. In effect, this retrains the speech recogniser on the observations, but on so little data and with such great uncertainty about the transcriptions that the parameter space needs to be constrained for robustness. Generic adaptation methods such as CMLLR [10] use linear transformations of model parameters, making as few assumptions about the mismatch as possible. From the perspective of adaptation, the difference between linear adaptation and methods for noise robustness is how the speech recogniser's parameter space is constrained. Methods for noise-robustness are parametrised through the distribution of the noise. The noise distribution usually has even fewer parameters than linear adaptation methods use. Therefore, adaptation for noise needs much less adaptation data (see e.g.[7] for a comparison). The reason for this is that the assumptions about the relationship of the speech, the noise and the observations mirror the true mismatch closely enough. In practice, therefore, it turns out that better modelling of the mismatch yields better and faster adaptation to noise.

Theory and practice, then, suggest that exact modelling of the noise-corrupted

---

<sup>1</sup> This is not to say that transforming feature before they enter the recogniser based on a different criterion than reconstructing the clean speech could not yield the best performance. Indeed, a performance-oriented implementation of this concept, transforming features to compensate speech recogniser models, has shown good results [33].

speech yields the optimal noise-robustness. However, considering methods for noise-robustness as adaptation also lends itself to the opposite argument. The distribution of the noise is estimated, e.g. with maximum likelihood estimation. This means it does not necessarily represent the actual noise, nor does it have to. Some of the difference between the workings of the model and the real environment can be absorbed by the parameters. Indeed, some work [23, 14] has found that tweaking the model arbitrarily can yield improvements in performance if the noise distribution is estimated for that model. This may have been an implicit argument in work proposing specific approximations to the model of the environment [13, 2]. However, previous work has not quantified the loss in performance from these approximations. This is only possible by comparing with the exact corrupted speech distribution.

This work will use a number of standard assumptions (specified in more detail in section 2). The speech and noise will be assumed Gaussian distributed. The *mismatch function*, which describes how the features of the speech, noise, and corrupted speech are related, will have a relatively standard form. Only additive noise will be considered, because convolutional noise is usually modelled with an offset to the feature vectors in the log-spectral domain, which would only convolute notation. To ensure independence from noise estimation schemes, background noise will be added to the speech audio artificially, and the noise model trained directly trained on the noise audio.

A relatively obscure aspect of the mismatch function is the uncertainty that arises from effects that speech recogniser features ignore but become important when the signal results from two sources (speech and noise in this case) that are additive in the frequency domain. There has recently been interest in how to model this in a mismatch function [4, 21], and this work will use a state-of-the-art model.

Because the relationship between speech, noise, and observations is non-linear, there exists no closed form for the distribution of the noise-corrupted speech even when speech and noise distributions are given. Model compensation methods often approximate the corrupted speech distribution parametrically, for example, with a Gaussian (section 3). However, the corrupted speech density only needs to be evaluated at points given by the observations. This work will find a method that approximates the real likelihood arbitrarily closely (section 5). It approximates the integral that the likelihood expression contains with a Monte Carlo algorithm (section 4). In coming close to the real likelihood it therefore becomes so slow that implementing a speech recogniser with it is infeasible.

It is, however, possible to make a much more fine-grained assessment of speech recogniser compensation, with a metric based on the  $\text{KL}$  divergence (section 6). In the limit, the new sampling methods will effectively give the point where the  $\text{KL}$  divergence is zero, which is otherwise not known. This calibration will make it possible to determine how far well-known compensation methods are from the ideal.

This work will examine how well the  $\text{KL}$  divergence predicts speech recogniser word error rate. It will compare different compensation schemes, and examine the effect of common approximations (section 7). This includes assuming the corrupted speech distribution Gaussian and diagonalising its covariance matrix, and approximations to the mismatch function.

## 2 The effect of the noise

This section lays down the model of the environment that this work will use: the speech and noise distributions, and how the speech and noise interact to form the corrupted speech. The assumptions here are standard for state-of-the-art speech recogniser robustness, though the model for the *phase factor* is more sophisticated.

### 2.1 Assumptions

Methods for noise robustness make assumptions about what the speech and the noise are like. Background noise generally alters the way people speak, because people aim to help the listener overcome the extra problems that the noise causes for decoding the meaning of the speaker. The change in people’s voice quality because of noise is known as the “Lombard effect” and it will be ignored in this work. Acoustic distortion can be divided into background noise, which is additive in the time domain, and channel noise, which is convolutional in the time domain. The convolutional noise will be left out in this work, because it would confuse notation, and assuming that it is known, it is straightforwardly modelled.

Many speech recognisers use feature vectors in the cepstral domain. Cepstral features are related to log-spectral features by the discrete cosine transform (DCT), which is a linear transformation. The reason the cepstral domain is often preferred is that the DCT goes a long way to decorrelating the features within a feature vector. However, correlations do not fully disappear, and especially under noisy conditions it becomes important to model them correctly [12]. For the purpose of finding the best possible distribution for the observations, therefore, it does not matter whether the models are in the log-spectral or in the cepstral domain: feature correlations need to be modelled anyway.

For the purpose of modelling the interaction between speech, noise, and observations, however, the log-spectral domain has an advantage: the interaction is per dimension. Therefore, this work will use the log-spectral domain throughout for the theory, and model feature correlations explicitly.

Speech recognisers also use dynamic coefficients, which indicate how coefficients change over time. They are computed as linear combinations of the static coefficients in a window. Feature enhancement can transform features either before or after adding dynamics. For model compensation methods, there is no choice: they have to compensate dynamic parameters. Often, extra approximations are used, such as the continuous time approximation for VTS compensation. However, it appears that the best strategy is to compensate the statics in the window separately [34, 35]. This work will therefore focus on static parameters.

For the recognition experiments, models will be in the cepstral domain. To generalise the compensation methods to dynamic parameters, they will be applied to *extended* feature vectors [34, 35]. This avoids additional approximations, and yields the best performance.

The model for the clean speech is normally extracted from a trained speech recogniser. Speech recognisers have a mixture of Gaussians associated with each state (which usually represents a sub-phone unit). Model compensation methods normally work on one Gaussian at a time, or sometimes one state-conditional mixture of Gaussians. A

model compensation technique called joint uncertainty decoding [25] represents a base class of speech recogniser Gaussians by one Gaussian for computational efficiency. It performs compensation for each base class and then applies the same transformation to all Gaussians in one base class. Model-based feature enhancement [6, 32] uses a simplified version of the speech recogniser, a mixture of Gaussians without sequence structure. In all of these cases, the basic unit of the clean speech model is a Gaussian. This work will therefore focus on that level, and represent the distribution of the clean speech  $\mathbf{x}$  by

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x), \quad (1)$$

where the parameter space is log-spectral and the covariance  $\boldsymbol{\Sigma}_x$  is full. The dependence on the component identity will not be written.

The noise model is often expressed in the same domain as the speech recogniser parameters. It usually has a Gaussian for the additive noise. Whether one Gaussian without time structure is a good model of the additive noise depends on the type of noise. However, in practice the parameters of the noise model have to be estimated, so that a simple model is an advantage. In this work, the noise model is therefore represented as

$$\mathbf{n} \sim \mathcal{N}(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n), \quad (2)$$

with known log-spectral domain parameters, and  $\boldsymbol{\Sigma}_n$  full.

## 2.2 The mismatch function

The relationship between the corrupted speech, the clean speech and the noise is central to noise-robust speech recognition. The term *mismatch function* is often used for the function that takes the speech and noise signals and returns the corrupted speech signal. Since this work uses log-spectral coefficients, the mismatch will be expressed in the log-spectral domain. The background noise is added to the speech in the time domain, but turn out more complicated in the log-spectral domain. The speech and the additive noise signals are symmetric in their influence on the corrupted speech signal, which the equations given here will bring out.

The relationship between the speech vector  $\mathbf{x}$ , the noise  $\mathbf{n}$ , and the observation  $\mathbf{y}$  will be derived and discussed in great detail in section 2.2.1. In the log-spectral domain, it will be modelled

$$\mathbf{exp}(\mathbf{y}) = \mathbf{exp}(\mathbf{x}) + \mathbf{exp}(\mathbf{n}) + 2\boldsymbol{\alpha} \circ \mathbf{exp}\left(\frac{1}{2}\mathbf{x} + \frac{1}{2}\mathbf{n}\right), \quad (3a)$$

where  $\mathbf{exp}(\cdot)$  and  $\circ$  denote element-wise exponentiation and multiplication. To express the observation as a function of the speech and noise, using  $\mathbf{log}(\cdot)$  for the element-wise logarithm,

$$\begin{aligned} \mathbf{y} &= \mathbf{log}\left(\mathbf{exp}(\mathbf{x}) + \mathbf{exp}(\mathbf{n}) + 2\boldsymbol{\alpha} \circ \mathbf{exp}\left(\frac{1}{2}\mathbf{x} + \frac{1}{2}\mathbf{n}\right)\right) \\ &\triangleq \mathbf{f}(\mathbf{x}, \mathbf{n}, \boldsymbol{\alpha}). \end{aligned} \quad (3b)$$

These forms will be used in this work because the symmetry between the clean speech and the additive noise will be an important consideration. However, (3b) is often rewritten to bring out the effect of the noise on the clean speech:

$$\mathbf{y} = \mathbf{x} + \mathbf{log}\left(1 + \mathbf{exp}(\mathbf{n} - \mathbf{x}) + 2\boldsymbol{\alpha} \circ \mathbf{exp}\left(\frac{1}{2}\mathbf{n} - \frac{1}{2}\mathbf{x}\right)\right). \quad (3c)$$

In all cases,  $\alpha$  is the *phase factor*, which arises from the phase difference between the two signals that are added (speech and noise). The phase information is discarded in the conversion to the log-spectral domain, so that with speech and noise models in that domain, the phase factor  $\alpha$  is a random vector, the distribution of which will be discussed below.

Before going into the details of how this mismatch function comes about, it is worth identifying a split in the literature on choosing the form of the mismatch function. Traditionally [13, 28, 2], the phase factor term has been ignored. Some papers [4, 21] have gone into mathematical depth to get as close as possible to the real mismatch. For example, they show the elements of  $\alpha$  to be between  $-1$  and  $+1$ .

Other papers have been more pragmatic. After all, model compensation for noise robustness is just a form of adaptation to the data. The parameters that in theory make up the noise model are usually estimated from the data, with the aim to maximise the likelihood of the adapted model. The difference between traditional linear transformation methods and methods for model compensation is therefore the space to which the adapted model are constrained. In both cases, it could be argued that the best choice for this space is the one that yields the lowest word error rate, which is not necessarily the mathematically correct one. The mismatch function is one element that determines this space.

In this vein, it is possible to show that the optimal value for the phase factor is the mathematically impossible  $\alpha_i = 2.5$  for the AURORA 2 corpus [24]. What has really happened is that the mismatch function has been tuned. The possibility of tuning the mismatch function to assume the features use a specific power of the spectrum (e.g. the magnitude or power spectrum) had been noted before [13]. Setting the phase factor to 2.5 has a very similar effect to assuming the power of the spectrum used to be 0.75 [14]. This parameter setting improves performance for AURORA 2, but not for other corpora. This illustrates that tweaking arbitrary parameters in the mismatch function, in effect adjusting the space in which the optimal adapted model is sought, can have an impact on word error rate in some cases.

The question that this work will try to answer, however, is how good the theoretically best noise compensation is, and it will therefore aim for mathematical correctness in the first instance. Thus, the following derivation of the mismatch function in the log-power-spectral domain will be as exact as possible.

### 2.2.1 Derivation of the mismatch function

In many practical cases, observations, clean speech models and a noise model are available in the cepstral or in the log-spectral domain. The following derivation therefore relates log-spectral vectors for the observation, the clean speech, and the additive noise. The power spectrum will be assumed here; appendix B generalises it to other factors. It follows [4, 21, 22].

In the time domain, the relationship between the corrupted speech  $y[k]$ , the clean speech  $x[k]$ , and the additive noise  $n[k]$  is simply [1]

$$y[k] = x[k] + n[k], \quad (4)$$

which in the frequency domain (after applying a Fourier transform) becomes a relation between complex numbers:

$$Y[k] = X[k] + N[k]. \quad (5)$$

To find the power spectrum, the absolute value of this complex value is squared:

$$\begin{aligned} |Y[k]|^2 &= |X[k] + N[k]|^2 \\ &= |X[k]|^2 + |N[k]|^2 + 2X[k]N[k] \cos \theta_k, \end{aligned} \quad (6)$$

where  $\theta_k$  is the angle in the complex plane between  $X[k]$  and  $N[k]$ . This relates to the phase difference at frequency  $k$  between the clean speech and the noise. Since there is no process in speech production that synchronises the phase to background noise, this angle is uniformly distributed, so that the expected value of the cosine of the angle is

$$\mathcal{E} \{ \cos \theta_k \} = 0. \quad (7)$$

This is often used as an argument to drop the cross-term of the speech and the noise in (6). Though not dropping this term complicates the derivation, the point here is to get as close to the real mismatch as possible, so (6) will be used as is.

To extract coefficients for speech recognition, the next step is to reduce the number of coefficients, by applying  $I$  filter bins to the power-spectral coefficients. There are usually 24 triangular bins. Let  $w_{ik}$  specify the contribution of the  $k$ th frequency to the  $i$ th bin. The mel-filtered power spectrum is then given by coefficients  $\bar{Y}_i$ :

$$\bar{Y}_i = \sum_k w_{ik} |Y[k]|^2 = \sum_k w_{ik} \left( |X[k]|^2 + |N[k]|^2 + 2 |X[k]| |N[k]| \cos \theta_k \right). \quad (8)$$

This value of the mel-filtered power spectrum for the corrupted speech can almost be defined in terms of values of the clean speech and the additive noise in the same domain:

$$\bar{X}_i = \sum_k w_{ik} |X[k]|^2; \quad (9a)$$

$$\bar{N}_i = \sum_k w_{ik} |N[k]|^2. \quad (9b)$$

Using these to replace the terms in the right-hand side of (8),

$$\bar{Y}_i = \sum_k w_{ik} |Y[k]|^2 = \bar{X}_i + \bar{N}_i + 2\alpha_i \sqrt{\bar{X}_i \bar{N}_i}, \quad (9c)$$

where  $\alpha_i$  has been defined as

$$\alpha_i \triangleq \frac{\sum_k w_{ik} |X[k]| |N[k]| \cos \theta_k}{\sqrt{\bar{X}_i \bar{N}_i}}. \quad (9d)$$

The definition of  $\alpha_i$  seems to be defined specifically to hide the term with coefficients in the magnitude spectrum  $X[k]N[k]$ . However, the next subsections will find properties of  $\alpha_i$  that are independent of the spectra of  $X[k]$ : first, that  $\alpha_i$  is constrained to  $[-1, +1]$ , and then that its distribution is approximately Gaussian.

The mel-power-spectral coefficients are usually converted to their logarithms so that  $y_i = \log(\tilde{Y}_i)$ ,  $x_i = \log(\tilde{X}_i)$ , and  $n_i = \log(\tilde{N}_i)$ . The mismatch expression in the log-spectral domain trivially becomes

$$\exp(y_i) = \exp(x_i) + \exp(n_i) + 2\alpha_i \exp\left(\frac{1}{2}x_i + \frac{1}{2}n_i\right). \quad (10)$$

The vector version of this is in (3a). Appendix B generalises this to features that use a different power, for example, the magnitude spectrum.

**2.2.1.1 The properties of  $\alpha_i$**  When  $\alpha_i$  was defined in (9d), it seemed arbitrary. However, two important observations about  $\alpha_i$  can be made.

First, it is possible to determine the range of  $\alpha_i$  [4]. Since a cosine is constrained to  $[-1, 1]$ , the following inequality holds:

$$|\alpha_i| \leq \frac{\sum_k w_{ik} |X[k]| |N[k]|}{\sqrt{\tilde{X}_i \tilde{N}_i}}. \quad (11)$$

It is possible to write the fraction in (11) as a normalised inner product of two vectors. The vectors are  $\tilde{X}_i$  and  $\tilde{N}_i$ , with entries

$$\tilde{X}_{ik} = \sqrt{w_{ik}} |X[k]|; \quad (12a)$$

$$\tilde{N}_{ik} = \sqrt{w_{ik}} |N[k]|. \quad (12b)$$

Then,  $\sqrt{\tilde{X}_i \tilde{N}_i}$  in (11) can be written as the norm of  $\tilde{X}_i$ , so that

$$\frac{\sum_k w_{ik} |X[k]| |N[k]|}{\sqrt{\tilde{X}_i \tilde{N}_i}} = \frac{\sum_k \sqrt{w_{ik}} |X[k]| \sqrt{w_{ik}} |N[k]|}{\|\tilde{X}_i\| \|\tilde{N}_i\|} = \frac{\tilde{X}_i^T \tilde{N}_i}{\|\tilde{X}_i\| \|\tilde{N}_i\|}, \quad (13)$$

which is a normalised inner product of two vectors with non-negative entries, which is always in  $[0, 1]$ . The inequality in (11) then shows that  $\alpha_i$  is constrained to  $[-1, 1]$ :

$$|\alpha_i| \leq \frac{\tilde{X}_i^T \tilde{N}_i}{\|\tilde{X}_i\| \|\tilde{N}_i\|} \leq 1. \quad (14)$$

Second, an approximation can decouple the distribution of  $\alpha_i$  from the values of  $|X[k]|$  and  $|N[k]|$  [21]. Removing magnitude-spectral terms from the equation is useful because they are not usually modelled explicitly in speech recognisers. The assumption is that for one frequency bin  $i$ , all  $|X[k]|$  have the same value, and similar for all  $|N[k]|$ . Since the bins overlap, this is known to be exactly true only if  $|X[k]|$  is equal for all  $k$ . However, since the bins are relatively narrow, it may be a reasonable approximation. By dividing both sides of the fraction in (9d) by  $|X[k]|$  and  $|N[k]|$ , assuming that they are the same for all  $k$ ,  $\alpha_i$  is approximated as

$$\begin{aligned} \alpha_i &= \frac{\sum_k w_{ik} |X[k]| |N[k]| \cos \theta_k}{\sqrt{\sum_k w_{ik} |X[k]|^2 \sum_k w_{ik} |N[k]|^2}} \\ &\simeq \frac{\sum_k w_{ik} \cos \theta_k}{\sqrt{(\sum_k w_{ik}) (\sum_k w_{ik})}} = \frac{\sum_k w_{ik} \cos \theta_k}{\sum_k w_{ik}}, \end{aligned} \quad (15)$$

$\alpha_i$  can thus be approximated as a weighted average of cosines over independently distributed uniform variables  $\theta_k$ . The distribution that this model produces is very close to the empirical distribution on various combinations of noises and signal-to-noise ratios on the AURORA 2 corpus [21].

---

**procedure** DRAW- $\alpha_i$ -SAMPLE( $i$ )  
**for** frequency  $k$  for which  $w_{ik} \neq 0$  **do**  
  sample  $\theta_k \sim \text{Unif}[-\pi, +\pi]$ ;  
  compute  $v_k = \cos \theta_k$ .  
Compute the sample  $\alpha_i = \frac{\sum_k w_{ik} v_k}{\sum_k w_{ik}}$ .

---

**Algorithm 1** Drawing a sample from  $p(\alpha_i)$ .

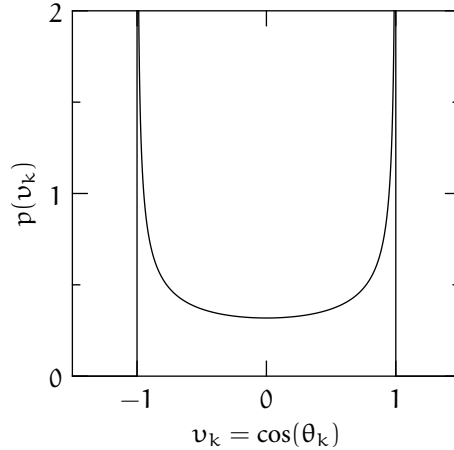
**2.2.1.2 The distribution of  $\alpha_i$**  The distribution of  $\alpha_i$  is most easily viewed by sampling from it. Drawing a sample is straightforward if three assumptions are used. Since there is no process in speech production that synchronises the speech phase with the noise phase at a specific frequency, the phase  $\theta_k$  is uniformly distributed. Also, the phase is assumed independently distributed for different frequencies  $k$ . Thirdly, the distribution of  $\alpha_i$  is approximated as in (15), removing the influence of particular value of the speech and noise signals.

Algorithm 1 shows how to sample from  $\alpha_i$ . Samples for  $\theta_k$  are drawn independently for all frequencies in one mel-bin. This straightforwardly yield samples for  $\cos \theta_k$ , which will be called  $v_k$ . The sample for  $p(\alpha_i)$  can be acquired by taking the weighted average over these.

It is also possible to find a parametric distribution of  $v_k = \cos \theta_k$ . It can be shown to be [21, 22]

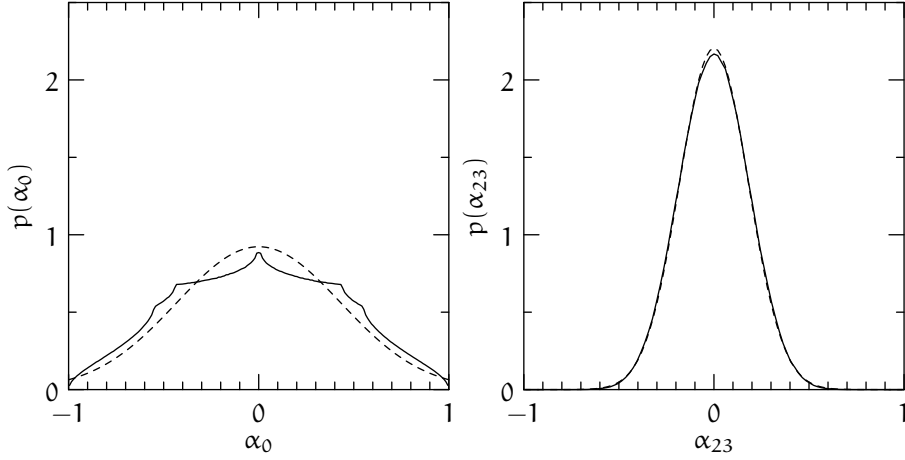
$$p(v_k) = \begin{cases} \frac{1}{\pi\sqrt{1-v_k^2}}, & |v_k| \leq 1, \\ 0, & \text{otherwise.} \end{cases} \quad (16)$$

This distribution is pictured in figure 1.



**Figure 1** The distribution of  $v_k = \cos \theta_k$  for one frequency  $k$ .

The distribution of  $\alpha_i$  is a weighted average of distributions over  $v_k$ 's. As the number of frequencies goes up, the central limit theorem says the distribution of  $\alpha_i$  becomes



(a) Bin 0 is the narrowest, so that  $\alpha_0$  has the least Gaussian-like distribution.

(b) Bin 23 is the widest, so that  $\alpha_{23}$  has the most Gaussian-like distribution.

**Figure 2** The distribution of  $\alpha_i$  for different mel-filter channels  $i$  (—), and their Gaussian approximations (— —).

closer to a Gaussian [4]. For lower-frequency bins, the number of frequencies that is summed over is smaller, so the distribution of  $\alpha_i$  is expected to be further away from a Gaussian. This effect can be seen in figure 2, which shows the distributions for two values of  $i$ . These distributions were found by sampling many times from  $\alpha_i$  using algorithm 1 on the preceding page. The dashed lines show Gaussian approximations. For  $\alpha_0$ , the Gaussian is least appropriate, but still a reasonable approximation. In this work, the distribution of  $\alpha_i$  will therefore be assumed Gaussian for each bin  $i$ .

The covariance of the Gaussian can be set to the second moment of the real distribution. It can be shown that, again assuming that all spectral coefficients in one filter bin are equal, that [21]

$$\sigma_{\alpha,i}^2 \triangleq \mathcal{E} \{ \alpha_i^2 \} = \frac{\sum_k w_{ik}^2}{2(\sum_k w_{ik})^2}. \quad (17)$$

This gives values very close to the actual variance of  $\alpha_i$  on various subsets of AURORA 2. This work will therefore approximate the distribution of  $\alpha_i$  as a Gaussian with

$$p(\alpha_i) \propto \begin{cases} \mathcal{N}(\alpha_i; 0, \sigma_{\alpha,i}^2) & \alpha_i \in [-1, +1]; \\ 0 & \text{otherwise.} \end{cases} \quad (18)$$

Evaluating the density at any point requires the normalisation constant  $1 / \int_{-1}^{+1} \mathcal{N}(\alpha; 0, \sigma^2) d\alpha$ , which could be approximated with an approximation to the Gaussian's cumulative distribution function. However, it is straightforward to draw samples from this distribution, by sampling from the Gaussian and rejecting any samples not in  $[-1, +1]$ .

### 2.3 Corrupted speech distribution

With the mismatch function  $\mathbf{f}$  in (3b), the observation vector is known if the vectors for the speech, noise, and the phase factor are known. If the distributions of the three inputs,  $p(\mathbf{x})$ ,  $p(\mathbf{n})$ , and  $p(\boldsymbol{\alpha})$ , are known, then the observation distribution can be trivially described using a Dirac delta at the point given by the mismatch function  $\mathbf{f}$ :

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \quad (19a)$$

$$= \int \int p(\mathbf{y}|\mathbf{x}, \mathbf{n}) p(\mathbf{n}) d\mathbf{n} p(\mathbf{x}) d\mathbf{x} \quad (19b)$$

$$= \int \int \int p(\mathbf{y}|\mathbf{x}, \mathbf{n}, \boldsymbol{\alpha}) p(\boldsymbol{\alpha}) d\boldsymbol{\alpha} p(\mathbf{n}) d\mathbf{n} p(\mathbf{x}) d\mathbf{x} \quad (19c)$$

$$= \int \int \int \delta_{\mathbf{f}(\mathbf{x}, \mathbf{n}, \boldsymbol{\alpha})}(\mathbf{y}) p(\boldsymbol{\alpha}) d\boldsymbol{\alpha} p(\mathbf{n}) d\mathbf{n} p(\mathbf{x}) d\mathbf{x}. \quad (19d)$$

This expression is exact, but it does not have a closed form. (It is still valid if  $\boldsymbol{\alpha}$  is fixed: then  $p(\boldsymbol{\alpha})$  is a Dirac delta at the fixed value.) Since the mismatch function  $\mathbf{f}$  is non-linear, there is no closed form for the distribution of  $\mathbf{y}$ . This work will introduce a transformation to the integral in (19d) that enables approximating it with a Monte Carlo method. Standard model compensation methods, however, approximate this distribution with a parametrical representation, for example a Gaussian. Each clean speech Gaussian can then be replaced by the corrupted speech Gaussian found using the parameters of the original Gaussian as statistics. When that is done, decoding can be as fast as normal.

#### 2.3.1 Sampling from the corrupted speech distribution

Though expressing the corrupted speech distribution parametrically is impossible, it is straightforward to draw samples  $\mathbf{y}^{(s)}$  from the distribution. This works by applying the mismatch function to samples from the distributions for  $\mathbf{x}$ ,  $\mathbf{n}$ , and  $\boldsymbol{\alpha}$ :<sup>2</sup>

$$\mathbf{x}^{(s)} \sim p(\mathbf{x}); \quad (20a)$$

$$\mathbf{n}^{(s)} \sim p(\mathbf{n}); \quad (20b)$$

$$\boldsymbol{\alpha}^{(s)} \sim p(\boldsymbol{\alpha}); \quad (20c)$$

$$\mathbf{y}^{(s)} = \mathbf{f}(\mathbf{x}^{(s)}, \mathbf{n}^{(s)}, \boldsymbol{\alpha}^{(s)}). \quad (20d)$$

Sampling from the corrupted speech distribution will be used in this work to train parametric distributions (DPMC and IDPMC), and to examine how well approximated distributions match the actual distribution.

## 3 Parametric representations for the corrupted speech distribution

There exist several methods for approximating the corrupted speech distribution. Finding a parametric form naturally appears as a subtask in model compensation, which

<sup>2</sup>This can be seen as a straightforward instantiation of Monte Carlo sampling, which section 4.1 will discuss.

replaces state-conditional clean speech distributions by estimated corrupted speech (“compensated”) distributions. This section will discuss various compensation methods.

Two schemes map one clean speech Gaussian to one corrupted speech Gaussian. VTS compensation (section 3.2) is a standard method that applies a first-order vector Taylor series approximation to the mismatch function, so that the corrupted speech becomes Gaussian. DPMC compensation (section 3.1) is a sampling approach that only makes the Gaussian approximation at the last instance, by training the Gaussian on samples. Iterative DPMC also uses samples, but trains a mixture of Gaussians with expectation–maximisation.

Two other schemes do not use a fixed parametric representation: they start the computation only when the observation has been seen. The Algonquin algorithm (section 3.3) extends the VTS approximation by iteratively updating the expansion point. It comes up with a different Gaussian for each clean speech Gaussian for each observation. It is also possible to approximate the integral over the speech and noise using a piecewise linear approximation (section 3.4).

### 3.1 Data-driven parallel model combination

Data-driven parallel model combination (DPMC) [13] finds a Gaussian distribution for the corrupted speech, with a Monte Carlo method. It approximates the distributions with samples and applies the correct mismatch function. The Gaussian assumption is made only when training the parameters on the samples. In the limit, it finds the optimal Gaussian distribution for the corrupted speech.

The original algorithm did not use phase factor  $\alpha$ ; however, the generalisation is straightforward. DPMC can be derived by approximating the integral over  $\mathbf{x}$ ,  $\mathbf{n}$ , and  $\alpha$  in (19d) by a sum over samples  $\mathbf{x}^{(s)} \sim p(\mathbf{x})$ ,  $\mathbf{n}^{(s)} \sim p(\mathbf{n})$ ,  $\alpha^{(s)} \sim p(\alpha)$ :

$$\begin{aligned} p(\mathbf{y}) &= \int \int \int \delta_{f(\mathbf{x}, \mathbf{n}, \alpha)}(\mathbf{y}) p(\alpha) d\alpha p(\mathbf{n}) d\mathbf{n} p(\mathbf{x}) d\mathbf{x} \\ &\simeq \frac{1}{S} \sum_{s=1}^S \delta_{f(\mathbf{x}^{(s)}, \mathbf{n}^{(s)}, \alpha^{(s)})}(\mathbf{y}) \triangleq \tilde{p}(\mathbf{y}). \end{aligned} \quad (21)$$

$\tilde{p}(\mathbf{y})$  is the empirical distribution, which has  $S$  delta spikes at positions  $\mathbf{y}^{(s)}$  obtained as detailed in (20d). The parametric distribution for the corrupted speech that DPMC finds is chosen to minimise the KL divergence with the empirical distribution  $\tilde{p}$ . This is equivalent to training the parametric distribution from the samples with maximum likelihood. Standard DPMC finds a Gaussian distribution for the corrupted speech. Its mean and covariance parameters are the mean and covariance of  $\mathbf{y}$  under the empirical distribution:

$$\boldsymbol{\mu}_y = \mathcal{E}_{\tilde{p}}\{\mathbf{y}\} = \frac{1}{S} \sum_{s=1}^S \mathbf{y}^{(s)}; \quad (22a)$$

$$\boldsymbol{\Sigma}_y = \mathcal{E}_{\tilde{p}}\{\mathbf{y}\mathbf{y}^T\} - \boldsymbol{\mu}_y \boldsymbol{\mu}_y^T = \frac{1}{S} \sum_{s=1}^S \mathbf{y}^{(s)} \mathbf{y}^{(s)T} - \boldsymbol{\mu}_y \boldsymbol{\mu}_y^T. \quad (22b)$$

Given a mismatch function and distributions for the speech, noise, and phase factor, this gives the optimal Gaussian parameters.

For the experiments, not only static, but also dynamic parameters need compensation. This work will use distributions over a window of consecutive static feature vectors, *extended* feature vectors [34, 35]. The mismatch function is applied to each time instance in the vector separately. This produces a corrupted-speech sample that can be converted to a sample with static and dynamic coefficients with the linear transformation that speech recogniser front-ends normally use. Training of the distribution works in the same way.

Iterative DPMC (IDPMC) also finds a parametric distribution that is close to the empirical distribution, but the distribution is a mixture of Gaussians associated with a speech recogniser state rather than a single Gaussian. The word “iterative” in the name of the scheme refers to the iterations of expectation–maximisation necessary to train a mixture of Gaussians. To draw corrupted speech samples, the same procedure applies as in section 2.3.1 on page 12. The clean speech model can be a state-conditional GMM. The corrupted speech GMM is then trained on the samples, without reference to the clean models, and is not restricted to have the same number of components as the clean speech GMM. At every iteration of expectation–maximisation, the parameters of components  $1, \dots, M$  are re-estimated using the maximum-likelihood estimates given the component–sample posteriors:

$$P(m|\mathbf{y}^{(s)}) \propto P(m)(\mathbf{y}^{(s)}|m); \quad (23a)$$

$$\boldsymbol{\mu}_y^{(m)} = \frac{1}{\sum_s P(m|\mathbf{y}^{(s)})} \sum_s P(m|\mathbf{y}^{(s)}) \mathbf{y}^{(s)}; \quad (23b)$$

$$\boldsymbol{\Sigma}_y^{(m)} = \frac{1}{\sum_s P(m|\mathbf{y}^{(s)})} \sum_s P(m|\mathbf{y}^{(s)}) \mathbf{y}^{(s)} \mathbf{y}^{(s)\top} - \boldsymbol{\mu}_y \boldsymbol{\mu}_y^\top. \quad (23c)$$

To increase the number of components, mixing up can be used: the heaviest component is split into two components with an offset, and a few iterations of expectation–maximisation training are run.

In the limit as the number of components  $M$  and the number of samples  $S$  go to infinity, the modelled distribution of the corrupted speech becomes equal to the real one. However, to train the mixture model well, the mean and covariance of each component need to be trained on a large enough portion of the samples. Therefore, the  $S$  must be kept roughly proportional to  $M$ , and the number of iterations of mixing up and expectation–maximisation increases linearly with  $M$  as well. In practice, then, the time complexity of IDPMC is  $\mathcal{O}(M^3)$ .

### 3.2 Vector Taylor series approximation

vts compensation [28, 2, 4] is a standard method for model compensation. vts compensation linearises the mismatch function  $\mathbf{f}$  in (3b). The most important result of this is that, given Gaussians for the clean speech, the noise, and the phase factor, the noise-corrupted speech also becomes Gaussian.

The first-order vector Taylor series approximation to the mismatch function  $\mathbf{f}$  with expansion point  $(\mathbf{x}_0, \mathbf{n}_0, \boldsymbol{\alpha}_0)$  is

$$\mathbf{f}_{\text{vts}}(\mathbf{x}, \mathbf{n}, \boldsymbol{\alpha}) = \mathbf{f}(\mathbf{x}_0, \mathbf{n}_0, \boldsymbol{\alpha}_0) + \mathbf{J}_x(\mathbf{x} - \mathbf{x}_0) + \mathbf{J}_n(\mathbf{n} - \mathbf{n}_0) + \mathbf{J}_\alpha(\boldsymbol{\alpha} - \boldsymbol{\alpha}_0), \quad (24a)$$

where the Jacobians for the clean speech, additive noise, and phase factor are

$$\mathbf{J}_x = \left. \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \right|_{\mathbf{x}_0, \mathbf{n}_0, \alpha_0}; \quad \mathbf{J}_n = \left. \frac{\partial \mathbf{y}}{\partial \mathbf{n}} \right|_{\mathbf{x}_0, \mathbf{n}_0, \alpha_0}; \quad \mathbf{J}_\alpha = \left. \frac{\partial \mathbf{y}}{\partial \alpha} \right|_{\mathbf{x}_0, \mathbf{n}_0, \alpha_0}. \quad (24b)$$

Appendix B gives expressions for these.

$\alpha$  is approximately Gaussian distributed but constrained to  $[-1, +1]$  (see section 2.2.1.2). To simplify the distribution of  $\mathbf{y}$ , the constraint can be ignored, so that  $\alpha \sim \mathcal{N}(\mathbf{0}, \Sigma_\alpha)$ .  $\mathbf{f}_{\text{vts}}$  in (24a) then is a sum of linearly transformed independently Gaussian distributed variables. These are also Gaussian. For example, for the clean speech:

$$\begin{aligned} \mathbf{x} &\sim \mathcal{N}(\boldsymbol{\mu}_x, \Sigma_x); \\ \mathbf{x} - \mathbf{x}_0 &\sim \mathcal{N}(\boldsymbol{\mu}_x - \mathbf{x}_0, \Sigma_x); \\ \mathbf{J}_x(\mathbf{x} - \mathbf{x}_0) &\sim \mathcal{N}(\mathbf{J}_x(\boldsymbol{\mu}_x - \mathbf{x}_0), \mathbf{J}_x \Sigma_x \mathbf{J}_x^T), \end{aligned} \quad (25)$$

and similar for the noise and the phase factor.

The linearised mismatch function  $\mathbf{f}_{\text{vts}}$  replaces  $\mathbf{f}$  in the delta function in (19d). The approximation for  $\mathbf{y}$  then is the sum of the mismatch function at the expansion point and the three Gaussians:

$$\begin{aligned} p(\mathbf{y}) &= \int \int \int \delta_{\mathbf{f}(\mathbf{x}, \mathbf{n}, \alpha)}(\mathbf{y}) p(\alpha) d\alpha p(\mathbf{n}) d\mathbf{n} p(\mathbf{x}) d\mathbf{x} \\ &\simeq \int \int \int \delta_{\mathbf{f}_{\text{vts}}(\mathbf{x}, \mathbf{n}, \alpha)}(\mathbf{y}) \mathcal{N}(\alpha; \mathbf{0}, \Sigma_\alpha) d\alpha \mathcal{N}(\mathbf{n}; \boldsymbol{\mu}_n, \Sigma_n) d\mathbf{n} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_x, \Sigma_x) d\mathbf{x} \\ &= \mathcal{N}(\mathbf{y}; \mathbf{f}(\mathbf{x}_0, \mathbf{n}_0, \alpha_0) + \mathbf{J}_x(\boldsymbol{\mu}_x - \mathbf{x}_0) + \mathbf{J}_n(\boldsymbol{\mu}_n - \mathbf{n}_0) + \mathbf{J}_\alpha(\mathbf{0} - \alpha_0), \\ &\quad \mathbf{J}_x \Sigma_x \mathbf{J}_x^T + \mathbf{J}_n \Sigma_n \mathbf{J}_n^T + \mathbf{J}_\alpha \Sigma_\alpha \mathbf{J}_\alpha^T). \end{aligned} \quad (26)$$

The expansion points are usually set to the means of the distributions for the clean speech, additive noise and phase factor, so that  $\boldsymbol{\mu}_x - \mathbf{x}_0 = \boldsymbol{\mu}_n - \mathbf{n}_0 = \mathbf{0} - \alpha_0 = \mathbf{0}$ . The parameters of the Gaussian approximation of the corrupted speech  $\mathbf{y}_{\text{vts}} \sim \mathcal{N}(\boldsymbol{\mu}_y, \Sigma_y)$  then become

$$\boldsymbol{\mu}_y = \mathbf{f}(\boldsymbol{\mu}_x, \boldsymbol{\mu}_n, \boldsymbol{\mu}_\alpha); \quad \Sigma_y = \mathbf{J}_x \Sigma_x \mathbf{J}_x^T + \mathbf{J}_n \Sigma_n \mathbf{J}_n^T + \mathbf{J}_\alpha \Sigma_\alpha \mathbf{J}_\alpha^T. \quad (27)$$

The most obviously useful effect of linearising the mismatch function is that the corrupted speech turns out Gaussian. There are also other advantages that arise from fixing the expansion points, so that the relationship between speech, noise, and corrupted speech becomes linear per component. The means of the noise model can be estimated with a fixed-point iteration [28] and the variance with a gradient-descent-based scheme [27]. Alternatively, since the first-order approximation makes the noise, speech, and corrupted speech jointly Gaussian, an EM approach [16, 9, 18] can be used. Also, it is possible to use adaptive training with it [26, 15]. These aspects make vts compensation very useful in practice.

Previous work has assumed the phase factor  $\mathbf{0}$  [2], or fixed to a different value, like 1 [27] or 2.5 [23]. A phase factor distribution has previously only been used for feature enhancement [4]. This work will apply vts with a Gaussian phase factor, and use it for model compensation.

Compared to using a distribution, assuming  $\alpha$  constant has two effects. One is that the term  $\mathbf{J}_\alpha \Sigma_\alpha \mathbf{J}_\alpha^T$  drops out from the covariance expression in (27). Since the entries

of the phase factor covariance are small, this changes the distributions only slightly. Since  $\Sigma_\alpha$  is constant across components and  $\mathbf{J}_\alpha$  changes only slightly between adjacent components, discrimination is hardly affected.

If  $\alpha$  is equal to its expected value,  $\mathbf{0}$ , then the mean of the compensated Gaussian does not change compared to when  $\alpha$  is assumed Gaussian. If  $\alpha$  is set to a higher value, the mean is overestimated. Also, Jacobians  $\mathbf{J}_x$  and  $\mathbf{J}_n$  move closer to  $\frac{1}{2}\mathbf{I}$  (see appendix B for the expressions).

In practice,  $\alpha$  is often assumed fixed but the noise model is estimated. This should subsume many of the effects that using a phase factor distribution would have had. This includes a wider compensated Gaussian and overestimation of the mode.

### 3.3 Algonquin

The Algonquin algorithm [8, 19] is an extension to vts compensation. Its most important conceptual addition is that it takes into account the observation vector. In the following discussion, when an actual observation is meant it will be indicated with  $\mathbf{y}_t$ . Whereas vts linearises the mismatch function at the expansion point given by the means of the prior distributions of the speech and the noise, the Algonquin algorithm updates the expansion point iteratively, finding the mode of the posterior of the speech and the noise. It can therefore be seen as an iterative approach to finding the Laplace approximation to the posterior.

The Algonquin algorithm uses a slightly simplified environment model compared to the one discussed in section 2.3, in (19d). The influence of the phase factor on the observation is captured by a Gaussian around the mode of the distribution for given  $\mathbf{x}$  and  $\mathbf{n}$ . Thus, that distribution is approximated as

$$p(\mathbf{y}|\mathbf{x}, \mathbf{n}) = \int \delta_{f(\mathbf{x}, \mathbf{n}, \alpha)}(\mathbf{y}) p(\alpha) d\alpha \simeq \mathcal{N}(\mathbf{y}; \mathbf{f}(\mathbf{x}, \mathbf{n}, \mathbf{0}), \mathbf{\Psi}), \quad (28)$$

where  $\mathbf{\Psi}$  is the fixed covariance that models the uncertainty around the mismatch function.

Figure 3 on the next page shows a one-dimensional example. The prior of the speech and the noise is given in the left panel. Their posterior distribution after having seen an observation  $\mathbf{y}_t$  is in the right panel. This posterior will be approximated with a Gaussian centred on its mode.

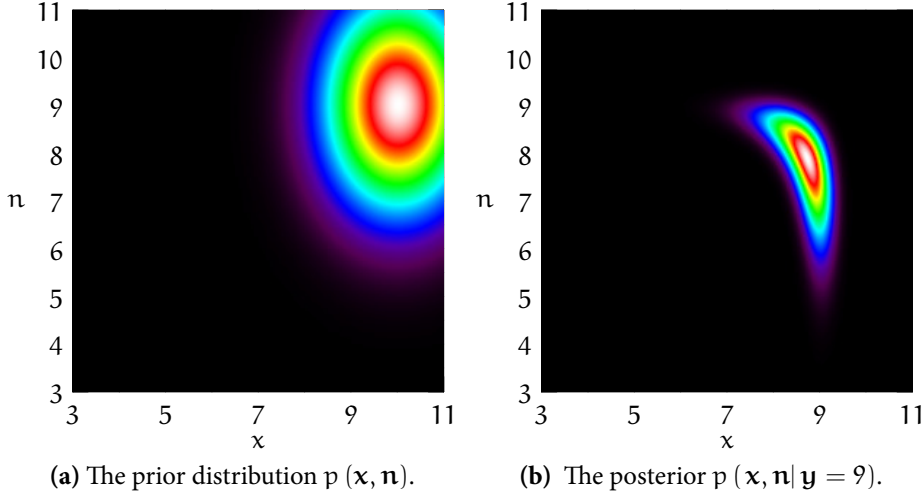
To deal with the non-linearity in the mismatch function, it is linearised (like in vts compensation). The Algonquin algorithm iteratively updates the linearisation point to the mode of the posterior distribution. The linearisation point in iteration  $k$  is denoted by  $(\mathbf{x}_0^{(k)}, \mathbf{n}_0^{(k)})$ . The linearised mismatch function in iteration  $k$  is

$$\mathbf{f}_{\text{vts}}^{(k)}(\mathbf{x}, \mathbf{n}) = \mathbf{f}(\mathbf{x}_0^{(k)}, \mathbf{n}_0^{(k)}, \mathbf{0}) + \mathbf{J}_x^{(k)}(\mathbf{x} - \mathbf{x}_0^{(k)}) + \mathbf{J}_n^{(k)}(\mathbf{n} - \mathbf{n}_0^{(k)}), \quad (29)$$

where the Jacobians are

$$\mathbf{J}_x^{(k)} = \left. \frac{d\mathbf{f}}{d\mathbf{x}} \right|_{\mathbf{x}_0^{(k)}, \mathbf{n}_0^{(k)}, \mathbf{0}}; \quad \mathbf{J}_n^{(k)} = \left. \frac{d\mathbf{f}}{d\mathbf{n}} \right|_{\mathbf{x}_0^{(k)}, \mathbf{n}_0^{(k)}, \mathbf{0}}. \quad (30)$$

For the first iteration  $k = 0$ , the linearisation point is set to  $(\boldsymbol{\mu}_x, \boldsymbol{\mu}_n)$ , so that the mismatch function is equivalent to the one used in vts compensation in (24a), leaving out the phase factor.



**Figure 3** The Algonquin-derived distribution of the clean speech and noise for  $\mathbf{x} \sim \mathcal{N}(10, 1)$ ;  $\mathbf{n} \sim \mathcal{N}(9, 2)$ ;  $\psi = 0.04$ ;  $\mathbf{y} = 9$ .

Using the linearised mismatch function in (29), the speech, the noise, and the observation become jointly Gaussian:

$$q_{\mathbf{y}_t}^{(k)} \left( \begin{bmatrix} \mathbf{x} \\ \mathbf{n} \\ \mathbf{y} \end{bmatrix} \right) = \mathcal{N} \left( \begin{bmatrix} \mathbf{x} \\ \mathbf{n} \\ \mathbf{y} \end{bmatrix}; \begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_n \\ \boldsymbol{\mu}_y^{(k)} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_x & \mathbf{0} & \boldsymbol{\Sigma}_{xy}^{(k)} \\ \mathbf{0} & \boldsymbol{\Sigma}_n & \boldsymbol{\Sigma}_{ny}^{(k)} \\ \boldsymbol{\Sigma}_{yx}^{(k)} & \boldsymbol{\Sigma}_{yn}^{(k)} & \boldsymbol{\Sigma}_y^{(k)} \end{bmatrix} \right). \quad (31)$$

Note that the parameters of the marginal of  $\mathbf{x}$  and  $\mathbf{n}$  ( $\boldsymbol{\mu}_x$ ,  $\boldsymbol{\mu}_n$ ,  $\boldsymbol{\Sigma}_x$ , and  $\boldsymbol{\Sigma}_n$ ) are given by the prior and do not depend on the iteration  $k$ . On the other hand, the covariance of the observation and the cross-covariances are found through the linearised mismatch function, which changes with every iteration. Those parameters are found as follows. Using the linearised mismatch function and assuming the error  $\mathcal{N}(\mathbf{0}, \boldsymbol{\Psi})$  of the mismatch function, the distribution of the corrupted speech is Gaussian  $\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}_y^{(k)}, \boldsymbol{\Sigma}_y^{(k)})$  with parameters similar to those for vts compensation in (27):

$$\boldsymbol{\mu}_y^{(k)} \triangleq \mathcal{E} \{ \mathbf{f}_{\text{vts}}^{(k)}(\mathbf{x}, \mathbf{n}) \} = \mathbf{f}(\mathbf{x}_0^{(k)}, \mathbf{n}_0^{(k)}, \mathbf{0}) + \mathbf{J}_x^{(k)}(\boldsymbol{\mu}_x - \mathbf{x}_0^{(k)}) + \mathbf{J}_n^{(k)}(\boldsymbol{\mu}_n - \mathbf{n}_0^{(k)}); \quad (32a)$$

$$\begin{aligned} \boldsymbol{\Sigma}_y^{(k)} &\triangleq \mathcal{E} \left\{ (\mathbf{f}_{\text{vts}}^{(k)}(\mathbf{x}, \mathbf{n}) - \boldsymbol{\mu}_y) (\mathbf{f}_{\text{vts}}^{(k)}(\mathbf{x}, \mathbf{n}) - \boldsymbol{\mu}_y)^\top \right\} + \boldsymbol{\Psi} \\ &= \mathcal{E} \left\{ \mathbf{J}_x^{(k)}(\mathbf{x} - \boldsymbol{\mu}_x) (\mathbf{J}_x^{(k)}(\mathbf{x} - \boldsymbol{\mu}_x))^\top + \mathbf{J}_n^{(k)}(\mathbf{n} - \boldsymbol{\mu}_n) (\mathbf{J}_n^{(k)}(\mathbf{n} - \boldsymbol{\mu}_n))^\top \right\} + \boldsymbol{\Psi} \\ &= \mathbf{J}_x^{(k)} \mathcal{E} \left\{ (\mathbf{x} - \boldsymbol{\mu}_x) (\mathbf{x} - \boldsymbol{\mu}_x)^\top \right\} \mathbf{J}_x^{(k)\top} + \mathbf{J}_n^{(k)} \mathcal{E} \left\{ (\mathbf{n} - \boldsymbol{\mu}_n) (\mathbf{n} - \boldsymbol{\mu}_n)^\top \right\} \mathbf{J}_n^{(k)\top} + \boldsymbol{\Psi} \\ &= \mathbf{J}_x^{(k)} \boldsymbol{\Sigma}_x \mathbf{J}_x^{(k)\top} + \mathbf{J}_n^{(k)} \boldsymbol{\Sigma}_n \mathbf{J}_n^{(k)\top} + \boldsymbol{\Psi}. \end{aligned} \quad (32b)$$

The cross-covariance between the speech and the observation and between the noise

and the observation can be derived similarly:

$$\begin{aligned}\boldsymbol{\Sigma}_{yx}^{(k)} &\triangleq \mathcal{E} \left\{ \left( \mathbf{f}_{\text{vts}}^{(k)}(\mathbf{x}, \mathbf{n}) - \boldsymbol{\mu}_y \right) (\mathbf{x} - \boldsymbol{\mu}_x)^\top \right\} \\ &= \mathcal{E} \left\{ \left( \mathbf{J}_x^{(k)} (\mathbf{x} - \boldsymbol{\mu}_x) \right) (\mathbf{x} - \boldsymbol{\mu}_x)^\top \right\} \\ &= \mathbf{J}_x^{(k)} \mathcal{E} \left\{ (\mathbf{x} - \boldsymbol{\mu}_x) (\mathbf{x} - \boldsymbol{\mu}_x)^\top \right\} = \mathbf{J}_x^{(k)} \boldsymbol{\Sigma}_x; \end{aligned} \quad (33a)$$

$$\boldsymbol{\Sigma}_{yn}^{(k)} \triangleq \mathbf{J}_n^{(k)} \boldsymbol{\Sigma}_n. \quad (33b)$$

Assuming the joint distribution of the speech, noise, and observations in (31), the posterior distribution of the speech and the noise conditioned on an observation  $\mathbf{y}_t$  follows from a standard result (derived in appendix A.3, (125c)). This approximation to the posterior distribution will be written  $q_{y_t}^{(k)}$ :

$$\begin{aligned} q_{y_t}^{(k)} \left( \begin{bmatrix} \mathbf{x} \\ \mathbf{n} \end{bmatrix} \right) &= \mathcal{N} \left( \begin{bmatrix} \mathbf{x} \\ \mathbf{n} \end{bmatrix}; \begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_n \end{bmatrix} + \begin{bmatrix} \boldsymbol{\Sigma}_{xy}^{(k)} \\ \boldsymbol{\Sigma}_{ny}^{(k)} \end{bmatrix} \boldsymbol{\Sigma}_y^{(k)-1} (\mathbf{y}_t - \boldsymbol{\mu}_y^{(k)}), \right. \\ &\quad \left. \begin{bmatrix} \boldsymbol{\Sigma}_x & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_n \end{bmatrix} - \begin{bmatrix} \boldsymbol{\Sigma}_{xy}^{(k)} \\ \boldsymbol{\Sigma}_{ny}^{(k)} \end{bmatrix} \boldsymbol{\Sigma}_y^{(k)-1} \begin{bmatrix} \boldsymbol{\Sigma}_{yx}^{(k)} & \boldsymbol{\Sigma}_{yn}^{(k)} \end{bmatrix} \right). \end{aligned} \quad (34)$$

Note that the speech and noise priors are not correlated, but the posteriors are.

Algonquin sets the expansion point for the next iteration to the mean of this approximation to the posterior:

$$\begin{aligned} \mathbf{x}_0^{(k+1)} &= \boldsymbol{\mu}_x + \boldsymbol{\Sigma}_{xy}^{(k)} \boldsymbol{\Sigma}_y^{(k)-1} (\mathbf{y} - \boldsymbol{\mu}_y^{(k)}); \\ \mathbf{n}_0^{(k+1)} &= \boldsymbol{\mu}_n + \boldsymbol{\Sigma}_{ny}^{(k)} \boldsymbol{\Sigma}_y^{(k)-1} (\mathbf{y} - \boldsymbol{\mu}_y^{(k)}), \end{aligned} \quad (35)$$

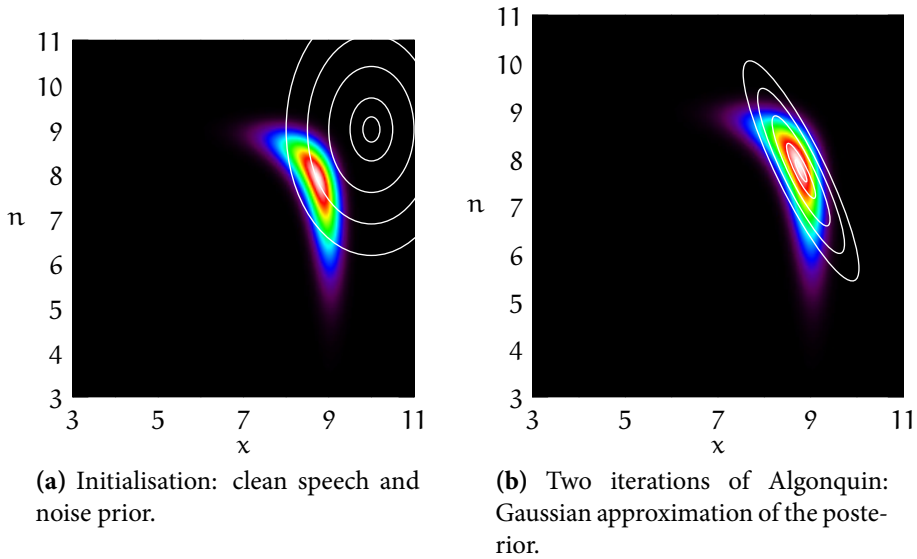
so that the expansion point is updated at every iteration, and the Gaussian approximation to the posterior is moved. There is no guarantee that the mode of the approximation converges to the mode of the real posterior: the algorithm may overshoot. A damping factor could be introduced for this, but this appears to slow down convergence without benefit [20].

After  $K$  iterations, the Gaussian approximation  $q$  to the distribution of  $\mathbf{y}$  is found from (32):

$$\begin{aligned} q_{y_t}^{(K)}(\mathbf{y}) &= \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}_y^{(K)}, \boldsymbol{\Sigma}_y^{(K)}) \\ &= \mathcal{N}(\mathbf{y}; \mathbf{f}(\mathbf{x}_0^{(K)}, \mathbf{n}_0^{(K)}, \mathbf{0}) + \mathbf{J}_x^{(K)} (\boldsymbol{\mu}_x - \mathbf{x}_0^{(K)}) + \mathbf{J}_n^{(K)} (\boldsymbol{\mu}_n - \mathbf{n}_0^{(K)}), \\ &\quad \mathbf{J}_x^{(K)} \boldsymbol{\Sigma}_x \mathbf{J}_x^{(K)\top} + \mathbf{J}_n^{(K)} \boldsymbol{\Sigma}_n \mathbf{J}_n^{(K)\top} + \boldsymbol{\Psi}). \end{aligned} \quad (36)$$

Figure 4 on the facing page shows a one-dimensional simulation of the Algonquin algorithm in  $(x, n)$ -space. The left panel shows the prior of the clean speech and the additive noise, and the right panel the Algonquin approximation to the posterior. Note that the priors of the clean speech and noise are not correlated, but the posterior is.

The problem with using this Gaussian approximation is that the effective distribution is not normalised. Even though  $q_{y_t}$  in (36) is a normalised Gaussian if estimated iteratively for one  $\mathbf{y}_t$ , in practice it would be estimated and then applied to the same  $q_{y_t}$ .



**Figure 4** *Iterations of Algonquin.*

Thus, in general,

$$\int q_{y_t}(\mathbf{y}_t) d\mathbf{y}_t \neq 1. \quad (37)$$

There are two ways of working around the problem that  $q_{y_t}(\mathbf{y}_t)$  is not a normalised distribution.

The original Algonquin algorithm [8] applies feature enhancement, which finds the minimum mean square error estimate of the clean speech. It uses an extension of model-based feature enhancement, which normally models the clean speech with a mixture of Gaussians. Algonquin extends this idea by at the same time as updating the observation distributions, finding an approximation to the component posteriors of the mixture of Gaussians. It is then possible to perform variational inference on the joint distribution of the clean speech component and the Gaussian approximation to the distribution of the observations [8]. The minimum mean square error estimate of the clean speech follows from the posterior responsibilities of the GMM components and their linearisation of the observation distribution.

Another approach uses a form of model compensation: it replaces the likelihood calculation for each component by a computation of the component’s “soft information score” [19, 20]. The original presentation derives this form from the feature enhancement version. The end result can be derived more directly, which the following will do before showing the equivalence to the original presentation. As in the original, this will assume that there is only one component per state (though the generalisation is straightforward).

Consider a speech recogniser, which aims to find the most likely state sequence  $\Theta$

from an observation sequence  $\mathbf{Y}$  with

$$\begin{aligned} P(\Theta|\mathbf{Y}) &= \frac{p(\mathbf{Y}|\Theta)P(\Theta)}{p(\mathbf{Y})} \\ &\propto p(\mathbf{Y}|\Theta)P(\Theta) = P(\theta_0) \prod_t P(\mathbf{y}_t|\theta_t)P(\theta_t|\theta_{t-1}). \end{aligned} \quad (38)$$

The normalisation constant  $p(\mathbf{Y})$  is normally ignored, because it does not change depending on the state sequence  $\Theta$ . An obvious way of approximating the component likelihood  $p(\mathbf{y}_t|\theta_t)$  would be to replace it by the Algonquin approximation for that component  $m$  and observation  $\mathbf{y}_t$ ,

$$p(\mathbf{y}_t|\theta_t = m) \simeq q_{y_t}^{(m)}(\mathbf{y}_t). \quad (39)$$

That this approximation to  $p(\mathbf{y}_t|m)$  is not necessarily normalised is not a problem for the Viterbi algorithm, though it does raise the question whether the likelihood approximations for different components can be compared.

The original presentation of Algonquin for model compensation [20, 19] derives this form (up to a constant factor) through a number of manipulations. This relates the form of models compensation to the one for feature enhancement. The normalisation constant in (38) is not ignored, but approximated with a mixture of Gaussians:

$$p(\mathbf{Y}) \simeq \prod_t p(\mathbf{y}_t). \quad (40)$$

This decouples the distribution of  $\mathbf{y}$  from the state sequence. This is substituted in in the expression that speech recognisers aim to maximise,  $P(\Theta|\mathbf{Y})$ . It is then rewritten by applying Bayes' rule, the approximation in (40), and again Bayes' rule:

$$\begin{aligned} P(\Theta|\mathbf{Y}) &= \frac{p(\mathbf{Y}|\Theta)P(\Theta)}{p(\mathbf{Y})} \simeq P(\theta_0) \prod_t \frac{p(\mathbf{y}_t|\theta_t)}{p(\mathbf{y}_t)} P(\theta_t|\theta_{t-1}) \\ &= P(\theta_0) \prod_t \frac{P(\theta_t|\mathbf{y}_t)p(\mathbf{y}_t)}{p(\mathbf{y}_t)P(\theta_t)} P(\theta_t|\theta_{t-1}) \\ &= P(\theta_0) \prod_t \frac{P(\theta_t|\mathbf{y}_t)}{P(\theta_t)} P(\theta_t|\theta_{t-1}). \end{aligned} \quad (41)$$

The expression that now replaces the likelihood computation for component  $\theta_t = m$  in the speech recogniser,  $P(m|\mathbf{y}_t)/P(m)$ , is then called the ‘‘soft information score’’. Its numerator,  $P(m|\mathbf{y}_t)$ , is approximated by its variational approximation<sup>3</sup>

$$P(m|\mathbf{y}_t) \simeq q_{y_t}(m) = \frac{q_{y_t}^{(m)}(\mathbf{y}_t)P(m)}{\sum_{m'} q_{y_t}^{(m')}(\mathbf{y}_t)P(m')}. \quad (42)$$

The soft information score therefore is approximated

$$\frac{P(m|\mathbf{y}_t)}{P(m)} \simeq \frac{q_{y_t}(m)}{P(m)} = \frac{q_{y_t}^{(m)}(\mathbf{y}_t)}{\sum_{m'} q_{y_t}^{(m')}(\mathbf{y}_t)P(m')}. \quad (43)$$

<sup>3</sup>See [20], equations (10.10), (10.15), and (8.18) for more details.

The normalisation term for one frame is the same across all components. Therefore, it does not have an effect on decoding, so that the likelihood is in essence replaced by unnormalised distribution  $q_{y_t}(\mathbf{y}_t)$ , as in (39). The question thus remains whether the unnormalised Algonquin approximations to the likelihood are comparable between difference components.

### 3.4 Piecewise linear approximation

The compensation methods in the previous sections have used a single Gaussian to represent the observation distribution. However, the actual observations are not Gaussian distributed even when the speech and noise are. It is possible to approximate the integral that gives the likelihood of the observation. [29] present a method for feature enhancement, but as a step in estimating the noise model the corrupted speech distribution is formulated in such a way that it can be approximated using a piecewise linear approximation. The model for the interaction of the speech, noise, and observation the original work uses is somewhat simpler than the one in section 2.3. The following will adhere to the original presentation, because it provides good insight in both the main idea and the main limitation.

The main idea is to transform the integral over the speech and the noise into another space. It then becomes easier to apply a piecewise linear approximation. The main limitation is that the method works on a single dimension, and uses log-spectral domain coefficients. In the log-spectral domain, coefficients are highly correlated. Appendix C.2 shows that it is theoretically possible to perform the transformation of the integral in more dimensions. However, if the piecewise linear approximation in the one-dimensional case requires 8 line segments, the  $d$ -dimensional case requires  $8^d$ . This makes the scheme infeasible for correlated feature vectors.

The scheme works as follows in one dimension. Speech  $x$ , noise  $n$ , and observation  $y$  are assumed deterministically related, with

$$\exp(y) = \exp(x) + \exp(n), \quad (44a)$$

as a one-dimensional version of (3a) where the phase factor is assumed 0. If  $y$  is observed to be  $y_t$ , and  $x$  is changed,  $n$  automatically changes too. A substitute variable  $u$  is therefore introduced to replace both  $x$  and  $n$  in the integration in the likelihood expression. It is defined

$$u = 1 - \exp(x - y_t), \quad (44b)$$

so that (see section C.1 for details)

$$n = y_t + \log(u); \quad (44c)$$

$$x = y_t + \log(1 - u). \quad (44d)$$

Given a speech coefficient, the distribution of the observation can be written in terms of the distribution of the noise. As this is a transformation of the feature space, it is a standard result (see section A.1) that a Jacobian is introduced:

$$p(y_t|x) = \left| \frac{dn(x, y_t)}{dy_t} \right| p(n(x, y_t)), \quad (45)$$

where  $p(\mathbf{n}(x, y_t))$  is the prior of  $\mathbf{n}$  evaluated at the point implied by the setting of  $x$  and  $y_t$ .

When the likelihood of  $y_t$  is expressed as an integral over  $x$ , it can be transformed into an integral over  $u$  as follows:

$$\begin{aligned} p(y_t) &= \int_{-\infty}^{y_t} p(y_t|x) p(x) dx \\ &= \int_{-\infty}^{y_t} \left| \frac{d\mathbf{n}(x, y)}{dy} \right|_{y_t} p(\mathbf{n}(x, y_t)) p(x) dx \\ &= \int_0^1 \left| \frac{d\mathbf{n}(x, y)}{dy} \right|_{y_t} p(\mathbf{n}(u, y_t)) \left| \frac{dx(u, y_t)}{du} \right| p(x(u, y_t)) du, \end{aligned} \quad (46)$$

where  $p(\mathbf{n}(u, y_t))$  is the prior of  $\mathbf{n}$  evaluated at the point implied by the setting of  $u$  and  $y_t$ , and similar for  $p(x(u, y_t))$ .

Appendix C gives the complete derivation. The likelihood can be rewritten

$$\begin{aligned} p(y_t) &= \exp\left(\frac{1}{2}\sigma_n^2 + \frac{1}{2}\sigma_x^2 - \mu_n - \mu_x + 2y_t\right) \\ &\int_0^1 \mathcal{N}(\log(u); \mu_n - \sigma_n^2 - y_t, \sigma_n^2) \mathcal{N}(\log(1-u); \mu_x - \sigma_x^2 - y_t, \sigma_x^2) du. \end{aligned} \quad (47)$$

It is  $\log(u)$  and  $\log(1-u)$  that can be approximated with piecewise linear functions. [29] use 8 line segments. For any observation  $y_t$ , the expression then becomes a sum of integrals of a fixed factor times an integral over a Gaussian, for which well-known approximations exist.

This derivation crucially depends on the assumption that coefficients can be considered separately. To model the likelihood well, the priors  $p(x)$  and  $p(\mathbf{n})$  need to model correlations between coefficients. In the cepstral domain where correlations are not usually modelled, they become more important as the signal-to-noise ratio drops [12]. But any generalisation of the derivation to vectors of cepstral coefficients will need to convert to log-spectral vectors anyway, and turn diagonal-covariance priors into full-covariance ones. Note that though the derivation in [29] is for log-spectral coefficients, the method is applied to cepstral coefficients.

Appendix C.2 gives the generalisation of the algorithm in [29] to  $d$ -dimensional log-spectral vectors. It turns out that the integral in (47) becomes an integral over  $\langle 0, 1 \rangle^d$ . This means that the 8 line segments for the single-dimensional case become  $8^d$  hyperplanes. It is infeasible to apply this to a standard 24-dimensional log-spectral feature space. Section 5.2 will therefore use a similar idea but a different transformation and a different approximation to the integral. It will use a Monte Carlo method, sequential importance re-sampling, to approximate the integral.

## 4 Multi-dimensional integration with importance sampling

The corrupted speech likelihood can be expressed analytically only as an integral (as in (19)). This integration can be approximated with sampling. Monte Carlo methods approximate a target density with a finite number of samples. Many Monte Carlo

methods can work with unnormalised densities, which for many applications is a useful feature. Markov chain Monte Carlo, for example, divides the value of the density at two points by each other, so that any normalisation constant cancels out, and can be disregarded. However, the value required here, the integral of a target density over the whole of a space, is the normalisation constant of the density. The samples themselves are merely a by-product.

This requirement rules out many Monte Carlo methods. One technique, called *importance sampling*, does return an approximation to the normalisation constant. It requires a (normalised) *proposal distribution* that samples can be drawn from and is close in shape to the target density. *Sequential importance sampling* is importance sampling over a multi-dimensional space. In itself, it is just importance sampling that deals explicitly with one dimension at a time. It becomes advantageous once *re-sampling* is introduced between dimensions. This removes low-weight samples and duplicates high-weight ones, so that the samples focus on the most interesting, high-probability regions of space.

Sequential sampling techniques are often presented as traversing through time. There is no reason, however, why the dimensions should represent time. In this work, dimensions will relate to log-powers in consecutive mel-bins and will be called just “dimensions”. This section follows the presentation of [5]. It will discuss implicitly multi-dimensional Monte Carlo and importance sampling. Then, re-sampling is introduced. Finally, section 4.4 on page 29 discusses the case where for some dimensions it is possible to draw from the target distribution, and for some it is not.

## 4.1 Monte Carlo

Monte Carlo methods approximate a target distribution with a finite number of samples. Denote the target probability distribution with  $\pi$ . If it is possible to draw  $S$  samples  $\mathbf{u}^{(s)} \sim \pi$ , the Monte Carlo approximation of  $\pi$  is the empirical distribution

$$\tilde{\pi} = \frac{1}{S} \sum_s \delta_{\mathbf{u}^{(s)}}, \quad (48)$$

where  $\delta$  denotes the Dirac delta. Using the empirical measure in the place of the target distribution, the expectation of any test function  $\phi$  under distribution  $\pi$  can be approximated as

$$\mathcal{E}\{\phi(\mathbf{u})\} = \int \pi(\mathbf{u})\phi(\mathbf{u}) \, d\mathbf{u} \approx \int \tilde{\pi}(\mathbf{u})\phi(\mathbf{u}) \, d\mathbf{u} = \frac{1}{S} \sum_s \phi(\mathbf{u}^{(s)}). \quad (49)$$

This equation will be used in this work to estimate empirical means (with  $\phi(\mathbf{u}) = \mathbf{u}$ ) and covariances (with  $\phi(\mathbf{u}) = \mathbf{u}\mathbf{u}^T$ ).

This straightforward Monte Carlo method requires, however, that it is possible to sample directly from the target distribution. When this is not the case, it is often possible to use *importance sampling*, which draws samples from a *proposal distribution* that is close to the target distribution, and then makes up for the difference.

### 4.1.1 Importance sampling

If it is impossible to sample from the target distribution, it can still be possible to approximate it by sampling from a proposal distribution  $\rho$  similar to the target distribu-

tion, and makes up for the difference by weighting the samples. This is called *importance sampling*. The weights also give an approximation to the normalisation constant. The process is analogous to the evaluation of a function under a distribution in (49). However,  $S$  samples  $\mathbf{u}^{(s)} \sim \rho$  are drawn from the proposal distribution, so that the empirical proposal distribution of it is, analogously to (48):

$$\tilde{\rho} = \frac{1}{S} \sum_s \delta_{\mathbf{u}^{(s)}}. \quad (50)$$

That samples are drawn from a proposal distribution and not directly from the target distribution makes it possible to use an unnormalised target density,  $\gamma$ . This is often useful if the normalisation constant of the target density is unknown. It does need to be possible to evaluate  $\gamma$  at any point. Importance sampling finds samples from the distribution at the same time as an approximation to the normalisation constant.  $\gamma$  is a scaled version of  $\pi$ :

$$\pi(\mathbf{u}) = \frac{\gamma(\mathbf{u})}{Z}, \quad (51a)$$

where the normalising constant is

$$Z = \int \gamma(\mathbf{u}) \, d\mathbf{u}. \quad (51b)$$

The proposal density needs to cover at least the area that the target distribution covers:

$$\pi(\mathbf{u}) > 0 \Rightarrow \rho(\mathbf{u}) > 0, \quad (52)$$

otherwise no samples will be drawn in some regions where  $\pi$  is non-zero.

The key to making up for the difference between proposal and target is the weight function  $w(\mathbf{u})$ . It gives the ratio between the target density and the proposal distribution:

$$w(\mathbf{u}) = \frac{\gamma(\mathbf{u})}{\rho(\mathbf{u})}. \quad (53)$$

Substituting  $w(\mathbf{u})\rho(\mathbf{u})$  for  $\gamma(\mathbf{u})$  in (51a) and (51b),

$$\pi(\mathbf{u}) = \frac{w(\mathbf{u})\rho(\mathbf{u})}{Z}; \quad (54a)$$

$$Z = \int w(\mathbf{u})\rho(\mathbf{u}) \, d\mathbf{u}. \quad (54b)$$

Now that the target distribution has been expressed in terms of the proposal distribution  $\rho$ , the proposal distribution can be replaced by its empirical version  $\tilde{\rho}$  in (50). This yields the empirical distribution to  $\pi$ ,  $\tilde{\pi}$ , with the samples from  $\rho$  weighted by their importance weight:

$$\tilde{\pi} = \frac{1}{S} \sum_s \frac{w(\mathbf{u}^{(s)})}{\bar{Z}} \delta_{\mathbf{u}^{(s)}} = \sum_s \bar{w}^{(s)} \delta_{\mathbf{u}^{(s)}}, \quad (55a)$$

where approximation to the normalisation constant is

$$\tilde{Z} = \int w(\mathbf{u}) \tilde{\rho}(\mathbf{u}) \, d\mathbf{u} = \frac{1}{S} \sum_s w(\mathbf{u}^{(s)}), \quad (55b)$$

and the normalised weights  $\bar{w}^{(s)}$  are

$$\bar{w}^{(s)} = \frac{w(\mathbf{u}^{(s)})}{\sum_{s'} w(\mathbf{u}^{(s')})}. \quad (55c)$$

$\tilde{\pi}$  is the normalised importance sampling approximation to target distribution  $\gamma$ , and  $\tilde{Z}$  is the corresponding approximation to the normalisation constant.

The expectation of a test function  $\phi(\mathbf{u})$  under  $\pi$  can be estimated analogously to (49), with  $\tilde{\pi}$  given by (55a):

$$\mathcal{E}\{\phi(\mathbf{u})\} = \int \pi(\mathbf{u}) \phi(\mathbf{u}) \, d\mathbf{u} \approx \int \tilde{\pi}(\mathbf{u}) \phi(\mathbf{u}) \, d\mathbf{u} = \frac{1}{S} \sum_s \bar{w}^{(s)} \phi(\mathbf{u}^{(s)}). \quad (56)$$

A degenerate case of importance sampling is when the proposal distribution is equal to the normalised target distribution  $\rho = \pi$ . If it is possible to draw samples from  $\pi$ , then using importance sampling is overkill. However, the next section will introduce sequential importance sampling, which samples from one dimension at a time. In that setting, it might be possible to draw from the target distribution for some dimensions, but not for others. In the simple importance sampling case with  $\rho = \pi$ , the weight function in (53) always yields the normalisation constant:

$$w(\mathbf{u}) = \frac{\gamma(\mathbf{u})}{\rho(\mathbf{u})} = Z. \quad (57)$$

Substituting this in (55a), the approximation  $\tilde{\pi}$  of the normalised target distribution  $\pi$  becomes

$$\tilde{\pi} = \frac{1}{S} \sum_s \frac{w(\mathbf{u}^{(s)})}{\tilde{Z}} \delta_{\mathbf{u}^{(s)}} = \frac{1}{S} \sum_s \delta_{\mathbf{u}^{(s)}}, \quad (58)$$

which is exactly the standard Monte Carlo empirical distribution in (48).

## 4.2 Sequential importance sampling

Sequential importance sampling is an instantiation of importance sampling that explicitly handles a multi-dimensional sample space. It steps through the dimensions one by one, keeping track of  $S$  samples, and extending them with a new dimension at every step. Considering this explicitly is useful because then the set of samples can be adjusted between dimensions. Section 4.3 will discuss re-sampling, which drops low-probability samples and multiplies high-probability samples, so that computational effort is focussed on high-probability regions.

Sequential importance sampling is a slight generalisation of the well-known particle filtering algorithm. It samples from a multi-dimensional distribution dimension by dimension, applying principles similar to those of importance sampling at every step.

The distribution must be factored into dimensions. In particle filtering, the dimensions are often time steps, and the factor for each dimension a distribution conditional on previous dimensions. However, for sequential importance sampling the per-dimension target distributions in sequential importance sampling need not be normalised or relate to valid probability distributions, as long as their product is equal to the target distribution.

In section 4.1 on page 23, the sample space was implicitly multi-dimensional. In this section, the the dimensions of the samples will be explicitly written. The space has  $d$  dimensions. Thus,  $\mathbf{u} \triangleq \mathbf{u}_{1:d}$ . The distributions  $\gamma$ ,  $\pi$ , and  $\rho$  will be factorised, as will  $Z$  and  $w$ .

To apply sequential importance sampling, it must be possible to factorise the target density  $\gamma$  into factors  $\gamma_i(\cdot|\cdot)$  for each dimension.  $\gamma_i(\cdot|\cdot)$  is therefore defined as

$$\gamma_i(\mathbf{u}_i|\mathbf{u}_{1:i-1}) = \frac{\gamma_i(\mathbf{u}_{1:i})}{\gamma_{i-1}(\mathbf{u}_{1:i-1})}. \quad (59a)$$

If for every  $i$ ,  $\gamma_i(\mathbf{u}_{1:i})$  is a marginal distribution of  $\mathbf{u}_{1:i}$ , then (59a) is an instantiation of Bayes' rule and  $\gamma_i(\mathbf{u}_i|\mathbf{u}_{1:i-1})$  is a conditional distribution. However, even though the notation used is  $(\cdot|\cdot)$ , there is no requirement for the factors to be conditionals or to be normalised. This is a generalisation of particle filtering, and indeed, a strength of sequential importance sampling.

The target density  $\gamma$  can be written as the product of factors  $\gamma_i$ . (59b) formulates the factorisation of  $\gamma$  recursively; (59c) writes out the recursion:

$$\gamma(\mathbf{u}) = \gamma_d(\mathbf{u}_{1:d}) = \gamma_{d-1}(\mathbf{u}_{1:d-1})\gamma_d(\mathbf{u}_d|\mathbf{u}_{1:d-1}) \quad (59b)$$

$$= \gamma_1(\mathbf{u}_1) \prod_{i=2}^d \gamma_i(\mathbf{u}_i|\mathbf{u}_{1:i-1}). \quad (59c)$$

The normalised variant of  $\gamma_i$  will be called  $\pi_i$  and defined analogous to  $\pi$  in the previous section:

$$\pi_i(\mathbf{u}_{1:i}) = \frac{\gamma_i(\mathbf{u}_{1:i})}{Z_i}; \quad (60a)$$

$$Z_i = \int \gamma_i(\mathbf{u}_{1:i}) d\mathbf{u}_{1:i}. \quad (60b)$$

The proposal distribution  $\rho$  is factorised similarly to the target distribution:

$$\rho_i(\mathbf{u}_i|\mathbf{u}_{1:i-1}) = \frac{\rho_i(\mathbf{u}_{1:i})}{\rho_{i-1}(\mathbf{u}_{1:i-1})}; \quad (61a)$$

$$\begin{aligned} \rho(\mathbf{u}) &= \rho_d(\mathbf{u}_{1:d}) = \rho_{d-1}(\mathbf{u}_{1:d-1})\rho_d(\mathbf{u}_d|\mathbf{u}_{1:d-1}) \\ &= \rho_1(\mathbf{u}_1) \prod_{i=2}^d \rho_i(\mathbf{u}_i|\mathbf{u}_{1:i-1}). \end{aligned} \quad (61b)$$

This makes it possible to draw samples  $\mathbf{u}_{1:d}^{(s)}$  dimension per dimension. For the first dimension,  $\mathbf{u}_1^{(s)} \sim \rho_1$ . Then,  $\mathbf{u}_i^{(s)}|\mathbf{u}_{1:i-1}^{(s)} \sim \rho_i$  for dimensions  $i = 2, \dots, d$ . Each proposal factor  $\rho_i$  approximates target factor  $\gamma_i$ .

---

**procedure** SEQUENTIAL-IMPORTANCE-SAMPLING( $\gamma, \rho$ )  
**for** dimension  $i = 1 \dots d$  **do**  
  **for** sample index  $s = 1 \dots S$  **do**  
    Sample  $\mathbf{u}_i^{(s)} \sim \rho_i(\mathbf{u}_i | \mathbf{u}_{1:i-1}^{(s)})$ ;  
    Compute weight  $w_i(\mathbf{u}_{1:i}^{(s)}) = w_{i-1}(\mathbf{u}_{1:i-1}^{(s)}) \frac{\gamma_i(\mathbf{u}_i^{(s)} | \mathbf{u}_{1:i-1}^{(s)})}{\rho_i(\mathbf{u}_i^{(s)} | \mathbf{u}_{1:i-1}^{(s)})}$ .  
  **return** weighted samples  $\{w_d(\mathbf{u}_{1:d}^{(s)}), \mathbf{u}_{1:d}^{(s)}\}$ .

---

**Algorithm 2** Sequential importance sampling

Computing the importance weight of a sample can also be done dimension per dimension. Decomposing the weight function in (53) recursively, in factors  $w_i(\cdot | \cdot)$  related to  $\gamma_i(\cdot | \cdot)$  and  $\rho_i(\cdot | \cdot)$ :

$$\begin{aligned} w_i(\mathbf{u}_i | \mathbf{u}_{1:i-1}) &= \frac{w_i(\mathbf{u}_{1:i})}{w_{i-1}(\mathbf{u}_{1:i-1})} = \frac{\gamma_i(\mathbf{u}_{1:i}) \rho_{i-1}(\mathbf{u}_{1:i-1})}{\rho_i(\mathbf{u}_{1:i}) \gamma_{i-1}(\mathbf{u}_{1:i-1})} \\ &= \frac{\gamma_i(\mathbf{u}_i | \mathbf{u}_{1:i-1})}{\rho_i(\mathbf{u}_i | \mathbf{u}_{1:i-1})}, \end{aligned} \quad (62a)$$

$$\begin{aligned} w(\mathbf{u}) = w_d(\mathbf{u}_{1:d}) &= w_{d-1}(\mathbf{u}_{1:d-1}) w_d(\mathbf{u}_d | \mathbf{u}_{1:d-1}) \\ &= w_1(\mathbf{u}_1) \prod_{i=2}^d w_i(\mathbf{u}_i | \mathbf{u}_{1:i-1}). \end{aligned} \quad (62b)$$

The empirical distribution of  $\rho_i$  then is found from  $S$  samples drawn from  $\rho_i$ , analogously to the approximation of  $\tilde{\rho}$  in (50):

$$\tilde{\rho}_i = \frac{1}{S} \sum_s \delta_{\mathbf{u}_{1:i}^{(s)}}. \quad (63)$$

Using this empirical distribution, the empirical normalisation constant and the normalised weights are

$$\tilde{Z}_i = \int w_i(\mathbf{u}_{1:i}) \tilde{\rho}_i(\mathbf{u}_{1:i}) d\mathbf{u}_{1:i} = \frac{1}{S} \sum_s w_i(\mathbf{u}_{1:i}^{(s)}); \quad (64a)$$

$$\bar{w}_i^{(s)} = \frac{w_i(\mathbf{u}_{1:i}^{(s)})}{\sum_{s'} w_i(\mathbf{u}_{1:i}^{(s')})}. \quad (64b)$$

The empirical distribution derived from  $\pi_i$  then is, analogously to (55a),

$$\tilde{\pi}_i = \sum_s \bar{w}_i^{(s)} \delta_{\mathbf{u}_{1:i}^{(s)}}. \quad (65)$$

The complete algorithm for sequential importance sampling is described in Algorithm 2.

The final normalisation constant could be approximated as the average of the weights in the last step, using (55b). However, in section 4.3 re-sampling will be introduced. This at every step removes some samples and duplicates others, and overall weights

will not be available. To overcome this problem, the normalising constant  $Z$  of  $\gamma$ , defined in (51b), can be factorised into terms  $Z_i/Z_{i-1}$ , which can be approximated at every step:

$$Z \triangleq Z_d = Z_{d-1} \frac{Z_d}{Z_{d-1}} = Z_1 \prod_{i=2}^d \frac{Z_i}{Z_{i-1}}. \quad (66)$$

The fraction  $Z_i/Z_{i-1}$  can be written in terms of the normalised density for dimension  $i-1$  and the proposal distribution and the weight function for dimension  $i$  (applying (60a), (62a), and (61a)) as

$$\begin{aligned} \frac{Z_i}{Z_{i-1}} &= \frac{\int \gamma_i(\mathbf{u}_{1:i}) \, d\mathbf{u}_{1:i}}{\int \gamma_{i-1}(\mathbf{u}_{1:i-1}) \, d\mathbf{u}_{1:i-1}} = \frac{\int \gamma_{i-1}(\mathbf{u}_{1:i-1}) \gamma_i(\mathbf{u}_i | \mathbf{u}_{1:i-1}) \, d\mathbf{u}_{1:i}}{\int \gamma_{i-1}(\mathbf{u}_{1:i-1}) \, d\mathbf{u}_{1:i-1}} \\ &= \int \pi_{i-1}(\mathbf{u}_{1:i-1}) w_i(\mathbf{u}_i | \mathbf{u}_{1:i-1}) \rho_i(\mathbf{u}_i | \mathbf{u}_{1:i-1}) \, d\mathbf{u}_{1:i} \\ &= \int \frac{\pi_{i-1}(\mathbf{u}_{1:i-1})}{\rho_{i-1}(\mathbf{u}_{1:i-1})} w_i(\mathbf{u}_i | \mathbf{u}_{1:i-1}) \rho_i(\mathbf{u}_{1:i}) \, d\mathbf{u}_{1:i}. \end{aligned} \quad (67)$$

This can then be approximated at every step  $i$  using the empirical distribution  $\tilde{\rho}_i$  from (63):

$$\frac{\widetilde{Z}_i}{\widetilde{Z}_{i-1}} = \frac{1}{S} \sum_{s=1}^S \bar{w}_{1:i-1}^{(s)} w_i(\mathbf{u}_i^{(s)} | \mathbf{u}_{1:i-1}^{(s)}). \quad (68)$$

It is straightforward to see that this yields a consistent estimate of  $Z_i/Z_{i-1}$ . Note that as long as the samples are not resampled, this computation yields the exact same  $\widetilde{Z}_d$  as (55b).

### 4.3 Re-sampling

A problem with importance sampling is that some samples will be in low-probability regions. As the number of dimensions grows, the number of high-probability samples tends to shrink exponentially. As a measure for this problem, the variance of the sample weights is often used. Sequential importance sampling as presented so far does not do anything to produce lower variances.

A technique that does help to focus on higher-probability regions, and therefore does produce lower-variance weights is *re-sampling*. Re-sampling can be applied at every step to find a new empirical measure with unweighted samples from a set of weighted samples.

The unweighted samples can be written as a set of weights and samples:  $\{\bar{w}_i^{(s)}, \mathbf{u}_{1:i}^{(s)}\}$ . The empirical distribution for dimensions  $1 \dots i$  was given in (65):

$$\tilde{\pi}_i = \sum_{s=1}^S \bar{w}_i^{(s)} \delta_{\mathbf{u}_{1:i}^{(s)}}. \quad (69)$$

Re-sampling aims to find an approximation to this distribution with unweighted samples. The conceptually simplest way of doing this is to draw  $S$  samples from  $\tilde{\pi}_i$  and construct a new empirical distribution

$$\hat{\pi}_i = \sum_{s=1}^S \frac{N_i^{(s)}}{S} \delta_{\mathbf{u}_{1:i}^{(s)}}, \quad (70)$$

**procedure** SEQUENTIAL-IMPORTANCE-RE-SAMPLING( $\gamma, \rho$ )  
**for** dimension  $i = 1 \dots d$  **do**  
  **for** sample index  $s = 1 \dots S$  **do**  
    Sample  $\mathbf{u}_i^{(s)} \sim \rho_i(\mathbf{u}_i | \hat{\mathbf{u}}_{1:i-1}^{(s)})$ ;  
    Compute incremental weight  $w_i(\mathbf{u}_i^{(s)} | \mathbf{u}_{1:i-1}^{(s)}) = \frac{\gamma_i(\mathbf{u}_i^{(s)} | \mathbf{u}_{1:i-1}^{(s)})}{\rho_i(\mathbf{u}_i^{(s)} | \mathbf{u}_{1:i-1}^{(s)})}$ .  
    Compute  $\frac{\tilde{Z}_i}{Z_{i-1}} = \frac{1}{S} \sum_s w_i(\mathbf{u}_i^{(s)} | \mathbf{u}_{1:i-1}^{(s)})$ .  
     $\{\hat{\mathbf{u}}_{1:i}^{(s)}\} \leftarrow \text{RESAMPLE}(\{w_i(\mathbf{u}_i^{(s)} | \mathbf{u}_{1:i-1}^{(s)})\}, \mathbf{u}_{1:i}^{(s)})$ .  
  Compute  $\tilde{Z} = \tilde{Z}_1 \sum_{i=2}^d \frac{\tilde{Z}_i}{Z_{i-1}}$ .  
**return**  $(\{\hat{\mathbf{u}}_{1:d}^{(s)}\}, \tilde{Z})$ .

---

**Algorithm 3** *Sequential importance re-sampling*

where  $N_i^{(s)}$  is the number of times sample  $\mathbf{u}_{1:i}^{(s)}$  was drawn from  $\hat{\pi}_i$ , the (integer) number of offspring of sample  $\mathbf{u}_{1:i}^{(s)}$ . This is called *multinomial re-sampling*. However, the only requirement to a re-sampling method is that the expected value of the number of offspring of a sample is proportional to its weight:  $\mathcal{E}\{N_i^{(s)}\} = S\bar{w}_i^{(s)}$ .

Another way of generating  $N_{1:i}^{(s)}$  uses *systematic re-sampling* [17]. This uses uniformly distributed  $z \sim \text{Unif}[0, \frac{1}{S}]$ . New sample  $s'$  then is set equal to original sample  $s$  where  $\sum_{j=1}^{s-1} \bar{w}_i^{(j)} \leq z + s' < \sum_{j=1}^s \bar{w}_i^{(j)}$ .

The new empirical distribution can also be described as a list of unweighted samples  $\{\frac{1}{S}, \hat{\mathbf{u}}_{1:i}^{(s)}\}$  that contains  $N_i^{(s)}$  copies of original sample  $\mathbf{u}_{1:i}^{(s)}$ . The distribution then is

$$\hat{\pi}_i = \frac{1}{S} \sum_{s=1}^S \delta_{\hat{\mathbf{u}}_{1:i}^{(s)}}. \quad (71)$$

This makes it straightforward to introduce re-sampling at every step of the sequential importance sampling algorithm as described in section 4.2. After drawing samples and computing their weights, the set of samples is resampled to yield equally weighted samples  $\{\frac{1}{S}, \hat{\mathbf{u}}_{1:i}^{(s)}\}$ . This set is then used when drawing samples for the next iteration. The complete algorithm for sequential importance re-sampling is described in Algorithm 3.

#### 4.4 Sampling from the target distribution

An extension of sequential importance re-sampling was foreshadowed in (58). It concerns the case where for some dimensions it is possible to sample from the normalised target distribution  $\pi_i(\mathbf{u}_i | \mathbf{u}_{1:i-1})$ . For such a dimension  $i$ , the proposal distribution can be set to  $\rho_i(\mathbf{u}_i | \mathbf{u}_{1:i-1}) = \pi_i(\mathbf{u}_i | \mathbf{u}_{1:i-1})$ . The incremental weight function from (62a) then returns the same value, the incremental normalisation constant, for each point:

$$w_i(\mathbf{u}_i | \mathbf{u}_{1:i-1}) = \frac{\gamma_i(\mathbf{u}_i | \mathbf{u}_{1:i-1})}{\rho_i(\mathbf{u}_i | \mathbf{u}_{1:i-1})} = \frac{Z_i}{Z_{i-1}}. \quad (72)$$

---

```

procedure HYBRID-SEQUENTIAL-IMPORTANCE-RE-SAMPLING( $\gamma, \rho$ )
  for dimension  $i = 1 \dots d$  do
    if  $\rho_i(\mathbf{u}_i | \hat{\mathbf{u}}_{1:i-1}^{(s)}) \propto \gamma_i(\mathbf{u}_i | \hat{\mathbf{u}}_{1:i-1}^{(s)})$  then
      for sample index  $s = 1 \dots S$  do
        Sample  $\hat{\mathbf{u}}_i^{(s)} \sim \rho_i(\mathbf{u}_i | \hat{\mathbf{u}}_{1:i-1}^{(s)})$ ;
        Set  $\frac{\tilde{Z}_i}{Z_{i-1}} = w_i(\mathbf{u}_i | \mathbf{u}_{1:i-1})$  (for any  $\mathbf{u}_{1:i-1}$ ).
    else
      for sample index  $s = 1 \dots S$  do
        Sample  $\mathbf{u}_i^{(s)} \sim \rho_i(\mathbf{u}_i | \hat{\mathbf{u}}_{1:i-1}^{(s)})$ ;
        Compute incremental weight  $w_i(\mathbf{u}_i^{(s)} | \mathbf{u}_{1:i-1}^{(s)}) = \frac{\gamma_i(\mathbf{u}_i^{(s)} | \mathbf{u}_{1:i-1}^{(s)})}{\rho_i(\mathbf{u}_i^{(s)} | \mathbf{u}_{1:i-1}^{(s)})}$ .
      Compute  $\frac{\tilde{Z}_i}{Z_{i-1}} = \frac{1}{S} \sum_s w_i(\mathbf{u}_i^{(s)} | \mathbf{u}_{1:i-1}^{(s)})$ .
       $\{\hat{\mathbf{u}}_{1:i}^{(s)}\} \leftarrow \text{RESAMPLE}(\{w_i(\mathbf{u}_{1:i}^{(s)})\})$ .
    Compute  $\tilde{Z} = \tilde{Z}_1 \sum_{i=2}^d \frac{\tilde{Z}_i}{Z_{i-1}}$ .
  return  $(\{\hat{\mathbf{u}}_{1:d}^{(s)}\}, \tilde{Z})$ .

```

---

**Algorithm 4** Hybrid sequential importance re-sampling

This is useful because it removes the need to compute the density of the distribution at any point, or to re-sample the set of samples. In the case where  $\gamma_i = \pi_i = \rho_i$ ,  $w_i(\mathbf{u}_i | \mathbf{u}_{1:i-1}) = 1$  by definition.

Algorithm 4 specifies how sequential importance re-sampling, in algorithm 3 on the preceding page, can be extended to deal with dimensions for which it is possible to sample from the target distribution, but its density at a point cannot be computed.

## 5 Asymptotically exact likelihood evaluation

This section uses a different approach to approximating the corrupted speech distribution than standard model compensation. Model compensation methods, like the ones discussed in section 3, find a parametric form for recognition. However, no expression for the full density is needed: while recognising speech only the likelihood of vectors that are observed is required. Therefore, in this section the likelihood is approximated for a specific observation  $\mathbf{y}_t$ . Substituting  $\mathbf{y}_t$  for  $\mathbf{y}$  in (19d),

$$p(\mathbf{y}_t) = \iiint \delta_{f(\mathbf{x}, \mathbf{n}, \boldsymbol{\alpha})}(\mathbf{y}_t) p(\boldsymbol{\alpha}) d\boldsymbol{\alpha} p(\mathbf{n}) d\mathbf{n} p(\mathbf{x}) d\mathbf{x}. \quad (73)$$

To approximate this integral over  $\mathbf{x}$ ,  $\mathbf{n}$ , and  $\boldsymbol{\alpha}$ , this work will apply importance sampling, which was discussed in section 4. Importance sampling requires a *proposal distribution*, the distribution that the algorithm draws from instead of the actual distribution. It makes up for the difference by assigning each sample a weight. The proposal distributions needs to match the target density well, otherwise the weights will have a high variance, and too many samples will be required to arrive at a good estimate. The number of samples required grows exponentially with the number of dimensions.

The following section will rewrite (73) as an integral over the speech  $\mathbf{x}$  and the noise  $\mathbf{n}$ . To apply importance sampling, the proposal distribution will be either the prior, or an Algonquin-derived approximation of the posterior. Both essentially use a joint Gaussian distribution over the speech and the noise. This will turn out a disadvantage for representing the non-linear relationship between the speech and the noise given the observation. Section 5.2 will then introduce a transformation that straightens the curve in  $(\mathbf{x}, \mathbf{n})$ -space by introducing a new variable  $\mathbf{u}$  that represents positions on that curve. The integral over  $\mathbf{x}$ ,  $\mathbf{n}$ , and  $\alpha$  can then be expressed as an integral over  $\mathbf{u}$  and  $\alpha$ . This new expression is still exact, but more amenable to being approximated with importance sampling. After considering the one-dimensional case, the higher-dimensional space will use the same techniques for each dimension, and apply sequential importance re-sampling.

### 5.1 Importance sampling from a Gaussian

This section approximates the integration over  $\mathbf{x}$  and  $\mathbf{n}$ . Importance sampling requires that the integrand can be evaluated at any point  $(\mathbf{x}, \mathbf{n})$ . For this, part of the observation likelihood needs to be rewritten more explicitly. The corrupted speech likelihood is ((19b) and (19d) with  $\mathbf{y}_t$  substituted for  $\mathbf{y}$ ):

$$p(\mathbf{y}_t) = \int \int p(\mathbf{y}_t | \mathbf{x}, \mathbf{n}) p(\mathbf{n}) d\mathbf{n} p(\mathbf{x}) d\mathbf{x} \quad (74a)$$

$$= \int \int \int \delta_{f(\mathbf{x}, \mathbf{n}, \alpha)}(\mathbf{y}_t) p(\alpha) d\alpha p(\mathbf{n}) d\mathbf{n} p(\mathbf{x}) d\mathbf{x}. \quad (74b)$$

The observation  $\mathbf{y}_t$  is not deterministic given the speech and the noise, because the phase factor  $\alpha$  is random. However,  $\alpha$  is deterministic given  $\mathbf{x}$ ,  $\mathbf{n}$ , and  $\mathbf{y}_t$ , so that  $p(\mathbf{y}_t | \mathbf{x}, \mathbf{n})$  can be written in terms of the distribution of the phase factor. The phase factor that a specific setting of  $\mathbf{x}$ ,  $\mathbf{n}$ , and  $\mathbf{y}_t$  implies will be written  $\alpha(\mathbf{x}, \mathbf{n}, \mathbf{y}_t)$ . It is a standard result (in appendix A.1 and, e.g., [3], 11.1.1) that transforming the space of a probability distribution requires multiplying by the determinant of the Jacobian:

$$p(\mathbf{y}_t | \mathbf{x}, \mathbf{n}) = \left| \frac{\partial \alpha(\mathbf{x}, \mathbf{n}, \mathbf{y})}{\partial \mathbf{y}} \right|_{\mathbf{y}_t} \cdot p(\alpha(\mathbf{x}, \mathbf{n}, \mathbf{y}_t)), \quad (75)$$

where  $p(\alpha(\mathbf{x}, \mathbf{n}, \mathbf{y}))$  denotes the density of  $p(\alpha)$  at the value of  $\alpha$  implied by  $\mathbf{x}$ ,  $\mathbf{n}$ , and  $\mathbf{y}$ .

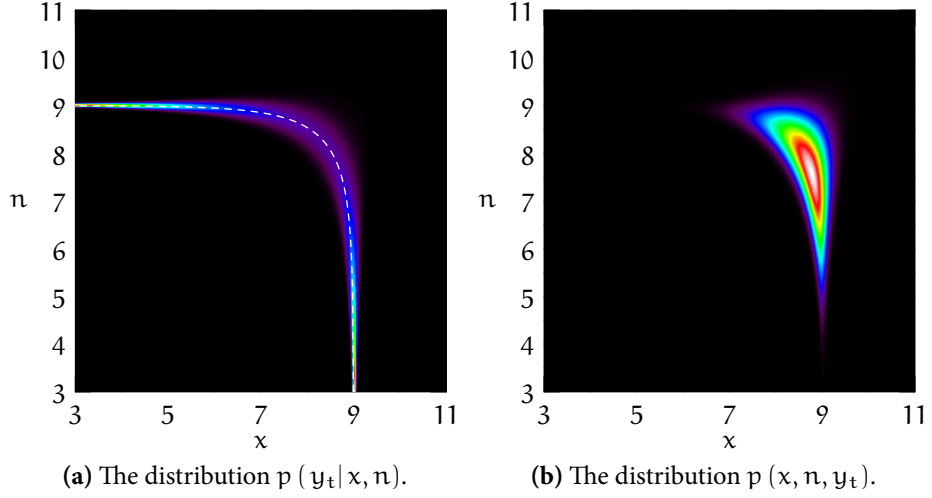
The value of the phase factor as a function of the other variables follows from (10). The relation is defined per coefficient (i.e. frequency bin)  $i$  of the variables:

$$\alpha_i = \frac{\exp(y_i) - \exp(x_i) - \exp(n_i)}{2 \exp(\frac{1}{2}x_i + \frac{1}{2}n_i)}, \quad (76a)$$

and its derivative with respect to  $y_i$  is

$$\frac{d\alpha(x_i, n_i, y_i)}{dy_i} = \frac{\exp(y_i)}{2 \exp(\frac{1}{2}x_i + \frac{1}{2}n_i)}. \quad (76b)$$

These make up the diagonal elements of the Jacobian  $\frac{\partial \alpha(\mathbf{x}, \mathbf{n}, \mathbf{y})}{\partial \mathbf{y}}$ , whose entries are 0 elsewhere.



**Figure 5** The distribution of the clean speech and noise for  $x \sim \mathcal{N}(10, 1)$ ;  $n \sim \mathcal{N}(9, 2)$ ;  $\alpha \sim \mathcal{N}(0, 0.04)$ ;  $y_t = 9$ .

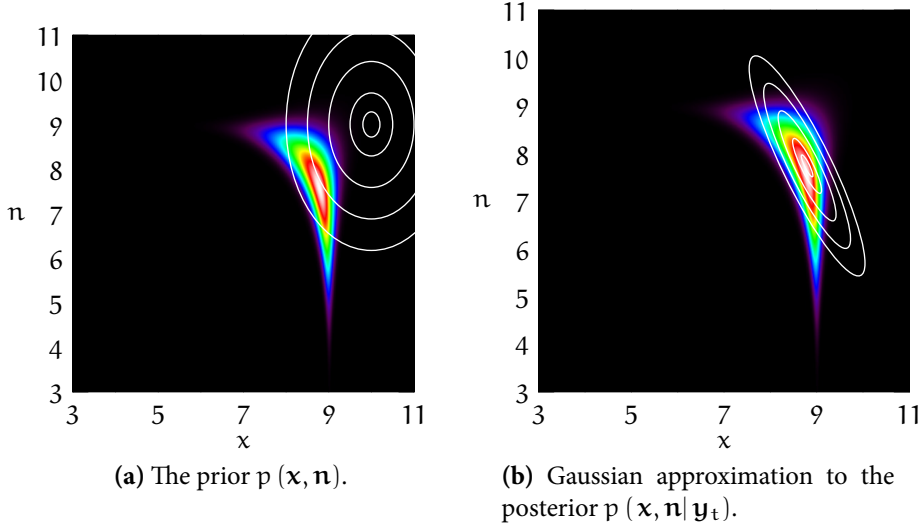
The the distribution of the corrupted speech in (74a) can then be written as

$$\begin{aligned}
 p(\mathbf{y}_t) &= \int \int p(\mathbf{y}_t | \mathbf{x}, \mathbf{n}) p(\mathbf{n}) d\mathbf{n} p(\mathbf{x}) d\mathbf{x} \\
 &= \int \int \left| \frac{\partial \alpha(\mathbf{x}, \mathbf{n}, \mathbf{y})}{\partial \mathbf{y}} \right|_{\mathbf{y}_t} \cdot p(\alpha(\mathbf{x}, \mathbf{n}, \mathbf{y}_t)) p(\mathbf{n}) d\mathbf{n} p(\mathbf{x}) d\mathbf{x} \\
 &= \int \int \left( \prod_i \frac{\exp(y_{t,i})}{2 \exp(\frac{1}{2}x_i + \frac{1}{2}n_i)} \right) \cdot p(\alpha(\mathbf{x}, \mathbf{n}, \mathbf{y}_t)) p(\mathbf{n}) p(\mathbf{x}) d\mathbf{n} d\mathbf{x} \\
 &\triangleq \int \int \gamma(\mathbf{x}, \mathbf{n}) d\mathbf{n} d\mathbf{x}. \tag{77}
 \end{aligned}$$

This expression is exact. Though the integrand, denoted with  $\gamma$ , is now straightforward to evaluate at any given point  $(\mathbf{x}, \mathbf{n})$  if  $p(\alpha)$  can be evaluated, the integral has no closed form. It can, however, be approximated using importance sampling with  $\gamma$  as the target density. Importance sampling requires a proposal distribution that is close to the target. Figure 5a illustrates a one-dimensional version of the density  $p(y_t | x, n)$ . The density lies around the curve that relates  $x$  and  $n$  for given  $\alpha$  and  $y_t$ , shown as a dashed line. This curve is given by  $\exp(x) + \exp(n) = \exp(y_t)$ . That the sum of the exponents of  $x$  and  $n$  is fixed causes a bend in the curve. Figure 5b shows the density in figure 5a multiplied by the priors of  $x$  and  $n$ , which gives  $p(x, n, y_t) = \gamma(x, n)$ .

To approximate the integral under  $\gamma$ , this section considers two options for the proposal distribution: the prior, and the Algonquin approximation to the posterior. Both are Gaussian.

A priori, the speech and the noise are independent. Given Gaussian prior distribu-



**Figure 6** The distribution of the clean speech and noise for  $\mathbf{x} \sim \mathcal{N}(10, 1)$ ;  $\mathbf{n} \sim \mathcal{N}(9, 2)$ ;  $\alpha \sim \mathcal{N}(0, 0.04)$ ;  $\mathbf{y} = 9$ .

tions for  $p(\mathbf{x})$  and  $p(\mathbf{n})$  (in (1) and (2), respectively), their joint prior becomes

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{n} \end{bmatrix} \sim \mathcal{N} \left( \begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_n \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_x & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_n \end{bmatrix} \right). \quad (78)$$

This Gaussian is shown in white lines on top of the actual posterior of  $\mathbf{x}$  and  $\mathbf{n}$  in figure 6a.

An alternative approach would be to use the Gaussian approximation to the posterior that the Algonquin algorithm (presented section 3.3) finds. Unlike the one in (78), this distribution does not model the speech and the noise as independent. Figure 6b shows it superimposed on the actual posterior.

Given either Gaussian proposal distribution  $\rho$ , importance sampling draws  $S$  samples  $(\mathbf{x}^{(s)}, \mathbf{n}^{(s)})$  from it and weights them to make up for the difference between proposal and target densities. This integral to be approximated is the normalisation constant of  $\gamma$ . The derivation is in section 4.1.1, and in particular (55b), but intuitively it can be written as

$$\int \int \gamma(\mathbf{x}, \mathbf{n}) \, d\mathbf{x} \, d\mathbf{n} = \int \int \frac{\gamma(\mathbf{x}, \mathbf{n})}{\rho(\mathbf{x}, \mathbf{n})} \rho(\mathbf{x}, \mathbf{n}) \, d\mathbf{x} \, d\mathbf{n} \simeq \sum_{s=1}^S \frac{\gamma(\mathbf{x}^{(s)}, \mathbf{n}^{(s)})}{\rho(\mathbf{x}^{(s)}, \mathbf{n}^{(s)})}, \quad (79)$$

$$(\mathbf{x}^{(s)}, \mathbf{n}^{(s)}) \sim \rho.$$

The fraction of the target and proposal densities,  $\gamma/\rho$ , gives the weight of the samples. This weight makes up for the difference between the two densities.

The main problem areas for either Gaussian as proposal distribution are the regions of space where proposal and target do not match well. Where the proposal distribution has a higher value than the target distribution, more samples will be drawn that are assigned lower weights: a waste of computational time. Conversely, where the proposal

distribution is much lower than the target distribution, samples will seldom be drawn, and when they do, they are assigned high weights. In this case, the number of samples that needs to be drawn to get sufficient coverage becomes very high.

These two problems are exacerbated by high dimensionality: for every dimension, either of these cases can occur. The probability that neither problem arises for a sample decreases exponentially, so that the number of samples required increases exponentially. Both approximations shown in figure 6 suffer from this problem, so that it is not feasible to apply them to a 24-dimensional log-spectral problem. The next section will therefore transform the space so that the target distribution per dimension can be approximated better.

## 5.2 Transformed-space sampling

The problem with the scheme in the previous section is the hard-to-approximate bend in the distribution of  $x$  and  $n$  given  $y_t$ . This section will overcome this by transforming the space, and then approximating the integral. Conceptually, this is similar to [29], which was discussed in section 3.4. However, there the approximation was constrained to one dimension. Here, the transformation is different, and the approximation uses sequential importance sampling. Initially, a one-dimensional space will be considered. Section 5.2.2 will discuss how to generalise this to the multi-dimensional case and how to deal with the additional complications that arise.

### 5.2.1 Single-dimensional

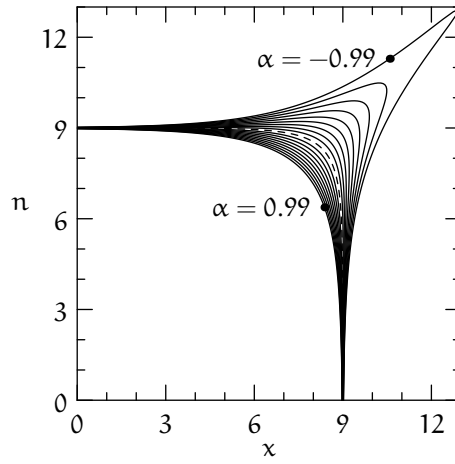
In one dimension, the mismatch function is ((10) without indices and with  $y_t$  substituted for  $y$ )

$$\exp(y_t) = \exp(x) + \exp(n) + 2\alpha \exp\left(\frac{1}{2}x + \frac{1}{2}n\right). \quad (80)$$

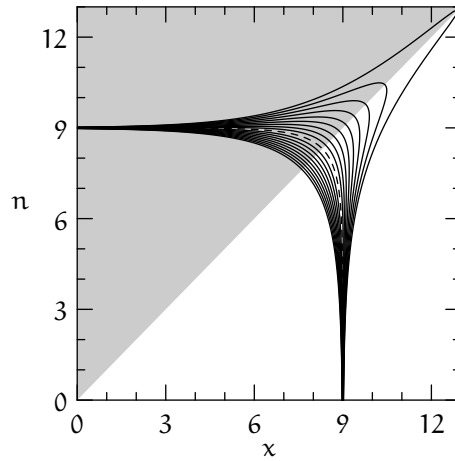
As discussed in section 2.2.1.1 on page 9,  $\alpha$  is a weighted average of cosines of the angle between the signals of the frequencies in one bin, and as such constrained to be between  $-1$  and  $+1$ .

Since this equality relates four variables deterministically, if three are known, then in many cases the fourth is known as well. The objective of this section is to find an approximation to  $p(y_t)$  for a given observation  $y_t$ . Since four variables are linked deterministically and one is known ( $y_t$ ), the integration will be over two variables. This was also the case in section 5.1. There, the obvious choice of integrating over  $x$  and  $n$  did not work out well because of the shape of the density in  $(x, n)$ -space. This section introduces a variable  $u$  that represents a pair  $(x, n)$  given  $y_t$  and  $\alpha$ . Changing  $u$  traverses the curve in  $(x, n)$ -space, so that the bend is not a problem any more. The integral will then be over  $\alpha$  and the new variable  $u$ .

A property of (80) that complicates the derivation of the transformed integral is that in some cases when three variables are known, the fourth can have two values. (80) is quadratic in  $\exp\left(\frac{1}{2}x\right)$  and  $\exp\left(\frac{1}{2}n\right)$ , so it can have two solutions for  $x$  and  $n$ . This can be seen in figure 7 on the facing page, which shows how  $x$  and  $n$  are related for various values of  $\alpha$ . For  $n = 10$ ,  $\alpha = 0.99$ , for example,  $x$  has two solutions.



**Figure 7** The relation between clean speech  $x$  and additive noise  $n$  for  $y_t = 9$  and evenly-spaced values of  $\alpha$ .



**Figure 8** The region of the  $(x, n)$  that the integral is explicitly derived for:  $x \leq n$ .

However, for given  $y_t$  and  $\alpha$ , it is possible to define one variable that unambiguously identifies a point on the curve. The substitute variable will be called  $u$  with

$$u = n - x. \quad (81)$$

The value of  $u$  is related to the signal-to-noise ratio. If  $u$  is a large negative number,  $x$  is close to  $y_t$  and  $n$  is also large and negative. This represents a very low signal-to-noise ratio. The converse is true if  $u$  is a large positive number: then  $n$  is close to  $y_t$  and  $x$  is large and negative.

The substitution will be used to define an integral that yields  $p(y_t)$ . First,  $p(y_t|\cdot)$  will be re-expressed using  $p(n|\cdot)$  or  $p(x|\cdot)$ . Since neither of these variables is known deterministically for all values of the other variables, the integral will be partitioned in two parts.  $n$  has one possible value given a setting for  $(x, y_t, \alpha)$  when  $x$  is constrained

to be smaller than  $n$ , which is the shaded region in figure 8 on the previous page. In the complementary region,  $x$  has one possible value given fixed  $(n, y, \alpha)$ . The likelihood can be written as a sum of both these regions:

$$p(y_t) = p(y_t, x \leq n) + p(y_t, n < x). \quad (82)$$

Because of the symmetry between these two regions, only the derivation for the region  $x \leq n$  will be given explicitly. The derivation for  $n < x$  is analogous.

The additive noise that follows from setting the variables  $x, y_t, \alpha$  will be denoted with  $n(x, y_t, \alpha)$ . This is deterministic in the region where  $x \leq n$ . The variable of the probability distribution will be changed from  $y_t$  to  $n$ . This requires multiplication by a Jacobian (see section A.1 on page 62 or, for example, [3], 11.1.1). This Jacobian, the partial derivative of  $n$  with respect to  $y$  and keeping  $x$  and  $\alpha$  fixed, will be written  $\left. \frac{\partial n(x, y, \alpha)}{\partial y} \right|_{y_t}$ , and be evaluated at  $y_t$ .

$$p(y_t, x \leq n | x, \alpha) = \left. \frac{\partial n(x, y, \alpha)}{\partial y} \right|_{y_t} \cdot 1(x \leq n) \cdot p(n(x, y_t, \alpha)). \quad (83)$$

Here,  $1(\cdot)$  denotes the indicator function, which evaluates to 1 if its argument is true, and 0 otherwise.  $p(n(x, y_t, \alpha))$  is the probability distribution of  $n$  evaluated at  $n(x, y_t, \alpha)$ , the value of  $n$  corresponding to the setting of  $(x, y_t, \alpha)$ .

The evaluation of the half of the likelihood for  $x \leq n$  can then be rewritten with (83) and by then replacing the variable of the integration by  $u$ . The predicate  $x \leq n$  is equivalent to  $0 \leq u$ .

$$\begin{aligned} p(y_t, x \leq n) &= \int p(\alpha) \int p(x) p(y_t, x \leq n | x, \alpha) dx d\alpha \\ &= \int p(\alpha) \int p(x) \cdot \left. \frac{\partial n(x, y, \alpha)}{\partial y} \right|_{y_t} \cdot 1(x \leq n) \cdot p(n(x, y_t, \alpha)) dx d\alpha \\ &= \int p(\alpha) \int_0^\infty \left. \frac{\partial x(u, y_t, \alpha)}{\partial u} \right| \cdot \left. \frac{\partial n(x, y, \alpha)}{\partial y} \right|_{y_t, x(u, y_t, \alpha)} \\ &\quad \cdot p(x(u, y_t, \alpha)) \cdot p(n(u, y_t, \alpha)) du d\alpha. \end{aligned} \quad (84)$$

Here,  $p(x(n, y_t, \alpha))$  is the probability distribution of  $x$  evaluated at  $x(n, y_t, \alpha)$ , the value of  $x$  corresponding to the setting of  $(n, y_t, \alpha)$ . Appendix D.1 on page 69 gives the derivations for the Jacobians in (84), and  $x(u, y_t, \alpha)$  and  $n(u, y_t, \alpha)$  for the region  $x \leq n$ . It appears (in (158c)) that the product of the Jacobians is  $-1$ . By substituting 1 for the absolute value of the product of the Jacobians in (158c) into (84), one half of the likelihood in (82) becomes

$$\begin{aligned} p(y_t, x \leq n) &= \int p(\alpha) \int_0^\infty \left. \frac{\partial x(u, y_t, \alpha)}{\partial u} \right| \cdot \left. \frac{\partial n(x, y, \alpha)}{\partial y} \right|_{y_t, x(u, y_t, \alpha)} \\ &\quad p(x(u, y_t, \alpha)) p(n(u, y_t, \alpha)) du d\alpha \\ &= \int p(\alpha) \int_0^\infty p(x(u, y_t, \alpha)) p(n(u, y_t, \alpha)) du d\alpha. \end{aligned} \quad (85a)$$

The integral of  $u$  is over the area where  $0 \leq u$ , i.e. where  $x \leq n$ . The equivalent integration over the region where  $u < 0$  could be derived by swapping  $x$  and  $n$ , and

replacing  $u$  by  $-u$ . Applying this to all derivations in section D.1 on page 69 and to (85a) yields the other half of the likelihood in (82). Because of the symmetry of  $n$  and  $x$  and because the Jacobians cancel out, the result is identical to (85a) save for the range of  $u$ :

$$p(y_t, n < x) = \int p(\alpha) \int_{-\infty}^0 p(x(u, y_t, \alpha)) p(n(u, y_t, \alpha)) du d\alpha. \quad (85b)$$

The sum of (85a) and (85b) yields the total likelihood of  $y_t$ . The integrand will be called  $\gamma$ .

$$\begin{aligned} p(y_t) &= p(y_t, x \leq n) + p(y_t, n < x) \\ &= \int p(\alpha) \int_{-\infty}^{\infty} p(x(u, y_t, \alpha)) p(n(u, y_t, \alpha)) du d\alpha \\ &\triangleq \int p(\alpha) \int \gamma(u|\alpha) du d\alpha \end{aligned} \quad (86)$$

$$\triangleq \iint \gamma(u, \alpha) du d\alpha. \quad (87)$$

This derivation is exact and holds for any form of priors for the speech and noise  $p(x)$  and  $p(n)$ . Just like after rewriting  $p(y_t)$  in (77), the integrand can be evaluated at any given point  $(u, \alpha)$ , assuming that  $p(\alpha)$  can be evaluated, but the integral has no closed form. The outer integral is straightforward to approximate with plain Monte Carlo (see section 4.1 on page 23). This works by drawing samples  $\alpha^{(s)}$  from  $p(\alpha)$  and averaging over approximations for the inner integral given  $\alpha^{(s)}$ . The problem with approximating the inner integral is that it is impossible to draw samples from  $\gamma(u|\alpha)$ . Therefore, importance sampling is necessary. This requires a proposal distribution  $\rho(u|\alpha)$  that it is possible to draw samples from, and is close to  $\gamma$ . Section 4.1.1 on page 23 has given a detailed description of importance sampling. However, intuitively the double integral can be replaced by a summation over  $S$  samples  $\alpha^{(s)}$  from  $p(\alpha)$  and corresponding samples for  $u^{(s)}$  drawn from  $\rho(u|\alpha^{(s)})$ :

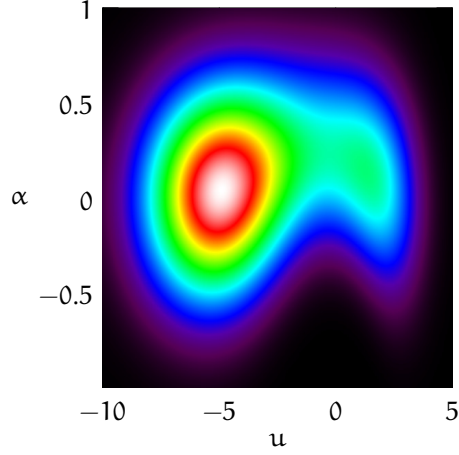
$$\begin{aligned} \int p(\alpha) \int \gamma(u|\alpha) du d\alpha &= \int p(\alpha) \int \frac{\gamma(u|\alpha)}{\rho(u|\alpha)} \rho(u|\alpha) du d\alpha \\ &\simeq \frac{1}{S} \sum_{s=1}^S \int \frac{\gamma(u|\alpha^{(s)})}{\rho(u|\alpha^{(s)})} \rho(u|\alpha^{(s)}) du \end{aligned} \quad (88a)$$

$$\simeq \frac{1}{S} \sum_{s=1}^S \frac{\gamma(u^{(s)}|\alpha^{(s)})}{\rho(u^{(s)}|\alpha^{(s)})}, \quad (88b)$$

$$\alpha^{(s)} \sim p(\alpha), \quad u^{(s)} \sim \rho(u|\alpha^{(s)}).$$

The next section will detail the shape of  $\gamma(u|\alpha^{(s)})$  and find appropriate forms for  $\rho(u|\alpha^{(s)})$  for it.

**5.2.1.1 The shape of the integrand** To apply importance sampling, a proposal distribution is required, which will be tailored to the parameters of the target distribution. Samples will be drawn from the proposal distribution which therefore by definition must be normalised. The target density  $\gamma(u, \alpha)$ , however, does not need to be normalised, nor can it be. The importance sampling algorithm finds samples, their



**Figure 9** The density of  $\gamma(u, \alpha) = p(\alpha) \gamma(u|\alpha)$  for  $y = 9, x \sim \mathcal{N}(7, 1), n \sim \mathcal{N}(4, 4), \sigma_\alpha^2 = 0.13$ .

weights, and an approximation to  $\gamma$ 's normalisation constant. The interest here is in the integral, which is the normalisation constant. The approximation of it is the average over the weights. Since the weights are the fraction of the target and the proposal densities, the closer the densities' shapes, the better the approximation. This section will find proposal distributions with well-matching shapes. The scaling of the density graphs in this section will be arbitrary.

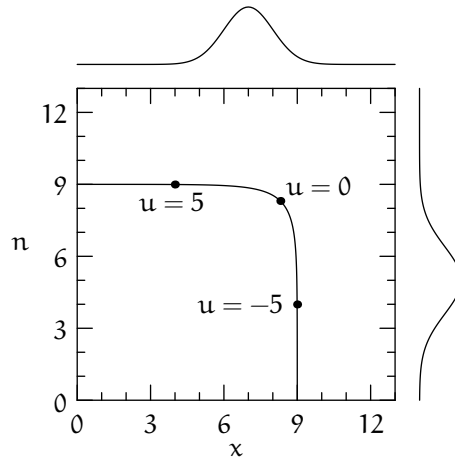
So far, the derivation has not assumed any specific distributions for  $x$  or  $n$ , and has been valid for any distribution. However, for different distributions, different proposal functions are required. With Gaussian for the speech and noise, the integrand becomes

$$\gamma(u, \alpha) = p(\alpha) \mathcal{N}(x(u, \alpha, y_t); \mu_x, \sigma_x^2) \mathcal{N}(n(u, \alpha, y_t); \mu_n, \sigma_n^2). \quad (89)$$

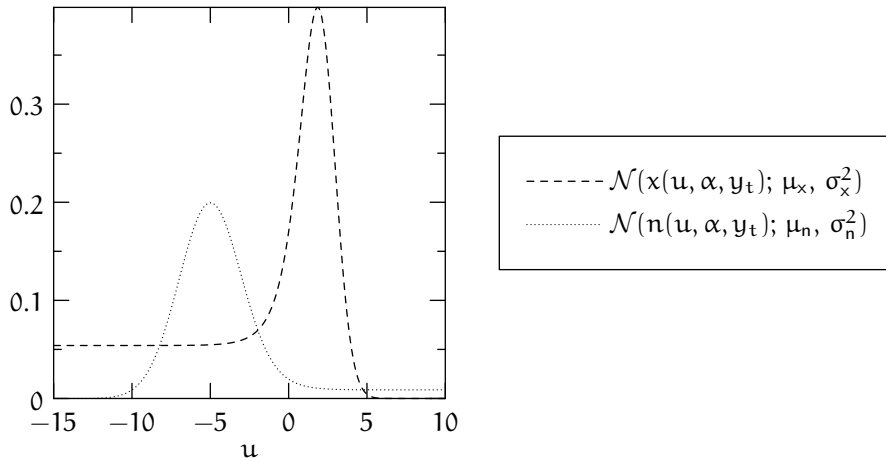
Figure 9 gives an example of the target distribution  $\gamma(u, \alpha)$ . As shown in (88b), samples for one dimension,  $\alpha$ , can be drawn from the correct distribution. It is the other dimension,  $u$ , that requires importance sampling, and therefore a proposal distribution  $\rho$ . The following examples will assume the mode of  $p(\alpha)$ ,  $\alpha = 0$ , and consider representative shapes for  $\gamma(u|\alpha = 0)$ .

$\gamma(u|\alpha)$  consists of a factor  $\mathcal{N}(x(u, \alpha, y_t); \mu_x, \sigma_x^2)$  related to the clean speech and a factor  $\mathcal{N}(n(u, \alpha, y_t); \mu_n, \sigma_n^2)$  related to the noise. Both terms are Gaussians, but the variables of the Gaussians ( $x$  and  $n$ ) are non-linear functions of  $u$ . Figure 10 on the facing page depicts the relationship between  $x$  and  $n$ . Different values of  $u$  represent different positions on the curve. When  $u$  is negative,  $n$  tends towards  $u$  and  $x$  tends towards  $y$ . When  $u$  is positive,  $x$  tends towards  $-u$  and  $n$  tends towards  $y$ . Around  $u = 0$  there is a soft cut-off. This graph provides an intuitive connection to noise masking schemes that assume that either the speech, or the noise dominates. This would yield a curve with a sharp angle, so that always either the speech or the noise equals the observation.

The two factors of  $\gamma(u|\alpha)$  are the Gaussians depicted on top and on the side of the graph. They are evaluated at the appropriate values of  $x$  and  $n$ . When the Gaussians are plotted with respect to  $u$ , the soft cut-off leads to a Gaussian distribution that is



**Figure 10** Values of  $x, n$  for  $y_t = 9, \alpha = 0$ , and varying  $u$ . At the top of the frame  $\mathcal{N}(x(u, \alpha, y_t); \mu_x, \sigma_x^2)$  in  $\gamma$ ; right  $\mathcal{N}(n(u, \alpha, y_t); \mu_n, \sigma_n^2)$ .



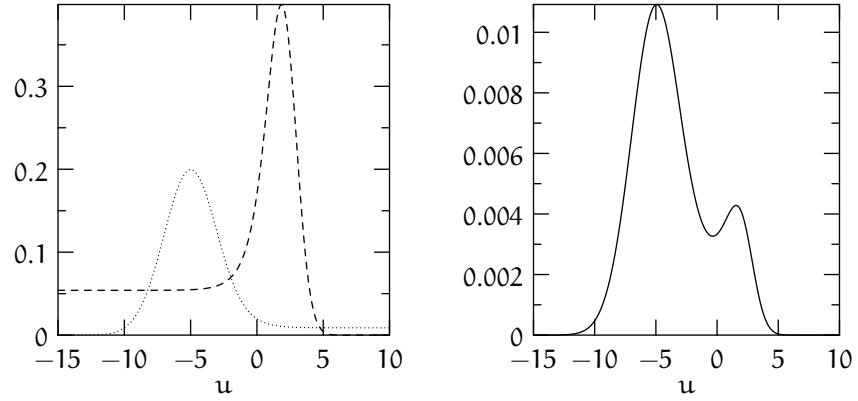
**Figure 11** The factors of  $\gamma(u | \alpha = 0)$  separately for  $y_t = 9, x \sim \mathcal{N}(7, 1), n \sim \mathcal{N}(4, 4)$ .

infinitely extended on one side. Figure 11 illustrates the shape of the two factors. As  $u$  tends to  $-\infty$ ,  $x$  tends to  $y$ . In this example,  $\mathcal{N}(x(u, \alpha, y_t); \mu_x, \sigma_x^2)$  therefore tends to

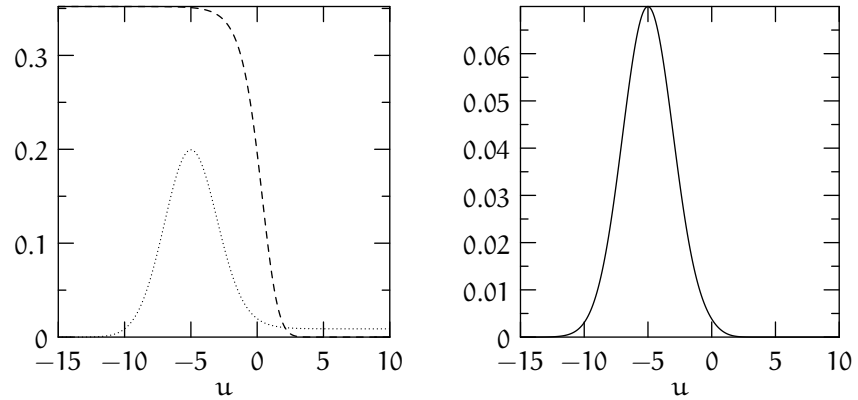
$$\mathcal{N}(x(u, \alpha, y_t); \mu_x, \sigma_x^2) \rightarrow \mathcal{N}(y; \mu_x, \sigma_x^2) = \mathcal{N}(9; 7, 1) = e^{-2}/\sqrt{2\pi} \approx 0.054. \quad (90)$$

$\mathcal{N}(x(u, \alpha, y_t); \mu_x, \sigma_x^2)$  in figure 11 therefore is a Gaussian-like distribution with a soft cut-off on its left tail, so that it converges to a non-zero constant. Analogously,  $\mathcal{N}(n(u, \alpha, y_t); \mu_n, \sigma_n^2)$  is Gaussian-like but cut off at its right tail, where it converges to a non-zero constant.

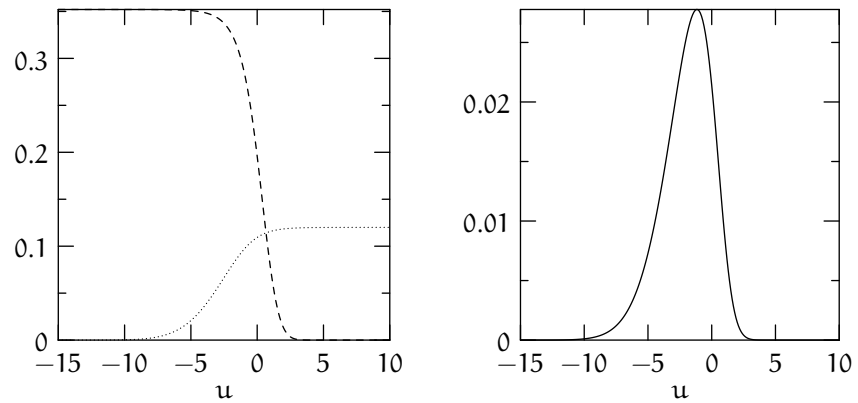
Figure 12 on the following page shows examples for the three types of shape of  $\gamma(u | \alpha = 0)$ . Each time, the left graph contains the shape of the two factors, and the



(a)  $\gamma$  for  $y = 9, x \sim \mathcal{N}(7, 1), n \sim \mathcal{N}(4, 4)$ .



(b)  $\gamma$  for  $y = 9, x \sim \mathcal{N}(9.5, 1), n \sim \mathcal{N}(4, 4)$ .



(c)  $\gamma$  for  $y = 9, x \sim \mathcal{N}(9.5, 1), n \sim \mathcal{N}(10, 10)$ .

**Figure 12**  $\gamma(u | \alpha = 0)$  for different cases: left the two factors, right their product.

right graph their product. Figure 12a uses the example from figure 11. The integral, the product of two cut-off Gaussians, is bimodal. When  $u$  tends to  $\pm\infty$ , one term of  $\gamma$  tends to a non-zero constant, but the other one tends to 0.  $\gamma$  therefore tends to 0 as well.<sup>4</sup>

In figure 12b,  $\mu_x > y$ , so that the graph of  $\mathcal{N}(x(u, \alpha, y_t); \mu_x, \sigma_x^2)$  is cut off right of its maximum. The product is similar to  $\mathcal{N}(n(u, \alpha, y_t); \mu_n, \sigma_n^2)$ , except that the right tail goes to zero. The result is almost Gaussian.

In the last example, figure 12c,  $\mu_x$  as well as  $\mu_n$  are greater than  $y$ , so that both Gaussians are cut off before their maximum. Their product has a lop-sided Gaussian-like shape around the point of the soft cut-off, by definition  $u = 0$ .

**5.2.1.2 Importance sampling from  $\gamma(u|\alpha)$**  Since  $\gamma(u|\alpha)$  cannot be integrated directly, an approximation is necessary. Importance sampling can provide this. It requires a proposal distribution that exists everywhere where  $\gamma(u|\alpha)$  exists, but may differ in magnitude, or overestimate its cover.

A proposal distribution that it is simple to draw a sample from is a Gaussian mixture model. To find a mixture of Gaussian densities that approximates  $\gamma$ , the three types of  $\gamma$  from figure 12 are considered. Because  $\gamma$  has these different shapes, the approximation will be different for each of these cases. the proposal distributions must be defined over  $u$ , so that it is useful to find the value of  $u$  corresponding to a specific setting of  $x$ ,  $\alpha$ , and  $y_t$  (and  $n$ ,  $\alpha$ , and  $y_t$ ). This will be denoted  $u(x, \alpha, y_t)$  (and  $u(n, \alpha, y_t)$ ). The expressions are derived in appendix D.4, (176) and (177f).

Figure 13 on the next page shows the proposal distributions. Their magnitudes are scaled to make the areas under the target and proposal densities equal. The proposal distribution is chosen differently depending on the mean of the terms of  $\gamma$  as follows:

1.  $\mu_x < y$  and  $\mu_n < y$ . This produces a shape of  $\gamma$  as in figure 12a. The shape of  $\gamma$  being close to the product of two Gaussians, a Gaussian mixture model with the parameters of these two Gaussians would form a good proposal distribution.

As a proposal distribution, a mixture of two Gaussians is chosen with means at the approximate modes, and covariances set to  $\sigma_x^2$  and  $\sigma_n^2$ , respectively. These Gaussians are illustrated in figure 13a. They approximate the terms  $\mathcal{N}(n(u, \alpha, y_t); \mu_n, \sigma_n^2)$  and  $\mathcal{N}(x(u, \alpha, y_t); \mu_x, \sigma_x^2)$  with

$$\mathcal{N}(u; u(\mu_n, \alpha, y), \sigma_n^2); \quad (91a)$$

$$\mathcal{N}(u; u(\mu_x, \alpha, y), \sigma_x^2). \quad (91b)$$

As was seen in figure 11 on page 39, each Gaussian is essentially scaled by the extended tail of the other one. The weights of the Gaussians of the proposal distribution can be set to the value that the tail of the other one converges to, which can be computed as in (90):

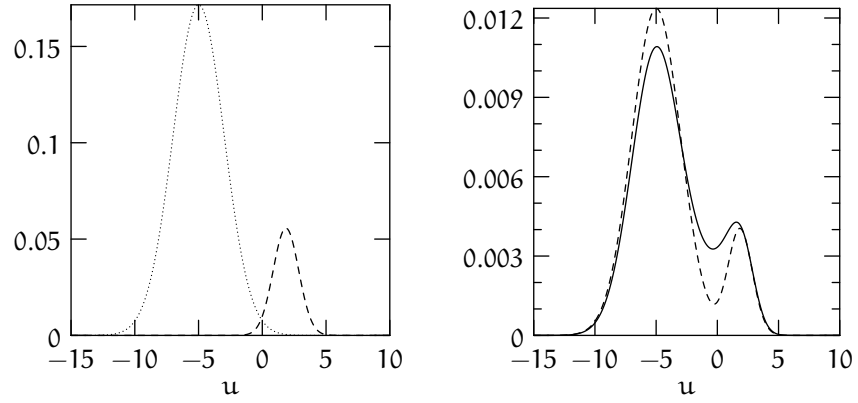
$$c_n \propto \mathcal{N}(y; \mu_x, \sigma_x^2); \quad (92a)$$

$$c_x \propto \mathcal{N}(y; \mu_n, \sigma_n^2). \quad (92b)$$

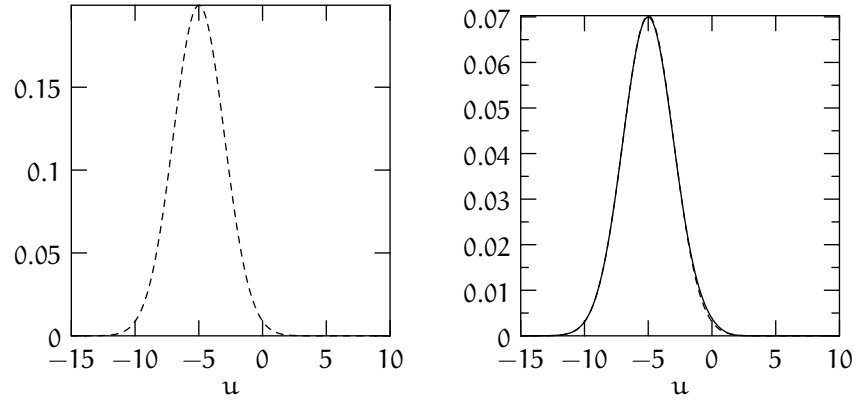
These weights are normalised so that they sum to 1. The distribution becomes

$$\rho = c_n \mathcal{N}(u; u(\mu_n, y), \sigma_n^2) + c_x \mathcal{N}(u; u(\mu_x, y), \sigma_x^2). \quad (93)$$

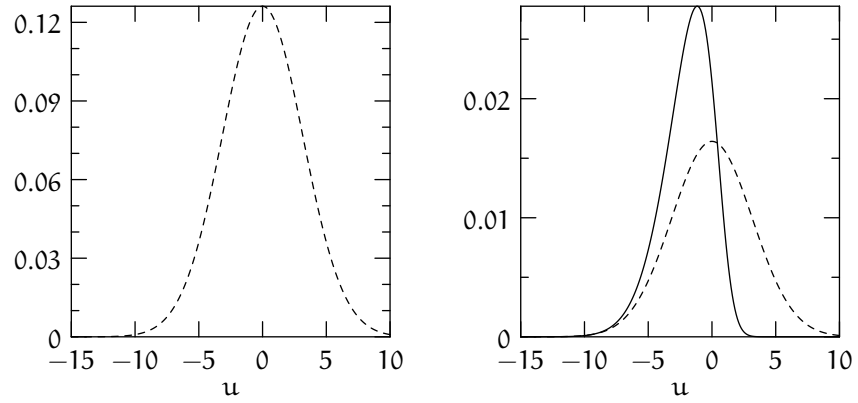
<sup>4</sup>This must be true also because  $\int \gamma(u|\alpha = 0) du$  is equal to  $p(y_t)$  evaluated at a point, which cannot be infinite if either  $\sigma_x^2$  or  $\sigma_n^2$  is non-zero.



(a)  $\gamma$  for  $y = 9, x \sim \mathcal{N}(7, 1), n \sim \mathcal{N}(4, 4)$ .



(b)  $\gamma$  for  $y = 9, x \sim \mathcal{N}(9.5, 1), n \sim \mathcal{N}(4, 4)$ .



(c)  $\gamma$  for  $y = 9, x \sim \mathcal{N}(9.5, 1), n \sim \mathcal{N}(10, 10)$ .

**Figure 13** The proposal distribution for  $\gamma(u)$  for different cases: left the components of the proposal distribution, right  $\gamma$  (solid line) and the proposal distribution (dashed line, scaled arbitrarily).

2.  $\mu_x > y$  and  $\mu_n < y$  (or its mirror image,  $\mu_x < y$  and  $\mu_n > y$ ). Figure 12b on page 40 has shown that  $\mathcal{N}(x(u, \alpha, y_t); \mu_x, \sigma_x^2)$  is cut off before its peak, and converges to its maximum in the limit as  $u \rightarrow -\infty$ . This results in a Gaussian distribution except for one tail. The proposal distribution is therefore set to this Gaussian:

$$\rho = \mathcal{N}(u; u(\mu_n, y), \sigma_n^2). \quad (94)$$

Figure 13b on the facing page shows the near-perfect match of this proposal distribution.

3.  $\mu_n > y$  and  $\mu_x > y$ . Both terms of  $\gamma$  are cut off before their peaks, resulting in a shape as in figure 12c on page 40. The product is a distribution around  $u = 0$  with Gaussian-like tails, one derived from  $\mathcal{N}(n(u, \alpha, y_t); \mu_n, \sigma_n^2)$  and another one derived from  $\mathcal{N}(x(u, \alpha, y_t); \mu_x, \sigma_x^2)$ . The proposal distribution is therefore set to a Gaussian with mean 0. Its variance is set to the largest of the variances of the speech and the noise:

$$\rho = \mathcal{N}(u; 0, \max(\sigma_n^2, \sigma_x^2)). \quad (95)$$

As figure 13c on the facing page shows, this provides good coverage but over-estimation on part of the space. This means that some samples will receive a very low weight.

Thus, by transforming the space of the integration from  $(x, n)$  to  $(u, \alpha)$ , much better proposal distributions for importance sampling can be found. The sample weights will therefore vary less, so that good approximations to the integral will be found with a much smaller number of samples. The next section will extend the techniques applied in this chapter to the multi-dimensional case.

### 5.2.2 Multi-dimensional

In this work, the log-spectral domain is used. This has the advantage that the interaction between the speech and the noise can be modelled separately per dimension. However, there are strong correlations between log-spectral coefficients. Therefore, the Gaussian priors for the speech and the noise have full covariance matrices. This section will build on the techniques used in the previous section. It will generalise the transformation of the integral to multi-dimensional  $(u, \alpha)$ . Rather than normal importance sampling, it will then apply sequential importance re-sampling to approximate the integral.

The relations between the single-dimensional variables in the previous section hold per dimension for the multi-variate case. The substitute variable  $u$  that is introduced to represent a point  $(x, n)$  given observation  $y_t$  and phase factor vector  $\alpha$  is therefore defined as

$$u = n - x. \quad (96)$$

The expressions for  $x(u, \alpha, y)$  and  $n(u, \alpha, y)$ , the values for the speech and noise that result from setting  $(u, \alpha, y)$ , are given in appendix D.2.

There was a complication in transforming the one-dimensional integral in section 5.2.1 from  $(x, n)$  to  $(u, \alpha)$ : for some  $x$ , multiple values for  $n$  were possible, and

vice versa. Because transforming the integral needed a deterministic link, the integral was split into two parts, for two regions of  $(\mathbf{x}, \mathbf{n})$ -space. In the multi-dimensional case it is necessary to do this for each of the dimensions. Appendix D.2 gives the full derivation. The integration is therefore first split up into conditional distributions per dimension  $i$ , and then into regions. The integrals for the two regions over  $(x_i, n_i)$  are rewritten as an integral over  $(u_i, \alpha_i)$ . Collapsing the dimensions (see equation (167)) then yields an unsurprising generalisation of (87):

$$p(\mathbf{y}_t) = \int p(\boldsymbol{\alpha}) \int p(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t)) p(\mathbf{n}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t)) d\mathbf{u} d\boldsymbol{\alpha} \quad (97a)$$

$$\triangleq \int \int \gamma(\mathbf{u}, \boldsymbol{\alpha}) d\mathbf{u} d\boldsymbol{\alpha}, \quad (97b)$$

and it is convenient to factorise the integrand  $\gamma(\mathbf{u}, \boldsymbol{\alpha})$  as

$$\gamma(\mathbf{u}, \boldsymbol{\alpha}) = \gamma(\boldsymbol{\alpha})\gamma(\mathbf{u}|\boldsymbol{\alpha}); \quad (97c)$$

$$\gamma(\boldsymbol{\alpha}) = p(\boldsymbol{\alpha}); \quad (97d)$$

$$\gamma(\mathbf{u}|\boldsymbol{\alpha}) = p(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t)) p(\mathbf{n}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t)). \quad (97e)$$

This derivation is valid whatever the form of the speech and noise priors,  $p(\mathbf{x})$  and  $p(\mathbf{n})$ . In this work, they are Gaussians with (repeated from (1) and (2))

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x); \quad \mathbf{n} \sim \mathcal{N}(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n), \quad (98)$$

with full covariance matrices  $\boldsymbol{\Sigma}_x$  and  $\boldsymbol{\Sigma}_n$ .  $\gamma(\mathbf{u}|\boldsymbol{\alpha})$  then becomes

$$\gamma(\mathbf{u}|\boldsymbol{\alpha}) = \mathcal{N}(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t); \boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x) \mathcal{N}(\mathbf{n}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t); \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n), \quad (99a)$$

so that

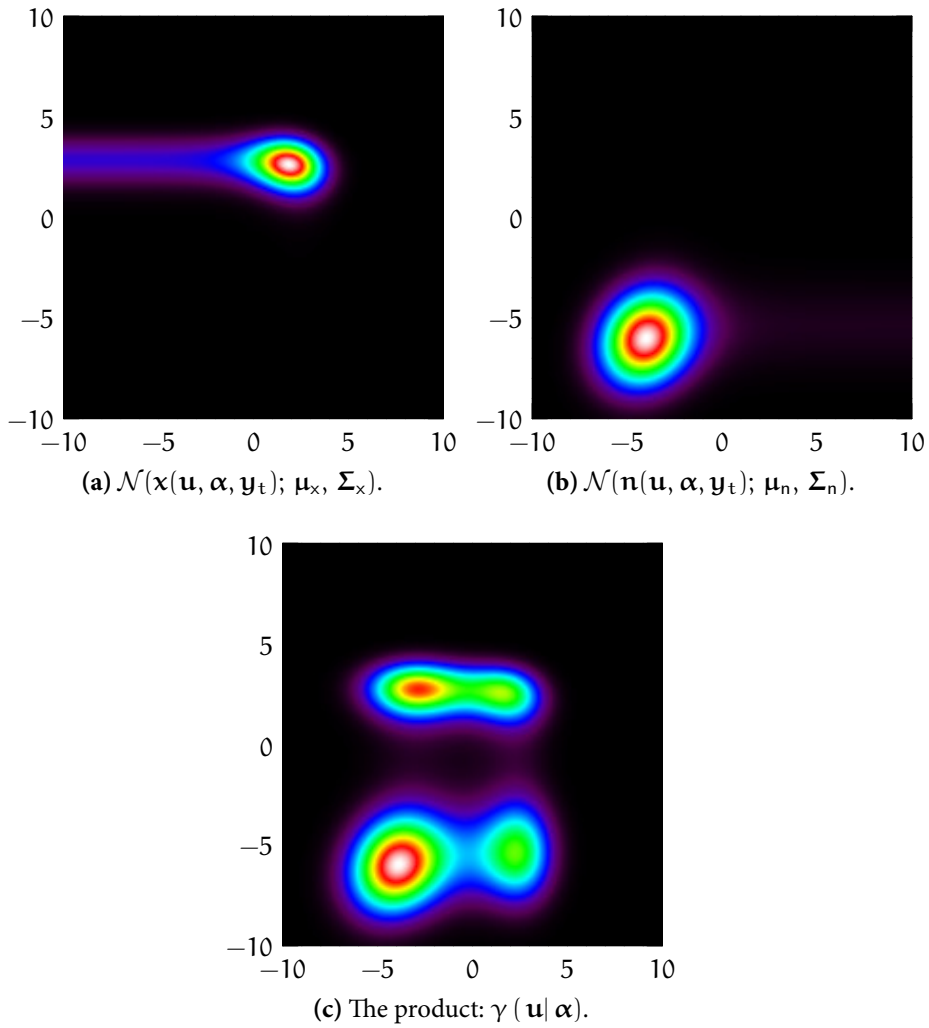
$$\gamma(\mathbf{u}, \boldsymbol{\alpha}) = p(\boldsymbol{\alpha}) \mathcal{N}(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t); \boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x) \mathcal{N}(\mathbf{n}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t); \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n). \quad (99b)$$

To approximate this integral, conventional importance sampling, discussed in full detail in section 4.1.1, could be used. Importance sampling finds weighted samples that approximate a target density, and, as a by-product, the density's normalisation constant. The result of the integral in (97b) is the normalisation term of  $\gamma$ . Just like in section 5.1, the integration over two variables can intuitively be approximated by drawing samples  $(\mathbf{u}^{(s)}, \boldsymbol{\alpha}^{(s)})$  from a proposal distribution  $\rho$ :

$$\begin{aligned} \int \int \gamma(\mathbf{u}, \boldsymbol{\alpha}) d\mathbf{u} d\boldsymbol{\alpha} &= \int \int \frac{\gamma(\mathbf{u}, \boldsymbol{\alpha})}{\rho(\mathbf{u}, \boldsymbol{\alpha})} \rho(\mathbf{u}, \boldsymbol{\alpha}) d\mathbf{u} d\boldsymbol{\alpha} \\ &\simeq \frac{1}{S} \sum_{s=1}^S \frac{\gamma(\mathbf{u}^{(s)}, \boldsymbol{\alpha}^{(s)})}{\rho(\mathbf{u}^{(s)}, \boldsymbol{\alpha}^{(s)})}, \quad (\mathbf{u}^{(s)}, \boldsymbol{\alpha}^{(s)}) \sim \rho. \end{aligned} \quad (100)$$

For this to work well in practice, the proposal distribution needs to be close to the target density, the integrand.

Figure 14 illustrates how the shape of the integrand  $\gamma(\mathbf{u}|\boldsymbol{\alpha} = \mathbf{0})$  generalises to two dimensions of  $\mathbf{u}$ . The principles are the same as the one-dimensional case in figure 11 on page 39. Figure 14a contains the factor of  $\gamma$  deriving from the speech prior,  $\mathcal{N}(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t); \boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)$ , and figure 14b the same for the noise prior. They are again



**Figure 14** The integrand  $\gamma(\mathbf{u} | \boldsymbol{\alpha})$  for  $\boldsymbol{\alpha} = \mathbf{0}$ : the two factors, and their product.

Gaussians with a soft cut-off, this time in two directions. By choosing the slightly contrived parameter setting

$$\mathbf{x} \sim \mathcal{N}\left(\begin{bmatrix} 7 \\ 6.3 \end{bmatrix}, \begin{bmatrix} 1 & -0.1 \\ -0.1 & 0.5 \end{bmatrix}\right); \quad \mathbf{n} \sim \mathcal{N}\left(\begin{bmatrix} 5 \\ 3 \end{bmatrix}, \begin{bmatrix} 2 & 0.3 \\ 0.3 & 2 \end{bmatrix}\right); \quad \mathbf{y}_t = \begin{bmatrix} 9 \\ 9 \end{bmatrix}, \quad (101)$$

the product of the two factors, in figure 14c, turns out to have four maxima. In general, for  $d$  dimensions, the integrand can have  $2^d$  modes. Even though this is unlikely to occur often in practice, it is hard to formulate a proposal distribution for importance sampling. The proposal distribution would need to be close to integrand, and it must be possible to draw samples from it. A mixture of Gaussians, for example, could need as many components as the integrand has maxima. However, rather than applying normal importance sampling, the integrand could be factorised in dimensions for sequential importance sampling.

**5.2.2.1 Per-dimension sampling** Section 4.2 on page 25 has discussed the theory of sequential importance sampling. In itself, it is an instantiation of importance sampling for multiple dimensions. First, it draws a set of samples from a distribution over the first dimension, and assigns the samples a weight. Then, for each dimension it extends every partial sample with a value drawn given the value for previous dimensions of that sample. The advantage of this formulation is that between dimensions it allows for re-sampling: duplicating some samples from the set and removing other ones. This concentrates the samples on the higher-probability areas.

To be able to apply sequential importance sampling, the target density needs to be factorised into dimensions. If the feature space is  $d$ -dimensional, the integration is over  $2d$  dimensions:  $\alpha_1, \dots, \alpha_d, u_1, \dots, u_d$ . It is important to realise that there is no need for the factors to represent conditional probability distributions, normalised or not. It is true that the most informative weights after each dimension  $i$  would arise if factors  $1 \dots i$  combined to form the (potentially unnormalised) marginal of partial sample  $u_{1:i}$ . Re-sampling would then be most effective. By definition, the factors would be (unnormalised) conditionals. However, this is not a requirement, and this work will compare two different factorisations, both of which should be close to the actual conditionals.

Since the phase factor coefficients are independent (see section 2.2.1.1 on page 9), for  $\gamma(\boldsymbol{\alpha})$  an obvious factorisation is

$$\gamma(\boldsymbol{\alpha}) = \gamma_1(\alpha_1) \cdots \gamma_d(\alpha_d) = \prod_{i=1}^d \gamma_i(\alpha_i), \quad \gamma_i(\alpha_i) = p(\alpha_i). \quad (102a)$$

Other factorisations are possible, but since the phase factor coefficients are independent and it is possible to sample directly from  $p(\alpha_i)$  (see section 2.2.1.1), this factorisation makes most sense.

Since the  $u_i$  are not independent, the factors of  $\gamma(\mathbf{u} | \boldsymbol{\alpha})$  must take the full partial sample into account:

$$\begin{aligned} \gamma(\mathbf{u} | \boldsymbol{\alpha}) &= \gamma_1(u_1 | \alpha_1) \gamma_2(u_2 | u_1, \alpha_{1:2}) \gamma_3(u_3 | u_{1:2}, \alpha_{1:3}) \cdots \gamma_d(u_d | u_{1:d-1}, \alpha_{1:d}) \\ &= \prod_{i=1}^d \gamma_i(u_i | u_{1:i-1}, \alpha_{1:i}). \end{aligned} \quad (102b)$$

The notation of these factors  $\gamma_i(\mathbf{u}_i | \mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i})$  does not mean that they necessarily have to be related to conditional distributions. They can be any function of the variables before and after the bar, as long as their product  $\gamma(\mathbf{u} | \boldsymbol{\alpha})$  yields the correct result. Indeed, the next subsections will present two choices for the factorisation. Both apply the same factorisation to both terms of  $\gamma$  in parallel. The first, which will be called “postponed factorisation”, each factor only incorporates the dimensions that are sampled from, and leave other terms for later in the process. The second, which will be called “quasi-conditional factorisation”, factorises the two Gaussians separately into conditional distributions per dimension.

The form of the speech and noise prior in this work are standard Gaussians, so that the factorisation of  $\gamma(\mathbf{u} | \boldsymbol{\alpha})$  must satisfy (combining (99a) and (102b)):

$$\prod_{i=1}^d \gamma_i(\mathbf{u}_i | \mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i}) = \mathcal{N}(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t); \boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x) \mathcal{N}(\mathbf{n}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t); \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n). \quad (103)$$

**5.2.2.2 Postponed factorisation** The Gaussian terms in (103) can be written as products with every element of the precision matrices. The precision matrix is the inverse covariance:  $\boldsymbol{\Lambda}_x = \boldsymbol{\Sigma}_x^{-1}$ . Its elements are denoted  $\lambda_{x,ij}$ . It is possible to postpone the terms until both dimensions to be related are known. This requires some manipulation, the details of which are in appendix D.3. The integrand is rewritten in (171), and the factors  $\gamma_i$  are defined as (from (172))

$$\begin{aligned} \gamma_1(\mathbf{u}_1 | \boldsymbol{\alpha}_1) &= |2\pi\boldsymbol{\Sigma}_y|^{-\frac{1}{2}} |2\pi\boldsymbol{\Sigma}_x|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\lambda_{n,11}(\mathbf{n}(\mathbf{u}_1, \boldsymbol{\alpha}_1, \mathbf{y}_{t,1}) - \boldsymbol{\mu}_{n,1})^2 \right. \\ &\quad \left. - \frac{1}{2}\lambda_{x,11}(\mathbf{x}(\mathbf{u}_1, \boldsymbol{\alpha}_1, \mathbf{y}_{t,1}) - \boldsymbol{\mu}_{x,1})^2\right); \end{aligned} \quad (104a)$$

$$\begin{aligned} \gamma_i(\mathbf{u}_i | \mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i}) &= \exp\left(-\frac{1}{2}\lambda_{x,ii}(\mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) - \boldsymbol{\mu}_{x,i})^2 - (\mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) - \boldsymbol{\mu}_{x,i}) \boldsymbol{\nu}_{x,i} \right. \\ &\quad \left. - \frac{1}{2}\lambda_{n,ii}(\mathbf{n}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) - \boldsymbol{\mu}_{n,i})^2 - (\mathbf{n}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) - \boldsymbol{\mu}_{n,i}) \boldsymbol{\nu}_{n,i}\right), \end{aligned} \quad (104b)$$

where the terms containing coordinates of lower dimensions  $\mathbf{u}_{1:i-1}$  are defined as (in (170b))

$$\boldsymbol{\nu}_{x,i} = \sum_{j=1}^{i-1} \lambda_{x,ij}(\mathbf{x}(\mathbf{u}_j, \boldsymbol{\alpha}_j, \mathbf{y}_{t,j}) - \boldsymbol{\mu}_{x,j}); \quad \boldsymbol{\nu}_{n,i} = \sum_{j=1}^{i-1} \lambda_{n,ij}(\mathbf{n}(\mathbf{u}_j, \boldsymbol{\alpha}_j, \mathbf{y}_{t,j}) - \boldsymbol{\mu}_{n,j}). \quad (105)$$

Note again that  $\gamma_i(\mathbf{u}_i | \mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i})$  is not a conditional distribution.

At every dimension, a proposal distribution  $\rho_i$  is required for importance sampling. This distribution needs to have a shape similar to  $\gamma_i$ . There is, however, no need to match the amplitude of  $\gamma_i$ . This is convenient when rewriting  $\gamma_i$  to find the shape of the distribution, as appendix D.3 does.

The factors in (104b) turn out to be proportional to two Gaussian distributions that are functions of  $\mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i})$  and  $\mathbf{n}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i})$  (from (174)):

$$\begin{aligned} \gamma_i(\mathbf{u}_i | \mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i-1}) &\propto \mathcal{N}\left(\mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}); \boldsymbol{\mu}_{x,i} - \frac{\boldsymbol{\nu}_{x,i}}{\lambda_{x,ii}}, \lambda_{x,ii}^{-1}\right) \\ &\cdot \mathcal{N}\left(\mathbf{n}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}); \boldsymbol{\mu}_{n,i} - \frac{\boldsymbol{\nu}_{n,i}}{\lambda_{n,ii}}, \lambda_{n,ii}^{-1}\right). \end{aligned} \quad (106)$$

This expression has the same shape as the one-dimensional integrand in (89) in section 5.2.1, and the same proposal distribution as discussed in section 5.2.1.2 can be used.

**5.2.2.3 Quasi-conditional factorisation** Another factorisation decomposes both Gaussians in (99b) into the conditionals that would be valid for a Gaussian. Appendix A.3 decomposes a Gaussian distribution over two vectors into the marginal of the first vector times the distribution of the second conditional on the first. However, since the density  $\gamma$  is the product of two Gaussians with two different non-linear variables, the factors here are not normalised or proportional to conditional probabilities.

Crucially, the derivation in (122) does not rely on the input variable or normalisation. It is therefore possible to find a factorisation of both speech and noise Gaussians in parallel, even if the factors are not exactly conditionals. For this, every factor of  $\gamma_{1:i}$  is factorised recursively (only the left-hand term is shown):

$$\begin{aligned} \mathcal{N}(\mathbf{x}_{1:i}; \boldsymbol{\mu}_{x,1:i}, \boldsymbol{\Sigma}_{x,1:i,1:i}) &= \mathcal{N}(\mathbf{x}_{1:i-1}; \boldsymbol{\mu}_{x,1:i-1}, \boldsymbol{\Sigma}_{x,1:i-1,1:i-1}) \\ &\quad \mathcal{N}(x_i; \boldsymbol{\mu}_{x,i|1:i-1}(\mathbf{x}_{1:i-1}), \sigma_{x,i|1:i-1}^2), \end{aligned} \quad (107a)$$

where the parameters for  $\mathbf{x}_i$  dependent on  $\mathbf{x}_{1:i-1}$  are

$$\boldsymbol{\mu}_{x,i|1:i-1}(\mathbf{x}_{1:i-1}) = \boldsymbol{\mu}_{x,i} - \boldsymbol{\Sigma}_{x,i,1:i-1} [\boldsymbol{\Sigma}_{x,1:i-1,1:i-1}]^{-1} (\mathbf{x}_{1:i-1} - \boldsymbol{\mu}_{x,1:i-1}); \quad (107b)$$

$$\sigma_{x,i|1:i-1}^2 = \sigma_{x,i,i}^2 - \boldsymbol{\Sigma}_{x,i,1:i-1} [\boldsymbol{\Sigma}_{x,1:i-1,1:i-1}]^{-1} \boldsymbol{\Sigma}_{x,1:i-1,i}. \quad (107c)$$

Note that the variance  $\sigma_{x,i|1:i-1}^2$  is not a function of  $\mathbf{x}$ , so that it needs to be computed only once for all samples. Also, in computing the inverses of  $\boldsymbol{\Sigma}_{x,1:i,1:i}$  for  $i = 1 \dots d$ , their structure can be exploited. A block-wise inversion can be used. Block-wise inversion is a technique often applied to take advantage of known structure in blocks of the matrix, for example, when a block is diagonal. The trick here, however, is that the intermediate results are useful. If the the inverse of a matrix with one column removed from the right and one row removed from the bottom,  $\boldsymbol{\Sigma}_{x,1:i-1,1:i-1}$  is known, the inverse of the matrix with the extra row and column,  $\boldsymbol{\Sigma}_{x,1:i,1:i}$ , can be computed in  $\mathcal{O}(i^2)$  time. An incremental implementation that yields  $[\boldsymbol{\Sigma}_{x,1:i,1:i}]^{-1}$  for  $i = 1 \dots d$  thus has a time complexity of only  $\mathcal{O}(d^3)$ , the same as inverting only the full covariance matrix.

The fully factorised formulation of the left-hand term of  $\gamma$  is

$$\begin{aligned} \mathcal{N}(\mathbf{x}_{1:d}; \boldsymbol{\mu}_{x,1:d}, \boldsymbol{\Sigma}_{x,1:d,1:d}) &= \mathcal{N}(x_1; \boldsymbol{\mu}_{x,1}, \sigma_{x,1,1}^2) \\ &\quad \prod_{i=2}^d \mathcal{N}(x_i; \boldsymbol{\mu}_{x,i|1:i-1}(\mathbf{x}_{1:i-1}), \sigma_{x,i|1:i-1}^2). \end{aligned} \quad (108)$$

The analogous factorisation can be applied to the right-hand term of  $\gamma(\mathbf{u})$ . (99a) can then be factorised

$$\begin{aligned} \gamma(\mathbf{u}|\boldsymbol{\alpha}) &= \mathcal{N}(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t); \boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x) \mathcal{N}(\mathbf{n}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t); \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n) \\ &= \mathcal{N}(x(\mathbf{u}_1, \boldsymbol{\alpha}_1, \mathbf{y}_1); \mu_1, \sigma_{1,1}^2) \mathcal{N}(n(\mathbf{u}_1, \boldsymbol{\alpha}_1, \mathbf{y}_1); \mu_1, \sigma_{1,1}^2) \\ &\quad \prod_{i=2}^d \mathcal{N}\left(x(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_i); \boldsymbol{\mu}_{x,i|1:i-1}(x(\mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i-1}, \mathbf{y}_{1:i-1})), \sigma_{x,i|1:i-1}^2\right) \\ &\quad \mathcal{N}\left(n(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_i); \boldsymbol{\mu}_{n,i|1:i-1}(n(\mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i-1}, \mathbf{y}_{1:i-1})), \sigma_{n,i|1:i-1}^2\right). \end{aligned} \quad (109)$$

The factors of  $\gamma$  then become<sup>5</sup>

$$\begin{aligned} \gamma_i(\mathbf{u}_i|\mathbf{u}_{1:i-1}) &= \mathcal{N}\left(x(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_i); \boldsymbol{\mu}_{x,i|1:i-1}(x(\mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i-1}, \mathbf{y}_{1:i-1})), \sigma_{x,i|1:i-1}^2\right) \\ &\quad \mathcal{N}\left(n(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_i); \boldsymbol{\mu}_{n,i|1:i-1}(n(\mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i-1}, \mathbf{y}_{1:i-1})), \sigma_{n,i|1:i-1}^2\right), \end{aligned} \quad (110)$$

where  $\boldsymbol{\mu}_{x,i|1:i-1}(x_{1:i-1})$  and  $\sigma_{x,i|1:i-1}^2$  are defined in (107a). Again, the factors have the form of density as the one-dimensional  $\gamma$  in (89), so that the proposal distribution proposed in section 5.2.1.2 can be used.

**5.2.2.4 Applying sequential importance re-sampling** Whichever factorisation of  $\gamma(\mathbf{u}|\boldsymbol{\alpha})$  is chosen, the application of sequential importance re-sampling is the same. The integral  $\int \int \gamma(\mathbf{u}, \boldsymbol{\alpha}) d\boldsymbol{\alpha} d\mathbf{u}$ , the value of interest, is the normalisation constant of  $\gamma(\mathbf{u}, \boldsymbol{\alpha})$ , which in section 4.1.1 was called  $Z$ . To find  $Z$  by stepping through dimensions, in section 4.2 it was expressed as a sequence of incremental normalisation constants  $Z_i/Z_{i-1}$ . Given a sample set  $\{(\mathbf{u}_{1:i-1}^{(s)}, \boldsymbol{\alpha}_{1:i-1}^{(s)})\}$ , the approximation of the incremental normalisation constant is <sup>6</sup> (when re-sampling is used)

$$\frac{\widetilde{Z}_i}{Z_{i-1}} = \frac{1}{S} \sum_{s=1}^S \frac{\gamma_i(\boldsymbol{\alpha}_i^{(s)}) \gamma_i(\mathbf{u}_i^{(s)}|\mathbf{u}_{1:i-1}^{(s)}, \boldsymbol{\alpha}_{1:i}^{(s)})}{\rho_i(\boldsymbol{\alpha}_i^{(s)}) \rho_i(\mathbf{u}_i^{(s)}|\mathbf{u}_{1:i-1}^{(s)}, \boldsymbol{\alpha}_{1:i}^{(s)})}, \quad (111)$$

where samples  $\boldsymbol{\alpha}_i^{(s)}$  are drawn from proposal distribution  $\rho_i(\boldsymbol{\alpha}_i^{(s)})$  and samples  $\mathbf{u}_i^{(s)}$  from the appropriate  $\rho_i(\mathbf{u}_i|\mathbf{u}_{1:i-1}^{(s)}, \boldsymbol{\alpha}_{1:i}^{(s)})$ .

The shape of the density  $\gamma(\mathbf{u}_i|\mathbf{u}_{1:i-1}^{(s)}, \boldsymbol{\alpha}_{1:i}^{(s)})$  depends on the current partial sample  $(\mathbf{u}_{1:i-1}^{(s)}, \boldsymbol{\alpha}_{1:i}^{(s)})$  and the type of factorisation. The factorisations in sections 5.2.2.2 and 5.2.2.3 both result in factors of the form

$$\begin{aligned} \gamma_i(\mathbf{u}_i|\mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i}) &= \mathcal{N}(x(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}); \hat{\boldsymbol{\mu}}_{x,i}, \hat{\boldsymbol{\sigma}}_{x,i}) \\ &\quad \cdot \mathcal{N}(n(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}); \hat{\boldsymbol{\mu}}_{n,i}, \hat{\boldsymbol{\sigma}}_{n,i}), \end{aligned} \quad (112)$$

where the parameters  $(\hat{\boldsymbol{\mu}}_{x,i}, \hat{\boldsymbol{\sigma}}_{x,i}, \hat{\boldsymbol{\mu}}_{n,i}, \hat{\boldsymbol{\sigma}}_{n,i})$  depend on the type of factorisation and the current partial sample  $(\mathbf{u}_{1:i-1}^{(s)}, \boldsymbol{\alpha}_{1:i}^{(s)})$ . Appropriate proposal distributions for this type

<sup>5</sup> This formulation assumes (in  $\gamma_1$ ) that multiplying a  $1 \times 0$  matrix by a  $0 \times 0$  matrix by a  $0 \times 1$  matrix yields a  $1 \times 1$  matrix  $[0]$ .

<sup>6</sup> Since samples for one dimension of  $\boldsymbol{\alpha}$  and one of  $\mathbf{u}$  are drawn, this could be written  $Z_{2i}/Z_{2i-2}$ .

of density have been discussed in section 5.2.1.2. These distributions over  $u_i$  take the form of one Gaussian or a mixture of two. They are therefore straightforward to draw a sample from and slot into (111).

The density  $\gamma_i(\alpha_i)$  is set to  $p(\alpha_i)$  defined in (18), which has a Gaussian shape, but constrained to  $[-1, 1]$ . It is straightforward to draw a sample directly from this distribution, by sampling from the Gaussian until the sample is in  $[-1, 1]$  (see section 2.2.1.2). Therefore,  $\rho_i(\alpha_i)$  can be set to  $\gamma_i(\alpha_i)$ . This means that  $\gamma_i(\alpha_i)/\rho_i(\alpha_i)$  in (111) becomes 1, and hybrid sequential importance re-sampling of algorithm 4 on page 30 can be applied.

*Re-sampling* was discussed in section 4.3. In short, it duplicates higher-weight samples from the sample set and removes lower-weight ones between every dimension. This reduces the variance of the sample weights; conceptually, it focuses effort on higher-probability regions.

When using re-sampling, the order in which the dimensions are traversed becomes important. The longer ago samples for one dimension were drawn, the more likely they are to have duplicate entries for that dimension. The sample set will therefore be less varied for earlier dimensions. This is not a big problem when, as here, the interest is not in the samples, but in the normalisation constant. However, when drawing samples for one dimension, it still makes sense to have last drawn the dimensions which the new dimension depends on most.

For example, it might seem obvious to draw  $\alpha_1^{(s)} \dots \alpha_d^{(s)}$  first, and then go through  $u_1^{(s)} \dots u_d^{(s)}$ . However, in  $i-1$  rounds of re-sampling, the set of samples  $\alpha_1^{(s)} \dots \alpha_d^{(s)}$  may become considerably less varied.  $u_i^{(s)}$ s for higher  $i$  may be drawn with only a few or one unique  $\alpha_i^{(s)}$ , which limits the accuracy of the approximation of the normalisation constant. In this work, the order of traversal is therefore  $\alpha_1, u_1, \alpha_2, u_2, \dots, \alpha_d, u_d$ . This works best if  $u_i$  and  $u_j$  are less dependent when  $j-i$  is greater. The order in which samples for the dimensions are drawn could also be determined on the fly by considering the values of the entries of  $\Sigma_x$  and  $\Sigma_n$ , but this work does not investigate this.

This section has discussed a transformation of the integral that gives the corrupted speech likelihood  $p(\mathbf{y}_t)$ , two different factorisations of the integrand, and how to apply sequential importance sampling to approximate the integral. The estimate from the sampling scheme is consistent, but not unbiased. This means that for a small sample cloud, the approximated value for  $p(\mathbf{y}_t)$  may be generally overestimated or underestimated. However, as the sample cloud size increases, the resulting value converges to the real likelihood.

## 6 Distance to the actual distribution

It is standard practice in speech recognition research to judge speech recognition methods by word error rates. However, this paper has the explicit aim of modelling the predicted corrupted speech distribution as accurately as possible. There is a more direct way of testing performance on this criterion: the Kullback-Leibler divergence between the predicted distribution and the approximation.

The Kullback-Leibler (KL) divergence measures how far a distribution  $q$  is from a distribution  $p$ . The KL divergence is always non-negative; it is 0 if and only if the distributions are the same. It has therefore been used to find how well speech recogniser

compensation matches the real distribution (e.g. [13]). Minimising a KL divergence between distributions is a frequent subtask in inference algorithms.

This paper will use the KL divergence to assess compensation for the corrupted speech distribution resulting from combining one speech Gaussian with one noise Gaussian. As in the theory section, the distributions will just be over statics, to remove the dependence on any additional approximations for the dynamics, and for speed. The KL divergence to real distribution  $p$  from approximation  $q$  over  $\mathbf{y}$  is defined as

$$\mathcal{KL}(p\|q) = \int p(\mathbf{y}) \log \frac{p(\mathbf{y})}{q(\mathbf{y})} d\mathbf{y}. \quad (113)$$

In this paper,  $q$  is an approximation to the noise-corrupted speech distribution, found for example with VTS, DPMC, or transformed-space sampling. The problem in computing this divergence is the one that motivates this whole paper: the real distribution  $p$  has no closed form, and neither does  $\mathcal{KL}(p\|q)$ . This problem can be worked around in two steps.

First, the KL divergence can be decomposed as

$$\mathcal{KL}(p\|q) = \mathcal{H}(p\|q) - \mathcal{H}(p), \quad (114a)$$

where the cross-entropy of  $p$  and  $q$  is defined as

$$\mathcal{H}(p\|q) = - \int p(\mathbf{y}) \log q(\mathbf{y}) d\mathbf{y} \quad (114b)$$

and the entropy of  $p$  as

$$\mathcal{H}(p) = - \int p(\mathbf{y}) \log p(\mathbf{y}) d\mathbf{y}. \quad (114c)$$

The entropy of  $p$  is constant when only  $q$  changes. The cross-entropy is then equal to the KL divergence up to a constant. For comparing different approximations  $q$ , therefore, the cross-entropy  $\mathcal{H}(p\|q)$  suffices. It does not, however, give an absolute divergence. When  $q$  becomes equal to  $p$ , the cross-entropy becomes equal to the entropy, but the latter cannot be computed for the noise-corrupted speech distribution.<sup>7</sup>

The second problem is that  $\mathcal{H}(p\|q)$  cannot be computed either. However, it is straightforward to draw samples from  $p$ : section 2.3.1 has shown the algorithm. Then, plain Monte Carlo (see section 4.1) can approximate the integration over  $\mathbf{y}$  by a sum over  $S$  samples drawn from  $p$ . The cross-entropy in (114b) can be seen as the expectation of  $\log q(\mathbf{y})$  under  $p$ , and approximated (using (49))

$$\mathcal{H}(p\|q) = - \int p(\mathbf{y}) \log q(\mathbf{y}) d\mathbf{y} \simeq - \frac{1}{S} \sum_s \log q(\mathbf{y}^{(s)}), \quad \mathbf{y}^{(s)} \sim p(\mathbf{y}). \quad (115)$$

Another way of looking at this expression is as the negative log-likelihood of  $q$  for the set of samples  $\{\mathbf{y}^{(s)}\}$ . This gives another motivation for using this as a metric

---

<sup>7</sup> The entropy could be rewritten to

$$\mathcal{H}(p) = \int p(\mathbf{y}) \log p(\mathbf{y}) d\mathbf{y} = \int \left( \iint p(\mathbf{x}, \mathbf{n}, \mathbf{y}) d\mathbf{n} d\mathbf{x} \right) \log \left( \iint p(\mathbf{x}, \mathbf{n}, \mathbf{y}) d\mathbf{n} d\mathbf{x} \right) d\mathbf{y},$$

which has no obvious closed form.

for comparing compensation methods. Note that when  $q$  is the transformed-space sampling method from this work, for every sample  $\mathbf{y}^{(s)}$  another level of sampling takes place inside the evaluation of  $q(\mathbf{y}^{(s)})$ .

The cross-entropy results in the next section will use this Monte Carlo approximation. It has one caveat: the distribution  $q$  is assumed to be normalised, and if not, then the result is not valid. This means that the Algonquin approximation, which does not yield a normalised distribution over  $\mathbf{y}$ , cannot be assessed in this way. As pointed out in section 5.2.2, the likelihood approximation of transformed-space sampling is biased, but consistent. This means that as the size of its sample cloud increases,  $q$  converges to being normalised.

## 7 Experiments

This work has aimed to find an accurate approximation to the corrupted speech likelihood. To assess the performance, this work will initially consider the cross-entropy to the real distribution for individual components. This allows a fine-grained assessment of the approximations that methods for noise-robustness make. This chapter will consider the effects of parameterisations of the noise-corrupted speech distributions, such as assuming it Gaussian (section 7.2.1), and diagonalising the covariance matrix of this Gaussian (section 7.2.2). It will also investigate the effects of a common approximation to the mismatch function: assuming the phase factor  $\alpha$  to be fixed (section 7.2.3).

### 7.1 Set-up

This work uses a noise-corrupted version of the Resource Management task both for evaluation the cross-entropy and the word error rate. The Resource Management task is a medium-vocabulary task, with a 1000-word vocabulary. Operations Room noise from the NOISEX-92 database was added artificially with the noise scaled to yield SNRS of 20 dB and 14 dB. The clean training data contains 109 speakers reading 3990 sentences, 3.8 hours of data. State-clustered cross-word triphone models with 6 components per mixture were built using the HTK RM recipe. All results are averaged over three of the four available test sets, Feb89, Oct89, and Feb91, a total of 30 test speakers and 900 utterances.

All experiments use a noise model trained directly on the noise as added to the speech audio. This eliminates the influence of the noise estimation algorithm. As discussed in the introduction, in practice methods for noise robustness can estimate a noise model on little training data compared to generic adaptation techniques. This is because their model of the environment matches the actual environment to some degree. This work examines how close that model can be without considering how to estimate the noise model.

This work uses the mismatch function presented in section 2.2 as the real one. This assumes that there is no convolutional noise, which would be additive in the log-spectral domain anyway. It also assumes that the phase factor is Gaussian but constrained to  $[-1, 1]$ , and independent per time frame and per spectral coefficient. The variances  $\sigma_\alpha^2$  of the phase factor are found from the actual filterbank weights for the Resource Management system, as discussed in section 2.2.1.2. The schemes that sample from the phase factor distribution, DPMC and transformed-space sampling, use the

exact same distribution. *vts* requires the distribution to be Gaussian and ignores the constraint on the coefficients.

For the cross-entropy experiments, the full-covariance noise and speech Gaussians are both over 24 log-spectral coefficients. The one for the noise is trained directly on the noise audio. The speech distribution is taken from a trained Resource Management system, single-pass retrained to find Gaussian in the log-spectral domain. A low-energy speech component<sup>8</sup> is chosen, to represent the part of the utterance where the low SNR causes recognition errors. The distance between the speech and the noise means, averaged over the log-spectral coefficients, corresponds to a 10 dB SNR. 5000 samples  $\mathbf{y}^{(s)}$  are drawn from the corrupted speech distribution. *DPMC* trains Gaussians on 50 000 samples. For *IDPMC*, the average number of samples per Gaussian component is also 50 000, so that the 8-component mixture, for example, is trained on 400 000 samples. Except where mentioned, the relative ordering of the approximation methods is the same for all combinations of speech and noise examined.

The speech recogniser experiments compare various model compensation methods. Previous work had not applied model compensation with a phase factor distribution. Here, the experiments for both *vts* and *DPMC* use a distribution over the phase factor  $\alpha$  of the mismatch function (section 7.2.3 examines the influence of the phase factor).

For *vts*, the standard approach [28, 2] fixes  $\alpha$  to 0. Previous work has modelled it with a distribution only for feature enhancement: to find a minimum mean square error estimate [4], or with a Kalman filter [21]. For model compensation, previous work has fixed  $\alpha$  to 1 [27], mathematically highly improbable, or 2.5 [23], mathematically impossible. This work, on the other hand, applies *vts* compensation with a distribution over  $\alpha$ .

*DPMC*, similarly, was originally presented with  $\alpha$  fixed to 0 [13]. However, since it trains a distribution on samples drawn from the corrupted speech distribution, it is straightforward to extend it so it uses a distribution for  $\alpha$ . The phase factor distribution to draw samples from was given in section 2.2.1.2.

For the recognition experiments, the number of samples per component for extended *DPMC* was set to 100 000. For iterative *DPMC*, the average number of samples was 100 000: for example, a 6-component mixture was trained on 600 000 samples.

It is standard practice in speech recognition to append delta and delta-delta coefficients to static observation vectors. These indicate the direction of change in the static coefficients and are computed from the static feature vectors in a window with a linear transformation. To find the distributions over dynamic features, model compensation methods often use additional approximations, such as the continuous time approximation for *vts*. However, it is more accurate to explicitly model the distribution over a window of static feature vectors, “extended” feature vectors. It is then possible then apply the linear transformation to find the distribution over statics, deltas and delta-deltas [34, 35]. Apart from yielding better accuracy, this also keeps compensation for dynamics most closely related to that for the statics. For robustness, the speech statistics have striped covariance matrices as discussed in [34, 35]. The distributions of the corrupted speech have full covariance matrices.

For *DPMC* and iterative *DPMC*, compensation with extended feature vectors works as follows [34, 35]. From the extended distributions over a window of statics, samples

<sup>8</sup>Tied state “st\_uh\_4\_3”, component 2.

are drawn for the speech, the noise, and the phase factor. These are combined, frame by frame, through the mismatch function, to produce a sample of a window of noise-corrupted speech. The linear transformation converts this sample into a sample with static and dynamic coefficients. The Gaussian (for DPMC) or the mixture of Gaussians (for IDPMC) is then trained on these samples as usual.

For VTS, the equations to compute the means and covariance for one frame of static parameters can be generalised to a window of static parameters. Importantly, off-diagonal covariance entries need to be compensated as well. This process, the details of which are in [34, 35], yields a Gaussian distribution over the statics of noise-corrupted speech in a window. The linear transformation can be applied to this Gaussian to yield a Gaussian over statics and dynamics.

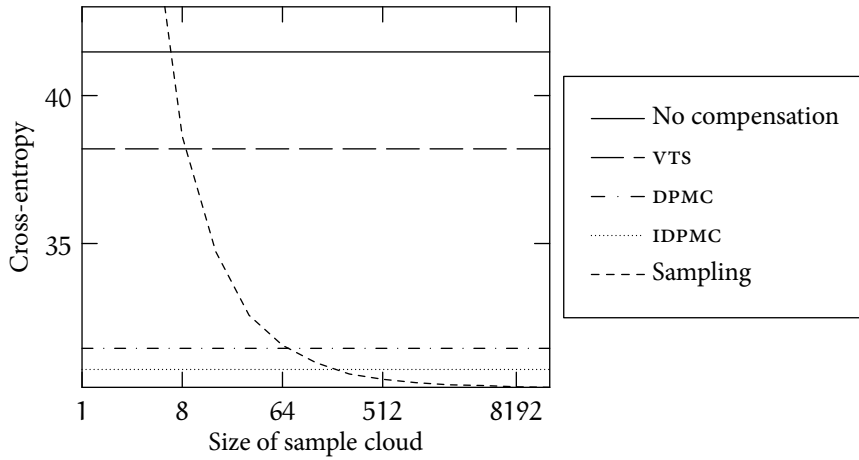
## 7.2 Results

If the speech and noise models represented the real distributions perfectly, then computing the corrupted speech distribution exactly would yield the best recognition performance. In practice, however, the models are imperfect and improving the KL divergence to the real distribution does not necessarily mean that the speech recognition accuracy will also improve. In this respect, assessing the quality of speech recognition compensation with the KL divergence is conceptually similar to assessing language models by their perplexities. The following sections will also relate cross-entropy results to word error rates.

However, not all methods discussed in this work can be assessed with both of these metrics. The Algonquin algorithm, discussed in section 3.3, yields a Gaussian approximation of the corrupted speech distribution specific to an observation. Used as a method to approximate the likelihood of observations, it therefore is not normalised. This makes it impossible to compute the cross-entropy for it. As discussed in section 5.2, the likelihood for transformed-space sampling is not normalised for small sample clouds, but converges to normalisation as the number of samples increases.

This work will not present word error rates for the transformed-space sampling method introduced in this work, because decoding with it is prohibitively slow. This is caused by a big conceptual difference between model compensation methods (e.g., VTS, DPMC, and IDPMC) on the one hand and transformed-space sampling on the other. Model compensation computes a parametric distribution, and once that is done, running a recogniser or computing a cross-entropy is no slower than without compensation. Transformed-space sampling, on the other hand, approximates the likelihood given an observation, and cannot pre-compute anything. For the recognition experiments in this work, just for the statics and with a decent-sized sample cloud of 512, it would run at roughly 20 million times real-time. This figure is based on an implementation that was not optimised for speed at all, but even with an optimised implementation running a speech recogniser with it would not be feasible.

However, the approximated likelihood of transformed-space sampling tends to the exact likelihood. In the following section it will become clear that it indicates the minimum value of the cross-entropy to the real distribution. This minimum is by definition where the KL divergence is 0. The distance to this point in a cross-entropy graph therefore shows how far compensation methods are from the ideal compensation.



**Figure 15** Cross-entropy to the corrupted speech distribution for transformed-space sampling and model compensation methods.

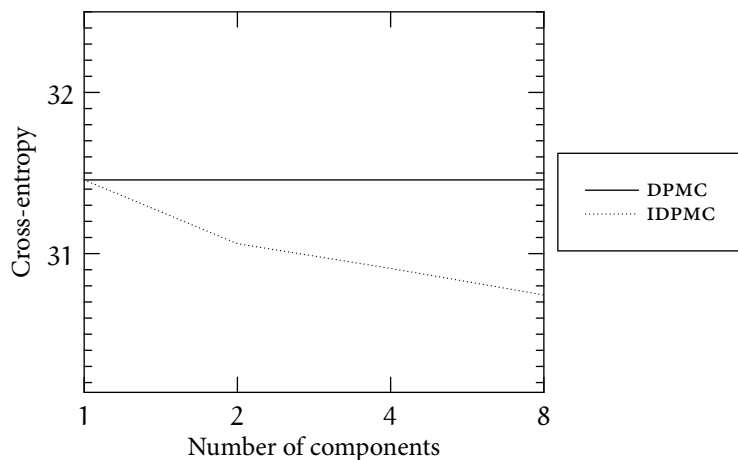
### 7.2.1 Compensation methods

The graph in figure 15 contains the cross-entropy for different compensation methods. The curved line indicates the cross-entropy between the real distribution and the transformed-space sampling method described in section 5.2, for increasing sample cloud size. The factorisation is the quasi-conditional factorisation from section 5.2.2.3. For distributions other than a three-dimensional toy example, the postponed factorisation discussed in section 5.2.2.2 showed a much slower convergence: in an earlier version of the experiment in figure 15 (without the phase factor), even with 16 384 samples it did not perform as well as the VTS-estimated Gaussian.

As the size of the sample cloud increases, the approximation of  $p(\mathbf{y}^{(s)})$  found with transformed-space sampling converges to the correct value. This means that the cross-entropy  $\mathcal{H}(p\|q)$  converges to the entropy  $\mathcal{H}(p)$ . The bottom of the graph is set to the point the curve in figure 15 converges to, which indicates the entropy of  $p$ . Since the KL divergence is defined (in (114a)) as  $\mathcal{KL}(p\|q) = \mathcal{H}(p\|q) - \mathcal{H}(p)$ , this is the point where the KL divergence is 0. Since the KL divergence cannot be negative, this point gives the optimum cross-entropy. It gives a lower-bound on how well the real corrupted speech distribution can be matched. The value of the cross-entropy for transformed-space sampling with 16 384 samples, 30.14, will also be the bottom for the other graphs in this section.

The line labelled “DPMC” in figure 15 indicates the best match to the real distribution possible with one Gaussian. The Monte Carlo approximation to the cross-entropy, section 6 has shown, is equivalent to the negative average log-likelihood on the samples. DPMC finds the Gaussian that maximises its log-likelihood on the samples it is trained on. If the sample sets for training and testing were the same, then DPMC would yield the mathematically optimal Gaussian. Though different sample sets are used (with 50 000 samples for training DPMC and 5000 samples for testing) the cross-entropy has converged. Any other Gaussian approximation will perform worse.

The state-of-the-art VTS compensation finds such a Gaussian analytically, and it is much faster. However, its cross-entropy to the real distribution is far from DPMC’s ideal



**Figure 16** Cross-entropy to the corrupted speech distribution for iterative DPMC.

Compensation	Shape	20 dB	14 dB
—		38.1	83.8
VTS, $\alpha = 1$	diag	8.6	17.3
EVTS		11.1	16.5
EDPMC		7.4	13.3
EIDPMC	full	6.9	12.0
EIDPMC + 6		6.2	11.1
EIDPMC + 12		6.5	11.3

**Table 1** Word error rates for various compensation schemes.

one.

Just like DPMC, IDPMC finds a distribution from samples, but it uses a mixture of Gaussians rather than one Gaussian. The mixture in the graph has 8 components trained on 400 000 samples, and comes close to the correct distribution. As the number of components increases from 1 to 8, keeping the average number of samples for components at 50 000, the cross-entropy decreases, as figure 16 illustrates. With an infinite number of components, it would yield the exact distribution. To correctly model the non-Gaussianity in 24 dimensions, however, a large number of components are necessary, which quickly becomes impractical.

To examine the link between the cross-entropy and the word error rate, recognition experiments are run. Improved modelling of the corrupted speech does not guarantee better discrimination, since speech and noise models are not necessarily the real ones. Since transformed-space sampling needs to be run separately for every observation vector for every speech component, it is too slow to use in a speech recogniser.

Table 1 contains word error rates at two signal-to-noise ratios for comparison with the cross-entropy results in figure 15. Results with the uncompensated system, trained on clean data, are in the top row. Below it, as a reference, is standard VTS. It sets the phase factor  $\alpha$  to 1, and finds diagonal-covariance compensation. Standard VTS uses the continuous time approximation to compensate delta- and delta-delta parameters.

This yields inaccurate compensation for off-diagonals. This is discussed in great depth in [34, 35]. Using block-diagonal statistics and compensation, word error rates for standard VTS are worse; 19.5 % and 38.5 %.

The bottom part of the table contains results on extended VTS (eVTS) and extended DPMC (eDPMC). As discussed in section 7.1, they use distributions over extended feature vectors [34, 35], which consist of the statics in a window that dynamic parameters are computed from. They also use a distribution over the phase factor  $\alpha$ . The covariances of the resulting distributions are not diagonalised.

eVTS performs less well than standard VTS at 20 dB. This is caused by the interaction of the phase factor with the vector Taylor series approximation, which section 7.2.3 will explore in more detail. At 14 dB, the more precise modelling does pay off. Compared to the uncompensated system extended VTS's performance improves much more (38.1 % to 11.1 %) than expected from its improvement in terms of the cross-entropy in figure 15. VTS compensation uses a vector Taylor series approximation around the speech and noise means. It therefore models the mode of the corrupted speech distribution better than the tails. This causes the majority of the improvement in discrimination.

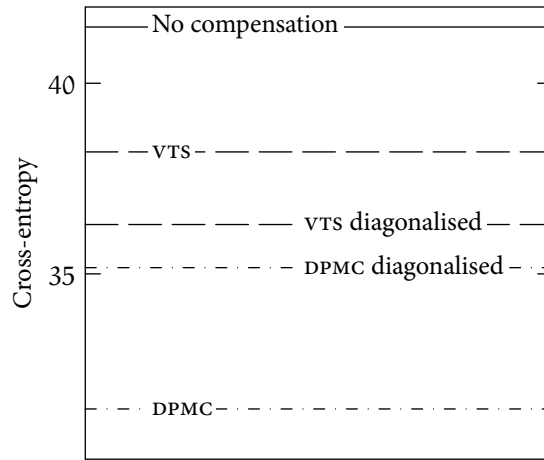
However, extended DPMC, which finds the optimal Gaussian given the speech and noise models, does yield better accuracy (7.4 %). Extended DPMC finds one corrupted speech Gaussian for one clean speech Gaussian. The cross-entropy experiment only used one clean speech Gaussian. Extended IDPMC (eIDPMC), however, trains a mixture of Gaussians from samples, which can be drawn from any distribution. For the recognition experiments, therefore, eIDPMC compensates one state-conditional mixture at a time. Replacing the 6-component speech distribution by a 6-component corrupted speech distribution, eIDPMC increases performance from eDPMC's 7.4 % to 6.9 %. By modelling the distribution better, with 12 components ("eIDPMC + 6"), performance increases further to 6.2 %. The corrupted speech distribution should be more precise as the number of components increases to 18 ("eIDPMC + 12"). However, even by increasing the number of samples by a factor of 2, to 3600 000, performance does not increase. This is felt to be caused by lack of robustness of the speech statistics, even though they have striped covariance matrices. Since in figure 16 the line for IDPMC tends towards the best possible cross-entropy, this is thought to be the best possible word error rate for these clean speech distributions and this noise model.

Going from a Gaussian trained with extended VTS to the optimal Gaussian to a mixture of Gaussians in general improves the precision of the corrupted speech model. This shows in the cross-entropy to the real distribution, and the same effects are observed in the word error rate. Better modelling of the corrupted speech distribution leads to better performance. The next sections will evaluate specific common approximations: diagonalising Gaussians' covariance matrices, and setting  $\alpha$  to a fixed value.

### 7.2.2 Diagonal-covariance compensation

The cepstral-domain Gaussians of speech recognisers are often diagonalised. This yields more robust estimates than full covariance matrices, and keeps decoding fast. To model some feature correlations while maintaining decoding speed, approaches that use structured precision matrices [11, 31] are possible.

For noisy conditions specifically, it has previously been observed that feature correlations change and it is advantageous to compensate for this. However, it turns out that modelling correlations for the wrong noise conditions is counter-productive [12]. Also,



**Figure 17** The effect of diagonalisation on the cross-entropy.

Compensation	Shape	20 dB	14 dB
—	diagonal	38.1	83.8
evts	diag	8.3	15.7
evts	full	11.1	16.5
edPMC	diag	7.5	14.9
edPMC	full	7.4	13.3

**Table 2** The effect of diagonalisation on the word error rate.

estimates for off-diagonal elements are much less robust to approximations [34, 35]. This section will relate these effects using the cross-entropy and speech recogniser accuracy.

Diagonalisation usually takes place in the cepstral domain. The theory and the cross-entropy experiments have used log-spectral-domain feature vectors. To emulate diagonalisation in the cepstral domain for log-spectral-domain features, therefore, the Gaussian is first converted to the cepstral domain with a DCT matrix. Normally, cepstral feature vectors are truncated to 13 elements. However, to be able to convert back to the log-spectral domain, here all 24 dimensions are retained. The Gaussian is then diagonalised. To be able to compare the log-likelihoods, the diagonalised Gaussians are converted back to the log-spectral domain with the inverse DCT.

Figure 17 compares the cross-entropy to the real distribution of diagonalised and non-diagonalised Gaussians found with DPMC and VTS. As explained in the previous section, DPMC by definition yields the optimal Gaussian, so it must result in the lowest cross-entropy, and diagonalising it makes it perform less well. That full-covariance VTS performs less well than its diagonalisation may come as a surprise. On this test case, apparently, the off-diagonals in the cepstral domain are not estimated well enough, so that diagonalising lends the distribution robustness.

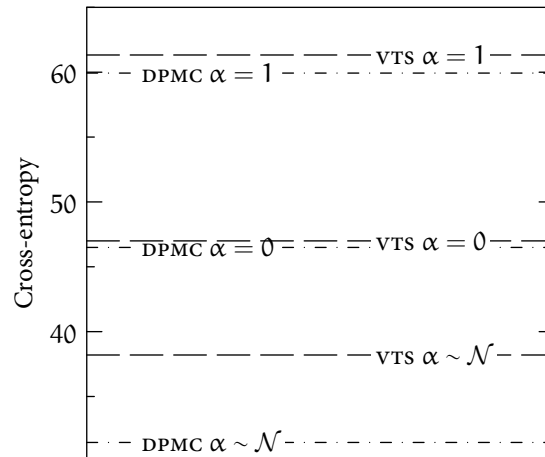
Table 2 investigates speech recognition performance when diagonalising Gaussian compensation. As in the previous section, the compensation methods use a phase factor distribution and extended feature vectors, to model the distributions as precisely

as possible. Here, as for the cross-entropy, diagonalising extended vTS compensation improves performance (e.g., 11.1 % to 8.3 %). The off-diagonal covariance entries are not estimated well, so that diagonalisation increases robustness. (The next section will relate this to the model for the phase factor  $\alpha$ .) However, eDPMC, which finds the optimal Gaussian compensation, does perform better when it is allowed to model correlations (7.5 % to 7.4 % at 20 dB). As expected from earlier work [34, 35], at a lower signal-to-noise ratio the correlations change more, so that modelling them becomes more important (14.9 % to 13.3 % at 20 dB).

### 7.2.3 Influence of the phase factor

Model compensation often assumes a mismatch function that is an approximation to the real one as presented in section 2.2. Traditionally the phase factor  $\alpha$ , which arises from the interaction between the speech and the noise in the complex plane, is assumed fixed. This section will look into the effect of the approximation, comparing Gaussian compensation with DPMC and vTS.

The phase factor distribution is a Gaussian around 0 (constrained to  $[-1, +1]$  in the case of DPMC). This corresponds to the model used to draw corrupted speech samples from for the cross-entropy. Previous work has not used DPMC with a phase factor distribution. vTS for feature enhancement has previously used such a distribution [4, 21]. For model compensation,  $\alpha$  has been set to fixed values [23, 27], but not to a distribution. Two settings for  $\alpha$  are of interest. The way it is traditionally presented [28, 2] is with  $\alpha = 0$ , which is the mode of the actual distribution. The second setting is  $\alpha = 1$ . This seems an arbitrary number, and is mathematically improbable. However, as appendix B shows, if the term with  $\alpha$  in the mismatch function is ignored and magnitude-spectrum feature vectors are used, this is equivalent to setting  $\alpha = 1$  for the power spectrum. This has been applied in previous work (e.g. [27]).



**Figure 18** The effect of the phase factor on Gaussian compensation.

Figure 18 shows the cross-entropy for DPMC and vTS with different models for the phase factor. Note that the vertical axis uses a larger scale than figure 15. The bottom of the graph is still set to the optimal cross-entropy acquired with transformed-space sampling. Both methods generate full covariance matrices. The diagonalised versions

show the same trends, with smaller distances between cross-entropies. DPMC with the model for  $\alpha$  matching the actual distribution (“DPMC  $\alpha \sim \mathcal{N}$ ”) yields the lowest cross-entropy by definition. vTS with a Gaussian model (“vTS  $\alpha \sim \mathcal{N}$ ”) is at some distance.

The obvious choice for fixing  $\alpha$  would be the mode of its actual distribution, 0. With that assumption, both DPMC and vTS end up further away from the ideal distribution. Note that though the cross-entropy lines for “DPMC  $\alpha = 0$ ” and “vTS  $\alpha = 0$ ” are close, the distributions are not necessarily similar. As expected, when  $\alpha$  is fixed to 1, the modelled distributions become even further away from the actual ones.

Scheme	$\alpha$	20 dB	14 dB
eDPMC	0	7.6	13.2
	1	8.0	14.7
	$\mathcal{N}$	7.4	13.3
eVTS	0	11.4	16.5
	1	8.7	14.9
	$\mathcal{N}$	11.1	16.5

**Table 3** *The effect of the phase factor on Gaussian compensation.*

Table 3 contains word error rates for the same contrasts. Again, it shows only full-covariance compensation. With diagonal covariances the trends are again the same but less pronounced. For eDPMC, the effect of different phase factor models is as expected. Whether  $\alpha$  is distributed around 0 or fixed to 0 mostly affects the covariances. Though this does have an effect on the cross-entropy, since the change to the covariance matrices is fairly uniform across components, this makes little difference for discrimination. However, setting  $\alpha$  to the unlikely value 1 affects performance negatively.

The results for vTS are more surprising. Again, there is little difference between setting  $\alpha$  to 0 and letting it be distributed around 0. For vTS, this by definition does not affect components’ means, but only their covariances. However, setting it to 1 does improve performance. This may be because overestimation of the mode (see section 3.2) improves modelling for some components. Preliminary results suggest that which value of  $\alpha$  yields the best cross-entropy varies with different distances between the speech and noise means. A possible hypothesis is that for different tasks, different settings for  $\alpha$  optimise compensation for components at a speech-noise distance where miscompensation is most likely to cause recognition errors. This would explain why the optimal  $\alpha$  is different for different corpora [14]. However, this is material for future research.

What the results here do show is that while modelling  $\alpha$  with a distribution reduces the distance to the actual distribution, as evidenced by the improving cross-entropy, discrimination is not helped. Section 3.2 has pointed out that the only effect of using a distribution for the phase factor over a fixed value at the distribution’s mode is a fairly equal bias on the covariance, which is unlikely to influence discrimination much. It has also discussed how in practice the noise estimation can subsume this bias. Using a distribution over the phase factor rather than a fixed value as is currently done, is therefore unlikely to cause gains in a practical speech recogniser.

## 8 Conclusion

This work has introduced a new technique for computing the likelihood of a corrupted speech observation vector. It does not use a parametric density, but rather a sampling method, which approximates the integral over speech, noise and phase factor that the likelihood consists of. Because the probability density has an awkward shape, the integral is first transformed. Then, sequential importance re-sampling deals with the high dimensionality. As the number of samples goes to infinity, this approximation comes arbitrarily close to the real likelihood.

Because the method cannot precompute distributions, it is too slow to embed in a speech recogniser. However, it is possible to find the KL divergence from approximations to the corrupted speech distribution to the real one up to a constant. The new method essentially gives the point where the KL divergence is 0, so it can be assessed how close compensation methods are to the ideal. For the recognition experiments, the compensation schemes work on extended feature vectors to provide the best compensation and to avoid further approximations. The KL divergence for different compensation methods appears to predict their word error rates well. One of these compensation methods is iterative data-driven parallel model combination (IDPMC), which takes impractically long to train but then is fast to run a speech recogniser with. An extension of iterative data-driven parallel model combination comes close to transformed-space sampling in terms of cross-entropy, and improves the word error rate substantially. Given the link between the cross-entropy and the word error rate, this should indicate the best possible performance with these speech and noise models.

Using the KL divergence technique, it also becomes possible to examine approximations to the mismatch function. These include assuming the corrupted speech distribution Gaussian, and diagonalising that Gaussian's covariance. One common approximation, assuming the phase factor fixed, has seen particular interest in recent years. This work introduces model compensation using a phase factor distribution for VTS, DPMC, and IDPMC. This turns out to have more effect on the cross-entropy than on discrimination. In particular, for VTS compensation setting the phase factor to a fixed value other than its mode appears to counter some effects of the vector Taylor series approximation at different signal-to-noise ratios.

## A Well-known equalities

The following useful equalities are well-known.

### A.1 Transforming variables of probability distributions

If variables  $\mathbf{x}$  and  $\mathbf{y}$  are deterministically linked, a probability distribution over  $\mathbf{x}$ ,  $p(\mathbf{x})$ , can be converted into one over  $\mathbf{y}$  with (see, for example, [3], 11.1.1)

$$p(\mathbf{y}) = p(\mathbf{x}) \left| \frac{\partial \mathbf{x}}{\partial \mathbf{y}} \right|. \quad (116)$$

### A.2 Matrix identities

The Woodbury identity relates three matrices  $\mathbf{A}$ ,  $\mathbf{B}$ ,  $\mathbf{C}$  (see e.g. [30], 3.2.2):

$$(\mathbf{A} + \mathbf{C}\mathbf{B}\mathbf{C}^T)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{C}(\mathbf{B}^{-1} + \mathbf{C}^T\mathbf{A}^{-1}\mathbf{C})^{-1}\mathbf{C}^T\mathbf{A}^{-1}. \quad (117)$$

The inverse of a block symmetric matrix is given by (see e.g. [30], 9.1.3)

$$\begin{bmatrix} \mathbf{A} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{B} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{D}^{-1} & -\mathbf{D}^{-1}\mathbf{C}^T\mathbf{B}^{-1} \\ -\mathbf{B}^{-1}\mathbf{C}\mathbf{D}^{-1} & \mathbf{E}^{-1} \end{bmatrix} \quad (118a)$$

$$= \begin{bmatrix} \mathbf{D}^{-1} & -\mathbf{A}^{-1}\mathbf{C}^T\mathbf{E}^{-1} \\ -\mathbf{E}^{-1}\mathbf{C}\mathbf{A}^{-1} & \mathbf{E}^{-1} \end{bmatrix}, \quad (118b)$$

where  $\mathbf{D} = \mathbf{A} - \mathbf{C}^T\mathbf{B}^{-1}\mathbf{C}$  is the Schur complement of the matrix with respect to  $\mathbf{B}$ , and  $\mathbf{E} = \mathbf{B} - \mathbf{C}\mathbf{A}^{-1}\mathbf{C}^T$  is the Schur complement of the matrix with respect to  $\mathbf{A}$ .

The determinant of the matrix is

$$\left| \begin{bmatrix} \mathbf{A} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{B} \end{bmatrix} \right| = |\mathbf{A}| \cdot |\mathbf{E}| = |\mathbf{B}| \cdot |\mathbf{D}|. \quad (119)$$

### A.3 Multi-variate Gaussian factorisation

It can be useful to decompose the evaluation of a multi-variate Gaussian into factors. An obvious choice of factors would be the actual distribution of one coefficient conditional on all previous ones. Straightforward derivations of this usually (e.g. [3]) assume that the Gaussian is normalised (so that constant factors can be dropped) and assume the input for the Gaussian is linear in the variable of interest (so that the integral over coefficients is constant). These assumptions, however, can not be made in this work. Below derivation therefore explicitly considers all constants.

Let  $q$  an unnormalised Gaussian density with parameters  $\mathbf{a}$  and  $\mathbf{b}$ ,

$$q\left(\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}\right) = \exp\left(-\frac{1}{2}\left(\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} - \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix}\right)^T \begin{bmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{bmatrix} \left(\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} - \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix}\right)\right), \quad (120)$$

where  $\Lambda$  is the precision matrix, the inverse of the covariance matrix  $\Sigma$ :

$$\Lambda = \begin{bmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{bmatrix} = \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix}^{-1} = \Sigma^{-1}. \quad (121)$$

From the expression for the inverse of a symmetric block matrix, given in (118), it follows that  $\Sigma_{aa}^{-1} = \Lambda_{aa} - \Lambda_{ab}\Lambda_{bb}^{-1}\Lambda_{ba}$ , which will be useful in the derivation below.

The density can be decomposed into a factor dependent on  $\mathbf{a}$  and one dependent on both  $\mathbf{a}$  and  $\mathbf{b}$ . The steps the derivation follows are (122a) expanding the terms; (122b) gathering terms containing  $\mathbf{b}$ ; (122c) completing the square and compensating for that; and finally (122d) simplifying.

$$\begin{aligned} q\left(\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}\right) &= \exp\left(-\frac{1}{2}\left(\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} - \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix}\right)^T \begin{bmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{bmatrix} \left(\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} - \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix}\right)\right) \\ &= \exp\left(-\frac{1}{2}(\mathbf{a} - \boldsymbol{\mu}_a)^T \Lambda_{aa} (\mathbf{a} - \boldsymbol{\mu}_a) - (\mathbf{b} - \boldsymbol{\mu}_b)^T \Lambda_{ba} (\mathbf{a} - \boldsymbol{\mu}_a) \right. \\ &\quad \left. - \frac{1}{2}(\mathbf{b} - \boldsymbol{\mu}_b)^T \Lambda_{bb} (\mathbf{b} - \boldsymbol{\mu}_b)\right) \\ &= \exp\left(-\frac{1}{2}(\mathbf{a} - \boldsymbol{\mu}_a)^T \Lambda_{aa} (\mathbf{a} - \boldsymbol{\mu}_a) - \mathbf{b}^T \Lambda_{ba} (\mathbf{a} - \boldsymbol{\mu}_a) + \boldsymbol{\mu}_b^T \Lambda_{ba} (\mathbf{a} - \boldsymbol{\mu}_a) \right. \\ &\quad \left. - \frac{1}{2}\mathbf{b}^T \Lambda_{bb} \mathbf{b} + \mathbf{b}^T \Lambda_{bb} \boldsymbol{\mu}_b - \frac{1}{2}\boldsymbol{\mu}_b^T \Lambda_{bb} \boldsymbol{\mu}_b\right) \quad (122a) \\ &= \exp\left(-\frac{1}{2}(\mathbf{a} - \boldsymbol{\mu}_a)^T \Lambda_{aa} (\mathbf{a} - \boldsymbol{\mu}_a) + \boldsymbol{\mu}_b^T \Lambda_{ba} (\mathbf{a} - \boldsymbol{\mu}_a) - \frac{1}{2}\boldsymbol{\mu}_b^T \Lambda_{bb} \boldsymbol{\mu}_b \right. \\ &\quad \left. - \frac{1}{2}\mathbf{b}^T \Lambda_{bb} \mathbf{b} + \mathbf{b}^T \Lambda_{bb} (\boldsymbol{\mu}_b - \Lambda_{bb}^{-1} \Lambda_{ba} (\mathbf{a} - \boldsymbol{\mu}_a))\right) \quad (122b) \\ &= \exp\left(-\frac{1}{2}(\mathbf{a} - \boldsymbol{\mu}_a)^T \Lambda_{aa} (\mathbf{a} - \boldsymbol{\mu}_a) + \boldsymbol{\mu}_b^T \Lambda_{ba} (\mathbf{a} - \boldsymbol{\mu}_a) - \frac{1}{2}\boldsymbol{\mu}_b^T \Lambda_{bb} \boldsymbol{\mu}_b \right. \\ &\quad \left. - \frac{1}{2}(\mathbf{b} - (\boldsymbol{\mu}_b - \Lambda_{bb}^{-1} \Lambda_{ba} (\mathbf{a} - \boldsymbol{\mu}_a)))^T \Lambda_{bb} (\mathbf{b} - (\boldsymbol{\mu}_b - \Lambda_{bb}^{-1} \Lambda_{ba} (\mathbf{a} - \boldsymbol{\mu}_a))) \right. \\ &\quad \left. + \frac{1}{2}(\boldsymbol{\mu}_b - \Lambda_{bb}^{-1} \Lambda_{ba} (\mathbf{a} - \boldsymbol{\mu}_a))^T \Lambda_{bb} (\boldsymbol{\mu}_b - \Lambda_{bb}^{-1} \Lambda_{ba} (\mathbf{a} - \boldsymbol{\mu}_a))\right) \quad (122c) \\ &= \exp\left(-\frac{1}{2}(\mathbf{a} - \boldsymbol{\mu}_a)^T (\Lambda_{aa} - \Lambda_{ab}\Lambda_{bb}^{-1}\Lambda_{ba}) (\mathbf{a} - \boldsymbol{\mu}_a) \right. \\ &\quad \left. - \frac{1}{2}(\mathbf{b}^T - (\boldsymbol{\mu}_b - \Lambda_{bb}^{-1} \Lambda_{ba} (\mathbf{a} - \boldsymbol{\mu}_a)))^T \Lambda_{bb} (\mathbf{b}^T - (\boldsymbol{\mu}_b - \Lambda_{bb}^{-1} \Lambda_{ba} (\mathbf{a} - \boldsymbol{\mu}_a)))\right) \\ &= \exp\left(-\frac{1}{2}(\mathbf{a} - \boldsymbol{\mu}_a)^T \Sigma_{aa}^{-1} (\mathbf{a} - \boldsymbol{\mu}_a)\right) \\ &\quad \exp\left(-\frac{1}{2}(\mathbf{b}^T - (\boldsymbol{\mu}_b - \Lambda_{bb}^{-1} \Lambda_{ba} (\mathbf{a} - \boldsymbol{\mu}_a)))^T \Lambda_{bb} (\mathbf{b}^T - (\boldsymbol{\mu}_b - \Lambda_{bb}^{-1} \Lambda_{ba} (\mathbf{a} - \boldsymbol{\mu}_a)))\right). \quad (122d) \end{aligned}$$

A normalised Gaussian can be factorised analogously. Using (119), the determinant of the block matrix  $\Sigma$  can be decomposed into the determinants of the covariance matrices of the two terms in (122):  $|\Sigma| = |\Sigma_{aa}| \cdot |\Lambda_{bb}^{-1}|$ . A normalised Gaussian then can

be decomposed into two normalised Gaussians:

$$\begin{aligned}
 & \mathcal{N} \left( \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}; \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{bmatrix} \right) \\
 &= |2\pi\boldsymbol{\Sigma}|^{-1/2} q \left( \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} \right) \\
 &= |2\pi\boldsymbol{\Sigma}_{aa}|^{-1/2} \cdot |2\pi\boldsymbol{\Lambda}_{bb}^{-1}|^{-1/2} \exp \left( -\frac{1}{2}(\mathbf{a} - \boldsymbol{\mu}_a)^\top \boldsymbol{\Sigma}_{aa}^{-1}(\mathbf{a} - \boldsymbol{\mu}_a) \right) \\
 &\quad \exp \left( -\frac{1}{2}(\mathbf{b}^\top - (\boldsymbol{\mu}_b - \boldsymbol{\Lambda}_{bb}^{-1}\boldsymbol{\Lambda}_{ba}(\mathbf{a} - \boldsymbol{\mu}_a)))^\top \boldsymbol{\Lambda}_{bb}(\mathbf{b}^\top - (\boldsymbol{\mu}_b - \boldsymbol{\Lambda}_{bb}^{-1}\boldsymbol{\Lambda}_{ba}(\mathbf{a} - \boldsymbol{\mu}_a))) \right) \\
 &= \mathcal{N}(\mathbf{a}; \boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a) \mathcal{N}(\mathbf{b}; \boldsymbol{\mu}_b - \boldsymbol{\Lambda}_{bb}^{-1}\boldsymbol{\Lambda}_{ba}(\mathbf{a} - \boldsymbol{\mu}_a), \boldsymbol{\Lambda}_{bb}^{-1}) \\
 &= \mathcal{N}(\mathbf{a}; \boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a) \mathcal{N}(\mathbf{b}; \boldsymbol{\mu}_b - \boldsymbol{\Sigma}_{ba}\boldsymbol{\Sigma}_{aa}^{-1}(\mathbf{a} - \boldsymbol{\mu}_a), \boldsymbol{\Sigma}_{bb} - \boldsymbol{\Sigma}_{ba}\boldsymbol{\Sigma}_{aa}^{-1}\boldsymbol{\Sigma}_{ab}). \quad (123)
 \end{aligned}$$

If the density  $q$  is a probability distribution and  $\mathbf{a}$  and  $\mathbf{b}$  are distributed according to it:

$$\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} \sim \mathcal{N} \left( \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{bmatrix} \right), \quad (124)$$

then the two factors in (123) are the marginal probability distributions of  $\mathbf{a}$  and the distribution of  $\mathbf{b}$  conditional on  $\mathbf{a}$ , so that

$$\mathbf{a} \sim \mathcal{N}(\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a); \quad (125a)$$

$$\mathbf{b} | \mathbf{a} \sim \mathcal{N}(\boldsymbol{\mu}_b - \boldsymbol{\Lambda}_{bb}^{-1}\boldsymbol{\Lambda}_{ba}(\mathbf{a} - \boldsymbol{\mu}_a), \boldsymbol{\Lambda}_{bb}^{-1}) \quad (125b)$$

$$\sim \mathcal{N}(\boldsymbol{\mu}_b - \boldsymbol{\Sigma}_{ba}\boldsymbol{\Sigma}_{aa}^{-1}(\mathbf{a} - \boldsymbol{\mu}_a), \boldsymbol{\Sigma}_{bb} - \boldsymbol{\Sigma}_{ba}\boldsymbol{\Sigma}_{aa}^{-1}\boldsymbol{\Sigma}_{ab}). \quad (125c)$$

This is a standard result. Note that the distribution of  $\mathbf{a}$  is more easily expressed in terms of the joint's covariance matrix, and the distribution of  $\mathbf{b} | \mathbf{a}$  in terms of the precision matrix.

## B Domain-specific mismatch function

The mismatch in the log-spectral domain was given in (3a). It assumed that the features  $y_i, x_i, n_i$  used the power spectrum. This section will write the power  $\beta$  applied to the spectral coefficients explicitly as  $y_i^{(\beta)}, x_i^{(\beta)}, n_i^{(\beta)}$ , so that (3a) becomes:

$$\exp(y_i^{(2)}) = \exp(x_i^{(2)}) + \exp(n_i^{(2)}) + 2\alpha_i \exp\left(\frac{1}{2}x_i^{(2)} + \frac{1}{2}n_i^{(2)}\right). \quad (126)$$

The expression for the mismatch relating vectors in domains with different powers than 2 derives from this using an assumption about the mel-filtered spectrum. A mel-filtered spectral coefficient is a weighted sum of spectral coefficients to the power of  $\beta$  (see (9c)), which can be assumed equal to the power of the sum:

$$\tilde{Y}_i^{(\beta)} = \sum_k w_{ik} |Y[k]|^\beta \simeq \left( \sum_k w_{ik} |Y[k]| \right)^\beta. \quad (127)$$

The log-spectral coefficients are found by taking the logarithm of this, so that

$$y_i^{(\beta)} = \log(\tilde{Y}_i^{(\beta)}) \simeq \beta \log \left( \sum_k w_{ik} |Y[k]| \right). \quad (128)$$

It is easy to see that coefficients acquired from the  $\beta$ th-power domain are then assumed related to those using a power of 2 by

$$\mathbf{y}_i^{(\beta)} = \frac{\beta}{2} \mathbf{y}_i^{(2)}; \quad \mathbf{x}_i^{(\beta)} = \frac{\beta}{2} \mathbf{x}_i^{(2)}; \quad \mathbf{n}_i^{(\beta)} = \frac{\beta}{2} \mathbf{n}_i^{(2)}. \quad (129)$$

For the log-magnitude-spectrum ( $\beta = 1$ ), for example, coefficients  $\mathbf{y}_i, \mathbf{x}_i, \mathbf{n}_i$ , are smaller by a factor of 2. Therefore, (126) can be generalised to any power  $\beta$  by making up for the power:

$$\exp\left(\frac{2}{\beta} \mathbf{y}_i^{(\beta)}\right) = \exp\left(\frac{2}{\beta} \mathbf{x}_i^{(\beta)}\right) + \exp\left(\frac{2}{\beta} \mathbf{n}_i^{(\beta)}\right) + 2\alpha_i \exp\left(\frac{1}{\beta} \mathbf{x}_i^{(\beta)} + \frac{1}{\beta} \mathbf{n}_i^{(\beta)}\right), \quad (130a)$$

or

$$\mathbf{y}_i^{(\beta)} = \frac{\beta}{2} \log\left(\exp\left(\frac{2}{\beta} \mathbf{x}_i^{(\beta)}\right) + \exp\left(\frac{2}{\beta} \mathbf{n}_i^{(\beta)}\right) + 2\alpha_i \exp\left(\frac{1}{\beta} \mathbf{x}_i^{(\beta)} + \frac{1}{\beta} \mathbf{n}_i^{(\beta)}\right)\right) \quad (130b)$$

or

$$\begin{aligned} \mathbf{y}_i^{(\beta)} &= \frac{\beta}{2} \log\left(\exp\left(\frac{2}{\beta} \mathbf{x}_i^{(\beta)}\right) \right. \\ &\quad \left. \cdot \left(1 + \exp\left(\frac{2}{\beta} \mathbf{n}_i^{(\beta)} - \frac{2}{\beta} \mathbf{x}_i^{(\beta)}\right) + 2\alpha_i \exp\left(\frac{1}{\beta} \mathbf{n}_i^{(\beta)} - \frac{1}{\beta} \mathbf{x}_i^{(\beta)}\right)\right)\right) \\ &= \mathbf{x}_i^{(\beta)} + \frac{\beta}{2} \log\left(1 + \exp\left(\frac{2}{\beta} \mathbf{n}_i^{(\beta)} - \frac{2}{\beta} \mathbf{x}_i^{(\beta)}\right) + 2\alpha_i \exp\left(\frac{1}{\beta} \mathbf{n}_i^{(\beta)} - \frac{1}{\beta} \mathbf{x}_i^{(\beta)}\right)\right). \end{aligned} \quad (130c)$$

Derivatives of this are

$$\frac{d\mathbf{y}_i^{(\beta)}}{d\mathbf{x}_i^{(\beta)}} = \frac{\exp\left(\frac{2}{\beta} \mathbf{x}_i^{(\beta)}\right) + \alpha_i \exp\left(\frac{1}{\beta} \mathbf{x}_i^{(\beta)} + \frac{1}{\beta} \mathbf{n}_i^{(\beta)}\right)}{\exp\left(\frac{2}{\beta} \mathbf{x}_i^{(\beta)}\right) + \exp\left(\frac{2}{\beta} \mathbf{n}_i^{(\beta)}\right) + 2\alpha_i \exp\left(\frac{1}{\beta} \mathbf{x}_i^{(\beta)} + \frac{1}{\beta} \mathbf{n}_i^{(\beta)}\right)}; \quad (131a)$$

$$\frac{d\mathbf{y}_i^{(\beta)}}{d\mathbf{n}_i^{(\beta)}} = \frac{\exp\left(\frac{2}{\beta} \mathbf{n}_i^{(\beta)}\right) + \alpha_i \exp\left(\frac{1}{\beta} \mathbf{x}_i^{(\beta)} + \frac{1}{\beta} \mathbf{n}_i^{(\beta)}\right)}{\exp\left(\frac{2}{\beta} \mathbf{x}_i^{(\beta)}\right) + \exp\left(\frac{2}{\beta} \mathbf{n}_i^{(\beta)}\right) + 2\alpha_i \exp\left(\frac{1}{\beta} \mathbf{x}_i^{(\beta)} + \frac{1}{\beta} \mathbf{n}_i^{(\beta)}\right)} = 1 - \frac{d\mathbf{y}_i^{(\beta)}}{d\mathbf{x}_i^{(\beta)}}; \quad (131b)$$

$$\frac{d\mathbf{y}_i^{(\beta)}}{d\alpha_i^{(\beta)}} = \frac{\beta \exp\left(\frac{1}{\beta} \mathbf{x}_i^{(\beta)} + \frac{1}{\beta} \mathbf{n}_i^{(\beta)}\right)}{\exp\left(\frac{2}{\beta} \mathbf{x}_i^{(\beta)}\right) + \exp\left(\frac{2}{\beta} \mathbf{n}_i^{(\beta)}\right) + 2\alpha_i \exp\left(\frac{1}{\beta} \mathbf{x}_i^{(\beta)} + \frac{1}{\beta} \mathbf{n}_i^{(\beta)}\right)}. \quad (131c)$$

Some implementations of vts compensation (e.g. [27]) have used magnitude-spectrum features ( $\beta = 1$ ), but assumed the mismatch function was simply

$$\exp(\mathbf{y}_i^{(1)}) = \exp(\mathbf{x}_i^{(1)}) + \exp(\mathbf{n}_i^{(1)}). \quad (132)$$

It is interesting to see the effect of these assumptions. By converting this back to power-spectral features,

$$\exp\left(\frac{1}{2} \mathbf{y}_i^{(2)}\right) = \exp\left(\frac{1}{2} \mathbf{x}_i^{(2)}\right) + \exp\left(\frac{1}{2} \mathbf{n}_i^{(2)}\right); \quad (133a)$$

$$\begin{aligned} \exp(\mathbf{y}_i^{(2)}) &= \left(\exp\left(\frac{1}{2} \mathbf{x}_i^{(2)}\right) + \exp\left(\frac{1}{2} \mathbf{n}_i^{(2)}\right)\right)^2 \\ &= \exp(\mathbf{x}_i^{(2)}) + \exp(\mathbf{n}_i^{(2)}) + 2 \exp\left(\frac{1}{2} \mathbf{x}_i^{(2)}\right) \exp\left(\frac{1}{2} \mathbf{n}_i^{(2)}\right). \end{aligned} \quad (133b)$$

This is exactly equivalent to the real mismatch function, in (126), with  $\alpha = 1$ . This means that when [27] perform vts compensation with vectors in the magnitude domain and ignore the phase term, as in (132), this is equivalent to assuming  $\alpha = 1$  on log-power-spectral features. Also, when the noise model is ML-estimated, with the same mismatch function used for decoding, then the noise model parameters will subsume much of the difference between model and reality.

## C Transforming the likelihood for piecewise linear approximation

This section follows the transformation of the expression for the likelihood presented in [29]. The explanation of the idea behind it is in section 3.4. Section C.1 discusses the single-dimensional case as in the original paper. Section C.2 generalises it to more dimensions.

### C.1 Single-dimensional

The interaction between the log-spectral coefficients of the speech  $x$ , the noise  $n$ , and the observation  $y$  is assumed to be

$$\exp(y) = \exp(x) + \exp(n). \quad (134)$$

$y$  is set to its observed value,  $y_t$ .

The substitute variable introduced to replace the integration over  $x$  and  $n$  is defined

$$u = 1 - \exp(x - y_t), \quad (135a)$$

so that

$$\begin{aligned} n &= \log(\exp(y_t) - \exp(x)) = y_t + \log(1 - \exp(x - y_t)) \\ &= y_t + \log(u); \end{aligned} \quad (135b)$$

$$x = y_t + \log(1 - u). \quad (135c)$$

Two useful derivatives for transforming the integral are the following. The derivative of  $n$  with respect to  $y$  while keeping  $x$  fixed is

$$\frac{dn(x, y)}{dy} = \frac{\exp(y)}{\exp(y) - \exp(x)} = \frac{1}{1 - \exp(x - y)} = \frac{1}{u}. \quad (136a)$$

The notation  $n(x, y)$  is used to indicate the value of  $n$  that the setting of the other two variables  $(x, y)$  implies. Similarly, the derivative of  $x$  with respect to  $u$  while keeping  $y_t$  fixed is

$$\frac{dx(u, y_t)}{du} = \frac{-1}{1 - u}. \quad (136b)$$

As explained in section 3.4 (and see also section A.1), the transformation of the integral in the likelihood expression uses the absolute values of the two derivatives.

$$\begin{aligned} p(y_t) &= \int_{-\infty}^{y_t} p(y_t | x) p(x) dx \\ &= \int_{-\infty}^{y_t} \left| \frac{dn(x, y)}{dy} \right|_{y_t} \left| p(n(x, y_t)) p(x) \right| dx \\ &= \int_0^1 \left| \frac{dn(x, y)}{dy} \right|_{y_t} \left| p(n(u, y_t)) \left| \frac{dx(u, y_t)}{du} \right| p(x(u, y_t)) \right| du \\ &= \int_0^1 \frac{1}{u} \mathcal{N}(y_t + \log(u); \mu_n, \sigma_n^2) \frac{1}{1 - u} \mathcal{N}(y_t + \log(1 - u); \mu_x, \sigma_x^2) du. \end{aligned} \quad (137)$$

Rewriting only the left-hand term of the integrand, noting that  $\frac{1}{u} = \exp(-\log(u))$ ,

$$\begin{aligned}
 & \frac{1}{u} \mathcal{N}(\log(u) + y_t; \mu_n, \sigma_n^2) \\
 &= \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{(\log(u) + y_t - \mu_n)^2}{2\sigma_n^2} - \log(u)\right) \\
 &= \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{(\log(u) + y_t - \mu_n + \sigma_n^2)^2}{2\sigma_n^2} + \frac{1}{2}\sigma_n^2 + y_t - \mu_n\right) \\
 &= \exp\left(\frac{1}{2}\sigma_n^2 + y_t - \mu_n\right) \mathcal{N}(\log(u); \mu_n - \sigma_n^2 - y_t, \sigma_n^2). \tag{138}
 \end{aligned}$$

The right-hand side of the integrand can be rewritten in a similar way, so that the likelihood expression becomes

$$\begin{aligned}
 p(y_t) &= \exp\left(\frac{1}{2}\sigma_n^2 + \frac{1}{2}\sigma_x^2 - \mu_n - \mu_x + 2y_t\right) \\
 & \int_0^1 \mathcal{N}(\log(u); \mu_n - \sigma_n^2 - y_t, \sigma_n^2) \mathcal{N}(\log(1-u); \mu_x - \sigma_x^2 - y_t, \sigma_x^2) du. \tag{139}
 \end{aligned}$$

By approximating  $\log(u)$  and  $\log(1-u)$  with a piecewise linear function [29], the integral can be written as a sum of integrals over part of a Gaussian and a constant factor.

## C.2 Multi-dimensional

That the derivation above can use scalars crucially relies on two assumptions. The assumption that the  $i$ th coordinate of the clean speech only influences the  $i$ th coordinate of the corrupted speech is only valid in the log-spectral domain. The assumption that the coordinates of both the clean speech and the corrupted speech are uncorrelated is marginally valid in the cepstral domain, and invalid in the log-spectral domain. The following generalises the derivation above to a vector of MFCCs. MFCCs are related to log-spectral coefficients by a linear transformation. As long as the distributions in the log-spectral domain are not assumed uncorrelated, therefore, a derivation in the log-spectral domain can be used.

The relation of the clean speech, noise, and corrupted speech for every dimension is the same as the single-dimensional case in (134), so that for vectors:

$$\mathbf{exp}(\mathbf{y}) = \mathbf{exp}(\mathbf{x}) + \mathbf{exp}(\mathbf{n}). \tag{140}$$

Again,  $\mathbf{y}$  is set to its observed value,  $\mathbf{y}_t$ .

The coefficients of the substitute variable  $\mathbf{u}$  are defined as in (135), so that in vector notation,

$$\mathbf{u} = \mathbf{1} - \mathbf{exp}(\mathbf{x} - \mathbf{y}_t), \tag{141}$$

so that

$$\mathbf{n} = \mathbf{y}_t + \mathbf{log}(\mathbf{u}); \tag{142}$$

$$\mathbf{x} = \mathbf{y}_t + \mathbf{log}(\mathbf{1} - \mathbf{u}), \tag{143}$$

where  $\mathbf{1}$  is a vector with all entries set to 1.

The absolutes of the derivatives that the transformation of the feature space results in in one-dimensional space generalise to determinants of partial derivatives. Since the relationships between speech, noise, substitute variable, and observation are element-by-element in log-spectral space, the partial derivatives are diagonal. The generalisations of the derivatives in (136) therefore is (note that  $\mathbf{u} \in \langle 0, 1 \rangle$ )

$$\left| \frac{\partial \mathbf{n}(\mathbf{x}, \mathbf{y})}{\partial \mathbf{y}} \right| = \left| \prod_i \frac{dn(x_i, y_i)}{dy_i} \right| = \left| \prod_i \frac{1}{1 - \exp(x_i - y_i)} \right| = \prod_i \frac{1}{u_i}; \quad (144)$$

$$\left| \frac{\partial \mathbf{x}(\mathbf{u}, \mathbf{y}_t)}{\partial \mathbf{u}} \right| = \left| \prod_i \frac{dx(u_i, y_{t,i})}{du_i} \right| = \prod_i \frac{1}{1 - u_i}. \quad (145)$$

The additive noise and the clean speech are distributed as

$$\mathbf{n} \sim \mathcal{N}(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n); \quad \mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x). \quad (146)$$

The likelihood of  $\mathbf{y}_t$  generalises (137):

$$\begin{aligned} p(\mathbf{y}_t) &= \int p(\mathbf{y}_t | \mathbf{x}) p(\mathbf{x}) d\mathbf{x} \\ &= \int \left| \frac{\partial \mathbf{n}(\mathbf{x}, \mathbf{y})}{\partial \mathbf{y}} \right|_{\mathbf{y}_t} p(\mathbf{n}(\mathbf{x}, \mathbf{y})) p(\mathbf{x}) d\mathbf{x} \\ &= \int_{\langle 0, 1 \rangle^d} \left| \frac{\partial \mathbf{n}(\mathbf{x}, \mathbf{y})}{\partial \mathbf{y}} \right|_{\mathbf{y}_t} p(\mathbf{n}(\mathbf{u}, \mathbf{y})) \left| \frac{\partial \mathbf{x}(\mathbf{u}, \mathbf{y}_t)}{\partial \mathbf{u}} \right| p(\mathbf{x}(\mathbf{u}, \mathbf{y}_t)) d\mathbf{u} \\ &= \int_{\langle 0, 1 \rangle^d} \left( \prod_i \frac{1}{u_i} \right) \mathcal{N}(\mathbf{y}_t + \mathbf{log}(\mathbf{u}); \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n) \\ &\quad \left( \prod_i \frac{1}{1 - u_i} \right) \mathcal{N}(\mathbf{y}_t + \mathbf{log}(\mathbf{1} - \mathbf{u}); \boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x) d\mathbf{u}. \end{aligned} \quad (147)$$

Noting that

$$\prod_i \frac{1}{u_i} = \exp\left(-\sum_i \log(u_i)\right) = \exp(-\mathbf{log}(\mathbf{u})^T \mathbf{1}); \quad (148)$$

$$\begin{aligned} \mathcal{N}(\mathbf{log}(\mathbf{u}) + \mathbf{y}_t; \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n) &= \\ &|2\pi\boldsymbol{\Sigma}_n|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{log}(\mathbf{u}) + \mathbf{y}_t - \boldsymbol{\mu}_n)^T \boldsymbol{\Sigma}_n^{-1} (\mathbf{log}(\mathbf{u}) + \mathbf{y}_t - \boldsymbol{\mu}_n)\right), \end{aligned} \quad (149)$$

the left term in (147) becomes (generalising (138))

$$\begin{aligned} &\left( \prod_i \frac{1}{u_i} \right) \mathcal{N}(\mathbf{log}(\mathbf{u}) + \mathbf{y}_t; \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n) \\ &= |2\pi\boldsymbol{\Sigma}_n|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{log}(\mathbf{u}) + \mathbf{y}_t - \boldsymbol{\mu}_n)^T \boldsymbol{\Sigma}_n^{-1} (\mathbf{log}(\mathbf{u}) + \mathbf{y}_t - \boldsymbol{\mu}_n) - \mathbf{log}(\mathbf{u})^T \mathbf{1}\right) \\ &= |2\pi\boldsymbol{\Sigma}_n|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{log}(\mathbf{u}) + \mathbf{y}_t - \boldsymbol{\mu}_n + \boldsymbol{\Sigma}_n \mathbf{1})^T \boldsymbol{\Sigma}_n^{-1} (\mathbf{log}(\mathbf{u}) + \mathbf{y}_t - \boldsymbol{\mu}_n + \boldsymbol{\Sigma}_n \mathbf{1})\right. \\ &\quad \left. + \frac{1}{2} \mathbf{1}^T \boldsymbol{\Sigma}_n \mathbf{1} + \mathbf{1}^T \mathbf{y}_t - \mathbf{1}^T \boldsymbol{\mu}_n\right) \\ &= \mathcal{N}(\mathbf{log}(\mathbf{u}); \boldsymbol{\mu}_n - \mathbf{y}_t - \boldsymbol{\Sigma}_n \mathbf{1}, \boldsymbol{\Sigma}_n) \exp\left(\frac{1}{2} \mathbf{1}^T \boldsymbol{\Sigma}_n \mathbf{1} + \mathbf{1}^T \mathbf{y}_t - \mathbf{1}^T \boldsymbol{\mu}_n\right). \end{aligned} \quad (150)$$

Applying the same process to the right term, the likelihood of  $\mathbf{y}_t$  becomes

$$p(\mathbf{y}_t) = \exp\left(\frac{1}{2}\mathbf{1}^\top \boldsymbol{\Sigma}_n \mathbf{1} + \frac{1}{2}\mathbf{1}^\top \boldsymbol{\Sigma}_x \mathbf{1} - \mathbf{1}^\top \boldsymbol{\mu}_n - \mathbf{1}^\top \boldsymbol{\mu}_x + 2 \cdot \mathbf{1}^\top \mathbf{y}_t\right) \int_{\langle 0,1 \rangle^d} \mathcal{N}(\mathbf{log}(\mathbf{u}); \boldsymbol{\mu}_n - \boldsymbol{\Sigma}_n \mathbf{1} - \mathbf{y}_t, \boldsymbol{\Sigma}_n) \mathcal{N}(\mathbf{log}(\mathbf{1} - \mathbf{u}); \boldsymbol{\mu}_x - \boldsymbol{\Sigma}_x \mathbf{1} - \mathbf{y}_t, \boldsymbol{\Sigma}_x) d\mathbf{u}. \quad (151)$$

In the single-dimensional case, the integral is approximated with 8 line segments. In this case, the approximation would use  $8^d$  hyperplanes. Since  $\mathbf{u}$  has as many dimensions as there are filterbank coefficients, a piecewise linear approximation is infeasible.

## D Transforming the likelihood for transformed-space sampling

To approximate the integral in the expression for the likelihood of the observation, this work uses sequential importance re-sampling. A number of transformations of the integral are required, some of the details of which are in this appendix.

The detailed derivation of the transformation of the single-dimensional version of the integral is in section D.1. The generalisation of this transformation to the multi-dimensional case is in section D.2. One of the two factorisations of the multi-dimensional integrand that this work presents is detailed in section D.3. The form of the proposal distribution that approximates the single-dimensional integrand and the factors of the multi-dimensional integrand is in section D.4.

### D.1 Terms of the integrand for transformed-space sampling

In section 5.2 on page 34, one half of a one-dimensional version of the corrupted speech likelihood is rewritten to (repeated from (84)):

$$p(\mathbf{y}_t, x \leq n) = \int p(\alpha) \int_0^\infty \left| \frac{\partial x(\mathbf{u}, \mathbf{y}_t, \alpha)}{\partial \mathbf{u}} \right| \cdot \left| \frac{\partial n(x, \mathbf{y}, \alpha)}{\partial \mathbf{y}} \right|_{\mathbf{y}_t, x(\mathbf{u}, \mathbf{y}_t, \alpha)} \cdot p(x(\mathbf{u}, \mathbf{y}_t, \alpha)) \cdot p(n(\mathbf{u}, \mathbf{y}_t, \alpha)) d\mathbf{u} d\alpha, \quad (152a)$$

where (repeated from (81))

$$\mathbf{u} = \mathbf{n} - x. \quad (152b)$$

Because the derivations of the Jacobians and of  $x(\mathbf{u}, \mathbf{y}_t, \alpha)$  and  $n(\mathbf{u}, \mathbf{y}_t, \alpha)$  are long, they are given here.

The mismatch function is (repeated from (80))

$$\exp(\mathbf{y}_t) = \exp(x) + \exp(n) + 2\alpha \exp\left(\frac{1}{2}x + \frac{1}{2}n\right). \quad (153)$$

To express  $n$  as a function of  $x, \mathbf{y}_t, \alpha$ , (153) can be rewritten to

$$\exp(n) + 2\alpha \exp\left(\frac{1}{2}x\right) \exp\left(\frac{1}{2}n\right) = \exp(\mathbf{y}_t) - \exp(x); \quad (154a)$$

$$\left(\exp\left(\frac{1}{2}n\right) + \alpha \exp\left(\frac{1}{2}x\right)\right)^2 = \exp(\mathbf{y}_t) - \exp(x) + \left(\alpha \exp\left(\frac{1}{2}x\right)\right)^2. \quad (154b)$$

This is where it becomes useful that the computation is restricted to the region where  $x \leq n$  so that  $n$  has only one solution. Since  $-1 \leq \alpha$  (as shown in section 2.2.1.1), the squared expression on the left-hand side,  $\exp(\frac{1}{2}n) + \alpha \exp(\frac{1}{2}x)$ , is always non-negative. Therefore,

$$\exp(\frac{1}{2}n) = -\alpha \exp(\frac{1}{2}x) + \sqrt{\exp(y_t) - \exp(x) + \alpha^2 \exp(x)}; \quad (154c)$$

$$n = 2 \log \left( -\alpha \exp(\frac{1}{2}x) + \sqrt{\exp(y_t) + \exp(x) (\alpha^2 - 1)} \right). \quad (154d)$$

To express  $n$  as a function of  $u, y_t, \alpha$ , (153) can be rewritten with  $x = n - u$  from (152b):

$$\begin{aligned} \exp(y_t) &= \exp(n - u) + \exp(n) + 2\alpha \exp(\frac{1}{2}n - \frac{1}{2}u + \frac{1}{2}n) \\ &= \exp(n) (1 + \exp(-u) + 2\alpha \exp(-\frac{1}{2}u)); \end{aligned} \quad (155a)$$

$$\exp(n) = \frac{\exp(y_t)}{1 + \exp(-u) + 2\alpha \exp(-\frac{1}{2}u)}; \quad (155b)$$

$$n = y_t - \log(1 + \exp(-u) + 2\alpha \exp(-\frac{1}{2}u)). \quad (155c)$$

Similarly,  $x$  can be expressed as a function of  $u, y_t, \alpha$  by rewriting (153) with  $n = u + x$  from (152b):

$$\begin{aligned} \exp(y_t) &= \exp(x) + \exp(u + x) + 2\alpha \exp(\frac{1}{2}x + \frac{1}{2}u + \frac{1}{2}x) \\ &= \exp(x) (1 + \exp(u) + 2\alpha \exp(\frac{1}{2}u)); \end{aligned} \quad (156a)$$

$$\exp(y_t - x) = 1 + \exp(u) + 2\alpha \exp(\frac{1}{2}u); \quad (156b)$$

$$x = y_t - \log(1 + \exp(u) + 2\alpha \exp(\frac{1}{2}u)). \quad (156c)$$

Because  $u$  was chosen to relate  $x$  and  $n$  symmetrically, (155c) and (156c) are the same except that  $u$  is replaced by  $-u$ .

An equality that will come in useful derives from (156b):

$$\begin{aligned} \sqrt{\exp(y_t) + \exp(x) (\alpha^2 - 1)} &= \exp(\frac{1}{2}x) \sqrt{\exp(y_t - x) + (\alpha^2 - 1)} \\ &= \exp(\frac{1}{2}x) \sqrt{\exp(u) + 2\alpha \exp(\frac{1}{2}u) + \alpha^2} \\ &= \exp(\frac{1}{2}x) (\exp(\frac{1}{2}u) + \alpha). \end{aligned} \quad (157)$$

The Jacobians in (84) are derivatives of (156c) and (154d):

$$\frac{\partial x(u, y_t, \alpha)}{\partial u} = -\frac{\exp(u) + \alpha \exp(\frac{1}{2}u)}{1 + \exp(u) + 2\alpha \exp(\frac{1}{2}u)} = -\frac{\exp(\frac{1}{2}u) (\exp(\frac{1}{2}u) + \alpha)}{1 + \exp(u) + 2\alpha \exp(\frac{1}{2}u)}; \quad (158a)$$

$$\begin{aligned}
 & \frac{\partial n(x, y, \alpha)}{\partial y} \\
 &= \frac{2}{\sqrt{\exp(y) + \exp(x)(\alpha^2 - 1) - \alpha \exp(\frac{1}{2}x)}} \cdot \frac{\exp(y)}{2\sqrt{\exp(y) + \exp(x)(\alpha^2 - 1)}} \\
 &= \frac{\exp(y)}{(\exp(\frac{1}{2}x)(\exp(\frac{1}{2}u) + \alpha) - \alpha \exp(\frac{1}{2}x)) \exp(\frac{1}{2}x)(\exp(\frac{1}{2}u) + \alpha)} \\
 &= \frac{\exp(y - x)}{\exp(\frac{1}{2}u)(\exp(\frac{1}{2}u) + \alpha)} = \frac{1 + \exp(u) + 2\alpha \exp(\frac{1}{2}u)}{\exp(\frac{1}{2}u)(\exp(\frac{1}{2}u) + \alpha)}. \quad (158b)
 \end{aligned}$$

When these are multiplied, as in the integral in (152a), they drop out against each other, except for the negation:

$$\frac{\partial x(u, y_t, \alpha)}{\partial u} \frac{\partial n(x, y, \alpha)}{\partial y} \Big|_{y_t} = -1. \quad (158c)$$

This does not seem to be an intrinsic property of the process.

## D.2 Transforming the space of the multi-dimensional integral

The transformation of the integral that returns the likelihood of the corrupted-speech observation is more laborious for multiple dimensions than for a single dimension. The derivation uses three steps. First, the integral is split into separate dimensions. Then, each of the integrals for one dimension is rewritten similarly to appendix D.1. Finally, the dimensions are collated.

The full expression for the likelihood of observation  $\mathbf{y}_t$  is (repeated from (19c))

$$p(\mathbf{y}_t) = \iiint p(\mathbf{y}_t | \mathbf{x}, \mathbf{n}, \alpha) p(\mathbf{x}) p(\mathbf{n}) p(\alpha) d\mathbf{x} d\mathbf{n} d\alpha. \quad (159)$$

Just like in the single-dimensional case, the integration over the clean speech and the additive noise will be rewritten as an integral over a substitute variable. For each dimension, this substitution is the same as the one in (81). In multiple dimensions, the substitute variable  $\mathbf{u}$  also relates the speech  $\mathbf{x}$  and the noise  $\mathbf{n}$  symmetrically:

$$\mathbf{u} = \mathbf{x} - \mathbf{n}. \quad (160)$$

However, the transformation of the integral will work one dimension at a time. Per dimension, the derivation will be split in two regions which use symmetric derivations, like in section 5.2.1. Again, the derivation will be explicitly given only for  $x_i \leq n_i$ , with  $n_i < x_i$  completely analogous. By formulating (159) recursively, it can be transformed one scalar at a time. The following marginalises out one variable at a time, starting with  $\alpha_i$ :

$$\begin{aligned}
 & p(\mathbf{y}_{t,i:d}, x_i \leq n_i | \mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \alpha_{1:i-1}) \\
 &= \int p(\alpha_i | \alpha_{1:i-1}) p(\mathbf{y}_{t,i:d}, x_i \leq n_i | \mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \alpha_{1:i}) d\alpha_i, \quad (161a)
 \end{aligned}$$

where, marginalising out  $x_i$ ,

$$\begin{aligned} & p(\mathbf{y}_{t,i:d}, x_i \leq n_i | \mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i}) \\ &= \int p(x_i | \mathbf{x}_{1:i-1}) p(\mathbf{y}_{t,i:d}, x_i \leq n_i | \mathbf{x}_{1:i}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i}) dx_i, \end{aligned} \quad (161b)$$

where, with the restriction  $x_i \leq n_i$  subsumed in the range of the integration over  $n_i$ ,

$$\begin{aligned} & p(\mathbf{y}_{t,i:d}, x_i \leq n_i | \mathbf{x}_{1:i}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i}) \\ &= \int_{x_i}^{\infty} p(n_i | \mathbf{n}_{1:i-1}) p(\mathbf{y}_{t,i:d} | \mathbf{x}_{1:i}, \mathbf{n}_{1:i}, \boldsymbol{\alpha}_{1:i}) dn_i. \end{aligned} \quad (161c)$$

The integrals in (161c) and in (161b) can then be re-expressed as one integral over the substitute variable.

First, the integral over  $n_i$  in (161c) can be written without the integral. This is because given the clean speech, additive noise, and phase factor for one dimension, the corrupted speech for that dimension is deterministic:

$$p(\mathbf{y}_{t,i} | x_i, n_i, \alpha_i) = \delta_{f(x_i, n_i, \alpha_i)}(\mathbf{y}_{t,i}). \quad (162)$$

The variable of the Dirac delta in (162) can be transformed using the Jacobian (see (116) in appendix A.1):

$$\begin{aligned} & p(\mathbf{y}_{t,i:d}, x_i \leq n_i | \mathbf{x}_{1:i}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i}) \\ &= \int_{x_i}^{\infty} p(n_i | \mathbf{n}_{1:i-1}) p(\mathbf{y}_{t,i} | x_i, n_i, \alpha_i) p(\mathbf{y}_{t,i+1:d} | \mathbf{x}_{1:i}, \mathbf{n}_{1:i}, \boldsymbol{\alpha}_{1:i}) dn_i \\ &= \int_{x_i}^{\infty} p(n_i | \mathbf{n}_{1:i-1}) \delta_{f(x_i, n_i, \alpha_i)}(\mathbf{y}_{t,i}) p(\mathbf{y}_{t,i+1:d} | \mathbf{x}_{1:i}, \mathbf{n}_{1:i}, \boldsymbol{\alpha}_{1:i}) dn_i \\ &= \int_{x_i}^{\infty} p(n_i | \mathbf{n}_{1:i-1}) \cdot \left| \frac{dn(x_i, \alpha_i, y_i)}{dy_i} \right|_{y_{t,i}} \cdot \delta_{n(x_i, \alpha_i, y_{t,i})}(n_i) \\ &\quad p(\mathbf{y}_{t,i+1:d} | \mathbf{x}_{1:i}, \mathbf{n}_{1:i}, \boldsymbol{\alpha}_{1:i}) dn_i \\ &= \left| \frac{dn(x_i, \alpha_i, y_i)}{dy_i} \right|_{y_{t,i}} \cdot \mathbf{1}(x_i \leq n_i) p(n(x_i, \alpha_i, y_{t,i}) | \mathbf{n}_{1:i-1}) \\ &\quad p(\mathbf{y}_{t,i+1:d} | \mathbf{x}_{1:i}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i}, n_i = n(x_i, \alpha_i, y_{t,i})). \end{aligned} \quad (163)$$

The next step is to substitute this result into (161b), and then replace the variable of the integral from  $x_i$  to  $u_i$ . The Jacobians that result from this are exactly the same as the ones in section 5.2.1 on page 34, in (84). Since the product of their absolutes is therefore again 1, they drop out.

$$\begin{aligned} & p(\mathbf{y}_{t,i:d}, x_i \leq n_i | \mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i}) \\ &= \int p(x_i | \mathbf{x}_{1:i-1}) \left| \frac{dn(x_i, \alpha_i, y_i)}{dy_i} \right|_{y_{t,i}} \mathbf{1}(x_i \leq n_i) p(n(x_i, \alpha_i, y_{t,i}) | \mathbf{n}_{1:i-1}) \\ &\quad p(\mathbf{y}_{t,i+1:d} | \mathbf{x}_{1:i}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i}, n_i = n(x_i, \alpha_i, y_{t,i})) dx_i \end{aligned}$$

$$\begin{aligned}
 &= \int_0^\infty \left| \frac{dx(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_i)}{d\mathbf{u}_i} \right| \cdot \left| \frac{dn(\mathbf{x}_i, \boldsymbol{\alpha}_i, \mathbf{y}_i)}{d\mathbf{y}_i} \right|_{\mathbf{y}_{t,i}} \\
 &\quad p(\mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) | \mathbf{x}_{1:i-1}) p(\mathbf{n}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) | \mathbf{n}_{1:i-1}) \\
 &\quad p(\mathbf{y}_{t,i+1:d} | \mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i}, \mathbf{x}_i = \mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}), \mathbf{n}_i = \mathbf{n}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i})) \\
 &\quad d\mathbf{u}_i \\
 &= \int_0^\infty p(\mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) | \mathbf{x}_{1:i-1}) p(\mathbf{n}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) | \mathbf{n}_{1:i-1}) \\
 &\quad p(\mathbf{y}_{t,i+1:d} | \mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i}, \mathbf{x}_i = \mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}), \mathbf{n}_i = \mathbf{n}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i})) \\
 &\quad d\mathbf{u}_i, \tag{164}
 \end{aligned}$$

Substituting this into (161a) gives one half of the likelihood:

$$\begin{aligned}
 &p(\mathbf{y}_{t,i:d}, \mathbf{x}_i \leq \mathbf{n}_i | \mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i-1}) \\
 &= \int p(\boldsymbol{\alpha}_i | \boldsymbol{\alpha}_{1:i-1}) \int_0^\infty p(\mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) | \mathbf{x}_{1:i-1}) p(\mathbf{n}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) | \mathbf{n}_{1:i-1}) \\
 &\quad p(\mathbf{y}_{t,i+1:d} | \mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i}, \mathbf{x}_i = \mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}), \mathbf{n}_i = \mathbf{n}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i})) \\
 &\quad d\mathbf{u}_i d\boldsymbol{\alpha}_i. \tag{165}
 \end{aligned}$$

This gives half the likelihood, because it is constrained to  $\mathbf{x}_i \leq \mathbf{n}_i$ . The other part, for  $\mathbf{n}_i < \mathbf{x}_i$ , has the exact same derivation with  $\mathbf{x}_i$  and  $\mathbf{n}_i$  swapped, and  $\mathbf{u}_i$  replaced by  $-\mathbf{u}_i$ . This is exactly the same as the single-dimensional case in appendix D.1. The full likelihood, expressed recursively, then combines integrals over  $\mathbf{u}_i \in [0, \infty)$  and  $\mathbf{u}_i \in \langle -\infty, 0)$ :

$$\begin{aligned}
 &p(\mathbf{y}_{t,i:d} | \mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i-1}) \\
 &= p(\mathbf{y}_{t,i:d}, \mathbf{x}_i \leq \mathbf{n}_i | \mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i-1}) \\
 &\quad + p(\mathbf{y}_{t,i:d}, \mathbf{n}_i < \mathbf{x}_i | \mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i-1}) \\
 &= \int p(\boldsymbol{\alpha}_i | \boldsymbol{\alpha}_{1:i-1}) \int p(\mathbf{x}_i(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) | \mathbf{x}_{1:i-1}) p(\mathbf{n}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) | \mathbf{n}_{1:i-1}) \\
 &\quad p(\mathbf{y}_{t,i+1:d} | \mathbf{x}_{1:i-1}, \mathbf{n}_{1:i-1}, \boldsymbol{\alpha}_{1:i}, \mathbf{x}_i = \mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}), \mathbf{n}_i = \mathbf{n}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i})) \\
 &\quad d\mathbf{u}_i d\boldsymbol{\alpha}_i. \tag{166}
 \end{aligned}$$

This recursive formulation is straightforward to unroll to

$$\begin{aligned}
 p(\mathbf{y}_t) &= p(\mathbf{y}_{t,1:d}) \\
 &= \int \left[ \prod_{i=1}^d p(\boldsymbol{\alpha}_i | \boldsymbol{\alpha}_{1:i-1}) \right] \int \left[ \prod_{i=1}^d p(\mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) | \mathbf{x}_{1:i-1}) \right] \\
 &\quad \left[ \prod_{i=1}^d p(\mathbf{n}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) | \mathbf{n}_{1:i-1}) \right] d\mathbf{u} d\boldsymbol{\alpha} \\
 &= \int p(\boldsymbol{\alpha}) \int p(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t)) p(\mathbf{n}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t)) d\mathbf{u} d\boldsymbol{\alpha}, \tag{167}
 \end{aligned}$$

where, analogously to the one-dimensional case,  $p(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t))$  denotes the value of the prior of  $\mathbf{x}$  evaluated at the value of  $\mathbf{x}$  implied by the values of  $(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t)$ , and similar

for  $p(\mathbf{n}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t))$ . To find these values for  $\mathbf{x}$  and  $\mathbf{n}$ , the relations in (156c) and (155c) apply per dimension:

$$\mathbf{x} = \mathbf{y}_t - \log(\mathbf{1} + \exp(\mathbf{u}) + 2\boldsymbol{\alpha} \circ \exp(\frac{1}{2}\mathbf{u})); \quad (168a)$$

$$\mathbf{n} = \mathbf{y}_t - \log(\mathbf{1} + \exp(-\mathbf{u}) + 2\boldsymbol{\alpha} \circ \exp(-\frac{1}{2}\mathbf{u})). \quad (168b)$$

In section 5.2.2, the full integrand is called  $\gamma(\mathbf{u}, \boldsymbol{\alpha})$ , and the integral is approximated with sequential importance sampling. Note that this derivation holds for any form of priors for the speech and noise  $p(\mathbf{x})$  and  $p(\mathbf{n})$ .

### D.3 Postponed factorisation of the integrand

This section presents a factorisation of the integrand  $\gamma(\mathbf{u} | \boldsymbol{\alpha})$ . It should result in factors  $\gamma_i$  so that (repeated from (103))

$$\gamma(\mathbf{u} | \boldsymbol{\alpha}) = \mathcal{N}(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t); \boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x) \mathcal{N}(\mathbf{n}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t); \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n). \quad (169)$$

The two Gaussians on the right-hand side have the same structure. The factorisation here will only explicitly consider the term deriving from the speech prior; the one deriving from the noise prior factorises analogously. A multi-variate Gaussian relates all elements in its input vector through the inverse covariance matrix, the precision matrix  $\boldsymbol{\Lambda}_x$ . The derivation writes these explicitly. The elements of  $\boldsymbol{\Lambda}_x = \boldsymbol{\Sigma}_x^{-1}$  are denoted with  $\lambda_{x,ij}$ .

$$\begin{aligned} & \mathcal{N}(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t); \boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x) \\ &= |2\pi\boldsymbol{\Sigma}_x|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t) - \boldsymbol{\mu}_x)^\top \boldsymbol{\Lambda}_x (\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t) - \boldsymbol{\mu}_x)\right) \\ &= |2\pi\boldsymbol{\Sigma}_x|^{-\frac{1}{2}} \exp\left(-\frac{1}{2} \sum_{i=1}^d \sum_{j=1}^d (x(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) - \mu_{x,i}) \lambda_{x,ij} (x(\mathbf{u}_j, \boldsymbol{\alpha}_j, \mathbf{y}_{t,j}) - \mu_{x,j})\right) \\ &= |2\pi\boldsymbol{\Sigma}_x|^{-\frac{1}{2}} \exp\left(\sum_{i=1}^d \left[ -\frac{1}{2} \lambda_{x,ii} (x(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) - \mu_{x,i})^2 \right. \right. \\ &\quad \left. \left. - (x(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) - \mu_{x,i}) \sum_{j=1}^{i-1} \lambda_{x,ij} (x(\mathbf{u}_j, \boldsymbol{\alpha}_j, \mathbf{y}_{t,j}) - \mu_{x,j}) \right]\right) \\ &= |2\pi\boldsymbol{\Sigma}_x|^{-\frac{1}{2}} \prod_{i=1}^d \exp\left( -\frac{1}{2} \lambda_{x,ii} (x(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) - \mu_{x,i})^2 - (x(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) - \mu_{x,i}) \nu_{x,i} \right), \end{aligned} \quad (170a)$$

where the term containing coordinates of lower dimensions  $\mathbf{u}_{1:i-1}$  is

$$\nu_{x,i} = \sum_{j=1}^{i-1} \lambda_{x,ij} (x(\mathbf{u}_j, \boldsymbol{\alpha}_j, \mathbf{y}_{t,j}) - \mu_{x,j}). \quad (170b)$$

When drawing  $u_i$  for dimension  $i$ , the coordinates of lower dimensions  $\mathbf{u}_{1:i-1}$  are known.

After applying the same factorisation to the noise term, the complete integrand in (169) can be written

$$\begin{aligned}
 \gamma(\mathbf{u}|\boldsymbol{\alpha}) &= \mathcal{N}(\mathbf{x}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t); \boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x) \mathcal{N}(\mathbf{n}(\mathbf{u}, \boldsymbol{\alpha}, \mathbf{y}_t); \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n) \\
 &= |2\pi\boldsymbol{\Sigma}_x|^{-\frac{1}{2}} |2\pi\boldsymbol{\Sigma}_n|^{-\frac{1}{2}} \prod_{i=1}^d \exp\left( \right. \\
 &\quad \left. -\frac{1}{2}\lambda_{x,ii} (\mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) - \boldsymbol{\mu}_{x,i})^2 - (\mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) - \boldsymbol{\mu}_{x,i}) \boldsymbol{\nu}_{x,i} \right. \\
 &\quad \left. -\frac{1}{2}\lambda_{n,ii} (\mathbf{n}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) - \boldsymbol{\mu}_{n,i})^2 - (\mathbf{n}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) - \boldsymbol{\mu}_{n,i}) \boldsymbol{\nu}_{n,i} \right), \tag{171}
 \end{aligned}$$

where  $\boldsymbol{\nu}_{n,i}$  is defined analogously to (170b). The factors are then defined as (it is arbitrary which factor takes the constant determiners)

$$\begin{aligned}
 \gamma_1(\mathbf{u}_1|\boldsymbol{\alpha}_1) &= |2\pi\boldsymbol{\Sigma}_y|^{-\frac{1}{2}} |2\pi\boldsymbol{\Sigma}_x|^{-\frac{1}{2}} \exp\left( -\frac{1}{2}\lambda_{n,11} (\mathbf{n}(\mathbf{u}_1, \boldsymbol{\alpha}_1, \mathbf{y}_{t,1}) - \boldsymbol{\mu}_{n,1})^2 \right. \\
 &\quad \left. -\frac{1}{2}\lambda_{x,11} (\mathbf{x}(\mathbf{u}_1, \boldsymbol{\alpha}_1, \mathbf{y}_{t,1}) - \boldsymbol{\mu}_{x,1})^2 \right); \tag{172a}
 \end{aligned}$$

$$\begin{aligned}
 \gamma_i(\mathbf{u}_i|\mathbf{u}_{1:i-1}, \boldsymbol{\alpha}_{1:i}) &= \exp\left( -\frac{1}{2}\lambda_{x,ii} (\mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) - \boldsymbol{\mu}_{x,i})^2 - (\mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) - \boldsymbol{\mu}_{x,i}) \boldsymbol{\nu}_{x,i} \right. \\
 &\quad \left. -\frac{1}{2}\lambda_{n,ii} (\mathbf{n}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) - \boldsymbol{\mu}_{n,i})^2 - (\mathbf{n}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) - \boldsymbol{\mu}_{n,i}) \boldsymbol{\nu}_{n,i} \right), \tag{172b}
 \end{aligned}$$

To find a proposal distribution for the resulting density, it can be rewritten so that it is easily related to the one-dimensional  $\gamma$  in section 5.2.1, for which good proposal distributions were discussed in section 5.2.1.2. Again, the following rewrites only part of the term related to the speech prior; the noise term works completely analogously. Since for importance sampling it is the shape rather than the height of the density that the proposal distribution needs to match, the following disregards constant factors (note the use of  $\propto$ ), which are additive within the  $\exp(\cdot)$ . A technique sometimes called ‘‘completing the square’’ helps find the shape of the term related to the clean speech in (172). This derivation is similar to the derivation of the parameters of a conditional Gaussian distribution (see e.g. [3]). Taking one factor from (170a),

$$\begin{aligned}
 &\exp\left( -\frac{1}{2}\lambda_{x,ii} (\mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) - \boldsymbol{\mu}_{x,i})^2 - (\mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) - \boldsymbol{\mu}_{x,i}) \boldsymbol{\nu}_{x,i} \right) \\
 &\propto \exp\left( -\frac{1}{2}\lambda_{x,ii} (\mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}))^2 + \lambda_{x,ii}\boldsymbol{\mu}_{x,i}\mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) - \boldsymbol{\nu}_{x,i}\mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) \right) \\
 &= \exp\left( -\frac{1}{2}\lambda_{x,ii} (\mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}))^2 + \lambda_{x,ii} \left( \boldsymbol{\mu}_{x,i} - \frac{\boldsymbol{\nu}_{x,i}}{\lambda_{x,ii}} \right) \mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) \right) \\
 &\propto \exp\left( -\frac{1}{2}\lambda_{x,ii} \left[ \mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}) - \left( \boldsymbol{\mu}_{x,i} - \frac{\boldsymbol{\nu}_{x,i}}{\lambda_{x,ii}} \right) \right]^2 \right) \\
 &\propto \mathcal{N}\left( \mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i}); \boldsymbol{\mu}_{x,i} - \frac{\boldsymbol{\nu}_{x,i}}{\lambda_{x,ii}}, \lambda_{x,ii}^{-1} \right). \tag{173}
 \end{aligned}$$

By rewriting the additive noise term in the same manner, the factors in (172) turn out to be proportional to two Gaussian distributions that are functions of  $\mathbf{x}(\mathbf{u}_i, \boldsymbol{\alpha}_i, \mathbf{y}_{t,i})$

and  $n(\mathbf{u}_i, \alpha_i, \mathbf{y}_{t,i})$ :

$$\begin{aligned} \gamma_i(\mathbf{u}_i | \mathbf{u}_{1:i-1}, \alpha_{1:i-1}) &\propto \mathcal{N}\left(x(\mathbf{u}_i, \alpha_i, \mathbf{y}_{t,i}); \mu_{x,i} - \frac{\gamma_{x,i}}{\lambda_{x,ii}}, \lambda_{x,ii}^{-1}\right) \\ &\cdot \mathcal{N}\left(n(\mathbf{u}_i, \alpha_i, \mathbf{y}_{t,i}); \mu_{n,i} - \frac{\gamma_{n,i}}{\lambda_{n,ii}}, \lambda_{n,ii}^{-1}\right). \end{aligned} \quad (174)$$

This expression has the same shape as the one-dimensional integrand in (89) in section 5.2.1.

#### D.4 Terms of the proposal distribution

Finding the proposal distribution uses  $u(x, y, \alpha)$ , the value for  $u$  that follows from fixing the other variables. Where this is necessary,  $u > 0$ . This is equivalent to  $x \leq n$ , which is the area that this expression for  $n$  is valid for (repeated in (154d)):

$$n = 2 \log \left( -\alpha \exp\left(\frac{1}{2}x\right) + \sqrt{\exp(y_t) + \exp(x)(\alpha^2 - 1)} \right), \quad (175)$$

so that  $u$  can be found with

$$u = n - x = 2 \log \left( -\alpha + \sqrt{\exp(y_t - x) + \alpha^2 - 1} \right). \quad (176)$$

The mirror image of this expression is  $u(n, y, \alpha)$ , which fixes  $n$  rather than  $x$ , and is required only for  $u < 0$ . This expression can be found by rewriting (155b):

$$\exp(n) = \frac{\exp(y_t)}{1 + \exp(-u) + 2\alpha \exp(-\frac{1}{2}u)}; \quad (177a)$$

$$\exp(-u) + 2\alpha \exp(-\frac{1}{2}u) + 1 = \exp(y_t - n); \quad (177b)$$

$$\left(\exp(-\frac{1}{2}u) + \alpha\right)^2 - \alpha^2 + 1 = \exp(y_t - n); \quad (177c)$$

$$\left(\exp(-\frac{1}{2}u) + \alpha\right)^2 = \exp(y_t - n) + \alpha^2 - 1. \quad (177d)$$

From  $u < 0$ , it follows that  $\exp(-\frac{1}{2}u) \geq 1$  and  $\exp(-\frac{1}{2}u) + \alpha \geq 0$ , so that

$$\exp(-\frac{1}{2}u) + \alpha = \sqrt{\exp(y_t - n) + \alpha^2 - 1}; \quad (177e)$$

$$u = -2 \log \left( -\alpha + \sqrt{\exp(y_t - n) + \alpha^2 - 1} \right). \quad (177f)$$

---

## Bibliography

- [1] Alex Acero. *Acoustical and Environmental Robustness in Automatic Speech Recognition*. PhD thesis, Carnegie Mellon University, 1990.
- [2] Alex Acero, Li Deng, Trausti Kristjansson, and Jerry Zhang. HMM adaptation using vector Taylor series for noisy speech recognition. In *Proceedings of ICSLP*, volume 3, pages 229–232, 2000.
- [3] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [4] Li Deng, Jasha Droppo, and Alex Acero. Enhancement of log Mel power spectra of speech using a phase-sensitive model of the acoustic environment and sequential estimation of the corrupting noise. *IEEE Transactions on Speech and Audio Processing*, 12(2):133–143, 2004.
- [5] A. Doucet and A.M. Johansen. A tutorial on particle filtering and smoothing: fifteen years later. Technical report, Department of Statistics, University of British Columbia, December 2008.
- [6] Yariv Ephraim. A minimum mean square error approach for speech enhancement. In *Proceedings of ICASSP*, pages 829–832, 1990.
- [7] F. Flego and M. J. F. Gales. Incremental predictive and adaptive noise compensation. In *Proceedings of ICASSP*, pages 3837–3840, 2009.
- [8] Brendan J. Frey, Li Deng, Alex Acero, and Trausti Kristjansson. ALGONQUIN: Iterating Laplace’s method to remove multiple types of acoustic distortion for robust speech recognition. In *Proceedings of Eurospeech*, pages 901–904, 2001.
- [9] Brendan J. Frey, Trausti T. Kristjansson, Li Deng, and Alex Acero. ALGONQUIN: Learning dynamic noise models from noisy speech for robust speech recognition. In *Proceedings of NIPS*, 2001.
- [10] M. J. F. Gales. Maximum likelihood linear transformations for HMM-based speech recognition. *Computer Speech and Language*, 12(2):75–98, 1998.
- [11] M. J. F. Gales. Semi-tied covariance matrices for hidden Markov models. *IEEE Transactions on Speech and Audio Processing*, 7(3):272–281, 1999.
- [12] M. J. F. Gales and R. C. van Dalen. Predictive linear transforms for noise robust speech recognition. In *Proceedings of ASRU*, pages 59–64, 2007.
- [13] Mark J. F. Gales. *Model-Based Techniques for Noise Robust Speech Recognition*. PhD thesis, Cambridge University, 1995.
- [14] M.J.F. Gales and F. Flego. Discriminative classifiers with adaptive kernels for noise robust speech recognition. *Computer Speech and Language*, 24(4):648–662, 2010.
- [15] O. Kalinli, M.L.Seltzer, and A.Acero. Noise adaptive training using a vector Taylor series approach for noise robust automatic speech recognition. In *Proceedings of ICASSP*, pages 3825–3828, 2009.

- [16] Do Yeong Kim, Chong Kwan Un, and Nam Soo Kim. Speech recognition in noisy environments using first-order vector Taylor series. *Speech Communication*, 24:39–49, 1998.
- [17] G. Kitagawa. Monte Carlo filter and smoother for non-Gaussian non-linear state space models. *Journal of Computational and Graphical Statistics*, 5:1–25, 1996.
- [18] Trausti Kristjansson, Brendan Frey, Li Deng, and Alex Acero. Joint estimation of noise and channel distortion in a generalized EM framework. In *Proceedings of ASRU*, 2001.
- [19] Trausti T. Kristjansson and Brendan J. Frey. Accounting for uncertainty [*sic*] in observations: a new paradigm for robust automatic speech recognition. In *Proceedings of ICASSP*, pages 61–64, 2002.
- [20] Trausti Thor Kristjansson. *Speech Recognition in Adverse Environments: a Probabilistic Approach*. PhD thesis, University of Waterloo, 2002.
- [21] Volker Leutnant and Reinhold Haeb-Umbach. An analytic derivation of a phase-sensitive observation model for noise robust speech recognition. In *Proceedings of Interspeech*, pages 2395–2398, 2009.
- [22] Volker Leutnant and Reinhold Heab-Umbach. An analytic derivation of a phase-sensitive observation model for noise robust speech recognition. Technical report, Universität Paderborn, Faculty of Electrical Engineering and Information Technology, September 2009.
- [23] Jinyu Li, Li Deng, Dong Yu, Yifan Gong, and Alex Acero. High-performance HMM adaptation with joint compensation of additive and convolutive distortions via vector Taylor series. In *Proceedings of ASRU*, pages 65–70, 2007.
- [24] Jinyu Li, Dong Yu, Li Deng, Yifan Gong, and Alex Acero. A unified framework of HMM adaptation with joint compensation of additive and convolutive distortions. *Computer Speech and Language*, 23:389–405, 2009.
- [25] H. Liao and M. J. F. Gales. Uncertainty decoding for noise robust speech recognition. In *Proceedings of Interspeech*, 2005.
- [26] H. Liao and M. J. F. Gales. Adaptive training with joint uncertainty decoding for robust recognition of noisy data. In *Proceedings of ICASSP*, volume IV, pages 389–392, 2007.
- [27] Hank Liao. *Uncertainty Decoding for Noise Robust Speech Recognition*. PhD thesis, Cambridge University, 2007.
- [28] Pedro J. Moreno. *Speech Recognition in Noisy Environments*. PhD thesis, Carnegie Mellon University, 1996.
- [29] Tor André Myrvoll and Satoshi Nakamura. Minimum mean square error filtering of noisy cepstral coefficients with applications to ASR. In *ICASSP*, pages 977–980, 2004.
- [30] K. B. Petersen and M. S. Pedersen. *The matrix cookbook*, Nov 2008.

- 
- [31] K. C. Sim and M. J. F. Gales. Basis superposition precision matrix modelling for large vocabulary continuous speech recognition. In *Proceedings of ICASSP*, volume 1, pages 801–804, 2004.
- [32] Veronique Stouten. *Robust automatic speech recognition in time-varying environments*. PhD thesis, Katholieke Universiteit Leuven, 2006.
- [33] R. C. van Dalen, F. Flego, and M. J. F. Gales. Transforming features to compensate speech recogniser models for noise. In *Proceedings of Interspeech*, pages 2499–2502, 2009.
- [34] R. C. van Dalen and M. J. F. Gales. Extended vts for noise-robust speech recognition. Technical Report CUED/F-INFENG/TR.636, Cambridge University Engineering Department, September 2009.
- [35] Rogier C. van Dalen and Mark J. F. Gales. Extended vts for noise-robust speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 2011.