Diploma work

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Kalman Filtering and Speech Enhancement

Bc. Jan Kybic

23 January 1998

This report can be obtained from http://cmp.felk.cvut.cz/~kybic/dipl

Abstract

The enhancement of noisy speech is a challenging research field with numerous applications. In the presented work we focus on the case of speech signal corrupted by slowly varying, non-white, additive noise, when only a corrupted signal is available. First, the survey of speech enhancement, identification and filtering methods is presented. Second, a new speech enhancement algorithm based on *Kalman smoothing*, spectral minima tracking, state-space identification and all-pole modelling is proposed. Its main novelty is the use of bidirectional fast two-pass smoothing and also the combination of minima tracking and Kalman filtering techniques. The intended application of this algorithm is the suppression of noise in a running car environment for hands-free mobile telephony. Its performance is compared to traditional methods: it is found that it can give better results at the expense of execution speed. Its usability in the speech recognition setting and the effect of changes of various parameters is demonstrated.

A conventional serial as well as a parallel version (using *Parallel Virtual Machine*) of the algorithm discussed were developed. Sources are available in *C* and *MATLAB*.

Keywords: speech enhancement, noise reduction, Kalman filtering, smoothing, spectral subtraction, system identification, parameter estimation, all-pole modelling, state-space identification, total least squares, structured total least squares, Hankel total least squares, singular value decomposition, spectral minima tracking, speech recognition, hands-free mobile telephony, Matlab, C, PVM, parallel computing.

AMS classification: 62M10, 62M20, 60G35, 68Q22, 93B30.

Abstrakt

Odstraňování šumu z řečového signalu je významným interdisciplinárním problémem s mnoha praktickými aplikacemi. Tato práce se soustřeďuje na případ řeči znehodnocené pomalu se měnícím, barevným aditivním šumem, je-li k dispozici pouze zašuměný signál. V úvodu je předložen přehled existujících metod zvýrazňovani řeči, identifikace lineárních dynamických systémů a stochastické filtrace. Následuje popis nového algoritmu zvýrazňování řeči založeného na existujících metodách dvouprůchodového Kalmanova vyhlazováni, odhadu šumu na principu sledováni minim, stavové identifikace a autoregresivního modelování. Jak použití Kalmanova vyhlazováni, tak i jeho kombinace s metodou sledování minim, je v této oblasti novinkou. Algoritmus byl navržen pro použití jako součást mobilního telefonu v automobilu. Závěrem je uveden popis provedených testů prokazujících, že nový algoritmus dáva za jistých okolností lepší výsledky než alternativní metody, i když na úkor delší výpočetní doby. Součástí práce je implementace téměř všech zmiňovaných metod a klasická i paralelní verze nového algoritmu.

Prohlášení

Tímto dávám svou diplomovou práci *Kalman Filtering and Speech Enhancement* k dispozici elektrotechnické fakultě ČVUT Praha a souhlasím s tím, aby byla použita podle potřeby.

Prohlašuji rovněž, že jsem tuto práci vypracoval samostatně a uvedl všechny použité prameny.

Lausanne, datum

Jan Kybic

English translation of the declaration above:

I hereby put my diploma thesis entitled *Kalman Filtering and Speech Enhancement* at the disposal of the Faculty of Electrical Engineering of the Czech Technical University in Prague and I grant it the right to use this work as needed.

I also declare that my diploma thesis is a result of my solely individual effort and that I have quoted all references used.

Contents

Glossary

(1)	reference to an equation number 1
[1]	reference to a book or article, see *References* at page 69
⟨x⟩	syntactical unit
[x]	optional syntactical unit

x_{ij}	component of matrix $\boldsymbol{X}$ in i^{th} row and j^{th} column
$\hat{x}$	estimate of x
$\tilde{x}$	$x - \hat{x}$
$\breve{x}$	windowed x
$\overline{x}$	averaged, smoothed x
x'	quantity related to x
x^*	true or ideal value
$x^{\triangleright}$	rectified value
$x^{\spadesuit}$	tunable parameters
$\boldsymbol{x}$	vector
$\boldsymbol{X}$	matrix
$\boldsymbol{X}_{m\times n}$	matrix with given size
$\boldsymbol{X}^{\diamond}_{m\times n}$	matrix cut to given size
$\boldsymbol{X}^{\uparrow}$	$\boldsymbol{X}$ without its top row
$\boldsymbol{X}^{\downarrow}$	$\boldsymbol{X}$ without its bottom row
$\boldsymbol{X}^{1/2}$	lower-triangular Cholesky factor
$\boldsymbol{X}^{T/2}$	upper-triangular Cholesky factor
$\Vert\boldsymbol{X}\Vert_F$	Frobenius norm, $\sum\sum x_{ij}^2$
$\{x(t)\}_{t=a}^{b}$	sequence $x(a), x(a+1), \dots, x(b)$
$\{\alpha\,\beta\}$	sequence, concatenation of α and β, set
$[x_1, \dots, x_n]$	elements composing a vector $\boldsymbol{x}$
$\propto$	is linearly proportional to
$\approx$	approximately equals
$\widehat{=}$	corresponds to
$\stackrel{!}{=}$	is to be (e.g. minimal)
$a \sim b$	ranging from a to b

$\boldsymbol{a}$	autoregressive model parameters
b	autoregressive model parameter
e	Euler constant $2.782818\ldots$
$\boldsymbol{e}$	prediction error
g	gain
j	imaginary unit
k	time lag, frequency bin number, frame number
m	dimensionality of the output
n	system order
r	dimensionality of the input
s	window overlap factor
t	(discrete) time
u, $\boldsymbol{u}$	system input
v, $\boldsymbol{v}$	process noise
w, $\boldsymbol{w}$	measurement noise, Fourier factor, weight factor
$\boldsymbol{x}$	system state

y, $\boldsymbol{y}$	system output
y_{N}	noise signal
y_{S}	speech signal
y_{X}	sum of speech and noise, measured signal
$\boldsymbol{A}$	system transition matrix
$\boldsymbol{B}$	system input matrix
$\mathcal{C}$	autocorrelation operator
$C_x(k)$	autocorrelation of x at lag k
$\boldsymbol{C}$	system output matrix, autocorrelation matrix
$\mathcal{D}^t$	data available at time t
$\boldsymbol{D}$	system feedforward matrix
F	complex Fourier spectrum, Newton method goal function
$\mathcal{F}$	Fourier transform operator
$\boldsymbol{F}$	backward gain in Kalman smoothing, left Vandermonde factor
$\boldsymbol{G}$	diagonal Vandermonde factor
$\boldsymbol{H}$	right Vandermonde factor
I	unitary matrix
J	minimisation criterion
$\boldsymbol{K}$	factor of Kalman gain
L	optimality criterion
L_F	Frobenius norm, $\sum\sum x_{ij}^2$
L_2	2-norm $\max_i \sigma_i$
L_∞	Chebyshev ∞-norm, $\max_{ij} \vert x_{ij}\vert$
$\boldsymbol{L}$	Kalman gain
P	power spectrum
M	buffer length for minimum tracking
$\mathcal{N}$	set of natural numbers
$\mathcal{O}(x)$	asymptoticaly proportional to x
$\boldsymbol{P}$	covariance of the state estimate
$\boldsymbol{Q}$	covariance of the process noise
$\mathcal{R}$	set of real numbers
$\boldsymbol{R}$	covariance of the output noise
S	amplitude spectrum
T	number of samples, window size
$\mathcal{Z}$	set of integers, $\mathcal{Z}$-transform
$\boldsymbol{Z}$	time step advance signal matrix

α	smoothing factor
β	secondary smoothing factor
γ	bias factor for noise estimation
φ	phase spectrum
δ	oversubtraction factor
$\delta(k)$	Kronecker delta (discrete unit impulse)
$\boldsymbol{\lambda}$	auxiliary state in Kalman smoothing, Lagrange coefficients
$\boldsymbol{\nu}$	innovation sequence
ξ	equation error
π	$3.141592\ldots$
σ	singular value
ω	frequency
Ω	penalty
Δx	change, improvement of x
Υ	rearrange into vector
Γ	Euler function
$\boldsymbol{\Gamma}$	prediction error sequence covariance factor, TLS factor

cov	covariance, $\mathcal{E}\left\{\tilde{x}\,\tilde{x}^T\right\}$
$\mathcal{E}\{\cdot\}$	expected value
AR	autoregressive (model)
AROE	autoregressive output-error (model)
ARMAX	autoregressive moving-average (model)
ASS	amplitude spectral subtraction
CTLS	constrained total least squares
DARE	discrete algebraic Riccati equation
dBA	logarithmic unit of acoustic intensity
DFT	discrete Fourier transform
FIFO	first in, first out
FFT	fast Fourier transform
HMM	hidden Markov model
HTLS	Hankel TLS
IDFT	inverse discrete Fourier transform
IFFT	inverse fast discrete Fourier transform
KF	Kalman filter
LMS	(minimum) linear mean square (estimate)
LS	least squares
MEM	maximum entropy
ML	maximum likelihood (estimate)
MS	(minimum) mean square (estimate)
MV	minimum variance
OE	output error (model)
PEM	prediction error method
PVM	Parallel Virtual Machine library
PSS	power spectral subtraction
QP	quadratic programming
RASTA	relative spectral (speech enhancement method)
SISO	single-input single-output model
SNR	signal to noise ratio
SVD	singular value decomposition
SR	square root (Kalman filter)
STLN	structured total least norm
STLS	structured total least squares
TLS	total least squares
TSTLS	Toeplitz structured total least squares
VD	Vandermonde decomposition
VAD	voice activity detector
Hankel	(matrix) with equal elements on anti-diagonals
monaural	single microphone, single input
Toeplitz	(matrix) with equal elements on diagonals

Preface

If I have seen farther than others, it is because I was standing on the shoulders of giants.

— Isaac Newton

The purpose of this report is to describe my diploma project on *Kalman filtering* and its application to *speech enhancement*. It provides an overview over the field of speech enhancement so that my new method could be placed into the proper framework. Given the space available, it was not possible to make this report completely self-contained. However, I tried to include all the information necessary to enable an interested reader to start experimenting and enhancing my algorithms straight away.

Most of the ideas presented here are, of course, not original. I have typically just reformulated generally known ideas and applied them to the particular problem. An attempt was made to acknowledge the original sources and I apologize for any omissions. I am deeply indebted to all the numerous researchers that made this work possible.

Acknowledgements

I wish to express my thanks to *Ing. Petr Pollák, CSc.* who kindly supervised this work. As a significant part of our project was carried out while staying at *Ecole Polytechnique Fédérale de Lausanne*, my deepest gratitude goes to everybody who made this stay possible, namely *Prof. Martin Hasler* and all his collaborators for creating a truly enjoyable and productive environment. The powerful computers, *Internet* connection and a well equipped library at hand made my work much easier.

Finally, many thanks to *Sabine Van Huffel* and *Philippe Lemmerling* from *Katholieke Universiteit Leuven* for letting me share their reports and peruse their programs.

Contact

If you find this report interesting, please get in touch with us:

			Ing. Petr Pollák, CSc.
	Bc. Jan Kybic		FEL ČVUT
	Součkova 985/11		Technická 2
	Praha 6		Praha 6
	Czech Republic		Czech Republic
e-mail:	kybic@cmp.felk.cvut.cz	*e-mail:*	pollak@feld.cvut.cz

The source text of this report and the source code for implementing various algorithms discussed are available from http://cmp.felk.cvut.cz/˜kybic/dipl as well as on the diskettes accompanying the original of this report.

Legal Matters

Disclaimer

I make no warranties, expressed or implied, that this report and/or the accompanying programs are free of errors.

License

1. Introduction

The enhancement of speech corrupted by noise is an important problem with numerous applications ranging from suppression of environmental noise for communication systems and hearing aids, enhancing the quality of old records, to preprocessing for speech recognition systems.

Depending on the application, the properties of the noise as well as the nature of the corruption vary. However, for the purpose of this work, we concentrate on the case of *slowly varying, coloured additive* noise. The means of improvements in the acquisition phase (e.g., using better or multiple microphones) are not considered. Instead, the speech enhancement algorithm itself is concentrated upon. The availability of the noisy speech signal in digital form is assumed.

The intended application of the algorithm developed is a *hands-free* car telephony system. As the use of conventional handsets can easily distract driver's attention, it is advantageous from the safety point of view (as well as more comfortable) to use a microphone and speakers attached to a suitable point in the car. However, such a microphone is usually considerably far from the speaker and picks up a significant amount of an environmental noise. A noise enhancer is therefore needed to improve the speech quality.

To further reduce the distraction associated with making phone calls while the car is in motion, a fully *voice-driven* telephone system is envisaged, using a *speech recognition* system. Traditionally, though, speech recognition systems do not perform well in the presence of a background noise and it is thus advantageous to employ a noise enhancement algorithm at a *preprocessing* stage. Subsequent reflections can be found in [63],[20],[17].

A brief overview of the field of speech enhancement, Kalman filtering and autoregressive model identification is given. The author implemented and tested all the described methods for which enough information was available; the implementation is available as a part of this work. Some analysis and comments on these methods and their limitations are included, based on theoretical studies, experiments and personal experience. However, time and space constraints sometimes prevented the author to treat the subject with a rigorousness it deserved. A *new* speech enhancement algorithm combining the existing spectral estimation, noise estimation, spectral subtraction, system identification and Kalman filtering algorithms is presented. The implementation of this algorithm as well as the problems encountered is described. Finally, an evaluation of the performance of this algorithm in comparison with traditional spectral subtraction based algorithms, its usability in the speech recognition setting and the effect of various parameter changes are given as well as a summary of its advantages, deficiencies and possibilities for future enhancements.

The new algorithm is unique in combining fast two-pass bidirectional Kalman smoothing with spectral minima tracking for speech enhancement purposes. This — together with the availability of a working implementation, detailed description, and a literature survey — should make this work a good basis for further improvements; some suggestions are included in the text.

2. State of the Problem

In this chapter, the problem is stated more precisely and some existing solutions are outlined. More information will be given in the following chapters on topics relevant to the new algorithm.

2.1. Problem Specification

The situation is pictured in *figure 1*. The task is to design a filter (denoted by $\boxed{?}$) to produce an estimate $\hat{y}_S$ of a speech signal y_S given the sum y_X of speech and noise signals. ($y_X = y_S + y_N$). The ultimate goal is to achieve the highest possible *intelligibility*. This is, however, hard to describe mathematically. Instead, an alternative *measure of quality* of the estimate will be used — the mean square value of the estimation error e .

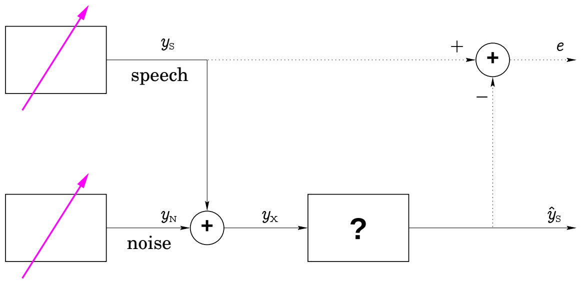


Figure 1.

This model neglects the transfer characteristics of the medium, or rather views it as a part of the two systems generating the speech and noise signals.

2.2. Design Requirements

The filter to be built, in order to be usable in the given situation, must be:

- **monaural** — it has just one input to get all the information needed.
- **fast** — real time performance is required for many applications, but the speed is important even for off-line processing.
- **robust** — the algorithm must be able to run unsupervised for extended periods of time, no matter what the input data are. Should it ever fail, it must be able to fail gracefully (for example by switching to an identity transformation) and restart as soon as possible.
- **adaptable** — it must be able to adapt automatically to different speakers and different kinds of noise.

2.3. Speech Characteristics

A speech signal is composed of *phonemes*. Depending on the language, there are usually a few tenths of phonemes. The signal can be regarded as stationary for some phonemes, notably vowels. Other phonemes, like ‘d’, must be further decomposed into *sub-phonemes* and only those can be then considered as stationary. Generally, a speech signal is said to be *quasi-stationary*, i.e., it can be divided into (almost) stationary segments. The longest segments can be up to 300 ms long, the shortest last only a few milliseconds.

Some phonemes, called *voiced*, are highly periodic, others, called *unvoiced*, are rather stochastic.

For most phonemes, the energy is concentrated between 300∼3000 Hz, depending on the speaker. The high frequency part is more pronounced for unvoiced phonemes. The peaks in the spectrum are called *formants*. Vowels have mostly two major formants.

2.3.1. Speech Modelling

It appears (see [62] and [53]) that an *autoregressive model* (further referred to as an AR model) is particularly suitable for modelling a speech signal. The motivation comes from a simplified picture of a vocal tract as a lossless tube built of adjoining cylinders of different diameters.

The AR model should be ideally driven by *pulse train* for voiced phonemes and by *white noise* for unvoiced ones. However, our experiments show (as well as [38],[19],[56],[47]) that a stochastically excited (i.e., white noise driven) model AR can generate also the voiced parts of the speech reasonably well by moving the poles close towards the unit circle (in the $\mathcal{Z}$ plane).

2.4. Noise Characteristics

We call it noise because we know nothing about it.
— Bishnu Atal, [6]

The noise is always a *stochastic signal* and can be often considered as long time *stationary*. For our application, the noise in a car can be regarded as stationary for time periods of 0.5∼2 s [22]. Its main contributions usually come from the engine (rather periodic noise on low frequencies), the tyres (less periodic, higher frequencies), the fan (almost stochastic, wide-band middle and high frequency noise) and the aerodynamic noise (random, wide-band high frequency noise). Their exact frequencies and energies vary depending on the current driving mode, the car and many other factors. Generally, the total noise spectrum ranges from about 50∼10 kHz. The noise intensities change between 50∼90 dBA.

Due to physical constraints we can safely assume that speech and noise are *independent* and *uncorrelated*. The physiological phenomena of speech parameter changes induced by noise (*Lombard effect*) are too long-term to be influential.

2.4.1. Noise Modelling

Given the stochastic nature and the variety of noise, it is difficult to find a good model for it. Usually, white noise driven AR model is used for its simplicity. Sometimes, the noise is modelled simply as Gaussian white noise.

2.5. Spectral Subtraction

Many speech enhancement algorithm are based on the scenario illustrated in *figure 2* and nicknamed *spectral subtracting*. The input signal y_X is first divided into equal length frames (usually 256 samples) with $^1/_2$ or $^3/_4$ overlap. A weighting function is applied, usually Hamming (or Hanning) window [30]. Frequency spectra of each frame (P_X) and of the noise in each frame (P_N) are estimated. A speech spectrum estimate (P_S) is calculated using a *subtraction rule* and converted into the time domain. Finally, the frames are assembled to get the output $\hat{y}_S$.

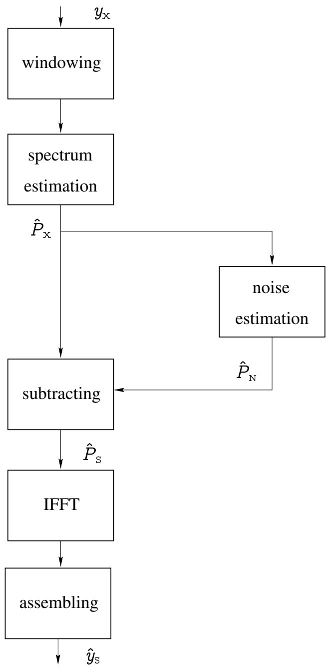


Figure 2.

2.5.1. Voice Activity Detector

Voice activity detector (abbreviated VAD) is an important part of many speech enhancement algorithms. Its purpose is to determine whether a given frame contains the speech or not. Many detectors are available based on different principles, e.g., detecting sudden changes in energy, spectral or cepstral distances. However, none of them seems to work reliably under low SNR conditions [49]. This seriously hinders the performance of all algorithms based on it. For an example of a modern and, according to authors, relatively robust implementation, see [71].

2.5.2. Spectrum Estimation

Most speech enhancement algorithms operate in the frequency domain on segments of data. This is called *batch* processing as opposed to *iterative* processing. A spectral estimation algorithm is used to transform the data from the time domain into the frequency domain. In most cases, the FFT (Fast Fourier Transform) is used preceded by windowing, combined with periodogram averaging [25],[30],[40],[15],[62]. Parametric spectral estimation methods based on autoregressive models,

equivalent to the *maximum entropy* (MEM) spectrum estimate [25], are suggested [38],[23],[62]. These methods can considerably reduce the variance of the estimate. Unfortunately, as the sum of speech and noise is no longer an AR process, they often introduce a significant systematic modelling error. Other methods such as the *minimum variance* (MV) spectrum estimate [25] can also be considered. The MV estimate is essentially an average of MEM estimates and thus also imposes a model on the data. Moreover, it is usually very conservative. See section 4.2.4 (p. 31) for details.

2.5.3. Noise Spectrum Estimation

All algorithms working in the frequency domain need to estimate the noise spectrum in order to perform the filtering. It is possible to average the noise spectrum over speech free segments as indicated by the VAD [49]. A more elaborate solution based on minima tracking was proposed in [40] (see section 6.4.2 (p. 45) for details) and a simplified version appeared in [15].

Attempts to use feature tracking algorithms have so far not succeeded, at least according to the author's knowledge. For an example, see [61],[12].

2.5.4. Speech Spectrum Estimation

Once we have the estimates of the input signal power spectrum $\hat{P}_X$ and the noise power spectrum $\hat{P}_N$ we can proceed to estimate the speech power spectrum $\hat{P}_S$. We shall henceforth use S to denote the amplitude spectrum and φ the phase spectrum, i.e., $P(\omega) = S(\omega)^2$, $S(\omega) \in \mathcal{R}_0^+$. As for the true, mutually independent spectra it holds $P_X(\omega) = P_S(\omega) + P_N(\omega)$, the straightforward approach called *power spectral subtraction* [23] gives

$$\hat{P}_S(\omega) = \hat{P}_X(\omega) - \hat{P}_N(\omega) \tag{1}$$

As for the phases, we usually leave them unchanged:

$$\hat{\varphi}_S(\omega) = \hat{\varphi}_X(\omega) \tag{2}$$

Another approach, called *amplitude spectral subtraction* or *magnitude subtraction*, [7] can be derived from the equal phase assumption (2):

$$\hat{S}_S(\omega) = \hat{S}_X(\omega) - \hat{S}_N(\omega) \tag{3}$$

Both methods can be unified in the *generalized spectral subtraction* method [49]:

$$\hat{S}_S(\omega) = \left| \hat{S}_X(\omega)^a - b\,\hat{S}_N(\omega)^a \right|^{1/a} \tag{4}$$

Alternatively, we can reformulate the spectral subtraction methods using a *gain factor* [23]:

$$\hat{S}_S(\omega) = g(\omega)\hat{S}_X(\omega) \quad \hat{P}_S(\omega) = g(\omega)^2 \hat{P}_X(\omega) \tag{5}$$

where for the power *resp.* amplitude spectral subtraction:

$$g_{\mathrm{PSS}}(\omega) = \sqrt{1 - \frac{\hat{P}_{\mathrm{N}}(\omega)}{\hat{P}_{\mathrm{X}}(\omega)}} \tag{6}$$

$$\textit{resp.} \quad g_{\mathrm{ASS}}(\omega) = 1 - \sqrt{\frac{\hat{P}_{\mathrm{N}}(\omega)}{\hat{P}_{\mathrm{X}}(\omega)}} \tag{7}$$

An often considered modification results in *modified power spectral subtraction* method [5],[40],[23]:

$$g_{\mathrm{MPSS}}(\omega) = \sqrt{1 - \delta(\omega)\frac{\hat{P}_{\mathrm{N}}(\omega)}{\hat{P}_{\mathrm{X}}(\omega)}} \tag{8}$$

where δ is called an *oversubtraction factor*. Various heuristics have been proposed to calculate it [40],[23]. See also section 6.4.3 (p. 46) for details. Yet different gain factors have been suggested in [16] and [15].

2.5.5. Rectification

Due to their physical meanings, $P_{\mathrm{S}}, S_{\mathrm{S}}, g \in \mathcal{R}_0^+$. However, the approximate methods from section 2.5.4 (p. 15) do not guarantee this. As a remedy, a *half wave rectification* can be used

$$\hat{S}_{\mathrm{S}}(\omega)^{\triangleright} = \max\left\{0, \hat{S}_{\mathrm{S}}(\omega)\right\} \tag{9}$$

as well as a *full wave rectification* [49]:

$$\hat{S}_{\mathrm{S}}(\omega)^{\triangleright} = \left|\hat{S}_{\mathrm{S}}(\omega)\right| \tag{10}$$

The rectification can be applied to any of P, S, g. Some refinements are suggested in [5] and [40].

2.5.6. Stochastic Approach to Speech Estimation

The speech spectrum estimation methods described so far have been only approximative, because no efficient exact methods are known. It is instructive to apply a stochastic approach to the problem in order to appreciate its difficulty. The reasoning sketched here is based on the periodogram smoothing spectrum estimation method (described in section 4.2.4 (p. 31)) but similar results hold generally. Many simplifying assumptions are used, such as the independence of speech and noise spectrum estimates and the mutual independence of their frequency components.

Assuming that the input time series $y_{\mathrm{S}}, y_{\mathrm{N}}$ are normally distributed random variables, it follows from the linearity of DFT that $\mathrm{Re}\left(F_{\mathrm{S/N/X}}\right)$ and $\mathrm{Im}\left(F_{\mathrm{S/N/X}}\right)$ are also normally distributed. Consequently, the components of the quantities $2m_{\mathrm{N}}\hat{P}_{\mathrm{N}}/P_{\mathrm{N}}$ resp. $2m_{\mathrm{X}}\hat{P}_{\mathrm{X}}/P_{\mathrm{X}}$ are χ^2 distributed with $2m_{\mathrm{N}}$ resp. $2m_{\mathrm{X}}$ degrees of freedom, where m is the effective number of periodograms used to obtain the estimate [62]. The task of estimating components of P_{S} can be now transformed to the task of estimating deterministic unknown P_{S} given random measurements $\hat{P}_{\mathrm{X}}, \hat{P}_{\mathrm{N}}$ of unknown deterministic random variables $P_{\mathrm{X}}, P_{\mathrm{N}}$. The probability density function (p.d.f) of $\hat{P}_{\mathrm{X}}$ is

$$p(\hat{P}_{\mathrm{X}}(\omega) = x | P_{\mathrm{X}}(\omega)) = \begin{cases} \dfrac{P_{\mathrm{X}} x^{m_{\mathrm{X}}-1} \mathrm{e}^{-x/2}}{2m_{\mathrm{X}} 2^{m_{\mathrm{X}}} \Gamma(m_{\mathrm{X}})}, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases} \tag{11}$$

and similarly for $\hat{P}_{\mathrm{N}}$. The procedure would now be to use the *Bayes formula* to calculate $p(P_{\mathrm{X}}(\omega) = x | \hat{P}_{\mathrm{X}}(\omega))$, where the *a priori* probability $p(P_{\mathrm{X}}(\omega) = x)$ must be guessed. Next, the probability density

$$p(P_{\mathrm{S}}(\omega) = x | \hat{P}_{\mathrm{X}}(\omega), \hat{P}_{\mathrm{N}}(\omega)) = \int_0^{\infty} p(P_{\mathrm{N}}(\omega) = y)\, p(P_{\mathrm{X}}(\omega) = x + y)\, \mathrm{d}y \tag{12}$$

can be evaluated and a suitable estimate computed, e.g., mean square:

$$\hat{P}_{\mathrm{S}}(\omega | \hat{P}_{\mathrm{X}}(\omega), \hat{P}_{\mathrm{N}}(\omega)) = \int_0^{\infty} x\, p(P_{\mathrm{S}}(\omega) = x | \hat{P}_{\mathrm{X}}(\omega), \hat{P}_{\mathrm{N}}(\omega))\, \mathrm{d}x \tag{13}$$

The main problem of this approach is that the integration involved cannot be performed analytically and a numerical integration is prohibitively slow. A feasible solution would be to approximate the input p.d.f. or the resulting function $\hat{P}_{\mathrm{S}}(\hat{P}_{\mathrm{X}}, \hat{P}_{\mathrm{N}})$. Alternatively, instead of aiming at a true statistically optimal estimate, a simplified criterion could be used, as in [16],[15].

2.5.7. Wiener Filter

If we assume that the speech and noise signals are independent and stationary (even though it is only approximately true), we can use the *non-causal Wiener filter* ([59],[28],[25]) to find a gain factor g_{W}. This would yield an optimal estimate of $\hat{S}_{\mathrm{S}}$ in the mean square sense when used in (5) if the assumptions held and if we had the true values of P_{X} and P_{N}.

$$g_{\mathrm{W}}(\omega) = \frac{P_{\mathrm{SX}}(\omega)}{P_{\mathrm{X}}(\omega)} = \frac{P_{\mathrm{S}}(\omega)}{P_{\mathrm{X}}(\omega)} = \frac{P_{\mathrm{X}}(\omega) - P_{\mathrm{N}}(\omega)}{P_{\mathrm{X}}(\omega)} \tag{14}$$

The equality $P_{\mathrm{SX}}(\omega) = P_{\mathrm{S}}(\omega)$ comes from the independence of y_{S} and y_{N} and from $y_{\mathrm{X}} = y_{\mathrm{S}} + y_{\mathrm{N}}$. Using the *certainty equivalence principle* ([59]) (i.e., presuming blindly that $y = f(x) \Rightarrow \hat{y} = f(\hat{x})$) we get:

$$\hat{g}_{\mathrm{W}}(\omega) = \frac{\hat{P}_{\mathrm{X}}(\omega) - \hat{P}_{\mathrm{N}}(\omega)}{\hat{P}_{\mathrm{X}}(\omega)} \tag{15}$$

In practice, we want to assure the non-negativity of $\hat{g}_{\mathrm{W}}$ e.g., by means of half-wave rectification.

Finally, we can use (5) to get $\hat{P}_{\mathrm{S}}$:

$$\hat{P}_{\mathrm{S}}(\omega) = \hat{g}_{\mathrm{W}}(\omega)^{\triangleright 2} \hat{P}_{\mathrm{X}}(\omega) = \frac{\max\left\{0, \hat{P}_{\mathrm{X}}(\omega) - \hat{P}_{\mathrm{N}}(\omega)\right\}^2}{\hat{P}_{\mathrm{X}}(\omega)} \tag{16}$$

2.5.8. Optimal Filter Limitations

It is instructive to observe that even under *optimal conditions*, i.e., if the signals were in fact stationary and independent and if we knew exactly their spectra, the Wiener estimate $\hat{P}_S(\omega)$ as given by (14,5,16) would be

$$\hat{P}_S(\omega) = P_S(\omega)\frac{P_S(\omega)}{P_X(\omega)} \tag{17}$$

which is *not* equal to $P_S(\omega)$ unless there is no noise. Heuristically speaking, the information lost in the mixing process *cannot* be retrieved and in absence of this information, the best estimate in the mean square sense we can get is *conservative* and *under-biased*.

Because Wiener estimate is optimal, *no other* estimation method can yield better results under the same conditions. Clearly, if the conditions are suboptimal, i.e., non-stationary, correlated or finite-length signals, the results can be expected to be even worse. Consequently, *no estimation method* can perform better in terms of mean square error (or equivalently SNR improvement) than *Wiener filter* as given by (16) or (17), unless additional information are available.

The result of the preceding paragraph permits us to calculate for example the maximum achievable SNR improvement for spectral subtraction based methods:

$$\triangle\mathrm{SNR}_{\max} = 10\log_{10}\frac{P_N}{P_e} = 10\log_{10}\frac{P_N}{P_S - \hat{P}_S} \tag{18}$$

where P_e is the power of the error signal $e = y_S - \hat{y}_S$, and the equality $P_e = P_S - \hat{P}_S$ holds because of the *orthogonality principle* [59]. Using (17) and integrating over ω gives after additional manipulations

$$\triangle\mathrm{SNR}_{\max} = 10\log_{10}\left(P_N \Big/ \int_0^\infty \frac{P_S(\omega)P_N(\omega)}{P_S(\omega)+P_N(\omega)}\,\mathrm{d}\omega\right) \tag{19}$$

Nevertheless, the filter (17) is hardly applicable in real situations because the Wiener filter assumptions do not hold and the quantities involved are unknown. Therefore, it pays out to find methods that would work even under these conditions. Moreover, the error minimisation is not always the ultimate goal, it is the speech quality improvement. Finally, remark that (19) does not assert anything about speech enhancement methods using an *a priori* information about the speech signal. And many methods *do* use such an information.

2.5.9. Iterative Improvement

> *Converging iterative methods do exist; this just isn't one of them.*
>
> — *Numerical Recipes, [50]*

An iterative improvement speech enhancement method for speech signal corrupted by white noise based on Wiener filtering and autoregressive modelling was proposed in [38] and quoted in [6]. The idea is as follows: Estimate initial AR parameters $\hat{\boldsymbol{a}}^0$ by the autocorrelation method (see section 4.4.1 (p. 34)), use $\hat{\boldsymbol{a}}^0$ and Wiener filtering to get the speech signal estimate $\hat{y}_S^1$, estimate AR parameters $\hat{\boldsymbol{a}}^1$ of $\hat{y}_S^1$, use $\hat{\boldsymbol{a}}^1$ and Wiener filtering to get the second speech signal estimate $\hat{y}_S^2$ etc., *ad infimum*.

The catch here is that by estimating the AR parameters of a filtered signal, we do *not* get a good estimate of the AR parameters of the original signal as $\hat{y}_S$ is necessarily a *biased* and *conservative* estimate of y_S. The algorithm as described converges to zero but particular implementations may behave differently or even diverge depending on the autocorrelation estimation procedure used. According to [21], an equivalent iterative method based on Kalman filtering "does *not* guarantee an improved estimate in terms of audible quality". However, the first few iterations of the algorithm *may* give better results than the autocorrelation method itself. The original article [38] mentions two iterations as optimal.

Despite of its deficiencies, the procedure described above can be used with good results if the SNR is high and if the algorithm speed is important in the particular application. Otherwise, one is better off using some output-error model parameter estimation procedure such as *prediction error methods* (PEM) [54] which are usually slower.

The author described in [34] another iterative approach to AR output-error model parameter estimation based on repetitive autocorrelation. However, it was discovered later that, for similar reasons as the above method, the model poles tend to converge to the unit circle (in $\mathcal{Z}$ plane) and so it is recommended to use just a small number of iterations when applied to the speech signal.

2.5.10. Extended Spectral Subtraction

Extended spectral subtraction method combining Wiener filtering and conventional spectral subtraction in a feedback algorithm is described in [58]. It is based on the assumption that fast changes in the signal spectrum represent speech while slower ones correspond to noise changes.

2.6. Kalman Filtering

Several attempts to use *Kalman filtering* (see next chapter) in speech enhancement algorithms have been made so far. While both Kalman and Wiener filtering can be made to optimally solve the *same* problem (linear minimum-mean-square-error estimate of a wide-sense stationary signal from noisy observations) and are hence *equivalent*, for other problems the Kalman filtering offers the following advantages:

- It is designed to work with *finite* data sets.
- It makes use of *models* of the speech and noise production processes.
- It can be made to work with *non-stationary* signals.

However, it is not yet clear what the best way of applying Kalman filter to speech enhancement is and most of the algorithms published so far have deficiencies calling for further improvements.

2.6.1. Published Algorithms

It was shown in [47] that Kalman filtering is a feasible approach and can outperform Wiener filtering. Noise is considered to be white. Speech AR model parameters are assumed to be *known* which is definitely not the case in real applications.

The approach described in [56] divides the signal into *subbands* using a quadrature-mirror filter bank and then applies Kalman filter in each subband. Both speech and noise are modelled as AR processes — noise does not need to be white. The speech model parameters are estimated either from clean speech by the autocorrelation method or by using an iterative method equivalent to the one described in section 2.5.9 (p. 18), sharing its deficiencies — reportedly only 2 or 3 iterations could be carried out and the input SNR had to be fixed at 5 dB. The report gives no indication as to how the noise parameters were obtained.

An iterative noise canceller based on [42],[43] is presented in [19]. The noise is assumed to be white. The *innovation sequence* is tested for whiteness, indicating optimal functioning of the filter. If it is not white, it is used to improve the Kalman gain towards the optimum. Unfortunately, the iterative improvement procedure is often unstable at low SNR conditions when the autocorrelation sequence estimate, needed for the computation, deviates significantly from the true value. Furthermore, the Kalman gain is kept constant during each frame which is suboptimal. See section 3.1.3 (p. 24) for additional comments.

A neural network model for speech generation trained by dual extended Kalman filter is employed in [70]. No justification for the non-linear system model is given. However, if it *is* appropriate it can, in theory, perform significantly better than linear models. Promising results are presented, though due to the approximative and rather *ad hoc* nature of the extended Kalman filter [59],[30], the interaction of the two Kalman filters running in parallel [45], and neural networks in general, *no* guaranties of performance or stability can be given.

The method described in [21] is similar to this work. However, this work was done independently — the author was not aware of the article [21] until recently. The algorithm relies on VAD to update noise parameters during speech pauses, limiting its performance for low SNRs. The suggestion of updating the noise model during long continuous speech sections through Kalman filter estimates suffers from the deficiencies outlined in section 2.5.9 (p. 18). An autocorrelation method is used for AR parameters estimation which does not perform very well on noisy data. A modified power spectral subtraction with half-way rectification is used to obtain the speech spectrum parameters. An alternative algorithm is proposed based on prefiltering an input signal with the inverse of the noise system followed by an ARMA system identification. Nevertheless, this algorithm is admittedly not working well due to a small size of the stationary period available for identification.

2.7. Other methods

Relative spectral (RASTA) techniques ([27] and related [3]) work by Wiener filtering transformed time trajectories of short-time power spectral bin magnitudes.

A noise canceller using signal delayed by one speech period and self-tuning FIR filter is presented in [55].

Various other methods are presented in a survey [6], many of them modelling the speech and noise processes using *Hidden Markov models* (HMM), usually needed to be trained on clean speech and therefore suitable for limited vocabulary and limited number of speakers only. Modern example of this kind of algorithms is given in [39].

3. Kalman Filtering

It is probably not an overstatement to assert that the Kalman filter represents the most widely applied and demonstrably useful result to emerge from the state variable approach of "modern control theory".

— Harold Sorenson, [57]

Kalman filter [59],[57],[1],[25] is an *optimal* linear minimum mean-square-error *state estimator* for stochastic linear systems in a state form. Given with the model and, possibly noisy, measurements of inputs and outputs, it provides an an optimal estimate of system states. If the noises involved are Gaussian, Kalman filter becomes an optimal mean-square-error estimator, i.e., not just among the linear estimators. Many formulations exist targeted for specific application. The particular approach presented here comes mainly from [59].

The main features of Kalman filtering are sequential operation, model-based approach and possible non-stationarity.

3.1. One-Step Prediction

One of the most popular applications for Kalman filtering is one-step prediction. A linear stochastic discrete system with n-dimensional state $\boldsymbol{x}$, m-dimensional output $\boldsymbol{y}$ and r-dimensional input $\boldsymbol{u}$ is described by the following equations:

$$\begin{aligned} \boldsymbol{x}(t+1) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{v}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) + \boldsymbol{w}(t) \end{aligned} \tag{20}$$

The matrices $\boldsymbol{A}_{n\times n}$, $\boldsymbol{B}_{n\times r}$, $\boldsymbol{C}_{m\times n}$ and $\boldsymbol{D}_{m\times r}$ are assumed to be known and time-invariant. The extension towards time-varying systems is straightforward [1] but makes the notation rather clumsy and is therefore omitted for clarity reasons here. Both input $\boldsymbol{u}$ and output $\boldsymbol{y}$ are measurable and known, unlike the state $\boldsymbol{x}$ which is "hidden" inside the system and must be thus estimated. And this is precisely a task for KF. The mutually uncorrelated discrete white noises $\boldsymbol{v}(t)$ and $\boldsymbol{w}(t)$ are called *process noise* and *measurement noise* and have the following known symmetric positive-definite covariance matrix

$$\operatorname{cov}\begin{bmatrix}\boldsymbol{v}(t)\\ \boldsymbol{w}(t)\end{bmatrix} = \mathcal{E}\left\{\begin{bmatrix}\boldsymbol{v}(t)\\ \boldsymbol{w}(t)\end{bmatrix}\cdot\begin{bmatrix}\boldsymbol{v}(t)\\ \boldsymbol{w}(t)\end{bmatrix}^T\right\} = \begin{bmatrix}\boldsymbol{Q}_{n\times n} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{R}_{m\times m}\end{bmatrix} \tag{21}$$

The Kalman filter is used to update an estimate $\hat{x}$ of the state $\boldsymbol{x}$ minimising in each step the trace $\operatorname{tr}\boldsymbol{P}(t)$ of the estimation error covariance matrix

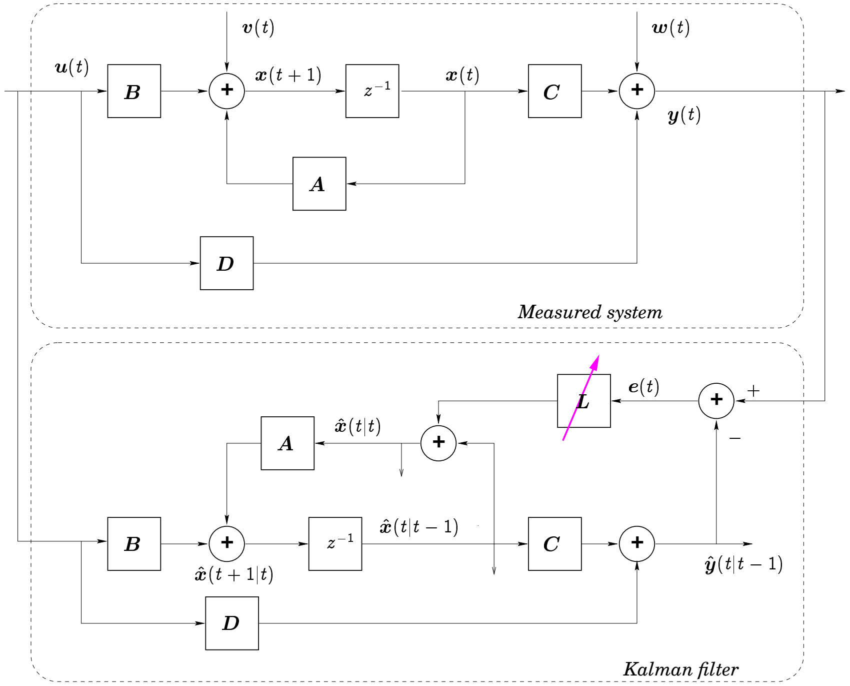


Figure 3.

$$\boldsymbol{P}(t) = \operatorname{cov} \boldsymbol{x}(t) = \mathcal{E}\left\{\tilde{\boldsymbol{x}}(t)\,\tilde{\boldsymbol{x}}^T(t)\right\} \quad \text{where} \quad \tilde{\boldsymbol{x}}(t) = \boldsymbol{x}(t) - \hat{\boldsymbol{x}}(t) \tag{22}$$

Figure 3 shows the structure of the compound consisting of a Kalman filter and the original system.

An *a priori* estimate $\boldsymbol{x}(t|t-1)$ — the best prediction of $\boldsymbol{x}(t)$ using all the available data *prior* to time t, denoted $\mathcal{D}^{t-1} = \{\boldsymbol{u}(1), \boldsymbol{u}(2), \ldots, \boldsymbol{u}(t-1), \boldsymbol{y}(1), \boldsymbol{y}(2), \ldots, \boldsymbol{y}(t-1)\}$ — with its associated covariance matrix $\boldsymbol{P}(t|t-1)$ are assumed to be known. The *data step* of the Kalman filter can be then used to incorporate the measurements $\boldsymbol{u}(t)$, $\boldsymbol{y}(t)$ from time t to get the *a posteriori* state estimate $\boldsymbol{x}(t|t)$.

$$\begin{aligned}
\hat{\boldsymbol{y}}(t|t-1) &= \boldsymbol{C}\hat{\boldsymbol{x}}(t|t-1) + \boldsymbol{D}\boldsymbol{u}(t) \\
\boldsymbol{e}(t) &= \boldsymbol{y}(t) - \hat{\boldsymbol{y}}(t|t-1) \\
\boldsymbol{L}(t) &= \boldsymbol{P}(t|t-1)\,\boldsymbol{C}^T\left(\boldsymbol{C}\boldsymbol{P}(t|t-1)\boldsymbol{C}^T + \boldsymbol{R}\right)^{-1} \\
\hat{\boldsymbol{x}}(t|t) &= \hat{\boldsymbol{x}}(t|t-1) + \boldsymbol{L}(t)\boldsymbol{e}(t) \\
\boldsymbol{P}(t|t) &= \boldsymbol{P}(t|t-1) - \boldsymbol{L}(t)\boldsymbol{C}\boldsymbol{P}(t|t-1)
\end{aligned} \tag{23}$$

The *a posteriori* estimate $\hat{\boldsymbol{x}}(t|t)$ with its associated covariance matrix $\boldsymbol{P}(t|t)$ is now the best linear estimate of $\boldsymbol{x}(t)$ using all the data upto and *including* data from time t, i.e., the estimate is based on data $\mathcal{D}^t = \{\boldsymbol{u}(1), \boldsymbol{u}(2), \ldots, \boldsymbol{u}(t), \boldsymbol{y}(1), \boldsymbol{y}(2), \ldots, \boldsymbol{y}(t)\}$. The recursion is closed using the *time step* of the Kalman filter:

$$\begin{aligned}
\hat{\boldsymbol{x}}(t+1|t) &= \boldsymbol{A}\hat{\boldsymbol{x}}(t|t) + \boldsymbol{B}\boldsymbol{u}(t) \\
\boldsymbol{P}(t+1|t) &= \boldsymbol{A}\boldsymbol{P}(t|t)\boldsymbol{A}^T + \boldsymbol{Q}
\end{aligned} \tag{24}$$

By providing a suitable initial estimate $\hat{\boldsymbol{x}}(1|0)$ and $\boldsymbol{P}(1|0)$ and alternating data steps (23) and time steps (24) it is possible to recursively calculate estimates $\hat{\boldsymbol{x}}(t)$ for all times t. Provided that the initial estimate is a true minimum mean-square-error estimate of the state $\boldsymbol{x}(1)$, all the subsequent estimates of $\boldsymbol{x}(t)$ will be the best linear estimates in the mean-square sense for the given data. If additionally the noises $\boldsymbol{v}$ and $\boldsymbol{w}$ are Gaussian, the estimates $\hat{\boldsymbol{x}}(t)$ will be optimal in the mean-square sense.

Note that the *Kalman gain* $\boldsymbol{L}(t)$ is independent of the measured data and can be thus computed in advance.

The covariance matrix $\boldsymbol{P}(t)$ can be proved to stay symmetric and positive definite. Furthermore, if the initial estimate $\boldsymbol{P}(1|0)$ is conservative in the sense $\boldsymbol{P}(1|0) \geq \boldsymbol{P}^*(1|0)$ (where $\boldsymbol{P}^*$ is the true covariance and the inequality is interpreted so as that the difference between the two matrices is positive-semidefinite) then $\boldsymbol{P}(t|t-1) \geq \boldsymbol{P}^*(t|t-1)$, i.e., the true covariances are *bounded* by the calculated ones [46] (also appeared in [57]).

The filtering procedure outlined above provides not only the current estimates of the system state and output but can be also employed to find the predicted future values and smoothed ancient values thereof. Alternative formulas for one-step Kalman prediction can be found in [59],[1],[57].

3.1.1. Innovation

Calculating the covariance of the prediction error sequence $\boldsymbol{e}$ and factorizing we get

$$\operatorname{cov} \boldsymbol{e} = \boldsymbol{C}\boldsymbol{P}(t|t-1)\boldsymbol{C}^T + \boldsymbol{R} = \boldsymbol{\Gamma}(t)\boldsymbol{\Gamma}^T(t)$$

The sequence $\boldsymbol{\nu}(t)_{m\times 1}$ defined by

$$\boldsymbol{e}(t) = \boldsymbol{\Gamma}(t)\boldsymbol{\nu}(t)$$

is called *innovation sequence*. If the filter is optimal, the innovation sequence is white and its covariance is unitary

$$\operatorname{cov} \boldsymbol{\nu} = \boldsymbol{I}_{m\times m}$$

For single-output case, innovation sequence and prediction error differ only in magnitude and the terms are therefore sometimes used interchangeably.

3.1.2. Steady-State Kalman Filter

Under fairly general assumptions (the precise formulation is too lengthy to be included here, see [59] for details) the sequence $\{\boldsymbol{P}(t+1|t)\}_{t=0}^{\infty}$ converges *exponentially* to a limit $\boldsymbol{P}_\infty$ and consequently also the Kalman gain converges to a limit $\boldsymbol{L}_\infty$. The convergence rate depends on variances of the measurement and process noises and on the speed of the system — it is faster for stronger measurement noise, weaker process noise and faster system. *Discrete-time algebraic Riccati equation* (DARE) holds for the limit values:

$$\begin{aligned} \boldsymbol{P}_\infty &= \boldsymbol{A}\boldsymbol{P}_\infty\boldsymbol{A}^T - \boldsymbol{A}\boldsymbol{P}_\infty\boldsymbol{C}^T\left(\boldsymbol{C}\boldsymbol{P}_\infty\boldsymbol{C}^T+\boldsymbol{R}\right)^{-1}\boldsymbol{C}\boldsymbol{P}_\infty\boldsymbol{A}^T+\boldsymbol{Q} \\ \boldsymbol{L}_\infty &= \boldsymbol{P}_\infty\boldsymbol{C}^T\left(\boldsymbol{C}\boldsymbol{P}_\infty\boldsymbol{C}^T+\boldsymbol{R}\right)^{-1} \end{aligned} \tag{25}$$

Approximating $\boldsymbol{L}(t)$ from (23) by $\boldsymbol{L}_\infty$ yields *steady-state Kalman filter*:

$$\begin{aligned} \hat{\boldsymbol{y}}(t|t-1) &= \boldsymbol{C}\hat{\boldsymbol{x}}(t|t-1)+\boldsymbol{D}\boldsymbol{u}(t) \\ \boldsymbol{e}(t) &= \boldsymbol{y}(t)-\hat{\boldsymbol{y}}(t|t-1) \\ \hat{\boldsymbol{x}}(t|t) &= \hat{\boldsymbol{x}}(t|t-1)+\boldsymbol{L}_\infty\boldsymbol{e}(t) \\ \hat{\boldsymbol{x}}(t+1|t) &= \boldsymbol{A}\hat{\boldsymbol{x}}(t|t)+\boldsymbol{B}\boldsymbol{u}(t) \end{aligned} \tag{26}$$

Using (26) instead of (23,24) can yield significant computational savings mainly if many samples are to be processed, because $\boldsymbol{P}$ does not have to be updated. However, it must be determined for each application separately whether the loss of precision is acceptable and if the time savings are significant.

The straightforward solution of DARE (25) is *iterative* [59]. It is simple but converges only linearly. More elaborate methods like *step-doubling* [1] or *Kleinmann algorithm* [59] converge in a quadratic fashion but require extensive calculations in each step. Purely algebraic DARE solution using *Hamilton matrix* eigen-decomposition also exists ([59], *MATLAB* function `dare`) but for most systems it is usually the slowest method.

3.1.3. Innovation Based Iterative Improvement

It is shown in [42],[43],[19] that if the steady state Kalman filter is not working optimally, because an inaccurate model was specified, the autocorrelation of the innovation sequence (which is now non-white) can be used to iteratively find an optimal Kalman gain $\boldsymbol{L}_\infty^*$. If only a limited amount of data is available (as it is always the case), the autocorrelation function of the innovation sequence must be estimated. According to experiments it seems that if the noise level is high, this estimate becomes unreliable and this prevents the algorithm from converging. See also section 2.6.1 (p. 19) for complementary description.

3.2. Fixed-Interval Smoothing

In the prediction setting, the Kalman filter provides the best estimate based on past data. If *future* data are also available, they can be used to further improve the estimate. The task of estimating states $\boldsymbol{x}(0)$, $\boldsymbol{x}(1)$,... $\boldsymbol{x}(T)$, given in advance all the available data $\mathcal{D}^T=\{\boldsymbol{u}(1),\boldsymbol{u}(2),\ldots,\boldsymbol{u}(t), \boldsymbol{y}(1),\boldsymbol{y}(2),\ldots,\boldsymbol{y}(T)\}$, as opposed to just past data, is termed fixed-interval *smoothing*.

3.2.1. Three-Pass Smoothing

To use Kalman filter to solve the smoothing problem, we may begin by *forward run*, described in section 3.1 (p. 21), followed by *backward run* obtained by applying the equations (23,24) on a *time-reversed* system. Finally, the results of the two runs are combined in a third pass [59]. This method requires little modification of the original Kalman filter algorithm but is computationally very demanding as several matrix inversions must be performed in each step.

3.2.2. Two-Pass Smoothing

A modification of the algorithm above requiring only two passes over the data can be found in [59] or [41] (included in [57], originally [52]). The forward run rests unchanged, the backward run is as follows:

$$\begin{aligned} \hat{\boldsymbol{x}}(t|T) &= \hat{\boldsymbol{x}}(t|t)+\boldsymbol{F}(t)\Big(\hat{\boldsymbol{x}}(t+1|T)-\hat{\boldsymbol{x}}(t+1|t)\Big) \\ \boldsymbol{P}(t|T) &= \boldsymbol{P}(t|t)-\boldsymbol{F}(t)\Big(\boldsymbol{P}(t+1|t)-\boldsymbol{P}(t+1|T)\Big)\boldsymbol{F}^T(t) \\ \boldsymbol{F}(t) &= \boldsymbol{P}(t|t)\boldsymbol{A}^T\boldsymbol{P}^{-1}(t+1|t) \end{aligned} \tag{27}$$

where the index T denotes estimates based on all available data. The recommended way to compute $\boldsymbol{F}(t)$ is to use the *Cholesky decomposition* of $\boldsymbol{P}(t+1|t)$ (guaranteed to exist because of the positive-definiteness), see [50]. It is both quicker and more accurate. Many times, $\boldsymbol{P}(t|T)$ does not have to be computed.

3.2.3. Fast Two-Pass Smoothing

The algorithm described by (27) still needs to perform (an equivalent of) matrix inversion of $\boldsymbol{P}$ in each step. This is avoided, with beneficial effects to performance, in another algorithm from [41] [1] (included in [57], based on [8],[9]) by introducing an auxiliary variable $\boldsymbol{\lambda}_{n\times 1}$

$$\boldsymbol{\lambda}(t)=\boldsymbol{P}^{-1}(t+1|t)\left(\hat{\boldsymbol{x}}(t+1|T)-\hat{\boldsymbol{x}}(t+1|t)\right) \tag{28}$$

The nice thing about $\boldsymbol{\lambda}$ is that it can be updated without the need of inverting $\boldsymbol{P}$

$$\begin{aligned} \boldsymbol{\lambda}(t-1) &= \boldsymbol{A}^T\boldsymbol{\lambda}(t)+\left(\boldsymbol{C}^T\boldsymbol{R}^{-1}\boldsymbol{C}\boldsymbol{L}(t)+\boldsymbol{K}(t)\right)_{n\times m}\cdot \\ &\quad\cdot\left(\boldsymbol{e}(t)-\boldsymbol{C}\boldsymbol{L}(t)\boldsymbol{e}(t)-\boldsymbol{C}\boldsymbol{P}(t|t)\boldsymbol{A}^T\boldsymbol{\lambda}(t)\right)_{m\times 1} \\ \text{where}\quad \boldsymbol{K}(t) &= \boldsymbol{C}^T\left(\boldsymbol{C}\boldsymbol{P}(t|t-1)\boldsymbol{C}^T+\boldsymbol{R}\right)^{-1} \end{aligned} \tag{29}$$

The forward run algorithm is extended to store the values of $\boldsymbol{P}(t|t)$, $\boldsymbol{L}(t)$ and $\boldsymbol{K}(t)$ (which is obtained as a by-product of calculating $\boldsymbol{L}$, see (23)). The backward run is initialized by setting $\boldsymbol{\lambda}(T)=\boldsymbol{0}$ and carried out by recursive application of (29) and (30).

$$\hat{\boldsymbol{x}}(t|T)=\hat{\boldsymbol{x}}(t|t)+\boldsymbol{P}(t|t)\boldsymbol{A}^T\boldsymbol{\lambda}(t) \tag{30}$$

This algorithm is very fast in comparison to the preceding ones but unfortunately exhibits far less numerical stability since the numerical errors tend to build up. Caution should be taken to check

[1] The update formula had to be changed slightly to work properly.

for possible divergence, see next section, and to switch to higher precision or more stable algorithm, if necessary.

3.3. Square-Root Covariance Kalman Filter

3.3.1. Divergence

Due to modelling errors, bad numerical conditioning and other numerical errors, Kalman filter algorithms can sometimes produce estimates much *less accurate* than the covariance matrix would have indicated, can become *unstable* (the state estimate increases exponentially beyond any bounds) and can fail due to the lack of non-negative-definiteness of the calculated matrix $\boldsymbol{P}$. This is generally called *divergence* [59],[1],[57]. It usually manifests itself by unnaturally small values of $\boldsymbol{P}$, $\boldsymbol{L}$ and too large a magnitude of $\hat{\boldsymbol{x}}$. Moreover, the prediction error sequence $\boldsymbol{e}$ fails to be white, zero-mean and have the expected covariance $\boldsymbol{R} + \boldsymbol{C}\boldsymbol{P}(t|t-1)\boldsymbol{C}^T$.

The remedy is to improve the model, artificially increase $\boldsymbol{R}$, limit $\boldsymbol{P}$, or — if the model is intrinsically badly conditioned — use numerically more stable algorithm.

3.3.2. Square-Root Filter

The main idea behind *square-root* (SR) Kalman filtering is to store the covariance matrix $\boldsymbol{P}$ in factorized form. This effectively *doubles* the available precision and makes the algorithm exceptionally stable. If it still fails, it is an indication that Kalman filtering is most likely inappropriate for the situation. Vast range of SR algorithms have been developed, varying in the type of factorisation and update formulas. See [59],[1] and survey [31] (appeared in [57]). Typically, an SR algorithm needs about 1.5 times the number of operations of the classical Kalman filter implementation. However, in cases when the classical implementation can be simplified due to system structure, no such optimisation is usually possible for the SR version and the corresponding increase in computational burden is substantial.

In the algorithm presented below, taken from [1], *Cholesky* factor $\boldsymbol{M}^T$ [50] of the covariance matrix is stored, instead of the full form $\boldsymbol{P}$.

$$\boldsymbol{P} = \boldsymbol{M}\boldsymbol{M}^T \tag{31}$$

where $\boldsymbol{M}$ is lower triangular. This ensures the non-negative definiteness of $\boldsymbol{P}$.

The *time step* is given be the following equations

$$\begin{aligned}
\hat{\boldsymbol{x}}(t+1|t) &= \boldsymbol{A}\hat{\boldsymbol{x}}(t|t) + \boldsymbol{B}\boldsymbol{u}(t) \\
\begin{bmatrix} \boldsymbol{M}^T(i+1|i) \\ \boldsymbol{0} \end{bmatrix}_{2n\times n} &= \boldsymbol{T} \begin{bmatrix} \boldsymbol{M}^T(i|i)\,\boldsymbol{A}^T \\ \boldsymbol{Q}^{T/2} \end{bmatrix}_{2n\times n} \\
\boldsymbol{Q} &= \boldsymbol{Q}^{1/2}\boldsymbol{Q}^{T/2} \\
\boldsymbol{I}_{2n\times 2n} &= \boldsymbol{T}\boldsymbol{T}^T
\end{aligned} \tag{32}$$

The matrix $\boldsymbol{Q}^{T/2}$ can be computed in advance by Cholesky factorisation, or, if $\boldsymbol{Q}$ is diagonal (which is often the case) by taking the square-root of the diagonal, element-by-element. The orthogonal matrix $\boldsymbol{T}$ does not have to be computed explicitly; all we need is to find an orthogonal transformation to transform a general matrix $\boldsymbol{X}$ to an upper triangular matrix $\boldsymbol{Y}$, i.e., $\boldsymbol{Y} = \boldsymbol{T}\boldsymbol{X}$. Gram-Schmidt methods, Householder transformation or Givens rotations can be used for this

task [50]. Realizing that $\boldsymbol{X} = \boldsymbol{T}^T\boldsymbol{Y}$, which describes a standard *QR-factorisation*, any of the readily available algorithms [50],[72] can be used. As $\boldsymbol{Y}^T\boldsymbol{Y} = \boldsymbol{X}^T\boldsymbol{X}$, Cholesky factorisation can be employed, too.

The *data step* is slightly more complicated:

$$\begin{aligned}
\hat{\boldsymbol{y}}(t|t-1) &= \boldsymbol{C}\hat{\boldsymbol{x}}(t|t-1) + \boldsymbol{D}\boldsymbol{u}(t) \\
\boldsymbol{e}(t) &= \boldsymbol{y}(t) - \hat{\boldsymbol{y}}(t|t-1) \\
\hat{\boldsymbol{x}}(t|t) &= \hat{\boldsymbol{x}}(t|t-1) + \boldsymbol{L}(t)\boldsymbol{e}(t) \\
\boldsymbol{L}(t) &= \boldsymbol{L}_\star(t)\boldsymbol{W}^{-1}(t) \\
\begin{bmatrix} \boldsymbol{W}^T(t) & \boldsymbol{L}_\star^T \\ \boldsymbol{0} & \boldsymbol{M}^T(t|t) \end{bmatrix} &= \boldsymbol{T}_\star \begin{bmatrix} \boldsymbol{R}^{T/2} & \boldsymbol{0} \\ \boldsymbol{M}^T(t|t-1)\,\boldsymbol{C}^T & \boldsymbol{M}^T(t|t-1) \end{bmatrix}_{(n+m)\times(n+m)} \\
\boldsymbol{R}_{m\times m} &= \boldsymbol{R}^{1/2}\boldsymbol{R}^{T/2} \\
\boldsymbol{I}_{(m+n)\times(m+n)} &= \boldsymbol{T}_\star\boldsymbol{T}_\star^T
\end{aligned} \tag{33}$$

As before, $\boldsymbol{R}^{T/2}$ can be computed in advance. The meaning of $\boldsymbol{W}$ is

$$\boldsymbol{W}(t)\boldsymbol{W}^T(t) = \boldsymbol{C}\boldsymbol{P}(t|t-1)\boldsymbol{C}^T + \boldsymbol{R}$$

To find the orthogonal triangularisation factor $\boldsymbol{T}_\star$, the same methods as in the time step can be used.

Further refinements, optimisations and alternative formulas can be found in [1],[31],[59]. According to [1], no set of filtering equations is always superior to the others. It is therefore necessary to evaluate their performance for the specific application.

3.3.3. Square-Root Smoothing

Square-root Kalman filter described by (32,33) can be effectively used for smoothing using the procedure (27), calculating $\boldsymbol{P}(t|t)$ from (31). As $\boldsymbol{P}(t+1|t)$ is already available in the square-root form, see the remark about Cholesky factorisation in section 3.2.2 (p. 25), the increase in complexity of square-root smoothing over square-root filtering is only modest while the benefit can be significant.

3.4. Extended Kalman Filter

Extended Kalman Filter (EKF) is a generalisation of an ordinary Kalman filter for non-linear systems, see [59], [57]. In each step, the general non-linear system is linearised around the current estimate of the state. Ordinary Kalman update of the state estimate is then calculated using the current linearised model. Note that this implies time-varying system matrices even if the underlying system is time-invariant.

Unfortunately, the linearization is rather an *ad hoc* approximation and may often be too coarse, especially if the initial estimate is imprecise, noise level is high or the system is extremely nonlinear. While EKF permits to use Kalman filtering even for non-linear systems and can be sometimes very efficient, experience shows that at many times it behaves poorly and divergence frequently occurs.

EKF is often used as a *parameter estimator*. Nevertheless, even if it converges, it usually converges much more slowly than corresponding alternative methods. See also section 4.5 (p. 36) for complementary description.

3.4.1. Neural Network Training

An interesting application of EKF is a training of feed-forward neural networks, [18]. It appears that states of a system modelled by a suitable non-linear neural network as well as the parameters of this network can be estimated by EKF without a pronounced danger of instability [70].

3.5. Parameter Optimisation

Parameters of a system we want to apply the Kalman filter on are often known only approximately. While the adverse effects of an erroneous model on the estimation error have not been thoroughly and completely investigated yet, it is known that they can be pronounced [46],[57]. As an alternative to improving the identification method for obtaining better model parameters beforehand, it is also possible to use the Kalman filter to iteratively ameliorate the model and thus the accuracy of the state estimate itself. Section 3.1.3 (p. 24) mentiones one such approach together with its limitations. Moreover, it is applicable only for steady-state Kalman filtering.

A better method exploits the quadratic optimality of Kalman estimate. Let us denote the unknown parameters needed to use Kalman filter as

$$\boldsymbol{\theta} = \{ \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{P}(1|0), \hat{\boldsymbol{x}}(1|0) \}$$

and the correct values as $\boldsymbol{\theta}^*$. Because the Kalman state estimate is optimal, it follows that

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}} \mathcal{E} \left\{ \boldsymbol{e}^T \boldsymbol{e} \right\}_{\forall\, t \in \langle 1, T \rangle} \tag{34}$$

The above formula is obviously not directly usable. To be rendered applicable, it must be approximated. A plausible version is

$$\hat{\boldsymbol{\theta}}^* = \arg\min_{\boldsymbol{\theta}} \sum_{t=1}^{T} \boldsymbol{e}^T(t)\, \boldsymbol{e}(t) \tag{35}$$

Any multidimensional minimum-finding method can be applied to solve (35) while the Kalman filter prediction error acts as the objective function. To speed it up, it is useful to note that $\hat{\boldsymbol{x}}(t) \propto \hat{\boldsymbol{x}}(1|0)$, which permits us to calculate $\hat{\boldsymbol{x}}^*(1|0)$ directly. Also observe, that multiplying $\boldsymbol{P}$, $\boldsymbol{Q}$, $\boldsymbol{R}$ by a constant scalar does not change $\hat{\boldsymbol{x}}$. Unfortunately, even after having applied these tricks, the process is still *very slow* — depending on the number of parameters to be optimised and the quality of the initial estimate, it can easily take several hours.

Experiments performed by the author show this method as very promising. However, much more investigation is definitely needed.

4. Identification

In order to use the Kalman filter, a *model* of the system must be known. A very brief survey of identification methods will be presented, with emphasis on the application in speech enhancement. For further information, it is refered to [59],[28],[62],[29],[25],[30],[54] and a comprehensive survey [2].

4.1. System Identification

The goal of a *stochastic system identification* is to find an "optimal" estimate of parameters of a stochastic system given its structure and a set of measurements of inputs and outputs of this system. Various criteria for optimality are employed, most frequently minimum *mean-square-error* (MS), minimum *linear* mean-square-error (LMS), minimum *variance* (MV), and maximum *likelihood* (ML). Other critaria are employed in *prediction error* methods (PEM). Instead of providing the values of inputs and outputs themselves, only their characteristics might be available, such as *spectrum* or *autocorrelation function*.

Only *batch* methods will be discussed here, i.e., it is assumed that all the data are available at once in advance. It is also assumed that the parameters remain *constant* during the estimation. Various techniques for time-varying parameter estimations can be found in [59],[62],[25].

4.1.1. Limits on Attainable Precision

Note that due to the stochastic nature of the system, it is of course never possible to get the *exact* values of the parameters. Moreover, the more complex the system structure is, the more difficult it is to obtain a suitable parameter estimate and the less precise the estimate. This effectively restricts the usability of complex models, if the amount and precision of data for parameter estimation is limited.

For some methods, the estimate of the uncertainty as represented by the covariance of the parameter estimates can be calculated. An important lower bound for this covariance for unbiased estimates is given by the *Cramér-Rao inequality*, [59],[28]. Unfortunately, it can obviously only be used if the stochastic parameters of the process are known, which is generally not the case.

4.2. Time-Series Analysis

In the view of the intended application only white-noise driven input-less single-output linear time-invariant system identification will be dealt with. The output of such a system is a sequence called *time series*, which can be both finite and infinite and will be denoted $y(t)$. The unit-variance zero-mean driving noise will be denoted $v(t)$

$$\mathcal{E}\{v(t)\} = 0, \quad \mathcal{E}\{v(t)v(t+\tau)\} = \delta(\tau) = \begin{cases} 1, & \text{if } \tau = 0 \\ 0, & \text{otherwise} \end{cases} \tag{36}$$

Following from the linearity, $\mathcal{E}\{y(t)\} = 0$. Because of the time-invariance, y is *stationary*, i.e., its properties do not change in time.

4.2.1. Autocorrelation

Important characteristic of y is its *autocorrelation* (the extension of the notation to the infinite case is straightforward)

$$C_y(k) = \mathcal{E}\{y(t)\cdot y(t+k)\}, \quad \text{denoted} \quad \{C_y(k)\}_{k=-T+1}^{T-1} = \mathcal{C}\{y(t)\}_{t=0}^{T-1} \tag{37}$$

Because of the stationarity of y, $C_y(k)$ does not depend on time t. It also follows that $C_y(k) = C_y(-k)$ [25]. If only one (finite length) realization of y is available, $C_y(k)$ must be estimated. According to [43], a reasonable estimate is

$$\hat{C}_y(k) = \frac{1}{T}\sum_{t=|k|}^{T-1} y(t)\, y(t-|k|) \tag{38}$$

The estimate (38) is biased for finite sample sizes,

$$\mathcal{E}\left\{\hat{C}_y(k)\right\} = (1-k/T)\, C_y(k) \tag{39}$$

However, it has been shown in [29] that the estimate (38) is preferable since it gives less mean-square error than the corresponding unbiased estimate. Heuristically speaking, the values of $\hat{C}_y(k)$ are less accurate for bigger k and it is therefore beneficial that they are smaller and have thus less influence.

It can be shown [48] that (38) is an asymptotically unbiased (as $T\to\infty$), normal and consistent estimate of C_y. An expression for the asymptotic covariance of $\hat{C}_y(k)$ has been given by Bartlett [4] (simplified):

$$\mathrm{cov}(\hat{C}_y(k), \hat{C}_y(l)) \approx \begin{cases} 0, & k \neq l \\ (1/T)\, C_y(0)^2, & k = l \neq 0 \\ C_y(0)^2, & k = l = 0 \end{cases} \tag{40}$$

4.2.2. Discrete Fourier Transform

Another commonly used characteristic of y is its (discrete) *spectrum* (sometimes called *frequency distribution*), obtainable by *discrete Fourier transform* (DFT)

$$\begin{aligned} F_y(k) &= \frac{1}{T}\sum_{t=0}^{T-1} y(t) w^{tk}, \quad \text{where} \quad w = e^{-j2\pi/T} \\ \text{denoted} \quad \{F_y(k)\}_{k=0}^{T-1} &= \frac{1}{T}\mathcal{F}\{y(t)\}_{t=0}^{T-1} \end{aligned} \tag{41}$$

The definition of $\mathcal{F}$ corresponds to the function `fft` in *MATLAB*. It is useful to define also the (discrete) *amplitude spectrum* $S_y(k) = |F_y(k)|$, the *phase spectrum* $\varphi_y(k) = \arg F_y(k)$ and the *power spectrum* $P_y(k) = |F_y(k)|^2$. All the discrete spectra are closely related to their continuous counterparts, e.g., $S_y(k) \widehat{=} S_y(\omega)|_{\omega=2k\pi/T}$.

DFT, as well as its inverse (called IDFT and denoted $\mathcal{F}^{-1}$), can be very efficiently (in $\mathcal{O}(T\log T)$ time) calculated by some of the numerous *fast Fourier transform* (FFT) algorithms ([50],[62], [30]).

4.2.3. Wiener-Khintchine Theorem

Instead of evaluating the sum (38), another autocorrelation function estimate can be obtained using a *Wiener-Khintchine theorem* [62],[30]

$$\begin{aligned} \left\{ \left\{\hat{C}_y(k)\right\}_{k=0}^{T/2} \left\{\hat{C}_y(k)\right\}_{k=T/2-1}^{1} \right\} &= \frac{1}{T}\mathcal{F}^{-1}\left\{ \left|\mathcal{F}\{y(t)\}_{t=0}^{T-1}\right|^2 \right\} \\ &= \frac{1}{T^2}\mathcal{F}\left\{ \left|\mathcal{F}\{y(t)\}_{t=0}^{T-1}\right|^2 \right\} \end{aligned} \tag{42}$$

(The right-hand side is a sequence symmetrical along the middle element.)

For moderate or long sample sets ($T \gtrsim 30$), the estimate (42) is quicker to compute than (38) (in $\mathcal{O}(T\log T)$ time as opposed to $\mathcal{O}(T^2)$) and yields almost the same results, usually well within the confidence limits of the estimate itself. Note that (42) gives effectively an autocorrelation sequence of the length $T/2+1$, while (38) gives all the T values but in practice this does not matter because the estimate $\hat{C}_y(k)$ for higher k is statistically unreliable anyway and should not be used [62].

4.2.4. Spectrum Estimation Methods

Some identification methods use a *frequency spectrum* for subsequent calculations. Since the estimate (41) has a rather high variance, it is often not directly usable.

Two of the main methods for spectrum estimation are periodogram and minimum entropy (MEM) estimate. In the *periodogram* method [59],[30],[25], the frame is first multiplied with a Hanning window (80) (or other suitable window, see [30]) and then the FFT (see section 4.2.2 (p. 30)) is applied. The estimate is further averaged or recursively smoothed (which is called *Welch* method)

$$\overline{P}_y(k) = \alpha\overline{P}_y(k) + (1-\alpha)P_y(k) \quad \forall\, k \tag{43}$$

In the *MEM* method [25], first the AR model parameters of the signal are estimated (e.g., using *Burg*'s method (section 4.4.5 (p. 35)) plus (56)) and then the calculated model spectrum is obtained using (48). The MEM method works very well if the process is autoregressive but does not give reliable results otherwise.

The *autocorrelation* (also called *correlogram*) method of spectrum estimation [30],[62] based on Wiener-Khintchine theorem (42) is difficult to use because the estimate (38) often yields invalid autocorrelation sequences under low SNR conditions.

The *minimum variance* (MV) estimate is given by [25]

$$\begin{aligned} \hat{P}_{\mathrm{MV}}(\omega) &= (p+1) \Big/ \left(\boldsymbol{w}^T(\omega)\, \boldsymbol{C}_{p\times p}^{-1}\, \boldsymbol{w}(\omega)\right) \\ \text{where} \quad \boldsymbol{w}(\omega) &= \left[1, e^{j\omega}, \ldots, e^{jp\omega}\right]^T \\ \boldsymbol{C}_{p\times p} &= \begin{bmatrix} C_y(0) & C_y(1) & \ldots & C_y(p-1) \\ C_y(1) & C_y(0) & \ldots & C_y(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ C_y(p-1) & C_y(p-2) & \ldots & C_y(0) \end{bmatrix} \end{aligned} \tag{44}$$

where p is the estimate order. The elements C_y must be approximated using the estimate (38) (or other); the unreliability of this estimate limits the usability of the MV method similarly to the previous case. MV estimate is a harmonic mean of MEM estimates of orders $0,\dots,p$ which accounts for its conservativeness.

Other parametric as well as non-parametric methods exist [62],[25],[30],[2], [54]. See also section 2.5.2 (p. 14) for an overview and section 6.4.1 (p. 45) for comments on implemenation.

4.3. Autoregressive System

In accordance with the conclusions of sections 2.3.1 (p. 13) and 2.4.1 (p. 13) only the autoregressive (AR) model parameter estimation is considered here. Such a white-noise driven single-input single-output (SISO) AR model can be described by the following equation:

$$y(t) = \sum_{i=1}^{n} a_i y(t-i) + bv(t), \quad b \geq 0 \tag{45}$$

where $v(t)$ is a unit-variance zero-mean white noise, as in (36).

As a shorthand, the set of parameters will be denoted $\boldsymbol{a} = (a_1, a_2, \dots, a_n)^T$ and $\boldsymbol{\theta} = (a_1, a_2, \dots, a_n, b)^T$.

4.3.1. State-Space Representation

The system (45) can be represented by the following state-space model with n-dimensional state $\boldsymbol{x}$

$$\begin{aligned}
\boldsymbol{x}(t) &= [y(t-n+1), y(t-n+2), \dots, y(t-1), y(t)]^T \\
\boldsymbol{A} &= \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ a_n & a_{n-1} & a_{n-2} & \dots & a_1 \end{bmatrix} \\
\boldsymbol{x}(t+1) &= \boldsymbol{A}\boldsymbol{x}(t) + b\boldsymbol{h}v(t), \qquad \boldsymbol{h} = [\underbrace{0, 0, \dots, 0, 1}_{n}]^T \\
y(t) &= \boldsymbol{h}^T \boldsymbol{x}(t)
\end{aligned} \tag{46}$$

This model is simple to construct and evaluate but might suffer from numerical instability for higher n. If this happens, other numerically more stable representations should be considered, such as lattice, cascade, Jordan etc., see [25],[60],[44].

4.3.2. Properties

The autocorrelation sequence of the system (45) satisfies the *Yule-Walker* equations [25],[62]

$$C_y(k) = b^2\delta(k) + \sum_{i=1}^{n} a_i C_y(k-i) \quad \forall\, k \in \mathcal{Z} \tag{47}$$

The discrete amplitude spectrum of the length $T+1$ of y is (following from the Wiener-Khintchine theorem)

$$\begin{aligned}
S_y(k) &= \frac{b}{\left|1 - \sum_{i=1}^{n} a_i w^{ik}\right|} \\
\text{calculated as} \quad \{S_y(k)\}_{k=0}^{T} &= b \Bigg/ \left|\mathcal{F}\left\{1, -a_1, -a_2, \dots, -a_n, \underbrace{0, 0, \dots, 0}_{(T-n-1)}\right\}\right|
\end{aligned} \tag{48}$$

(The division and absolute values are performed element by element.)

For the state-space formulation (46) the covariance matrix $\boldsymbol{P} = \operatorname{cov} x(t)$ satisfies the *Lyapunov equation* [59],[43]

$$\boldsymbol{P} = \boldsymbol{A}\boldsymbol{P}\boldsymbol{A}^T + b^2\boldsymbol{h}\boldsymbol{h}^T \tag{49}$$

The Lyapunov equation can be solved directly by transforming into a set of linear equations [59], iteratively, or by Schur decomposition (*MATLAB* function `dare`). All methods are rather time consuming — for our application (described later in more detail) solving Lyapunov equation and Kalman filtering one segment typically takes comparable time.

Using $\boldsymbol{P}$, a closed-form formula for C_y can be derived [43]

$$C_y(k) = \begin{cases} \boldsymbol{h}^T\boldsymbol{P}\boldsymbol{h} + b^2, & \text{if } k = 0 \\ \boldsymbol{h}^T\boldsymbol{A}^k\boldsymbol{P}\boldsymbol{h}, & \text{if } k > 0 \end{cases} \tag{50}$$

4.3.3. Indirect Estimation

There is an interesting similarity between (47) and (45). Under the assumption that the estimates $\hat{C}_y$ are unbiased and normal (which is asymptotically true), $\hat{C}_y$ can be viewed as an output of an input-less autoregressive output-error (AROE) system (47,55). Therefore, methods for estimating parameters of an AROE model directly from the signal can be used for estimating parameters of AR model from the autocorrelation sequence. Further approximation is to use AR model estimation techniques on the autocorrelation sequence.

Even though this use of an AR estimation algorithm is obviously suboptimal, the results are, in the author's experience, usually acceptable. A recursive extension of this approach is described in [34], see also section 2.5.9 (p. 18).

4.4. Least-Squares Estimators

In order to estimate parameters of the AR model (45) given either y or $\hat{C}_y$, the methods described in this section construct a set of linear equations and solve it (in the least-squares sense, if the set is overdetermined). Again, only a very limited subset of the most relevant available methods is presented here due to space constraints. The equations come from minimising various *least mean-square error* criteria such as the following least mean-square *forward prediction error* criterion.

$$\hat{\boldsymbol{a}} = \arg\min_{\boldsymbol{a}} \mathcal{E}\{y(t) - \hat{y}(t)\} \tag{51}$$

$$\hat{y}(t) = \sum_{i=1}^{n} a_i y(t-i) \tag{52}$$

Note that the parameter b in (45) must be estimated separately. See section 4.4.6 (p. 35) for details.

Generalising the predictor, the model structure and the optimality criterion leads to a wide class of, usually iterative, *prediction error methods* (PEM), see [54].

4.4.1. Autocorrelation Method

Using (47), exploring the symmetry of $C_y(k)$ and taking derivatives of (51) we get, after additional manipulations

$$\begin{bmatrix} \hat{C}_y(0) & \hat{C}_y(1) & \hat{C}_y(2) & \dots & \hat{C}_y(n-1) \\ \hat{C}_y(1) & \hat{C}_y(0) & \hat{C}_y(1) & \dots & \hat{C}_y(n-2) \\ \hat{C}_y(2) & \hat{C}_y(1) & \hat{C}_y(0) & \dots & \hat{C}_y(n-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \hat{C}_y(n-1) & \hat{C}_y(n-2) & \hat{C}_y(n-3) & \dots & \hat{C}_y(0) \\ \vdots & \vdots & \vdots & & \vdots \\ \hat{C}_y(N-1) & \hat{C}_y(N-2) & \hat{C}_y(N-3) & \dots & \hat{C}_y(N-n) \end{bmatrix} \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \hat{a}_3 \\ \vdots \\ \hat{a}_n \end{bmatrix} = \begin{bmatrix} \hat{C}_y(1) \\ \hat{C}_y(2) \\ \hat{C}_y(3) \\ \vdots \\ \hat{C}_y(n) \\ \vdots \\ \hat{C}_y(N) \end{bmatrix} \tag{53}$$

If the estimates $\hat{C}_y(k)$ are taken from (38) and if $N = n$ (which is the common case), the procedure described by (53) is commonly called *autocorrelation* method. As the matrix is then *symmetrical* and *Toeplitz*, fast algorithms exist capable of solving (53) in $\mathcal{O}(n^2)$ time, such us *Levinson-Durbin* algorithm, see [62],[28],[50],[25], which accounts for the popularity of this method. However, the autocorrelation method effectively weights the data with a window which seriously compromises the resulting model accuracy. Moreover, if the data is noisy, which is always the case in reality, the estimate is biased and statistically inefficient [43]. On the positive side, *stability* of the model is assured. Making $N > n$ is guaranteed to increase the accuracy of the model only if $N \ll T$, to make a good estimate $\hat{C}_y(k)$ by (38).

Similar to the autocorrelation method just described is the *covariance method* [62],[25]. However, in the author's experience, for moderate $N \gtrsim 100$ the additional computational burden is not justified by the, relatively small, estimate quality improvement.

4.4.2. Cayley-Hamilton Method

Cayley-Hamilton theorem states that a matrix is a root of its own characteristic polynomial. Applying this theorem on (50) (see [43]) yields after additional manipulation the following equation. Alternatively, the same result can be obtained from (47).

$$\begin{bmatrix} \hat{C}_y(1) & \hat{C}_y(2) & \hat{C}_y(3) & \dots & \hat{C}_y(n) \\ \hat{C}_y(2) & \hat{C}_y(3) & \hat{C}_y(4) & \dots & \hat{C}_y(n+1) \\ \hat{C}_y(3) & \hat{C}_y(4) & \hat{C}_y(5) & \dots & \hat{C}_y(n+2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \hat{C}_y(n) & \hat{C}_y(n+1) & \hat{C}_y(n+2) & \dots & \hat{C}_y(2n-1) \\ \vdots & \vdots & \vdots & & \vdots \\ \hat{C}_y(N) & \hat{C}_y(N+1) & \hat{C}_y(N+2) & \dots & \hat{C}_y(N+n-1) \end{bmatrix} \begin{bmatrix} \hat{a}_n \\ \hat{a}_{n-1} \\ \hat{a}_{n-2} \\ \vdots \\ \hat{a}_1 \end{bmatrix} = \begin{bmatrix} \hat{C}_y(n+1) \\ \hat{C}_y(n+2) \\ \hat{C}_y(n+3) \\ \vdots \\ \hat{C}_y(2n) \\ \vdots \\ \hat{C}_y(N+n) \end{bmatrix} \tag{54}$$

The matrix is known as the *Hankel* matrix of the system [43], however, it can be converted to the Toeplitz structure by reverting $\boldsymbol{a}$. Again, as in the previous case, it is usually set $N = n$ and (38) is used to estimate $\hat{C}_y$. Increase N if your data set (T) is large to gain some extra accuracy. It can be proved [43] that asymptotically (as $T \to \infty$) the estimate $\hat{\boldsymbol{a}}$ from (54) is *unbiased*, *normal* and *consistent* as the errors of estimating $\hat{\boldsymbol{a}}$ are linearly related to errors of estimating C_y. For moderate $T \approx 100$, the performance is similar to the autocorrelation method.

4.4.3. Output-Error Model

The principal advantage of (54) over (53) lies in the fact that (54) does not use $\hat{C}_y(0)$ and can be therefore used also for *output-error* models

$$y_\times = y + w \tag{55}$$

when only $y_\times$ is observable, y is an AR process (as in (45)) and w is a *white* measurement noise with (possibly) unknown variance but *independent* of the system driving noise v, defined by (36). Because of the whiteness and independence it holds

$$C_\times(k) = \begin{cases} C_y(k), & \text{if } k \neq 0 \\ C_y(0) + C_w(0), & \text{if } k = 0 \end{cases}$$

where $C_\times$ is the covariance of $y_\times$. It implies that $\hat{C}_y(0)$ is not readily available and therefore (54) must be used in place of (53).

Generally, the additive noise w will degrade the quality of the estimate of $\hat{C}_y(k)$ also for $k \neq 0$. This has serious adverse effects on both (53) and (54). One solution is to use the repetitive autocorrelation method [34] but it is not feasible for many applications, as already discussed in section 2.5.9 (p. 18).

4.4.4. Modified Covariance Method

The autocorrelation method minimises the *forward* prediction error (51). However, given all the data at once, there is no reason to prefer the forward direction over the backward direction. The *modified covariance method* (also called *forward-backward* method) combines the two and minimises the *sum* of forward *and* backward prediction errors. The formulas are rather lengthy and will not be given here, they can be found in [25]. The modified covariance method performs very well, usually much better than either autocorrelation or Cayley-Hamilton methods. Its disadvantages are the need to operate directly on the signal y and no guaranty of the model stability.

4.4.5. Burg's Method

The *Burg's* method [25] minimises the same criterion as the modified covariance method and operates directly on the signal y, too. However, in contrast to this method, Burg proposed to *sequentially* minimise the *reflection coefficients* ([25],[62]) starting from first order filter towards higher orders. Although the result is not globally optimal, it is usually close to the result given by modified covariance method and the model is in addition guaranteed to be *stable*. Burg's method requires only $\mathcal{O}(nT)$ operations which makes it fast enough for most applications. Sometimes, the estimate seems to be sensible to signal phase shifts.

4.4.6. Estimating b

Several methods are available for estimating the parameter b in (45). The power spectrum of the model can be calculated using (48) and then made equal to the directly calculated signal energy

$$\sum_{k=0}^{T} S_y^2(k) = \sum_{t=0}^{T} y^2(t) \tag{56}$$

Alternatively, an autocorrelation sequence $C_y(k)$ of the model output can be calculated either from (38),(42) or (50) and then fitted, in the least-squares sense, to the autocorrelation sequence $\hat{C}_y(k)$ estimated from data.

$$b = \arg\min_{b} \sum_{k=0}^{T'} \left(C_y(k) - \hat{C}_y(k) \right)^2 \tag{57}$$

where T' is the index of the last reliable estimate. Implementation of both methods should make use of the fact that $C_y(k) \propto b^2$, $S_y(k) \propto b^2$. Other methods, such as linearly fitting power spectrum etc., are of course possible, depending on what data are available. Least-squares fitting gives usually slightly better results at the expense of speed.

It is also possible to set b^2 equal to the power of the modelling error residual series [25],[12] giving

$$b^2 = \hat{C}_y(0) + \sum_{k=1}^{n} \hat{a}_i \hat{C}_y(i) \tag{58}$$

which is the quickest to compute but the estimate is often not very good.

4.5. Kalman Filter for Parameter Estimation

Extended Kalman filter can be used for parameter estimation by regarding the unknown parameters as additional states [59],[57], as already mentioned in section 3.4 (p. 27). However, the resulting combined model is non-linear which causes difficulties in assuring stability and convergence speed.

In some cases, though, it is possible to avoid the non-linearity in the estimation process and to use the Kalman filter effectively. A *dual* Kalman filter approach was proposed in [45], using one KF to estimate the parameters and a second one to estimate the state. It was shown in [24] that provided the stochastic parameters of an ARMAX (auto-regressive moving average) system are known, the rest of the parameters and the state can be estimated simultaneously by a linear Kalman filter.

4.6. State Space Approach

In the classical identification dealt with so far, the stochastic system parameters are determined from noisy observations and then used to find an estimate of the system states, for example by Wiener or Kalman filtering. In the *state space* approach [14], an estimate of the system states is calculated first and then used to determine system parameters. The observations are usually gathered in a matrix and then the closest matrix fulfilling constraints imposed by the system structure is calculated. State space methods are in many cases equivalent to classical methods, at least in the deterministic setting. For further information on algorithms and applications, see [14],[36],[13],[64],[69],[66],[37],[67],[35],[68],[10].

4.6.1. HTLS

Hankel TLS (HTLS) is a subspace-based method for estimating parameters of damped sinusoids in additive white noise [10],[11]. An application on the autocorrelation sequence estimates $\hat{C}_y(k)$ will be shown here. Alternatively, HTLS can be used on the output of input-less autoregressive output-error model, as in [10] or [36]. It can be assumed that the difference $\tilde{C}_y(k) = C_y(k) - \hat{C}_y(k)$ is white, see section 4.2.1 (p. 30). Furthermore, as a consequence of (47), $C_y(k)$ can be expressed as a sum of exponentially damped sinusoids.

The measurement data is arranged in a Hankel matrix. $\hat{C}_y(0)$ should be included if it is available, see section 4.4.3 (p. 35). Otherwise, start with $\hat{C}_y(1)$.

$$\boldsymbol{X}_{l\times m} = \begin{bmatrix} \hat{C}_y(0) & \hat{C}_y(1) & \hat{C}_y(2) & \dots & \hat{C}_y(m-1) \\ \hat{C}_y(1) & \hat{C}_y(2) & \hat{C}_y(3) & \dots & \hat{C}_y(m) \\ \hat{C}_y(2) & \hat{C}_y(3) & \hat{C}_y(4) & \dots & \hat{C}_y(m+1) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \hat{C}_y(l-1) & \hat{C}_y(l) & \hat{C}_y(l+1) & \dots & \hat{C}_y(l+m-2) \end{bmatrix} \tag{59}$$

The dimensions l, m must be greater than n. Moreover, $l+m-1$ may not exceed the length of the autocorrelation sequence available. It is recommended to choose $l \approx m$ because bigger matrix $\boldsymbol{X}$ results in greater accuracy.

A *signal matrix* $\boldsymbol{X}$ of a signal composed by n exponentially damped sinusoids can be decomposed using a *Vandermonde decomposition* (VD) [11] as follows

$$\begin{aligned} \boldsymbol{X} &= \boldsymbol{F}\boldsymbol{G}\boldsymbol{H}^T = \\ &= \begin{bmatrix} 1 & 1 & \dots & 1 \\ \zeta_1^1 & \zeta_2^1 & \dots & \zeta_n^1 \\ \zeta_1^2 & \zeta_2^2 & \dots & \zeta_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \zeta_1^{l-1} & \zeta_2^{l-1} & \dots & \zeta_n^{l-1} \end{bmatrix} \begin{bmatrix} g_1 & & & \\ & g_2 & & \\ & & \ddots & \\ & & & g_k \end{bmatrix} \begin{bmatrix} 1 & \zeta_1^1 & \dots & \zeta_1^{m-1} \\ 1 & \zeta_2^1 & \dots & \zeta_2^{m-1} \\ 1 & \zeta_3^1 & \dots & \zeta_3^{m-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \zeta_n^1 & \dots & \zeta_n^{m-1} \end{bmatrix} \end{aligned} \tag{60}$$

where $\zeta_i = \mathrm{e}^{-\alpha_i + j\omega_i}$ are the (complex) signal components with their respective normalised frequencies and damping factors and g_i are the corresponding (complex) amplitudes. A diagonal matrix $\boldsymbol{Z}$ relates the VD in consecutive time intervals

$$\boldsymbol{Z} = \begin{bmatrix} \zeta_1 & & & \\ & \zeta_2 & & \\ & & \ddots & \\ & & & \zeta_k \end{bmatrix}$$

$$\boldsymbol{F}^{\uparrow} = \boldsymbol{F}^{\downarrow}\boldsymbol{Z} \qquad \boldsymbol{H}^{\uparrow} = \boldsymbol{H}^{\downarrow}\boldsymbol{Z} \tag{61}$$

where $\uparrow$ resp. $\downarrow$ stand for deleting the top resp. bottom row.

It turns out ([11],[10]) that a SVD of $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$ has a very similar property

$$\boldsymbol{U}^{\uparrow} = \boldsymbol{U}^{\downarrow}\boldsymbol{Z}' \qquad \boldsymbol{V}^{\uparrow} = \boldsymbol{V}^{\downarrow}\boldsymbol{Z}'' \tag{62}$$

where $\boldsymbol{Z}'$, $\boldsymbol{Z}''$ are similarity transforms of $\boldsymbol{Z}$ and have therefore the *same eigenvalues*. This permits retrieving roots of the characteristic polynomial of the generating system by eigenvalue decomposition [50].

The HTLS estimation procedure can be summarised as follows:

- **(1)** — Arrange the (noisy) measured data in a matrix $\boldsymbol{X}$ according to (59).
- **(2)** — Choose a model order n. Compute rank n SVD approximation $\hat{\boldsymbol{X}}_{l\times m} = \boldsymbol{U}_{l\times n}\boldsymbol{S}_{n\times n}\boldsymbol{V}^T_{m\times n}$
- **(3)** — Use TLS (see next chapter) to retrieve $\boldsymbol{Z}'$ from (62).
- **(4)** — Find eigenvalues $\hat{\zeta}_1,\dots,\hat{\zeta}_n$ of $\boldsymbol{Z}'$.
- **(5)** — Calculate the desired system parameters, e.g., for an autoregressive system, solve the following equation for a_i

$$x^n - \sum_{i=1}^{n} a_i x^{n-i} = \prod_{i=1}^{n} x - \hat{\zeta}_i$$

In step 2, faster algorithms than general purpose SVD can be applied, see [68].

While HTLS does not fully exploit the structure present in the signal matrix $\boldsymbol{X}$ and is therefore clearly suboptimal, it still performs very well, usually with comparable accuracy to classical methods, such as Burg's algorithm.

4.6.2. Exact AR Modelling

Exact AR modelling (name taken from [36], also called Toeplitz structured total least squares — TSTLS) employs STLN approach (to be discussed in section 5.4 (p. 41)) for identification of input-less AROE systems (as (45) with $b=0$ plus (55)). It converts the identification task into a constrained optimisation problem shown below. Again, an application on the autocorrelation sequence will be shown, while a direct application on the input sequence is also possible [36].

The constrained optimisation problem is as follows: Given the noisy autocorrelation sequence $\{\hat{C}_y(k)\}_{t=0}^{T}$ and the model order n, find a sequence $\{C_y^*(k)\}_{t=0}^{T}$ and parameter vector $\boldsymbol{a}^*$ such that

$$\begin{aligned} \left[\{C_y^*(k)\}_{t=0}^{T};\boldsymbol{a}^*\right] &= \operatorname*{arg\,min}_{\left[\{C_y(k)\}_{t=0}^{T};\,\boldsymbol{a}\right]} \sum_{k=0}^{T}\left(\hat{C}_y(k)-C_y(k)\right)^2 \\ \text{while}\quad C_y(k) &= \sum_{i=1}^{n} a_i C_y(k-i) \quad \forall\, k=n,n+1,\dots,T \end{aligned} \tag{63}$$

If the noise $\{\tilde{C}_y(k)\}$ is a white Gaussian noise, the optimal solution of (63) is the *maximum-likelihood* (ML) estimate of $\{C_y(k)\}_{t=0}^{T}$ and $\boldsymbol{a}$. This makes the method very attractive. The exact AR modelling method is capable of retrieving AR model parameters even from a short and noisy signal sample and is, in terms of accuracy, usually superior to all the other methods presented in this chapter.

See section 5.4 (p. 41) for an overview of available methods for solving (63). The author found the *weighting method* to be the fastest and most easily applicable. Unfortunately it still typically needs thousands of iterations to arrive at an accuracy significantly higher than that of alternative methods. This makes the TSTLS method rather unsuitable for real-time processing. The same conclusion was obtained in [36]. For algorithm details, see section 6.4.4 (p. 47).

5. Total Least Squares

TLS represents a technique that synthetizes statistical and numerical methodologies for solving problems in many application areas.

— Gene H. Golub, Stanford University

Total Least Squares (TLS) approach is an important tool for state-space identification methods discussed in the preceding chapter as well as for many other tasks. It has been developed since the last century, known as *errors-in-variables* approach. However, it has not become widely used until the advent of cheap computing power. TLS allows to extract more precise information out of inaccurately measured data. For an overview, see [65].

5.1. Ordinary Linear Least Squares

The least-squares procedure for solving overdetermined set of linear equations $\boldsymbol{Ax}\approx\boldsymbol{b}$ (usually attributed to Johann Carl Friedrich Gauss) has been long used for finding a "best" approximate solution $\hat{\boldsymbol{x}}$ from noisy observations $\boldsymbol{b}$ by finding a correction $\Delta\boldsymbol{b}$ such that

$$\|\Delta\boldsymbol{b}\| \stackrel{!}{=} \min \quad \text{while} \quad \exists\,\hat{\boldsymbol{x}}\,;\ \boldsymbol{A}\hat{\boldsymbol{x}} = \boldsymbol{b}+\Delta\boldsymbol{b} \tag{64}$$

However, in many cases, for example in the system identification context, such as (53), the model (64) is *no longer* appropriate as not only $\boldsymbol{b}$ but $\boldsymbol{A}$ too, is noisy. The least-squares solution

$$\hat{\boldsymbol{x}} = \boldsymbol{A}^{\diamond\,-1}_{n\times n}\,(\boldsymbol{b}+\Delta\boldsymbol{b})^{\diamond}_{n\times 1} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b} \tag{65}$$

(diamonds denoting cutting out superfluous rows)

then becomes statistically inefficient and asymptotically *underbiased* estimate of $\boldsymbol{x}$, see [69].

5.2. Total Least Squares

The *total least squares* (TLS) approach assumes that *both* $\boldsymbol{A}$ and $\boldsymbol{b}$ are noisy and tries to find corrections $\Delta\boldsymbol{A}$ and $\Delta\boldsymbol{b}$ such that

$$\left\|\,[\Delta\boldsymbol{A};\Delta\boldsymbol{b}]\,\right\|_F \stackrel{!}{=} \min \quad \text{while} \quad \exists\,\hat{\boldsymbol{x}}\,;\ (\boldsymbol{A}+\Delta\boldsymbol{A})\,\hat{\boldsymbol{x}} = \boldsymbol{b}+\Delta\boldsymbol{b} \tag{66}$$

If the minimum exists, the problem is called *generic* and any $\hat{x}$ satisfying the condition (66) is called a total least square (TLS) solution of $\boldsymbol{Ax}\approx\boldsymbol{b}$. If the solution is not unique, the *minimum-norm* solution is usually chosen. The solution of a *non-generic* TLS problem can be defined by imposing additional constraints [69] but will not be discussed here.

TLS solution usually gives higher accuracy than the corresponding LS solution when the model is appropriate, i.e., all elements of $\boldsymbol{A}$ and $\boldsymbol{b}$ are corrupted by independent identically distributed (i.i.d.) white noise. It can be proved that under this assumption the TLS solution is a consistent estimator of $\boldsymbol{x}$, unlike the LS solution [65]. Unfortunately, TLS is rather sensitive to the presence of outliers in data. Using different and more robust norms (such as L_2 or L_∞) for measuring the perturbation in (66) might be recommended in these cases [65].

5.2.1. TLS solution

The generic TLS problem solution can be found as follows. First, calculate a SVD decomposition

$$[\boldsymbol{A};\boldsymbol{b}]_{m\times(n+1)} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T = \boldsymbol{U}\begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_{n+1} \end{bmatrix}\boldsymbol{V}^T \tag{67}$$

where $\boldsymbol{U}_{m\times(n+1)}$ and $\boldsymbol{V}_{(n+1)\times(n+1)}$ are orthogonal and $\boldsymbol{S}_{(n+1)\times(n+1)}$ is diagonal with its diagonal components σ (called *singular values*) sorted in non-ascending order ($\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{n+1}$). An algorithm for calculating SVD can be found in [50]. Assuming the singular values are distinct (which is almost always true for data from measurements), the TLS solution is given by

$$\hat{\boldsymbol{x}} = -\begin{bmatrix} v_{1,(n+1)} \\ v_{2,(n+1)} \\ \vdots \\ v_{n,(n+1)} \end{bmatrix} \Big/ v_{(n+1),(n+1)} \tag{68}$$

where $v_{i,(n+1)}$ are components of the last column of V. If the singular values are not distinct, (68) still gives a TLS solution but it may not be the minimum-norm solution. See the original algorithm in [69],[65] for an improvement.

If $v_{(n+1),(n+1)} = 0$ the problem is *non-generic* and no solution of (66) exists. This may be caused by incompatible equations or redundant variables.

Alternatively, by analysing (66), it can be seen that the TLS approach seeks a *minimum effort* (as measured by Frobenius norm) rank n approximation of the matrix $[\boldsymbol{A};\boldsymbol{b}]$. Following from the Eckart-Young-Mirsky *matrix approximation theorem* [65], this can be acomplished simply by zeroing the smallest singular value

$$[\boldsymbol{A}+\Delta\boldsymbol{A};\boldsymbol{b}+\Delta\boldsymbol{b}] = \boldsymbol{U}\begin{bmatrix} \sigma_1 & & & & \\ & \sigma_2 & & & \\ & & \ddots & & \\ & & & \sigma_n & \\ & & & & 0 \end{bmatrix}\boldsymbol{V}^T \tag{69}$$

The TLS solution can be then found as

$$\hat{\boldsymbol{x}} = (\boldsymbol{A}+\Delta\boldsymbol{A})^{\diamond\,-1}_{n\times n}\,(\boldsymbol{b}+\Delta\boldsymbol{b})^{\diamond}_{n\times 1} \tag{70}$$

where the matrix inversion can be calculated from the SVD decomposition, if needed. See [65] for details.

5.3. Multidimensional TLS

Unlike the ordinary LS approach for solving over-determined set of linear equations, it makes sense with TLS to consider also a multidimensional case

$$\boldsymbol{A}_{m\times n}\boldsymbol{X}_{n\times l} \approx \boldsymbol{B}_{m\times l} \tag{71}$$

Analogically to the one-dimensional case, we use SVD decomposition

$$[\boldsymbol{A};\boldsymbol{B}]_{m\times(n+l)} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T \tag{72}$$

and (under the same assumptions) the solution is given by

$$\hat{\boldsymbol{X}} = -\begin{bmatrix} v_{1,(n+1)} & \cdots & v_{1,(n+l)} \\ v_{2,(n+1)} & \cdots & v_{2,(n+l)} \\ \vdots & \ddots & \vdots \\ v_{n,(n+1)} & \cdots & v_{n,(n+l)} \end{bmatrix} \underbrace{\begin{bmatrix} v_{(n+1),(n+1)} & \cdots & v_{(n+1),(n+l)} \\ v_{(n+2),(n+1)} & \cdots & v_{(n+1),(n+l)} \\ \vdots & \ldots & \vdots \\ v_{(n+l),(n+1)} & \cdots & v_{(n+1),(n+l)} \end{bmatrix}^{-1}}_{\boldsymbol{\Gamma}} \tag{73}$$

Again, if $\boldsymbol{\Gamma}$ is singular, the problem is *nongeneric* and is probably not apt for linear modelling. See [65],[69] for detailed description of the algorithm.

5.4. Introducing Structure into TLS

While using TLS solution as a direct replacement in (53) or (54) works, it gives only a moderate or small gain in accuracy. The reason is that the corrections $\Delta\boldsymbol{A}$, $\Delta\boldsymbol{b}$ in the TLS approach do not take into account the inherent *structure* of the matrices of the equation set originating from the underlying problem. In (53), for example, it only makes sense if $\boldsymbol{A}+\Delta\boldsymbol{A}$ is Toeplitz. Many formulations of this problem have been made [37],[66],[67] — *Structured TLS* (STLS), *Structured Total Least Norm* (STLN), *Constrained TLS* (CTLS). All of them describe the same problem but some might be more appropriate than others for a specific application.

5.4.1. Structured Total Least Norm

The *Structured Total Least Norm* problem is formulated as follows. (See [37],[67] for alternative definitions.) Let $\boldsymbol{A}$, $\boldsymbol{B}$ be matrices with a known structure, i.e., some elements are guaranteed to be equal. The matrices are defined by means of a vector $\boldsymbol{\alpha} \in \mathcal{R}^p$ and a "copying" operation ϖ, such that $[\boldsymbol{A};\boldsymbol{B}] = \varpi(\boldsymbol{\alpha})$ is a bijection. The task is to find a correction $\Delta\boldsymbol{\alpha}$ so that

$$\|\Delta\boldsymbol{\alpha}\| \stackrel{!}{=} \min \quad \text{while} \quad \begin{array}{l} \exists\,\hat{\boldsymbol{X}}\,;\ (\boldsymbol{A}+\Delta\boldsymbol{A})\,\hat{\boldsymbol{X}} = \boldsymbol{B}+\Delta\boldsymbol{B} \\ [\boldsymbol{A}+\Delta\boldsymbol{A};\boldsymbol{B}+\Delta\boldsymbol{B}] = \varpi\left(\boldsymbol{\alpha}+\Delta\boldsymbol{\alpha}\right) \end{array} \tag{74}$$

Any norm can be used to measure the magnitude of $\Delta\boldsymbol{\alpha}$, though usually the quadratic norm is used.

Generally, a STLN problem must be solved iteratively. Due to its high dimensionality and high nonlinearity, the convergence is typically *very* slow. A good starting guess (e.g., by an alternative method) can speed up the process significantly; a bad starting guess can get the algorithm stuck in a local minimum.

5.4.2. STLN solution by Lagrange method

The STLN task (74) is an equality constrained quadratic minimisation problem. It can be converted into an unconstrained minimisation-maximisation problem using *Lagrange multipliers* by defining a new criterion

$$L = \|\Delta\boldsymbol{\alpha}\| + \boldsymbol{\lambda}^T \Upsilon \left(\boldsymbol{B} + \Delta\boldsymbol{B} - (\boldsymbol{A} + \Delta\boldsymbol{A})\,\hat{\boldsymbol{X}} \right) \tag{75}$$

where the operator Υ rearranges the residual error matrix into a vector. This minimisation-maximisation problem can be in turn converted into a task of solving a set of nonlinear equations

$$\frac{\partial L}{\alpha_i} = 0\ ,\ \forall i \qquad \frac{\partial L}{\lambda_j} = 0\ ,\ \forall j \qquad \text{denoted } \boldsymbol{F}(\boldsymbol{\alpha}, \boldsymbol{\lambda}) = \boldsymbol{0} \tag{76}$$

The equations (76) can be solved by *multidimensional Newton method* [50]. Difficulties were encountered in obtaining a reasonable initial estimate of $\boldsymbol{\lambda}$ but it was found that zero can be used safely. Unfortunately, as the initial estimate is almost always too far from the solution, the algorithm converges only *linearly* with the exception of last few steps. Care must be taken to adapt the step size properly. In [50] it is recommended that the search in each step be terminated immediately after finding a point when $\|\boldsymbol{F}\|$ is smaller than the last value. In contrary to this, faster convergence was experienced for this application when the search was carried out completely, until the minimum along a search vector was found. Otherwise, the steps were sometimes too small.

5.4.3. STLN Solution by Weighting Method

Weighting method [67],[35] transforms (74) into an approximately equivalent unconstrained problem

$$\left\| \ \Delta\boldsymbol{\alpha}\ ;\ \Omega\,\Upsilon \left(\boldsymbol{B} + \Delta\boldsymbol{B} - (\boldsymbol{A} + \Delta\boldsymbol{A})\,\hat{\boldsymbol{X}} \right) \right\| \overset{!}{=} \min \tag{77}$$

where Ω is a suitably large number. The weighting method is not very accurate because increasing Ω makes the problem (77) ill-conditioned [35]. The problem (77) can be solved (among others) by a *conjugate gradient* multidimensional minimisation method [50], also described in section 6.4.4 (p. 47).

5.4.4. STLN Solution by Linearization

Iterative algorithms described in [35] solve (74) by linearising the constraints and solving the resulting linearly constrained *quadratic programming* (QP) (see [73],[72]) problem in each step. Using modified criterion based on Karush-Kuhn-Tucker equations, superlinear convergence is attained in the vicinity of the minima. As the algorithm operates on matrices in their expanded form, it has very high memory and time consumption even for moderately sized problems and is therefore not applicable for the task at hand. As presented in its simple form, the algorithm can diverge and precautions must be taken.

6. Algorithm Overview

You know my methods. Apply them, and it will be instructive to compare results.
— Arthur C. Doyle, The Sign of the Four

This chapter describes the principal ideas of the new speech enhancement algorithm developed as a part of this diploma work. The actual implementation is the topic of the following chapter.

6.1. Parameters

The algorithm contains many tunable parameters and alternative methods. The parameter values suggested here work reasonably well for the specific signals used — linearly sampled at 8000 Hz with 16 bit precision consisting of an artificial mixture of speech (isolated digits) and noise recorded inside a steadily running car with average SNR of voiced segments around 0 dB. For other signals and/or other preferences, different parameters may give better results; some hints will be provided on their choice. The implementation was designed to make experiments easy. In this chapter, tunable parameters are marked by ♠. The kalmse program has command line options to change them (see section 7.2.3 (p. 52)). Other parameters must be changed in the source code.

6.2. Block Processing

The principal reason for the design decision of processing the input *frame-by-frame* is that no reliable recurrent noise parameter estimating algorithm is known to the author. Furthermore, using future data *improves* the estimate quality. Finally, very efficient algorithms based on FFT can be used.

The main disadvantage of block processing is that inter-frame parameter changes often cause artificially sounding artifacts. Additionally, as the stationarity regions in speech vary in length, fixed-length segmentation is clearly suboptimal.

6.3. Basic Structure

Basic structure of the algorithm is depicted in *figure 4*. It assumes the sound generating system as described in section 2.1 (p. 12).

Before any further processing, the input signal is *normalised* to have zero mean and unit maximum amplitude. When the speech enhancement is finished, the original scale of the signal is restored.

The input signal $y'_\times$ is cut into overlapping *windows* (also called *frames*) $y_\times$. Note that for the sake of notational simplicity, the symbol $y_\times$ was sometimes used for $y'_\times$ in earlier chapters. Given the window size $T = 256^\spadesuit$ and the window overlap factor $s = 4^\spadesuit$, the contents of k^{th} window is

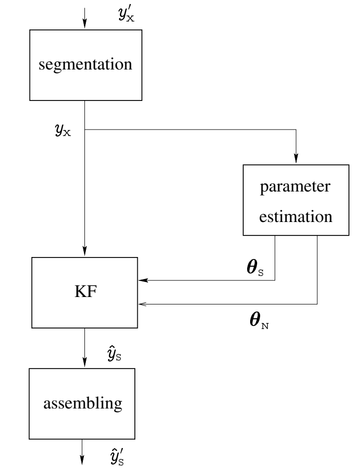


Figure 4. *Basic structure of the algorithm*

$$\left\{y^k_X(t)\right\}_{t=0}^{T-1} = \left\{y'_X(t)\right\}_{t=kT/s}^{kT/s+T-1} \tag{78}$$

In accordance with [30], the only sensible values of s were found to be 2 and 4. Parameters $\boldsymbol{\theta}_S$ resp. $\boldsymbol{\theta}_N$ of the AR models of speech resp. noise are estimated. The parameters remain *constant* during each frame. Kalman filter is used to find an estimate $\hat{y}_S$ of the speech in a given window. Finally, estimates $\hat{y}_S$ from each s adjacent overlapping windows are assembled to produce an output $\hat{y}'_S$ as follows. First, each frame is multiplied by a weight function $w(t)$.

$$\breve{y}^k_S(t) = w(t)\hat{y}^k_S(t) \qquad \text{for} \quad t = 0, \ldots, T-1 \tag{79}$$

Hanning window is used, i.e.,

$$w(t) = 1 - \cos(2\pi(t+1)/(T+1)) \tag{80}$$

In the end the weighted frames are added together.

$$\{\hat{y}_S(t)'\}_{t=0}^{\infty} = \sum_{k=0}^{\infty} \frac{1}{s} \{ \underbrace{0, 0, \ldots, 0}_{kT/s} \{\breve{y}^k_S(t)\}\ 0, 0, \ldots \} \tag{81}$$

6.4. Parameter Estimation

The structure of the parameter estimation can be seen in *figure 5*.

- **(1)** — The power spectrum P_X of the frame is estimated.
- **(2)** — The noise spectrum P_N is estimated from $\hat{P}_X$.
- **(3)** — The speech power spectrum estimate $\hat{P}_S$ is calculated from $\hat{P}_X$ and $\hat{P}_N$.
- **(4)** — Finally, AR parameters of speech and noise are retrieved from the spectra.

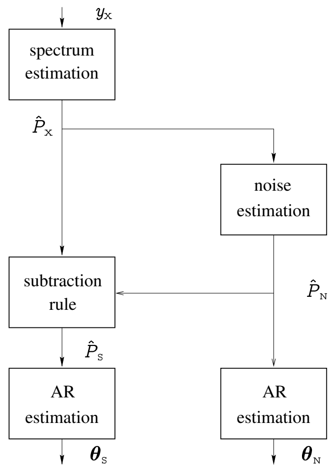


Figure 5. *Parameter estimation*

6.4.1. Frame Spectrum Estimation

Two spectrum estimation methods are available♠ — periodogram and MEM estimate, see section 4.2.4 (p. 31).

Under high to moderate SNR conditions ($\gtrsim 5\,\mathrm{dB}$), MEM estimate is superior to the periodogram, providing an estimate with much less variance. For lower SNR the systematic error (caused by imposing an incorrect AR model on the data) makes it no longer viable.

Either estimate is smoothed using (43), which effectively amounts to the Welch method [25],[12] of spectrum estimation. In fact, two smoothed estimates are calculated, using smoothing factors $\alpha_S = 0.75$♠ resp. $\alpha_N = 0.9$♠ to be used in speech and noise spectrum estimation respectively. Much less smoothing is necessary for the MEM estimate as compared with the periodogram. However, it seems to improve its stability. More smoothing is possible for the purpose of noise estimation (governed by α_N) while α_S must allow for sufficiently fast tracking of speech changes.

6.4.2. Frame Noise Estimation

A practically unchanged spectral minimum tracking based algorithm by *Martin* is used for noise spectrum estimation [40]. The principal idea is to store smoothed frame power spectra $\overline{P}^k_X$ (smoothing factor α_N) of last M frames in a circular buffer. The noise estimate is calculated as

$$\check{P}_{\rm N}^k = \gamma \min_{l=0}^{M-1} \overline{P}_{\rm X}^{k-l}$$
$$\hat{P}_{\rm N}^k = \beta \hat{P}_{\rm N}^{k-1} + (1-\beta)\check{P}_{\rm N}^k \tag{82}$$

(The calculation is done for each frequency bin separately.)

where the noise bias compensation factor γ must be determined experimentally [40] — higher γ results in higher noise suppression but also higher speech distortion. The buffer length M should represent a sufficiently long period to bridge the longest conceivable speech activity, yet short enough for the noise to remain approximately stationary — 0.5∼2.5 s is recommended. The secondary smoothing factor β helps to reduce short-term variations of the noise estimate, its value is not critical. Currently, $M = 64^♠$, $\gamma = 1.5^♠$, $\beta = 0.9^♠$ is used.

To reduce the computational complexity, the implementation does not in fact use a circular buffer of the length M. Instead, a smaller circular FIFO buffer b of length M_1 is used, denoting $M_1 M_2 = M$. An auxiliary variable P_a keeps track of the current minima of the $\overline{P}_{\rm X}$ spectra for each frequency bin. Every M_2 frames, the minima of the last M_2 frames stored in P_a is copied into b and P_a is reset. Finally, the estimate $\check{P}_{\rm N}^k$ is computed as

$$\check{P}_{\rm N}^k = \gamma \min \left\{ \min_{l=0}^{M_1-1} P_b^l \, , \; P_a \right\} \tag{83}$$

where P_b denotes the contents of the buffer b. Consequently, the operation count is reduced approximately by a factor of M_2^2 at the expense of an equivalent length M in (82) effectively varying between $M_1 M_2$ and $M_1 M_2 + M_2$. Hence, M_1 should not be smaller than about 5. As default $M_1 = 8^♠$, $M_2 = 8^♠$.

The noise spectrum estimation algorithm by *Doblinger* [15] was found to give very similar but slightly worse results for given signals. Moreover, it is more complex. Nevertheless, its implementation is available as a part of the ssub program.

6.4.3. Subtraction Rule

Several subtraction rules are available ♠ — simple power subtraction (1), power spectral subtraction with half wave rectification (1,9) or full wave rectification (1,10), Wiener filtering (16), and modified power spectral subtraction (8) with half wave rectification. The oversubtraction factor δ for the last method is calculated [40] from an estimated band SNR

$$\widehat{\rm SNR}(\omega) = 10 \log_{10} \min \left\{ q_{\rm H} \, , \; \max \left\{ q_{\rm L} \, , \; \frac{\hat{P}_{\rm X}(\omega) - \hat{P}_{\rm N}(\omega)}{\hat{P}_{\rm N}(\omega)} \right\} \right\} \tag{84}$$

Currently, $q_{\rm L} = 1$ and $q_{\rm H} = 100$. This automatically limits the SNR estimate into the $0 \sim 20$ dB range. The resulting δ is a linear interpolation

$$\delta = \delta_{\rm L} + (\delta_{\rm H} - \delta_{\rm L}) \frac{\widehat{\rm SNR} - 10 \log_{10} q_{\rm L}}{10 \log_{10} q_{\rm H} - 10 \log_{10} q_{\rm L}} \tag{85}$$

where $\delta_{\rm L} = 1^♠$ and $\delta_{\rm H} = 4^♠$ Using higher δ improves noise suppression at the expense of higher speech distortion. It is argued in [23] that smaller values such as $\delta \approx 0.5 \sim 0.9$, depending on other parameters, should be used.

The modified power spectral subtraction method is usually superior to the others and was made default. Alternatively, *Ephraim-Malah* gain factors described in [15] could have been used. However, according to the experiments carried out by the author, the difference in results was rather small, while the presented method is quicker and needs less memory.

6.4.4. Frame AR Parameter Estimation

The estimated noise and speech spectra $\hat{P}_{\rm N}$, $\hat{P}_{\rm S}$ are first converted to autocorrelation sequences $\hat{C}_{\rm N}$, $\hat{C}_{\rm S}$ of lengths $T/2$ using (42). To proceed, two methods are available ♠ — Burg's and exact AR modelling.

As a default, *Burg*'s algorithm (described in section 4.4.5 (p. 35)) is used to extract the AR parameters $\boldsymbol{a}$ of an autocorrelation sequence to be used as estimates of the AR parameters of the original process. See section 4.3.3 (p. 33) for justification. Subsequently (56) is used to estimate a parameter b , as defined by (45). Although the Burg's method is theoretically suboptimal, it proved to be fast and to perform better on the average than methods mentioned in the section 4.4 (p. 33).

Alternatively, *exact AR modelling* method is used. See section 4.6.2 (p. 38) for a description. The minimisation problem resulting from the weighting method (section 5.4.3 (p. 42)) is solved by conjugate gradient multidimensional minimisation [50] as follows. The criterion to be minimised given $\hat{C}_y$, n and T (see (63) and (77)) is

$$J = \sum_{k=0}^{T} \left(\hat{C}_y(k) - C_y(k) \right)^2 + \Omega \sum_{k=n+1}^{T} \xi_k^2$$
$$\text{where} \quad \xi_k = C_y(k) - \sum_{i=1}^{n} a_i C_y(k-i) \tag{86}$$

The ξ is called an *equation error*. The weight $\Omega = 100$ is a compromise — too small a weight results in a limited precision because the constraints are not met, too big a weight worsens the numerical conditioning of the problem which prolongs the calculation and may cause instability. Sensible values for Ω are approximately $1 \sim 10^7$. The gradient of $J(\{C_y\}, \boldsymbol{a})$ can be calculated as follows

$$\frac{\partial J}{\partial a_i} = -2\Omega \sum_{k=n}^{T} \xi_k C_y(k-i)$$
$$\frac{\partial J}{\partial C_y(k)} = 2(C_y(k) - \hat{C}_y(k)) + 2\Omega \left(\xi_k - \sum_{\substack{i=1 \\ k+i \le T}}^{T} a_i \xi_{i+k} \right) \tag{87}$$

The minimisation of (86) is performed sequentially starting from $C_y = \hat{C}_y$ and $\boldsymbol{a} = \boldsymbol{a}_{\rm Burg}$. Denoting the parameter vector as $\boldsymbol{x} = [C_y(0), \ldots, C_y(T), \boldsymbol{a}^T]^T$, the starting value will be called $\boldsymbol{x}_0$. Two other auxiliary vectors are defined and initialized as $\boldsymbol{g}_0 = \boldsymbol{h}_0 = -\nabla_{\boldsymbol{x}} J$.

The iterative step consists of [50]

$$\begin{aligned}
\lambda &= \arg\min_{\lambda} J(\boldsymbol{x}_i + \lambda \boldsymbol{h}_i) \\
\boldsymbol{x}_{i+1} &= \boldsymbol{x}_i + \lambda \boldsymbol{h}_i \\
\boldsymbol{g}_{i+1} &= -\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_{i+1}) \\
\gamma_i &= \frac{(\boldsymbol{g}_{i+1} - \boldsymbol{g}_i)^T \boldsymbol{g}_{i+1}}{\boldsymbol{g}_i^T \boldsymbol{g}_i} \\
\boldsymbol{h}_{i+1} &= \boldsymbol{g}_{i+1} + \gamma_i \boldsymbol{h}_i
\end{aligned} \tag{88}$$

The iteration stops when either $(J(\boldsymbol{x}_i) - J(\boldsymbol{x}_{i+1})) \;/\; J(\boldsymbol{x}_i) \leq 10^{-10}$ or after $100^{\spadesuit}$ iterations. This makes the method only a few times slower than Burg's method with comparable accuracy. To attain significantly higher precision, $10^3 \sim 10^5$ iterations are needed, which is mostly too slow to be usable.

The one-dimensional minimisation of J for λ in (88) is performed by the following parabolic interpolation algorithm, inspired by [50]. As a shortcut, let us denote $f(\lambda) = J(\boldsymbol{x}_i + \lambda \boldsymbol{h}_i)$.

- **(1)** — If the magnitude of $\boldsymbol{h}_i$ exceeds 10, it is scaled down to this magnitude. This is admittedly a hack and while it is not strictly needed, it speeds up the process if $\boldsymbol{h}_i$ is excessively large.
- **(2)** — Full step ($\lambda_1 = 1$) is taken. If $f(\lambda_1) < f(0)$, progressively larger steps are taken as long as $f(\lambda_i) < f(\lambda_{i-1})$. If $f(\lambda_1) > f(0)$, progressively shorter steps are taken until $f(\lambda_i) < f(0)$. As a result, the minimum of f is *bracketted* between three points $\lambda_{\rm L} < \lambda_{\rm M} < \lambda_{\rm H}$ such that the value of $f(\lambda_{\rm M})$ in the middle point is smaller than both $f(\lambda_{\rm L})$ and $f(\lambda_{\rm H})$.
- **(3)** — Parabolic approximation fitting $f(\lambda_{\rm L})$, $f(\lambda_{\rm M})$, and $f(\lambda_{\rm H})$ is attempted in order to find a new λ' closer to the minimum. The new λ' is forced into the 'bracket' ($\lambda_{\rm L} < \lambda' < \lambda_{\rm H}$), while care is taken not to evaluate f too close to the endpoints, to avoid numerical problems. If the interpolation fails because of colinearity, classical *golden ratio* search step is performed to find alternative λ'.
- **(4)** — The 'bracket' $\lambda_{\rm L} < \lambda_{\rm M} < \lambda_{\rm H}$ is updated according to the values of λ' and $f(\lambda')$, so that the bracketing is preserved while the bracket size $\lambda_{\rm H} - \lambda_{\rm L}$ decreases. The procedure beginning with point (3) is repeated as long as the absolute and relative bracket size ($\lambda_{\rm H} - \lambda_{\rm L}$ resp. $(\lambda_{\rm H} - \lambda_{\rm L})/\lambda_{\rm M}$) are greater than 10^{-20} resp. 10^{-6} and the number of iterations does not exceed 50.

This algorithm converges mostly super-linearly, usually in $10 \sim 20$ iterations. If used for different purposes when increased robustness is desired, *Brent's* method can be used at the expense of increased complexity [50].

6.5. Frame Filtering

In accordance with sections 2.1 (p. 12), 2.3.1 (p. 13) and 2.4.1 (p. 13), the following *double AR model* is assumed

$$\begin{aligned}
y_{\rm N}(t) &= \sum_{i=1}^{n_{\rm N}} a_{{\rm N}_i} y_{\rm N}(t-i) + b v_{\rm N}(t) \\
y_{\rm S}(t) &= \sum_{i=1}^{n_{\rm S}} a_{{\rm S}_i} y_{\rm S}(t-i) + b v_{\rm S}(t) \\
y_{\rm X}(y) &= y_{\rm N}(t) + y_{\rm S}(t) + \sqrt{r}\, w(t)
\end{aligned} \tag{89}$$

where $v_{\rm N}$, $v_{\rm S}$ and w are mutually uncorrelated zero-mean Gaussian white noises with unit variances. The variance of the measurement noise, denoted r, is, in the given setting, mainly caused by quantisation noise and modelling errors. It is advisable to set r a little higher because of its beneficial regularising effects. Too high r on the other hand makes the resulting estimate too conservative. Value of $r = 10^{-6\spadesuit}$ has been found to be a good compromise. The orders $n_{\rm N} = 8^{\spadesuit}$, $n_{\rm S} = 8^{\spadesuit}$ of noise resp. speech generating systems are used. Somewhat higher values might improve the performance at the expense of increased calculation, however for $n_{\rm S/N} \gtrsim 16$ the negative effects of identification errors often outweight the increased model precision.

In order to be able to use the Kalman filter, the model (89) is converted to the state-space form (20) by combining two models (52). The resulting $n = n_{\rm N} + n_{\rm S}$ dimensional state-space model with no deterministic input is

$$\begin{aligned}
\boldsymbol{x}(t+1) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{v}(t) \\
y_{\rm X}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + w'(t)
\end{aligned} \tag{90}$$

where

$$\begin{aligned}
\boldsymbol{x}(t) &= [\, y_{\rm N}(t - n_{\rm N} + 1), \dots, y_{\rm N}(t), y_{\rm S}(t - n_{\rm S} + 1), \dots, y_{\rm S}(t)\,]^T \\
\boldsymbol{C} &= [\underbrace{0, 0, \dots, 1}_{n_{\rm N}}, \underbrace{0, 0, \dots, 1}_{n_{\rm S}}] \\
\boldsymbol{A} &= \begin{bmatrix}
0 & 1 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \\
0 & 0 & 1 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \dots & 1 & 0 & 0 & 0 & \dots & 0 \\
\hat{a}_{{\rm N}_{n_{\rm N}}} & \hat{a}_{{\rm N}_{(n_{\rm N}-1)}} & \hat{a}_{{\rm N}_{(n_{\rm N}-2)}} & \dots & \hat{a}_{{\rm N}_1} & 0 & 0 & 0 & \dots & 0 \\
0 & 0 & 0 & \dots & 0 & 0 & 1 & 0 & \dots & 0 \\
0 & 0 & 0 & \dots & 0 & 0 & 0 & 1 & \dots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 1 \\
0 & 0 & 0 & \dots & 0 & \hat{a}_{{\rm S}_{n_{\rm S}}} & \hat{a}_{{\rm S}_{(n_{\rm S}-1)}} & \hat{a}_{{\rm S}_{(n_{\rm S}-2)}} & \dots & \hat{a}_{{\rm S}_1}
\end{bmatrix} \\
\boldsymbol{Q} &= \operatorname{cov} \boldsymbol{v} = \begin{bmatrix}
\underbrace{\begin{matrix} 0 & & & \\ & \ddots & & \\ & & 0 & \\ & & & b_{\rm N}^2 \end{matrix}}_{n_{\rm N}} & \\
& \underbrace{\begin{matrix} 0 & & & \\ & \ddots & & \\ & & 0 & \\ & & & b_{\rm S}^2 \end{matrix}}_{n_{\rm S}}
\end{bmatrix} \\
\boldsymbol{R} &= \operatorname{cov} w' = r
\end{aligned}$$

Given the observed signal values in the frame $y_{\rm X}$, fast bidirectional Kalman smoothing algorithm described in section 3.2.3 (p. 25) is used to extract the speech estimate $\hat{y}_{\rm S}$. The starting value $\hat{\boldsymbol{x}}(1|0)$ is set to zero. The initial covariance matrix is estimated as

$$\boldsymbol{P}(1|0) = \operatorname{cov} \boldsymbol{x} = \tag{91}$$

$$\begin{bmatrix}
\hat{C}_{\mathrm{N}}(0) & \hat{C}_{\mathrm{N}}(1) & \dots & \hat{C}_{\mathrm{N}}(n_{\mathrm{N}}-1) & & & & \\
\hat{C}_{\mathrm{N}}(1) & \hat{C}_{\mathrm{N}}(0) & \dots & \hat{C}_{\mathrm{N}}(n_{\mathrm{N}}-2) & & & & \\
\vdots & \vdots & \ddots & \vdots & & & & \\
\hat{C}_{\mathrm{N}}(n_{\mathrm{N}}-1) & \hat{C}_{\mathrm{N}}(n_{\mathrm{N}}-2) & \dots & \hat{C}_{\mathrm{N}}(0) & & & & \\
& & & & \hat{C}_{\mathrm{S}}(0) & \hat{C}_{\mathrm{S}}(1) & \dots & \hat{C}_{\mathrm{S}}(n_{\mathrm{S}}-1) \\
& & & & \hat{C}_{\mathrm{S}}(1) & \hat{C}_{\mathrm{S}}(0) & \dots & \hat{C}_{\mathrm{S}}(n_{\mathrm{S}}-2) \\
& & & & \vdots & \vdots & \ddots & \vdots \\
& & & & \hat{C}_{\mathrm{S}}(n_{\mathrm{S}}-1) & \hat{C}_{\mathrm{S}}(n_{\mathrm{S}}-2) & \dots & \hat{C}_{\mathrm{S}}(0)
\end{bmatrix}$$

which would be exact, if the autocorrelation function estimates were equal to the true values. Because of the structure of the system matrices, many simplifications are possible so that both the forward and backward runs take only $\mathcal{O}(Tn^2)$ operations.

As described in section 3.2.3 (p. 25), this Kalman smoothing algorithm is fast but not very robust. To overcome the problem, following measures are taken. If $\max \hat{y}_{\mathrm{S}} > 1.5 \max y_{\mathrm{X}}$, divergence is assumed. The divergence criterion seems to perform very well, divergence almost never escapes undetected, while seldom causing false alarms. If the condition above is triggered, the filtering is redone using the following alternative robust smoothing method.

New $\boldsymbol{P}(1|0)$ is calculated by iteratively solving Lyapunov equation $\boldsymbol{P} = \boldsymbol{APA}^T + \boldsymbol{Q}$, see section 4.3.2 (p. 32). The iteration is stopped when the L_∞ norm of the difference between successive iterations drops below $10^{-6}\,\|\boldsymbol{P}\|_\infty$. If $\boldsymbol{P}$ fails again to be positive- definite, the filtering is skipped, i.e., $\hat{y}_{\mathrm{S}} = y_{\mathrm{X}}$. Otherwise, *square-root* Kalman smoothing is started (described in section 3.3.3 (p. 27)), using Cholesky factorisation for triangularisation. If the smoothing fails, i.e., if the triangularisation cannot be performed for numerical reasons, the filtering is also skipped. However, this happens very rarely, unless the problem is truly ill-conditioned. Unfortunately, the square-root smoothing cannot be optimised in the same way as the previously mentioned fast Kalman smoothing. Consequently, square-root smoothing tends to be about $2n$ times slower.

While the default procedure described above works well, providing maximum speed most of the time and additional accuracy when needed, the user has an additional possibility♠ to perform only *forward* filtering (which speeds up the process approximately twice), or to force *always* the square-root implementation.

7. Implementation

This chapter describes the software developed as a part of this diploma work. It can be found either on a diskette accompanying the original of this report or alternatively on the Internet, at http://cmp.felk.cvut.cz/~kybic/dipl. From either source, you get an archive file kalmse-1.2.tgz. Unpack the archive file using programs gzip and tar. This creates a top directory kalmse-1.2/ with several subdirectories. These contain:

- *MATLAB* scripts used for experiments, testing and prototyping.
- *C* source code for the program kalmse, implementing the speech enhancement algorithm described in the previous section. The same source can also generate functionally equivalent parallel version kalmse_pvm.
- *C* source code for other speech enhancement algorithms implemented earlier by the author and used for performance comparison as well as additional auxiliary programs for speech detection, sound file format conversion, mixing and measurements. [1]
- Sample sound files.
- TEX source of this manuscript together with all the auxiliary fonts and macros used. The compiled *Postscript* version ready to print as well as reduced version with two pages on one A4 sheet are provided separately on the Internet. The source for slides used for presentation of this work is also included.
- Other useful scripts for various conversions, testing etc.

More complete description can be found later in this chapter. For license information, refer to page 10.

7.1. Requirements

The software was developed and tested on *Unix* workstations running *Linux* and *HP-UX* but as it was written with portability in mind, it should run on other operating systems as well. *C* compiler conforming to the ANSI specification with 32 bit integers is assumed. The *MATLAB* scripts were tested with *MATLAB* version 5.1. Some scripts might need minor changes if other version of *MATLAB* or its clone octave is used. Means of viewing *Postscript* files (e.g., programs ghostscript and ghostview) is useful. Having the TEX typesetting system installed is not necessary unless you want to recompile this manual. Sound card is, of course, helpful. On a PC running *Linux*, the built-in speaker will do too, although the sound quality is obviously poor. The program sox was found useful for converting between various sound formats.

As the algorithms are rather computationally intensive (next chapter gives quantitative values) fast processor is an asset. If you have several networked computers at your disposition, it is strongly recommended to use the parallel version. For this, you need to install the *PVM* library [74]. Programs octave, ghostscript, ghostview, TEX, sox, and the PVM library are all available freely on the Internet.

[1] Some of these programs were developed jointly with Petr Pollák.

7.2. Program kalmse

The program kalmse implements the algorithm described in the previous chapter.

7.2.1. Installation

Go to the c/ subdirectory. Create Makefile using the provided Makefile.hp (for *HP-UX*) or Makefile.linux (for *Linux*) as a basis. No changes except setting a compiler name and options should be needed. Type make kalmse to compile the program kalmse or make to compile all programs at the same time. Facultatively, move the resulting executable(s) to a customary directory.

7.2.2. Sound File Format

The program kalmse, as well as other programs included, store the sound data as a sequence of 16 bit signed integers in machine representation without any header and with a default extension *bin*. The sampling frequency is assumed to be 8 kHz and the algorithm parameters are tuned for it but it is not an obligation. This format was chosen because it is the native format of the speech database used. While it is fast to process and memory-effective, it is generally *not* portable across different architectures. Scripts bin2wav resp. wav2bin in the bin/ directory translate the *bin* format to resp. from the *wav* format (readable by *MATLAB* and used in *MS Windows*) using the sox program. Based on these scripts, conversion to other formats should not be difficult.

7.2.3. Description

The program kalmse takes a sound file, performs speech enhancement on the data as described in the previous chapter and writes results into another file. When invoked without parameters, it displays usage information. (All other *C* programs follow this conventions.) The normal invocation is as follows

kalmse [options] ⟨input file⟩ ⟨output file⟩

Available options are listed below together with their acceptable and default values (in parentheses). For parameters mentioned in formulas of the previous chapter, the appropriate symbol is given. Floating point parameters are indicated by presence of a decimal point in the sample values. The rest of parameters take integer values.

–v 0∼9 (9) controls the *verbosity*. Only messages on verbosity level lower or equal to the level set will be printed. Level 1 is used for error messages, 2 for warnings, 3 for general information, 5 for progress indication and 9 for debug messages.
–nn 1∼30 (8) noise model order n_N .
–ns 1∼30 (8) speech model order n_S .
–wl 64,128,256,512 (256) window length T .
–wo 1,2,4,8 (4) window overlap factor s .
–la 1∼20 (8) number of frames for primary minimisation M_2 .
–lb 1∼20 (8) circular buffer length M_1 .
–an 0.0∼1.0 (0.9) smoothing factor α_N for noise estimation.
–as 0.0∼1.0 (0.75) smoothing factor α_S for speech estimation.
–be 0.0∼1.0 (0.9) smoothing factor β for additional smoothing of noise estimate.
–eb 0.0∼10.0 (1.5) noise estimate bias factor γ .
–mc 0.0∼10.0 (10^{-6}) relative measurement noise covariance r .
–mh 0.0∼10.0 (4) maximum oversubtraction factor δ_H .
–ml 0.0∼10.0 (1) minimum oversubtraction factor δ_L .
–sr 0∼4 (4) subtraction rule (see section 6.4.3 (p. 46)). 0 stands for simple power subtraction, 1 resp. 2 is half resp. full wave rectification, 3 is Wiener filtering and 4 modified power subtraction.
–im 0,1 (0) identification method: 0 Burg, 1 TSTLS (exact AR modelling).
–tm 0∼10^5 (100) maximum number of iterations for TSTLS, see section 6.4.4 (p. 47).

–kf 0,1,2 (1) Kalman filter type: 0 unidirectional, 1 fast bidirectional with transition to SR when needed, 2 always SR.
–fm 0,1 (0) spectrum estimation method: 0 MEM, 1 periodogram (smoothed FFT).

The implementation follows closely the description given in the previous chapter. Main source files are kalmse.c and kalmseflt.c but the program also links fft.c, burg.c, tstlsw.c, darkalm7.c and darkalm.c to perform particular computational tasks.

The program reads all the input into memory at once, performs the filtering and stores results to an output file. This permits easy preprocessing (mean removal and normalisation), minimises operation system overhead and simplifies parallel implementation, which uses the same skeleton. Given today's memory sizes, it should not be a limitation. Nevertheless, the program is modularised so that it be easy to modify to keep in memory only a few frames at a time.

All computation is done in double precision (54 bit mantissa) to avoid numerical problems. Even though for some operations single precision might be satisfactory, for identification and Kalman filtering the additional accuracy is essential. Despite of all precautions, numerical problems in the most difficult situations were still not completely eliminated. However, the program was written with robustness in mind; if a computation fails, alternative actions are taken. If all else fails, no filtering is performed for a given frame.

7.3. Parallel Version

To speed up the computation by using all available computer resources, parallel version kalmse_pvm was developed using the PVM package [74]. (This package implements a message passing paradigm across multiple architectures.) Except for the parallelism, kalmse_pvm and kalmse are completely equivalent. They are even produced from the same source code using conditional compilation. The only new command-line option added is **–pv**, to change the number of slave processes to be started; the default is 1. Usually, one slave process per processor is used.

The computation is organised using the one-master/multiple-slaves model. Master and slaves run the same executable, they are distinguished by command line options. The user-started master process creates a predefined number of slave processes, possibly on different machines. For each frame, the master performs all the calculations to obtain speech and noise spectra. It passes these spectra to an available slave, which performs identification and filtering. The results from all slaves are returned to the master, which combines them and ultimately writes the output file and stops the slaves. The scheduling policy is FIFO — new tasks are distributed to slaves in the same order as the slaves finish their previous assignments. Three tasks are kept in the input message queue of each slave lest network delays should interrupt the computation.

With the parallelisation just described, it has been observed that there is a significant speed-up for a small number of machines (details in section 8.6 (p. 62)). However, using more than about 10 hosts does not yield any additional benefit, as the master machine becomes a bottleneck. It is therefore recommended to use the fastest machine as the master.

Greater, albeit more complicated parallelisation would be possible by distributing also the spectrum estimation task between slaves. This has not been implemented so far. Moreover, it is not clear whether the additional communication burden would not cancel-out the gain.

7.3.1. Installation

PVM must be correctly installed on all your computers [74]. Edit the supplied Makefile.aimk to reflect your setup. Typing aimk should create a PVM-enabled executable kalmse_pvm in your preferred directory.

7.4. MEX Files

MEX files are compiled routines callable from *MATLAB*. Because of the compilation, they are faster than corresponding *MATLAB* scripts. The principal algorithms for kalmse were developed as follows.

- *MATLAB* script was written and tested, taking advantage of the fact that prototyping mathematical algorithms in *MATLAB* is relatively easy and fast.
- The algorithm was reimplemented in *C*.
- A wrapper function was created and MEX file was built.
- Because the *C* implementation could now be called from *MATLAB*, it was easy to compare results with the original *MATLAB* implementation and assure they match. This made debugging much easier.
- The unchanged *C* routine was incorporated into kalmse.

The wrapper files darkalm3c.c darkalmbc.c darkalmc.c darkalmsc.c lusolve.c and tstlswc.c reside in the c/ directory and a symbolic link is made into the m/ directory. Each MEX file is equivalent to a *MATLAB* function with similar name (see source code). Unfortunately, the MEX application interface has been changing with almost every version of *MATLAB*, some change to the wrapper functions might be therefore needed. Similarly, follow your documentation on how to compile MEX files. On my system (HP-UX, bash shell, *MATLAB* 5.1), the following commands did the job:

```
cd kalmse-1.2/m ; for i in *.c ; do cmex $i ; done
```

7.5. Archive Contents

This section describes rather briefly in alphanumerical order all files from the archive distributed as a part of this work. Besides, there are README files providing additional information. The principal files are set in **bold**. Conversely, files set in *slanted* type, as opposed to upright, are experimental or targeted to my specific installation. They might be interesting and inspiring to look at but might contain some deficiencies and should *not* be relied upon without prior inspection. To lesser extent, this applies to all other files and especially scripts; you are advised to skip through them at least briefly before use. Nevertheless, a significant effort has been put into making the kalmse and kalmse_pvm programs build seamlessly and work as intended. Files marked by ⋆ were not created by me, or not solely by me, although they have usually been modified. Files marked by ⋄ may not be contained in the archive and must be built from other files, mostly by invoking make in the respective directory.

README	Basic overview of the package. Most recent information.
LICENSE	License information.
bin/	Directory containing scripts and programs.
c/	Directory containg C source code.
data/	Directory containing sample sound files.
m/	Directory containg *Matlab* scripts.
pvm/	Directory devoted to auxiliary files for PVM.
tex/	TeX source code of this report and slides, other necessary files, and formatted output.
bin/README	Brief info.
bin/backdipl	Makes an archive of everything.
bin/bin2post	Convert *bin* format to the format used by the POST speech recognition system.
bin/bin2wav	Convert *bin* format to the *wav* format.
bin/calcres.tcl	Tcl script for wieghted average calculation.
bin/imarktcl	Tcl script generating index for TeX documents. Finally not used.
bin/kalmanfilter	Tcl script for testing effects of speech enhancement by the new algorithm on speech recognition.
bin/latex2tex	Awk script by S. Rogmann to render PS+LaTeX pictures from xfig usable in plain TeX documents. ⋆
bin/makenoisyall	Create all artificially mixed signals.
bin/makenoisypar	Create all artificially mixed signals for a given SNR.
bin/martinfilter	Tcl script for testing effects of speech enhancement by the Martin's algorithm on speech recognition.
bin/post2bin	Convert from the format used by the POST speech recognition system to the *bin* format.
bin/recogall	Top-level Tcl script driving the POST speech recognition system.
bin/spenh	Tcl script driving tests of speech enhancement algorithms.
bin/wav2bin	Convert *wav* format to the *bin* format.

c/README	Brief info.
c/Makefile	Link to a system specific Makefile.
c/Makefile.aimk	Link to a system specific Makefile.aimk, used by aimk to build kalmse_pvm.
c/Makefile*.hp	HP-UX specific version of Makefile*.
c/Makefile*.linux	Linux specific version of Makefile*.
c/burg.c	AR coefficients calculation using Burg's algorithm. ⋆
c/cepdet	Program implementing cepstral VAD. ⋄
c/cepdet.c	Main source code file for cepdet.
c/cepdist	Program implementing cepstral distance calculation. ⋄
c/cepdist.c	Main source code file for cepdist.
c/cepstr.c	Implementation of cepstral coefficients calculation. ⋆
c/cepstr.h	Header file for cepstr.c. ⋆
c/cholesk.c	Cholesky decomposition and backsubstitution.
c/darkalm.c	Fast bidirectional Kalman smoothing implementation.
c/darkalm3c.c	MEX wrapper for darkalm.c.
c/darkalm4.c	Two-pass Kalman smoothing using constant backward gain.
c/darkalm5.LU.c	Two-pass Kalman smoothing, LU-decomposition in each step.
c/darkalm5.c	Two-pass Kalman smoothing, Cholesky factorisation in each step.
c/darkalm6.c	SR Kalman smoothing.
c/darkalm7.c	SR Kalman smoothing. Slightly faster than darkalm6.c because most operations now work on the triangularized form.
c/darkalmbc.c	MEX wrapper for darkalm4.c.
c/darkalmc.c	MEX wrapper for darkalm5.c.
c/darkalmsc.c	MEX wrapper for darkalm7.c.
c/dobl.c	Implementation of Doblinger's speech enhancement algorithm. Part of the ssub program.
c/dobl.h	Header file for dobl.c.
c/enedet	Energy based VAD. ⋄
c/enedet.c	Source code for enedet.
c/fft.c	Two alternative FFT implementation.
c/fft.h	Header file for fft.c.
c/kalmse	Program kalmse performing Kalman filter based speech enhancement. ⋄
c/kalmse.c	Main source code file for kalmse (and kalmse_pvm).
c/kalmse.h	Header file for kalmse
c/kalmseflt.c	Implementation of speech enhancement algorithms for kalmse.
c/ludecomp.c	LU decomposition and backsubstitution.
c/lusolve.c	MEX wrapper for ludecomp.c.
c/martin.c	Implementation of Martin's speech enhancement algorithm. Part of the ssub program.
c/martin.h	Header file for martin.c.
c/mix	Program for mixing of speech and noise signals with desired SNR. ⋄
c/mix.c	Source code for mix. ⋆
c/mono2ste.c	Source code of a program mono2ste to convert two mono *bin* files into one stereo file.
c/snr	Program for SNR measurements. ⋄
c/snr.c	Source for snr.
c/so1ff.c	Implementation of two-step spectral subtraction, full-wave rectification speech enhancement algorithm. Part of ssub.
c/so1ff.h	Header file for so1ff.c.
c/ssub	Program ssub, combining several speech enhancement algorithms. Gives usage hints when started without parameters. ⋄
c/ssub.c	Main source file for ssub.
c/ste2mono.c	Source code of a program ste2mono to convert one stereo *bin* file into two mono file.
c/tstlsw1.c	Implementation of TSTLS (exact AR modelling) solution by weighting method and conjugate gradient search.
c/tstlswc.c	MEX wrapper for tstlsw1.c.
c/wav2sphr.c	Source code for a program wav2sphr, transforming *wav* files to the *sphere* format, used by some speech recognition software.
data/README	Brief info.
data/ct.wav	Car noise, 4 s.
data/p.wav	Isolated digits, 2.5 s.
data/xsig.wav	Speech plus noise, 12.5 s.
data/xsigout.wav	xsig.wav processed by kalmse.
m/README	Brief info.
m/.mexrc.sh	Script sourced by *MATLAB* when making MEX files. Usually not necessary unless you want to change e.g., compile flags.
m/a2c.m	Calculate autocorrelation sequence from AR model coefficients using (50).
m/a2imp.m	Returns impulse response of AR system.
m/a2s.m	Calculates amplitude spectrum of AR system.
m/a2step.m	Gives step response of AR system.
m/ar2tm.m	Assembles state-space transfer matrix from AR coefficients (52).
m/ard2tm.m	Assembles transfer matrix for dual AR system (90).
m/ard2tme.m	As ard2tm.m plus estimates initial covariance matrix.
m/arfr.m	Draws a frequency response of AR system.
m/arinv.m	Whitening filter, inverse to a given AR system.

m/arkalm.m	Two-pass Kalman smoothing of a signal from white-noise driven AR system corrupted by white noise.
m/arkalmr.m	Repetitive, iterative arkalm.m.
m/arxkalm.m	ARX Kalman filtering.
m/arxoekal.m	Improved arxkalm.m.
m/astab.m	Adjust given AR parameters, so that the model is stable.
m/burg.m	Burg's algorithm. ⋆
m/burg1.m	Corrected and changed burg.m, different output format.
m/c2R.m	Build Toeplitz matrix from autocorrelation coefficients.
m/c2a.m	Find AR model parameters using autocorrelation method.
m/c2ach.m	Find AR model parameters using Cayley-Hamilton method.
m/c2ache.m	As c2ach.m except uses all available autocorrelation coefficients.
m/c2achw.m	As c2ache.m except takes into account bias data.
m/c2achz.m	As c2ache.m except uses also autocorrelation at zero lag.
m/c2ps.m	Calculates power spectrum from autocorrelation.
m/chappr.m	Chi-square distribution approximation.
m/chshape.m	Matrix reorganization. ⋆
m/clhamre.m	Using repetitive autocorrelation and Cayley-Hamilton method to find AR model parameters.
m/clhamred.m	Like clhamre except provides results for all different number of iterations.
m/clhamres.m	Like clhamre except FFT is used to calculate autocorrelation.
m/corri.m	Compute autocorrelation for a signal with a reciprocal spectrum to given input signal.
m/corrn.m	Compute directly biased autocorrelation estimate of a sequence.
m/corrs.m	Use FFT to compute autocorrelation estimate.
m/corrss.m	Use smoothed periodograms for autocorrelation estimate.
m/corru.m	Like corrn except the estimate is unbiased.
m/corrw.m	Uses time warping for efficient autocorrelation estimate by FFT.
m/darkalm.m	Bidirectional two-pass Kalman smoothing for double AR model.
m/darkalm1.m	Like darkalm.m with stationary Kalman filter; constant gain in both runs.
m/darkalm2.m	Like darkalm.m but uses fast two-pass smoothing.
m/darkalm3.m	Like darkalm2.m but more optimised.
m/darkalm3c.c	Symbolic link to the MEX wrapper representing C version of darkalm3.m.
m/darkalm4.m	Like darkalm.m except constant gain is used for the backward run and prediction error is calculated.
m/darkalmbc.c	Symbolic link to the MEX wrapper representing C version of darkalm4.c.
m/darkalmc.c	Symbolic link to the MEX wrapper representing C version of darkalm.c.
m/darkalms.m	Covariance SR Kalman filtering (not smoothing).
m/darkalmsb.m	Covariance SR Kalman two-pass smoothing.
m/darkalmsc.c	Symbolic link to the MEX wrapper representing C version of darkalmsc.c.
m/darkalmu.m	Like darkalm.m but does only filtering, no smoothing.
m/dbode.m	Makes Bode plot of a given ARMA system.
m/demo1.m	Demonstrates repetitive autocorrelation AR identification method.
m/demo10.m	Kalman filtering demonstration. Covariance matrix estimated from autocorrelation.
m/demo2.m	Demonstrates Kalman filtering for signal separation.
m/demo3.m	Compares performance of several Kalman filter variants on real sound signals.
m/demo4.m	Like demo3.m, using artificial signals.
m/demo5.m	Demonstrates Kalman smoothing on real signals.
m/demo6.m	Like demo5.m except covariance matrix is estimated from autocorrelation.
m/demo7.m	Like demo6.m, using darkalms.m
m/demo8.m	Demonstrate spectral subtraction on real data.
m/demo9.m	Like demo6.m using darkalm3c.c
m/dlyap1.m	Solve Lyapunov equation algebraically. ⋆
m/dlyap2.m	Solve Lyapunov equation iteratively.
m/dpok3.m	AR identification in noise, Kalman filtering.
m/dpok4.m	Experiments with fltsnd1.m
m/dpok5.m	SNR improvement measurement for Kalman filtering.
m/dpok6.m	AR identification evaluation.
m/dpok7.m	Demonstration of alternative AR identification methods.
m/dyadr.m	Performs dyadic reduction. ⋆
m/fltseg.m	Performs complete identification and Kalman filtering for one segment of a speech+noise signal. Assumes white noise.
m/fltsgonc.m	Given a Kalman gain, apply the filter. Calculate corrected Kalman gain based on innovation.
m/fltsnd.m	Filter a given signal using iterative Kalman filtering as above. Uses kalmkee.m
m/fltsnd1.m	Filter a given signal using iterative Kalman filtering. Uses kalmkeb.m
m/fltsnd2.m	Like fltsnd1.m, uses clhamres.m for AR identification.
m/fltsnd3.m	Like fltsnd2.m, uses kalmkebsvd.m.
m/gccopts.sh	Script to configure MEX creation. ⋆
m/getnoise.m	Given autocorrelation and AR coefficients, estimate process and measurement noise energies by least-squares fitting.
m/getpns.m	Get power spectrum of the noise signal using Doblinger's algorithm.
m/getpns2.m	Get power spectrum of the noise signal using Martin's algorithm.
m/getpss.m	Estimate power spectrum of a signal segment-by-segment.
m/hanning1.m	Returns Hanning window coefficients. ⋆
m/havlena.m	Implements Havlena's simultaneous state and parameter estimation.

m/htls.m	Hankel TLS AR identification.
m/idarx.m	Direct LS AR identification.
m/idarxb.m	AR identification using modified covariance method (supplied by *MATLAB*). Also gives b.
m/idarxoe.m	Direct LS AROE identification.
m/iswhite.m	Autocorrelation whiteness test.
m/kalm1.m	Test of identification and Kalman filtering on an AROE model.
m/kalm2.m	Calls fltsgonc.m for an artificial signal.
m/kalmanf.m	Performs Kalman filtering for general linear, white noise driven system. Gives filtered output.
m/kalmanr.m	Like kalmanf.m but performs two-pass smoothing.
m/kalmanrs.m	Like kalmanr.m, plus returns all state estimates.
m/kalmanrsi.m	Like kalmars.m except it takes initial covariance matrix estimate.
m/kalmanrsi1.m	Reorganised version of kalmanrsi.m.
m/kalmans.m	Like kalmans.m, plus returns all state estimates.
m/kalmk.m	Performs steady-state Kalman filtering, given the state matrix and Kalman gain.
m/kalmke.m	Performs repetitively steady-state Kalman filtering as long as the innovation is non-white, updating Kalman gain in each iteration. Based on Mehra's articles [42],[43].
m/kalmke1.m	Similar to kalmke.m except only one trial filtering is performed. Uses calculated innovation sequence to iteratively find optimal Kalman gain.
m/kalmkeb.m	Finds process noise for Kalman filtering by optimising a prediction error, then calls kalmk.m.
m/kalmkebsvd.m	Like kalmkeb.m except it uses kalmksvd.m instead of kalmk.m.
m/kalmkee.m	Like kalmke.m but proceeds until convergence is attained.
m/kalmkee1.m	Streamlined version of kalmkee.m.
m/kalmkg.m	Steady-state Kalman filtering for AR systems with gradient gain improvement algorithm.
m/kalmkre.m	Like kalmkee.m, except uses weighting based on autocorrelation estimate accuracy to improve robustness.
m/kalmksvd.m	Like kalmk.m, in addition it finds optimal initial estimate of the state with respect to prediction error.
m/kalmkw.m	Steady-state Kalman filtering for AR systems with another gradient gain improvement algorithm.
m/kbfopt.m	Objective function for kalmkeb.m using kalmk.m.
m/kbfoptsvd.m	Objective function for kalmkeb.m using kalmksvd.m.
m/kgain.m	Calculate Kalman gain from system parameters.
m/kgainm.m	Like kgain.m but allows more general system description.
m/ldfaktor.m	Finds LD factorisation of a matrix. ⋆
m/loadwav.m	Loads *wav* file. Obsolete for recent *MATLAB*. ⋆
m/lusolve.c	Symbolic link to a MEX wrapper. Solves a set of linear equation via LD decomposition.
m/lyap1.m	Solves continuous-time Lyapunov equation. ⋆
m/memsp.m	Maximum entropy spectrum estimate as suggested by Burg.
m/mlaroe.m	Given AR signal distorted by additive noice, calculate ML estimate of its parameters. Very slow.
m/mlaroefn*	Objective functions for mlaroe.m.
m/mvsp.m	Minimum variance spectrum estimate.
m/pfft.m	Parametric (AR) estimate of power spectrum.
m/play.linux.m	Plays a vector on my Linux system as sound. Obsolete for recent *MATLAB*.
m/play.m	Another version of play.linux.m.
m/procsig1.m	Given a signal, performs windowing and speech parameter estimation. Parametric spectrum estimation.
m/procsig2.m	Given a signal, estimates speech and noise spectra in each segment using a method similar to Martin's.
m/procsig3.m	Given a signal, performs complete processing including identification and Kalman smoothing, returns speech estimate. May use darkalmc MEX.
m/procsig4.m	Processes a signal using Martin's algorithm.
m/procsig5.m	Like procsig3, using fast Kalman smoothing as default and SR implementation as a fall-back. Uses MEX files.
m/procsig6.m	Generalised version of procsig5.m, variant subtraction rules etc.
m/procsig7.m	Like procsig6.m except uses MEM spectrum estimation.
m/ps2c.m	Transforms power spectrum to autocorrelation.
m/pub.m	Exact AR estimation by P. Lemmerling. ⋆
m/rekurz.m	Recursive ARMA identification.
m/s2a.m	Given a spectrum, calculates AR parameters.
m/s2ae.m	Like s2a.m plus returns also autocorrelation.
m/s2aex.m	Like s2ae.m, uses TSTLS.
m/sarmax.m	Simulates stochastic ARMAX system.
m/savewav.m	Saves vector to *wav* file. Obsolete for recent *MATLAB*. ⋆
m/sfft.m	Computes smoothed periodogram.
m/spsig.mat	*MATLAB* 5.0 MAT file containing prepared real signals.
m/spsig4.mat	Like spsig.mat but readable by *MATLAB* 4.x.
m/subr.m	Alternative subtraction rule.
m/test?.m	Various tests of spectrum and autocorrelation estimation.
m/testsig1.m	Generates artificial test signals simulating speech and noise. 'Speech' is harmonic.
m/testsig2.m	Generates artificial test signals simulating speech and noise by means of AR models.
m/testsig3.m	Read and prepare real speech and noise signals for tests. Produced xsig.wav.
m/tls.m	Solves generic one-dimensional TLS problem.
m/tstls.m	Exact AR modeling by TSTLS solution. Using algorithm by Van Huffel, Park, Rosen.

m/tstls1.m	Uses quasi-Newton method to solve the TSTLS problem.
m/tstls2.m	Like tstls1.m, now minimising objective function along the Newton vector in each step.
m/tstls3.m	Like tstls2.m plus tries to find a good initial guess.
m/tstls4.m	Like tstls3.m but improved handling of unsuccessful steps.
m/tstls5.m	Different and more efficient implementation of tstls.m. Faster but needs more precise initial estimate.
m/tstls6.m	Yet different implementation for tstls.m.
m/tstlsw.m	Solves TSTLS problem by weighting method and conjugate gradient minimisation.
m/tstlswc.c	Symbolic link to the MEX wrapper representing C version of tstlsw.m.
m/ttls.m	Solves TSTLS identification by neglecting Toeplitz matrix structure.
m/x2a.m	Given a signal, compute AR parameters of the synthesising filter.
m/x2ar.m	Like x2a.m, but uses repetitive autocorrelation method.
m/yey.m	Returns anti-diagonal matrix.
pvm/hostfile	An example host file for PVM.
pvm/startpvmd	Script used to start PVM daemons.
tex/Makefile	Makefile to facilitate compiling of this report. If you are fortunate, typing 'make' should be enough.
tex/README	Description on how to compile this report and slides.
tex/abstract.tex	Source for abstract.
tex/algor.tex	Source for the chapter 'Algorithm Overview'.
tex/*.eps	Pictures as encapsulated Postscript, converted from *fig*, generated by *MATLAB* or obtained from other sources.
tex/*.fig	Source for all diagrams in xfig native format.
tex/.pstex**	Intermediary representation of xfig pictures.
tex/<picture>.tex	Text part of a diagram, generated by latex2tex.
tex/conclusions.tex	Source for the chapter 'Conclusions'.
tex/declaration.tex	Source for the initial declaration.
tex/dipl.aux	Auxiliary file generated by TEX.
tex/dipl.dvi	*Dvi* version of this report.
tex/dipl.log	TEX log file.
tex/dipl.ps	Postscript version of this report. ◇
tex/dipl.tex	Main TEX file for this report.
tex/dipl.toc	Automatically generated table of contents.
tex/diplmac.tex	Macros used to typeset this report. Actually a link to diplmac.10pt.tex.
tex/diplmac.10pt.tex	Truly used macros.
tex/diplmac..tex*	Previously used macros for different type sizes.
tex/diplpre1.tex	Short preliminary report.
tex/diplsmall.ps	Postscript version of this report, reduced by putting two pages on one sheet. Generated by pstops. ◇
tex/dplslds.tex	Source code for slides.
tex/dvipsrc	This file is to be stored either as ~/.dvipsrc or in a system-wide directory. It is used by dvips to find non-standard fonts. See also jk_psfonts.map.
tex/eplain.tex	Eplain format by Karl Berry. ⋆
tex/experiments.tex	Source for the chapter 'Experiments'.
tex/glossary.tex	Source for the 'Glossary'.
tex/identification.tex	Source for the chapter 'Identification'.
tex/implementation.tex	Source for the chapter 'Implementation'.
tex/introduction.tex	Source code for the chapter 'Introduction'.
tex/jk_psfonts.map	This file is to be used by dvips to find the non-standard fonts. You might not need it, if you have these fonts already installed.
tex/kalman.tex	Source for the chapter 'Kalman Filtering'.
tex/plaina4.tex	Sets A4 paper size. ⋆
tex/preface.tex	Source for the 'Preface'.
tex/psfonts.map	Symbolic link to jk_psfonts.map. Might be needed for some versions of dvips.
tex/*.pro	Postscript prologs from the PSTricks package by Timothy Van Zandt. Not needed if you have PSTricks installed. ⋆
tex/pst-node.tex	TEX source from the PSTricks package by Timothy Van Zandt. See remark above. ⋆
tex/pstricks.*	TEX sources from the PSTricks package by Timothy Van Zandt. See remark above. ⋆
tex/references.tex	Source for the 'References'.
tex/un*.*	*Afm, pfb, tfm,* and *vf* files for the non-standard Postscript fonts used, namely *Century Schoolbook, Nimbus Mono, Nimbus Sans*, all coming from URW and distributed under GPL. Not needed if you have these fonts already installed.
tex/state.tex	Source for the chapter 'Kalman Filtering'.
tex/tls.tex	Source for the chapter 'Total Least Squares'.

8. Experiments

This chapter describes experiments executed in order to assess performance of the newly developed speech enhancement algorithm in comparison with existing ones. Care must be taken, however, in interpreting the results. Firstly, the behaviour of the algorithm is highly dependent on the parameter values; some observations of the dependence are also included. Certain criterion can thus be improved at the expense of others. Secondly, the results are only meaningful for the specific test signal set. For other test signals, the performance may be different. As for the subjective criteria, it must be taken into account that they were evaluated only by the author, while for a statistically reliable results a much larger number of listeners would be needed.

The new algorithm as implemented in program kalmse was compared with double power spectral subtraction (PSS) (1,10) using an energy detector [49], Martin's algorithm [40], and Doblinger's algorithm [15] as implemented in program ssub. Unless stated otherwise, default parameters were used.

8.1. Test Signals

The 12 noise signals used were recorded inside different cars running at approximately constant speeds using microphones mounted near the upper rim of the windshield. The 9 speech signals were recorded using the same setting when the cars were stationary. The speech signals contain isolated Czech digits 0∼9 in random order pronounced by several mostly male speakers. After manual labelling of speech/non-speech regions, the signals were artificially mixed to obtain average segmental SNR in speech regions (denoted $\mathrm{spSNR}_{\mathrm{SEG}}$) of −10,−5,0,5, and 10 dB, thus yielding a total of $12 \times 9 \times 5 = 540$ test signals of the lengths of 6∼10 s.

The use of artificially mixed signals was motivated by the need of having the original clean speech signal available, so that the performance could be evaluated. Nevertheless, artificially mixed signals exhibit similar characteristics to 'real' signals encountered when recording speech in a running car, as far as spectrum and subjective acoustic perception are concerned; this was demonstrated by several experiments. However, it is known that speakers unconsciously react to high environmental noise which results in slight changes of speech parameters. It is known as *Lombard effect* [12].

It should also be noted that SNR of −10 dB represents rather harsh conditions that are not expected to occur too often in the target application. On the other hand, at the SNR of 10 dB the speech quality might be acceptable even without a speech enhancement. For most applications, an effective range of the input SNR can be assumed to be close to the −5∼5 dB range.

8.2. Criteria

The performance of all algorithms was compared using the following criteria:

- subjective speech quality, intelligibility, and distortion
- recognition accuracy improvement
- improvement of mean cepstral distance in speech segments
- global segmental SNR improvement ($\mathrm{gSNR}_{\mathrm{SEG}}$)
- segmental SNR improvement in speech segments ($\mathrm{spSNR}_{\mathrm{SEG}}$).

8.3. Recognition Accuracy

Recognition error improvement was tested on a speech recognition system *POST* [26] in cooperation with its author. However, in the given time we did not succeed to demonstrate clearly the benefits of speech enhancement for improving a recognition rate for noisy input signals. When confronted with a noisy signal, the recognition rate dropped almost to zero, regardless of whether the speech enhancement algorithm was applied. This was in accordance with our former experiments.

8.4. Cepstral Distance

As another signal quality measure, a mean *cepstral distance* between the given signal and the corresponding clean signal in speech segments was calculated [49]. The smaller this distance, the greater the similarity between these signals, and the higher the quality of the tested signal. The program cepdist was used to calculate the difference between these distances for a noisy input signal and the corresponding filtered signal. Cepstral distances were obtained by the following method: The signal was cut into segments, AR parameters of each segment found using Burg's algorithm and converted to cepstral coefficients [49]. An Euclidean distance between first 10 coefficients of the two signals was calculated and averaged across all segments containing speech.

Ideally, a speech enhancement algorithm should make a signal closer to the original undistorted speech, i.e., *decrease* the cepstral distance. Nevertheless, after having performed the tests on the aforementioned set of 540 test signals, it was found that for *all* algorithms, the cepstral distance *increased* significantly (by 20$\sim$100 % on the average). The cepstral distance decreased in less than 5 % of cases and never by more than 5%. Moreover, the standard deviation averaged about 100 % of the mean. The author infers that although it was demonstrated that *none* of the algorithms improves the cepstral distance improvement criterion, the data set was probably too small to make other conclusions.

8.5. Listening Tests

Example signals are shown in *figure 6*. The speech signal is a 5 s long sequence of Czech words 'dva osm jedna šest' (meaning 'two eight one six') pronounced by a male speaker. This signal was mixed with a noise from a car running with constant velocity, so that the resulting $\mathrm{spSNR}_{\mathrm{SEG}}$ were 0 dB.

The output of the power spectral subtraction method is rather naturally sounding speech corrupted by a large amount of bursting noise similar in nature to the original noise. The intelligibility has deteriorated and the sound is rather unpleasant to listen to. At lower SNR the algorithm performs very poorly because the VAD is no longer capable of working properly. At higher SNR 'musical noise' can be perceived.

The Martin's algorithm is very efficient and robust. At higher SNR, the noise suppression is almost ideal with light speech distortion. Noise-free pauses are impressive. As the SNR decreases, the

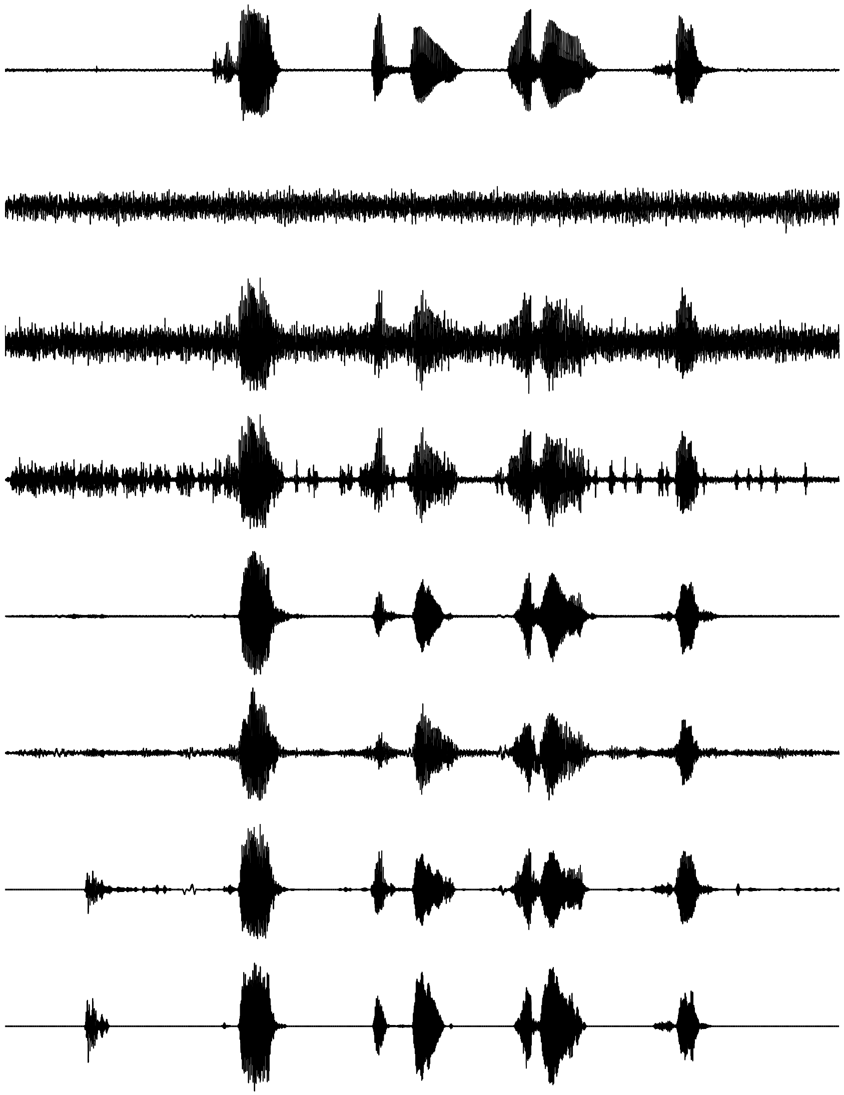


Figure 6. *From top to bottom: clean speech, noise, speech corrupted by noise, PSS, Martin's algorithm, Doblinger's algorithm, new algorithm with FFT, new algorithm with MEM. (The artifacts seen at the beginning of* kalmse *output in both cases are caused by the startup phase of the algorithm and do not appear later. They cannot be removed by simple means, because their shape depends on algorithm parameters.)*

signal seems to be still virtually noise-free but its distortion increases and low energy phonemes are gravely attenuated. The frequency characteristics of the speech are also changed, the speech could be described like coming from a deep well. The results are however very consistent and uniform.

The Doblinger's method does not achieve the high noise suppression of Martin's and artificially sounding 'musical noise' can be heard. On the other hand, speech distortion is slightly smaller and the algorithm also fails more gracefully with increasing SNR, the speech quality deteriorates gradually and rather slowly. Generally, both Martin's and Doblinger's algorithm perform well and the choice between them depends on personal preferences. Both have a number of parameters so that their performance can be tuned to specific applications.

The performance of kalmse algorithm with averaged periodogram is similar to Burg's and Doblinger's. Only moderate attenuation of low-energy phonemes is perceived while some background musical noise is unfortunately reintroduced. The result is rather pleasant although the filtering characteristics are apparently randomly varying around some mean values. It was also observed that the more stationary a phoneme is, the better it appears at the output.

The musical noise problem is completely overcome in the MEM variant of the method when no musical noise is present but at this SNR level the resulting speech distortion is severe.

The output of all examined methods except PSS is more pleasant to listen to than the original noisy input. However, no clear intelligibility improvement was noticed. The Martin's method and kalmse were found to be the best from the subjective speech quality point of view. As their speech and noise estimation procedures are based on similar principles they are consequently very close in performance. Martin's method has the advantage of higher robustness and introduces almost no audible residual noise. Although kalmse can be made to perform better, parameter tuning is often needed. Two good starting points for experimenting are the default parameters and '-fm 1 -mh 10 -ml 0.001 -eb 0.7 -be 0.99 -an 0.95'.

8.6. Algorithm Speed Comparison

The tests were performed on a HP 9000/712 workstation, model 712/100 with a 100 MHz PA-RISC processor, SPECfp95 index 36.3, running HP-UX version B.10.20. The programs were compiled using the *C* compiler cc provided with the system using maximum optimisation. Elapsed CPU time is shown for serial algorithms. The speed of parallel versions was measured on a cluster of 8 computers — 7 HP workstations of different kinds and one PC with PentiumPro at 200 MHz running Linux. All computers were being used by other users during the tests as well as the network which should be taken into account when interpreting the results. The total elapsed real times are shown, while total CPU time used on all machines were approximately the same as for the serial versions. The test signal was the same as for the listening test, i.e., about 5 s long.

Algorithm	serial	parallel
PSS	0.20 s	
Martin	0.55 s	
Doblinger	0.82 s	
kalmse + FFT	36.96 s	10.94 s
kalmse + MEM	33.86 s	10.34 s

8.7. SNR Improvements

The segmental SNR of a signal $\hat{y}_s$ with respect to the clean signal y_s was calculated according to the following formula

$$\mathrm{SNR}_{\mathrm{SEG}} = \frac{1}{N}\sum_{i=0}^{N-1} \max\left\{\mathrm{SNR}_{\min}\ ,\ \min\left\{\mathrm{SNR}_{\max}\ ,\ 10\log_{10}\frac{\sum_{t=Ti}^{Ti+T-1} y_s^2(t)}{\sum_{t=Ti}^{Ti+T-1}\left(\hat{y}_s(t)-y_s(t)\right)^2}\right\}\right\} \tag{92}$$

where the frame length $T = 128$ and averaging is done either across all frames (resulting in $\mathrm{gSNR}_{\mathrm{SEG}}$), or only across segments containing speech (giving $\mathrm{spSNR}_{\mathrm{SEG}}$), where the presence of speech is determined by human listener. For methods seeking to reproduce the original waveform, such as all methods tested, the $\mathrm{spSNR}_{\mathrm{SEG}}$ is believed to be highly correlated with subjective quality perception [12]. As $\mathrm{spSNR}_{\mathrm{SEG}}$ does not give any indication of the algorithm behaviour in speech pauses, $\mathrm{gSNR}_{\mathrm{SEG}}$ was used for this purpose. The serious deficiency of $\mathrm{gSNR}_{\mathrm{SEG}}$ is its dependence on somewhat arbitrary limit $\mathrm{SNR}_{\min}$ which may come into effect in speech pauses. Consequently, $\mathrm{gSNR}_{\mathrm{SEG}}$ values should be used for algorithm performance comparison only if applicable, if calculated by the same algorithm and if the input signals are identical.

Sample means and standard deviations of the output $\mathrm{gSNR}_{\mathrm{SEG}}$ and $\mathrm{spSNR}_{\mathrm{SEG}}$ for various input $\mathrm{spSNR}_{\mathrm{SEG}}$ are presented in the following table. All algorithms were tested on the complete set of data described in section 8.1 (p. 59). SNR values were calculated by the program snr, which uses $\mathrm{SNR}_{\min} = -300$.

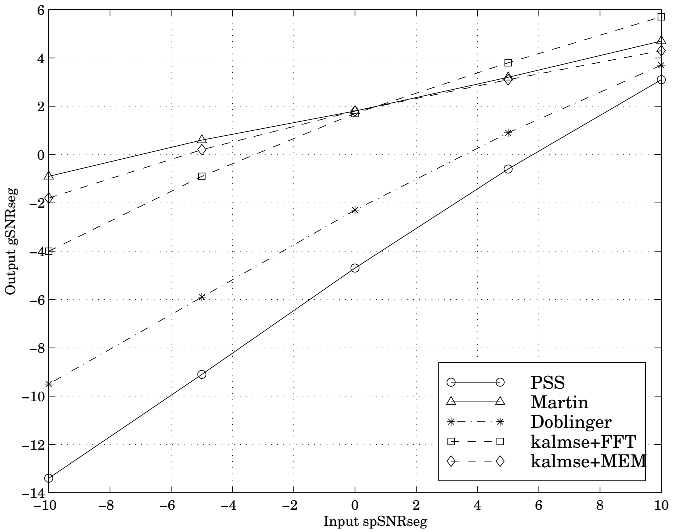
input $\mathrm{spSNR}_{\mathrm{SEG}}$[dB]

Algorithm	-10		-5		0	
PSS	$-13.4_{\pm1.4}$	$-4.3_{\pm0.6}$	$-9.1_{\pm1.4}$	$-0.6_{\pm0.5}$	$-4.7_{\pm1.4}$	$3.5_{\pm0.5}$
Martin	$-0.9_{\pm0.6}$	$1.0_{\pm0.4}$	$0.6_{\pm0.4}$	$2.5_{\pm0.4}$	$1.8_{\pm0.3}$	$4.5_{\pm0.5}$
Doblinger	$-9.5_{\pm3.5}$	$-2.4_{\pm1.7}$	$-5.9_{\pm3.0}$	$0.2_{\pm1.3}$	$-2.3_{\pm2.3}$	$3.2_{\pm1.2}$
kalmse + FFT	$-4.0_{\pm1.6}$	$0.5_{\pm0.7}$	$-0.9_{\pm1.2}$	$2.8_{\pm0.6}$	$1.7_{\pm0.9}$	$5.1_{\pm0.6}$
kalmse + MEM	$-1.8_{\pm1.8}$	$0.4_{\pm0.6}$	$0.2_{\pm1.3}$	$1.8_{\pm0.5}$	$1.8_{\pm0.9}$	$3.4_{\pm0.6}$
output	$\mathrm{gSNR}_{\mathrm{SEG}}$	$\mathrm{spSNR}_{\mathrm{SEG}}$				

Algorithm	$+5$		$+10$	
PSS	$-0.6_{\pm1.2}$	$7.6_{\pm0.4}$	$3.1_{\pm1.0}$	$11.7_{\pm0.3}$
Martin	$3.2_{\pm0.4}$	$7.3_{\pm0.5}$	$4.7_{\pm0.4}$	$10.4_{\pm0.5}$
Doblinger	$0.9_{\pm1.7}$	$6.6_{\pm1.2}$	$3.7_{\pm1.1}$	$10.2_{\pm1.2}$
kalmse + FFT	$3.8_{\pm0.6}$	$7.8_{\pm0.6}$	$5.7_{\pm0.5}$	$10.9_{\pm0.5}$
kalmse + MEM	$3.1_{\pm0.6}$	$5.3_{\pm0.7}$	$4.3_{\pm0.6}$	$7.5_{\pm0.7}$

Figure 7 presents the results from the preceding table in graphical form. It can be seen that in terms of output $\mathrm{gSNR}_{\mathrm{SEG}}$ kalmse (in the combination with averaged periodogram) performs better than other algorithms for $\mathrm{spSNR}_{\mathrm{SEG}} \gtrsim 0\,\mathrm{dB}$, while for lower input $\mathrm{spSNR}_{\mathrm{SEG}}$, Martin's algorithm is preferable. This is in accordance with listening tests, as unlike kalmse, Martin's algorithm indeed gives almost noise-free pauses between words.

As explained above, it is more justified to compare algorithm performance using $\mathrm{spSNR}_{\mathrm{SEG}}$. In terms of this criterion, Martin's algorithm gives best results for very low input $\mathrm{spSNR}_{\mathrm{SEG}}$, PSS for



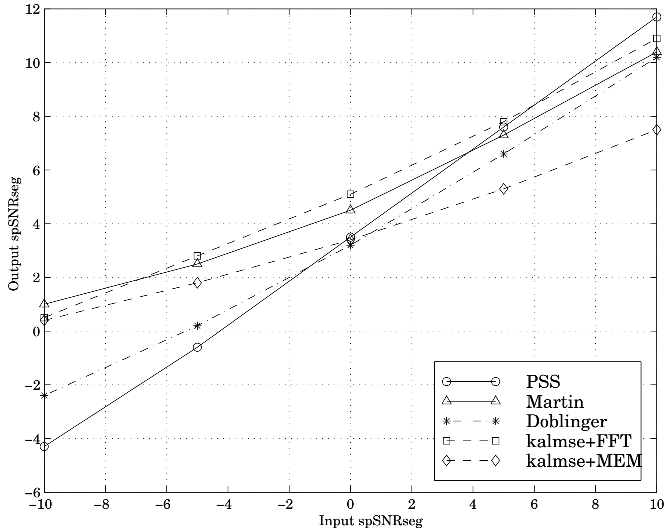


Figure 7. *The dependence of output gSNR$_{\text{SEG}}$ and spSNR$_{\text{SEG}}$ on input spSNR$_{\text{SEG}}$ for the algorithms tested.*

high input spSNR$_{\text{SEG}}$, while kalmse outperforms all others in the region in between. And as most application will probably process signals mainly from this range, it is a strong argument in favour of the new kalmse algorithm. It is also interesting that the kalmse method using periodograms outperforms the MEM variant in terms of SNR, while subjective comparison of speech quality sometimes yields the contrary.

8.8. Effect of Parameter Changes

This section describes how changes to various parameters influence the performance of the kalmse algorithm (using periodograms for spectrum estimation). As the number of combinations of possible parameter values, input characteristics, and evaluation methods surpasses all practicable limits, the results presented in this section come from only a relatively small number of tests. Moreover, the optimal parameter value are likely to depend on the nature of input signals. This dependence could not be studied as all signals available were approximately of the same kind.

If a different sampling speed f_s and a different resolution N_{BITS} is to be used, instead of 8000 Hz and 16 bit, the parameters should be modified so that the following quantities remain constant: T/f_s, $M_1 M_2/f_s$, $(1-\alpha_{N/S})\,f_s$, $(1-\beta)\,f_s$, $(1-\gamma)\,f_s$, $N_{\text{BITS}}/\log r$. The rest can remain unchanged.

For the choice of a subtraction rule, refer to section 6.4.3 (p. 46), while section 6.4.1 (p. 45) can give guidance on the proper spectrum estimation method and section 6.4.4 (p. 47) advises on the choice of an identification method.

8.8.1. Model Orders

Listening tests indicate increase in the output speech quality as the model order n_S increases. The improvement gradually diminishes and for $n_S \gtrsim 8$ becomes imperceptible. Similar behaviour is perceived for n_N, when little change was observed for $n_N \gtrsim 6$. Presumably, this threshold decreases as the noise signal gets more stochastic and *vice versa*.

Output spSNR$_{\text{SEG}}$ values for different n_S and n_N are given in the following tables. The input spSNR$_{\text{SEG}}$ was fixed at 5 dB. Each value given is a mean of 10 measurements on randomly chosen input signals and its standard deviation is 0.5$\sim$0.7 dB.

$n_N = 8$

n_S	2	4	6	8	10	12	14	16	18	20
spSNR$_{\text{SEG}}$[dB]	7.1	7.4	7.7	7.6	7.5	7.3	7.4	7.3	7.2	7.3

$n_S = 8$

n_N	2	4	6	8	10	12	14	16	18	20
spSNR$_{\text{SEG}}$[dB]	7.3	7.6	7.7	7.6	7.6	7.6	7.3	7.3	7.2	7.2

The decrease for higher n can be explained by increased difficulty of the identification.

8.8.2. Smoothing Factors and Circular Buffer Length

Increasing smoothing factors means longer time constant of the averaging filters, which decreases the variance of the estimate at the expense of its tracking ability and resolution. The smoothing factors α_N, β controlling noise estimation should be made larger if the noise is rather stationary and conversely. For all three smoothing factors α_S, α_N, β there are optimum values around which performance slowly deteriorates. This is illustrated in the following table showing the effect of changing α_N on the output spSNR$_{\text{SEG}}$. Input spSNR$_{\text{SEG}}$ $= 5\,\text{dB}$; all values have standard deviation 0.5$\sim$0.7 dB.

α_N	0.5	0.6	0.7	0.8	0.9	0.93	0.95
$\mathrm{spSNR_{SEG}}$ [dB]	6.7	7.3	7.6	7.7	7.5	7.3	6.5

Increasing effective length $M_1 M_2$ of the circular buffer (used for minimum tracking for noise estimation) has similar effects to increasing the smoothing factors — the estimate gets more precise but loses its ability to track short-time changes and may become outdated. Again, optimum value exists, around which the quality decreases.

8.8.3. Bias Factors

The behaviour of the algorithm with respect to noise bias factor γ and oversubtraction factors δ_H, δ_L is analogous. The greater they are, the greater suppression of unwanted noise is achieved. However, the low energy phonemes are suppressed too and noise distortion increases. While δ_L resp. δ_H control the lower resp. upper boundary for the suppression, γ has general effect. Typical effects can be seen in the following table, containing dependencies of output $\mathrm{spSNR_{SEG}}$ and $\mathrm{gSNR_{SEG}}$ on γ, as input $\mathrm{spSNR_{SEG}}$ is kept at 5 dB. There is an optimum value of γ with respect to $\mathrm{spSNR_{SEG}}$, corresponding to minimum speech distortion. There is also an optimum with respect to $\mathrm{gSNR_{SEG}}$; this optimum is higher because noise suppression in speech pauses increases monotonously with γ.

γ	0.7	0.9	1.1	1.3	1.5	1.7	1.9	2.1	2.3	2.5	2.7	2.9
$\mathrm{spSNR_{SEG}}$ [dB]	7.7	7.7	7.9	7.6	7.5	7.5	7.5	7.4	7.2	7.2	7.0	6.7
$\mathrm{gSNR_{SEG}}$ [dB]	1.7	2.6	3.2	3.4	3.7	3.8	3.9	3.9	3.9	3.9	3.8	3.7

9. Conclusions

Speech enhancement is an inherently complex interdisciplinary topic, exploiting the knowledge of physics (acoustics and mechanics), stochastic control theory (system identification and optimal filtering), digital signal processing, numerical mathematics, computer science (efficient algorithm design) and other disciplines. Although it could not be fully demonstrated in this report due to the space constraints, the author enriched his knowledge in all the above mentioned areas. An extensive reference list was gathered.

An overview of the speech enhancement field focused on spectral subtraction techniques was presented. The basics of Kalman filtering, system identification and total least squares methods were stated.

A new speech enhancement algorithm for speech corrupted by slowly varying additive background noise based on Kalman filtering was developed. Existing spectral subtraction algorithms were taken as basis, parametric models of both noise and speech generating systems were estimated and passed to Kalman smoothing filter to obtain the speech signal estimate. According to our knowledge, this is the first attempt to use Kalman smoothing for speech enhancement purposes and also the first attempt to combine minimum tracking noise estimation, spectral subtraction, and Kalman filtering. It was demonstrated that this new algorithm can outperform alternative speech enhancement algorithms.

The algorithm was implemented and evaluated. The parallel version proves that speech enhancement algorithms can profit from parallel computing techniques.

Nevertheless, the rather modest quality improvement of the resulting speech over alternative methods would not by itself justify the significant increase in computational complexity. The author believes that the main advantage of using Kalman filtering over spectral subtraction and Wiener filtering is the freedom from the somewhat artificial assumptions about the speech and noise properties imposed by the two latter methods. Using Kalman filtering, it is natural to use adaptive segmentation of the input signal aiming at identifying the true stationarity regions of the speech. Another example would be using model parameters varying in time within frames or abandoning the frames completely.

These new possibilities were not exploited in the presented work. Instead, knowingly good legacy algorithms were used. Many alternatives were proposed but usually found unfeasible.

Although the Kalman filter is optimal in the linear setting, the extended Kalman filter applied on non-linear problems is only an approximation and suffers from numerous problems. If a full extension of the Kalman filtering theory into the non-linear domain were available, it would greatly simplify the solution of many important problems, including the identification-estimation task discussed here.

This work also demonstrates the use of a model based approach for speech enhancement. Provided a suitable model is available, model based techniques have the potential of being superior to alternative approaches, making advantage of the *a priori* information accumulated in the model structure. In the case studied, to identify the model parameters correctly and reliably was found to be the critical part of the algorithm. Less *ad hoc* techniques for separating speech and noise features are needed, perhaps based on stationarity and independence measures. Moreover, a suitable and efficient parameter estimation technique still remains to be find. Consequently, suboptimal methods had to be used as well as clearly oversimplified auto-regressive models.

Limits on the maximum attainable performance for uninformed speech enhancement algorithms were derived. To overcome these limits, a knowledge base consisting e.g., of typical phoneme characteristics must be incorporated into the algorithm, for which HMM can be a valuable tool.

Finally, it is important to keep in mind that for speech enhancement systems designed for human listener, it is the human listener who is the ultimate judge. The conversion to an optimisation problem is a design and engineering decision and must be always verified by listening tests. As far as enhancement systems for speech recognition are concerned, their design criterion should closely match the recogniser structure and therefore, according to the author, it seems unlikely that a single enhancement algorithm would perform well for both tasks.

Constant progress is being made in the domain of speech enhancement but it still remains a challenging and rewarding field full of problems waiting to be solved.

References

[1] B. Anderson and J. Moore, *Optimal Filtering,* Prentice Hall, 1979.

[2] K. Åström and P. Eykhoff, *System Identification — A Survey,* Automatica, Vol. 7, 1971.

[3] C. Avendano, H. Hermansky, M. Vis and A. Bayya, *Adaptive Speech Enhancement Using Frequency-Specific SNR Estimates,* Proceedings of EUROSPEECH'97.

[4] M. Bartlett, *An Introduction to Stochastic Processes,* Cambridge Univ. Press, London, 1962.

[5] M. Berouti, R. Schwartz and J. Makhoul, *Enhancement of Speech Corrupted by Acoustic Noise,* Proceedings of IEEE Conf. ASSP, April 1979.

[6] S. Boll, *Speech Enhancement in the 1980s: Noise Suppression with Pattern Matching,* S. Furui,M. Sondhi: Advances in Speech Signal Processing. Marcel Dekker, Inc. 1992.

[7] S. Boll, *Suppression of acoustic noise in speech using spectral subtraction,* IEEE Trans. Acoust., Speech, and Sig. Proc., No. 2, 1979.

[8] A. Bryson and M. Frazier, *Smoothing for Linear and Nonlinear Dynamic Systems,* TDR 63-119, Aero. Sys. Div., Wright-Patterson Air Force Base, Ohio 1963.

[9] A. Bryson and Y. Ho, *Applied Optimal Control,* Blaisdell, Waltham, Mass., 1969.

[10] H. Chen, S. Van Huffel, A. van den Boom, P. van den Bosch, *Extended HTLS methods for parameter estimation of multiple data sets modeled as sums of exponentials,* ftp://ftp.esat.kuleuven.ac.be/pub/SISTA/vanhuffel/reports/dsp97.ps.Z

[11] R. De Beer, *Quantitative In Vivo NMR,* http://dutnsic.tn.tudelft.nl:8080/c59_to_html/c59.html

[12] J. Deller, J. Proakis, J. Hansen, *Discrete-Time Processing of Speech Signals,* Prentice Hall, 1987.

[13] B. De Moor and G. Golub, *The Restricted Singular Value Decomposition: Properties and Applications,* ftp://ftp.esat.kuleuven.ac.be/pub/SISTA/demoor/reports/simaxrsvdreview.ps.Z

[14] B. De Moor, *Numerical Algorithms for State Space Subspace System Identification,* ftp://ftp.esat.kuleuven.ac.be/pub/SISTA/demoor/reports/kak.ps.Z

[15] G. Doblinger, *Computationally Efficient Speech Enhancement By Spectral Minima Tracking in Subbands,* proceedings of EUROSPEECH'95.

[16] Y. Ephraim and D. Malah, *Speech Enhancement using a Minimum Mean-Square Error Log-Spectral Amplitude Estimator,* IEEE Trans. Acoust., Speech and Sig. Proc., April 1985.

[17] G. Faucon and R. Le Bouquin-Jeannes, *Echo and Noise Reduction for Hands-Free Terminals — State of the Art,* Proceedings of EUROSPEECH'97.

[18] F. Fnaiech, D. Bastard, V. Buzenac, R. Settineri and M. Najim, *A fast Kalman filter based new algorithm for training feedforward neural networks,* in M. Holt et al.: Signal Processing VII, Theories and Applications, EASP 1994.

[19] M. Gabrea, E. Mandridake and M. Najim, *A Single Microphone Noise Canceller Based on Adaptive Kalman Filter,* research report.

[20] R. Haeb-Umbach, *Robust Speech Recognition of Wireless Networks and Mobile Telephony,* Proceedings of EUROSPEECH'97.

[21] P. Händel, *Kalman Filtering for Low Distortion Speech Enhancement in Mobile Communication,* http://www.syscon.uu.se/Staff/ph/1491SOER.ps

[22] P. Händel, *Lecture notes, 23.6.1997, Prague ČVUT,* http://frej.teknikum.uu.se/Staff/ph/ph.html

[23] P. Händel, *Low-Distortion Spectral Subtraction for Speech Enhancement,* http://www.syscon.uu.se/Staff/ph/eurosp95.ps

[24] V. Havlena, *Simultaneous Parameter Tracking and State Estimation in a Linear System,* Automatica, Vol. 29, No. 4, 1993.

[25] M. Hayes, *Statistical Digital Signal Processing and Modeling,* John Wiley and Sons, 1996.

[26] J. Hennebert, *Parallel Object-Oriented Speech Toolkit (POST),* http://circwww.epfl.ch/data/general/jean/post/post.html

[27] H. Hermansky, E. Wan and C. Avendano, *Speech Enhancement Based on Temporal Processing, ICASSP 1995,* http://www.ee.ogi.edu/~ericwan/pubs.html

[28] Z. Hrdina, *Statistická radiotechnika,* skriptum ČVUT, Praha, 1996.

[29] G. Jenkins and D. Watts, *Spectral Analysis and its Applications,* Holden Day Publ. San Francisco, 1968.

[30] R. Johansson, *System Modeling and Identification,* Prentice Hall, 1994.
[31] P. Kaminski, A. Bryson and S. Schmidt, *Discrete Square Root Filtering: A Survey of Current Techniques,* IEEE Trans. on Automatic Control, December 1971.
[32] R. Kashyap, *Maximum Likelihood Identification of Linear Dynamic Systems,* IEEE Trans. Automat. Contr., February 1970.
[33] S. Kay, *Modern Spectral Estimation, Theory and Application,* Prentice Hall, 1987.
[34] J. Kybic, *Identification of Autoregressive Time Series Using Repetitive Autocorrelation,* http://cmp.felk.cvut.cz/~kybic/software.html
[35] P. Lemmerling and S. Van Huffel, *Superlinear algorithms for solving the Structured Total Least Norm problem,* ftp://ftp.esat.kuleuven.ac.be/pub/SISTA/lemmerli/reports/int9544.ps.Z
[36] P. Lemmerling, I. Dologlou and S. Van Huffel, *Speech Compression based on exact modeling and Structured Total Least Norm optimization,* ftp://ftp.esat.kuleuven.ac.be/pub/SISTA/lemmerli/reports/int9731.ps.Z
[37] P. Lemmerling, S. Van Huffel and B. De Moor, *Structured Total Least Squares Problems: Formulations, Algorithms and Applications,* ftp://ftp.esat.kuleuven.ac.be/pub/SISTA/lemmerli/reports/int9658.ps.Z
[38] J. Lim and A. Oppenheim, *All-Pole Modeling of Degraded Speech,* IEEE Trans. Acoust., Speech, and Sig. Proc., June 1978.
[39] B. Logan and A. Robinson, *Enhancement and Recognition of Noisy Speech Within an Autoregressive Hidden Markov Model Framework Using Noise Estimates from the Noisy Signal,* http://svr-www.eng.cam.ac.uk/~btl/publications/icassp97.ps.Z
[40] R. Martin, *Spectral Subtraction Based on Minimum Statistics,* proceedings of EUSIPCO'94.
[41] J. Meditch, *A Survey of Data Smoothing for Linear and Nonlinear Dynamic Systems,* Automatica, March 1973.
[42] R. Mehra, *On the Identification of Variances and Adaptive Kalman Filtering,* IEEE Trans. Automat. Contr., April 1970.
[43] R. Mehra, *On-Line Identification of Linear Dynamic Systems with Applications to Kalman Filtering,* IEEE Trans. Automat. Contr., February 1971.
[44] S. Mitra and J. Kaiser, *Handbook for Signal Processing,* John Wiley and Sons, 1993.
[45] L. Nelson and E. Star, *The Simultaneous On-Line Estimation of Parameters and States in Linear Systems,* IEEE Trans. on Automatic Control, February 1976.
[46] T. Nishimura, *On the a priori Information in Sequential Estimation Problems,* IEEE Trans. on Automatic Control, April 1996.
[47] K. Paliwal and A. Basu, *A Speech Enhancement Method Based on Kalman Filtering,* proceedings of IEEE Int. Conf. Acoust. Speech, 1987.
[48] E. Parzen, *An Approach to Time-Series Analysis,* Ann. Math. Stat., vol. 32, December 1961.
[49] P. Pollák and P. Sovka, *Spectral Subtraction and Speech/Pause Detection,* internal report R95-1, ČVUT, Praha, 1995.
[50] W. Press, S. Teukolsky, W. Vetterling and B. Flannery, *Numerical Recipes in C,* Cambridge University Press, 1992.
[51] L. Rabiner and B. Juang, *Fundamentals of Speech Recognition,* Prentice Hall, 1993.
[52] H. Rauch, *Solutions to the linear smoothing problem,* IEEE Trans. on Automatic Control, AC-8 1963.
[53] T. Robinson, *Speech Analysis (lecture notes),* http://svr-www.eng.cam.ac.uk/~ajr/SpeechAnalysis/
[54] T. Söderström and P. Stoica, *System Identification,* Prentice Hall, 1988.
[55] M. Sambur, *Adaptive Noise Canceling for Speech Signals,* IEEE Trans. on Acoustics, Speech and Sig. Proc., October 1978.
[56] V. Sharma et al., *Adaptive Speech Enhancement Using a Subband Based Kalman Filter,* research paper.
[57] H. Sorenson, *Kalman Filtering: Theory and Application,* IEEE Press, 1985.
[58] P. Sovka, P. Pollák, J. Kybic, *Extended Spectral Subtraction,,* Proceedings of EUSIPCO'96, Trieste, ISBN 88-861 79-83-9.
[59] J. Štecha a V. Havlena, *Moderní teorie řízení,* skriptum ČVUT, Praha, 1996.
[60] F. Taylor, *Digital Filter Design Handbook,* Marcel Dekker, 1983.
[61] P. Tichavský and P. Händel, *Efficient Tracking of Multiple Sinusoids with Slowly Varying Parameters,* http://www.syscon.uu.se/Staff/ph/icassp93.ps
[62] J. Uhlíř and P. Sovka, *Číslicove zpracování signálů,* ČVUT.
[63] D. Van Compernolle, *Speech Recognition in the Car — From Phone Dialing to Car Navigation,* Proceedings of EUROSPEECH'97.

[64] S. Van Huffel and H. Zha, *The Restricted Total Least Squares Problem,* ftp://ftp.esat.kuleuven.ac.be/pub/SISTA/vanhuffel/reports/simax89.ps.Z
[65] S. Van Huffel and J. Vandewalle, *The Total Least Squares Problem: computational aspects and analysis,* Society for Industrial and Applied Mathematics (SIAM), 1991.
[66] S. Van Huffel, B. De Moor and H. Chen, *Relationship between Structured and Constrained TLS, with applications to Signal Enhancement,* ftp://ftp.esat.kuleuven.ac.be/pub/SISTA/vanhuffel/reports/mtns93.ps.Z
[67] S. Van Huffel, H. Park and J. Ben Rosen, *Formulation and Solution of Structured Total Least Norm Problems for Parameter Estimation,* ftp://ftp.esat.kuleuven.ac.be/pub/SISTA/vanhuffel/reports/ieeesp95.ps.Z
[68] S. Van Huffel, P. Lemmerling and L. Vanhamme, *Fast algorithms for signal subspace fitting with Toeplitz matrices and applications to exponential data modeling,* ftp://ftp.esat.kuleuven.ac.be/pub/SISTA/vanhuffel/reports/svdw94.ps.Z
[69] S. Van Huffel, *On the Significance of Nongeneric Total Least Squares Problems,* ftp://ftp.esat.kuleuven.ac.be/pub/SISTA/vanhuffel/reports/simax90.ps.Z
[70] E. Wan and A. Nelson, *Neural Dual Extended Kalman Filtering: Applications in Speech Enhancement and Monaural Blind Signal Separation,* IEEE Workshop and Neural Networks and Signal Processing, 1997.
[71] N. Yoma, F. McInnes, M. Jack, *Robust speech pulse detection using adaptive noise modelling and non-stationarity measure,* Proceedings of EUROSPEECH'97.
[72] —, *CLAPACK — Linear algebra routines in C,* http://www.netlib.org/clapack/
[73] —, *NEOS Guide,* http://www.mcs.anl.gov/otc/Guide/OptWeb/continuous/constrained/qprog/index.html
[74] —, *PVM Home Page,* http://www.epm.ornl.gov/pvm/pvm_home.html