Institut für Geodäsie und Geoinformation der Universität Bonn

Digital Filtering Algorithms for Decorrelation within Large Least Squares Problems

Inaugural–Dissertation zur
Erlangung des akademischen Grades
Doktor–Ingenieur (Dr.–Ing.)
der Hohen Landwirtschaftlichen Fakultät
der Rheinischen Friedrich–Wilhelms–Universität
zu Bonn

vorgelegt am 28. Januar 2008 von

Dipl.–Ing. Christian Siemes
aus Nettetal

Referent:	Prof. Dr. techn. W.–D. Schuh
1. Koreferent:	PD Dr.–Ing. Jürgen Kusche
2. Koreferent:	ao. Univ.–Prof. Mag. Dr. techn. Pail Roland

Tag der mündlichen Prüfung:	4. April 2008
Publikation:	Diese Dissertation ist auf dem Hochschulschriftenserver der ULB Bonn `http://hss.ulb.uni-bonn.de/diss_online` elektronisch publiziert.
Erscheinungsjahr:	2008

Abstract

The GOCE (Gravity Field and steady-state Ocean Circulation Explorer) mission is dedicated to the determination of the Earth's gravity field. During the mission period of at least one year the GOCE satellite will collect approximately 100 million highly correlated observations. The gravity field will be described in terms of approximately 70 000 spherical harmonic coefficients. This leads to a least squares adjustment, in which the design matrix occupies 51 terabytes while the covariance matrix of the observations requires 72 760 terabytes of memory. The very large design matrix is typically computed in parallel using supercomputers like the JUMP (Juelich Multi Processor) supercomputer in Jülich, Germany. However, such a brute force approach does not work for the covariance matrix. Here, we have to exploit certain features of the observations, e.g. that the observations can be interpreted as a stationary time series. This allows for a very sparse representation of the covariance matrix by digital filters.

This thesis is concerned with the use of digital filters for decorrelation within large least squares problems. First, it is analyzed, which conditions the observations must meet, such that digital filters can be used to represent their covariance matrix. After that, different filter implementations are introduced and compared with each other, especially with respect to the calculation time of filtering. This is of special concern, as for many applications the very large design matrix has to be filtered at least once. One special problem arising by the use of digital filters is the so-called warm-up effect. For the first time, methods are developed in this thesis for determining the length of the effect and for avoiding this effect. Next, a new algorithm is developed to deal with the problem of short data gaps within the observation time series. Finally, it is investigated which filter methods are best adopted for the application scenario GOCE, and several numerical simulations are performed.

Zusammenfassung

Die GOCE (Gravity Field and steady-state Ocean Circulation Explorer) Mission ist der Bestimmung des Erdschwerefeldes gewidmet. Während der Missionsdauer von mindestens einem Jahr wird der GOCE Satellit circa 100 Millionen hoch korrelierte Beobachtungen sammeln. Das Erdschwerefeld wird durch circa 70 000 sphärisch harmonische Koeffizienten beschrieben. Dies führt zu einem kleinste-Quadrate Ausgleich, wobei die Designmatrix 51 Terabytes benötigt während die Kovarianzmatrix der Beobachtungen 72 760 Terabytes erfordert. Die sehr große Designmatrix wird typischerweise parallel berechnet, wobei Supercomputer wie JUMP (Juelich Multi Processor) in Jülich (Deutschland) zum Einsatz kommen. Ein solcher Ansatz, bei dem das Problem durch geballte Rechenleistung gelöst wird, funktioniert bei der Kovarianzmatrix der Beobachtungen nicht mehr. Hier müssen bestimmte Eigenschaften der Beobachtungen ausgenutzt werden, z.B. dass die Beobachtungen als stationäre Zeitreihe aufgefasst werden können. Dies ermöglicht es die Kovarianzmatrix durch digitale Filter zu repräsentieren.

Diese Arbeit beschäftigt sich mit der Nutzung von digitalen Filtern zur Dekorrelation in großen kleinste-Quadrate Problemen. Zuerst wird analysiert, welche Bedingungen die Beobachtungen erfüllen müssen, damit digitale Filter zur Repräsentation ihrer Kovarianzmatrix benutzt werden können. Danach werden verschiedene Filterimplementierungen vorgestellt und miteinander verglichen, wobei spezielles Augenmerk auf die Rechenzeit für das Filtern gelegt wird. Dies ist von besonderer Bedeutung, da in vielen Anwendungen die sehr große Designmatrix mindestens einmal gefiltert werden muss. Ein spezielles Problem, welches beim Benutzen der Filter entsteht, ist der sogenannte Warmlaufzeiteffekt. Zum ersten Mal werden in dieser Arbeit Methoden entwickelt, um die Länge des Effekts zu bestimmen und um den Effekt zu vermeiden. Als Nächstes wird ein neuer Algorithmus zur Lösung des Problems von kurzen Datenlücken in der Beobachtungszeitreihe entwickelt. Schließlich wird untersucht, welche Filtermethoden man am besten für das Anwendungsszenario GOCE verwendet und es werden verschiedene numerische Simulationen durchgeführt.

Contents

1. Introduction

1.1 Motivation

Knowledge about the Earth's gravity field is one important component in the understanding of the Earth as a dynamic system. The mass distribution within the Earth system shapes the Earth's gravity field while any mass transports cause changes of the Earth's mass distribution. One special feature of interest is that the Earth's gravity field defines the geoid, a surface of equal gravity potential. This surface corresponds to the surface of the ocean at rest, i.e. without tides and currents. After the geoid is combined with altimetry data, which measures the actual height of the ocean surface, one can derive the ocean circulation. Besides ocean circulation there are many other fields which benefit from the accurate knowledge of the Earth's gravity field. Among them are geodynamics, sea level, ice mass balance and the global hydrologic water cycle. A good overview is given in Ilk et al. (2005).

The determination of the Earth's gravity field is a challenging task. In order to investigate the Earth's system on a global scale, one needs to collect precise and globally distributed observations. This can only be achieved by satellite missions. There are currently three satellite missions dedicated to the determination of the Earth's gravity field: CHAMP, GRACE and GOCE.

CHAMP CHAMP has been the first of these missions. In addition to measuring the Earth's gravity field, it was also designed to determine the Earth's magnetic field, ionosphere and troposphere (cf. Reigber et al. 2004). The CHAMP satellite was launched in Juli 2000. The measurement concept is satellite-to-satellite tracking in high-low mode (cf. for example Rummel et al. 2002). Its data allows for a resolution of the Earth's gravity field up to degree and order 60–70 in terms of spherical harmonics (cf. Mayer-Gürr et al. 2005, Reigber et al. 2003), which corresponds to the low frequency part of the Earth's gravity field.

GRACE The GRACE mission has a remarkable measurement concept: Two satellites fly 220 km apart, one behind the other on the same orbital track. This corresponds to satellite-to-satellite tracking in low-low mode. The gravity field is determined from measuring the distance between the two satellites, using GPS and a microwave ranging system. The twin satellites were launched in March 2002. The GRACE mission provides very precise measurements of the time variable gravity field as well as the low frequency parts of the static gravity field (cf. for example Mayer-Gürr 2006). Current gravity fields derived from GRACE data are resolved up to degree and order 140–180 (cf. ITG-GRACE03 2007, Reigber et al. 2005, Tapley et al. 2005).

GOCE The GOCE mission is dedicated to the very precise determination of the high frequency part of the static gravity field. The mission goal is to determine the Earth's gravity field at the highest possible spatial resolution and accuracy (cf. ESA 2000). The launch of the GOCE satellite is scheduled for May 2008. The GOCE satellite relies on a sensor fusion concept. The core instrument is a three-axis gradiometer, which consists of three orthogonally mounted pairs of accelerometers. This instrument will measure the high frequency part of the Earth's gravity field. The low frequency part of the Earth's gravity field will be determined by satellite-to-satellite tracking (SST) in high-low mode using the GPS and GLONASS (cf. ESA 1999).

The GRACE and GOCE mission can be considered as complementary (cf. Rummel et al. 2002). While the GRACE mission aims at the precise determination of the low frequency part of the static gravity field and the temporal variations of the gravity field, the GOCE mission has the goal to determine the high frequency part of the gravity field as precisely as possible. The processing of GRACE and GOCE mission data are both computationally very demanding. One reason is the very large number of observations, which are collected by the satellites. Another reason is the large number of parameters. Current gravity fields derived from GRACE data are described by approximately 33 000 spherical harmonic coefficients (cf. ITG-GRACE03 2007) resulting from a resolution up to degree and order 180. For gravity fields derived from GOCE data, the number of parameters may exceed 70 000 spherical harmonic coefficients, corresponding to a resolution of higher than degree and order 270. In both cases, the spherical harmonic coefficients are typically estimated by a least squares adjustment. The design matrices of both least squares adjustments need tens of terabytes of memory. Thus, both least squares adjustments can be characterized as large least squares adjustments. The computation of the design matrix is normally performed in parallel using computer clusters or supercomputers.

Besides the size of the design matrices within the least squares adjustment, the size of the covariance matrices is another problem. In case of the GOCE mission the number of observations will be approximately 100 000 000 resulting in a covariance matrix which would require more than 70 000 terabytes if it was fully stored. For such large matrices, parallel computing techniques are not sufficient to handle them anymore. Therefore, tailored algorithms are needed to incorporate the covariance matrices into the least squares adjustment.

In the case of the GOCE mission the SGG observations can be considered as stationary time series. Consequently, their covariance matrix is a Toeplitz matrix, i.e. a matrix, in which each descending diagonal from left to right is constant. For this reason it is possible to represent the covariance matrix of the SGG observations by digital filters. This thesis is concerned with the modeling of covariance matrices by digital filters and their integration within large least squares adjustments. Due to the large number of parameters, the extremely large number of observations and their high correlations, the SGG observations provide the best playground for this thesis. As the GOCE satellite has not been launched yet, the calculations within this thesis are based on simulated test data, resulting from an end-to-end simulator (cf. De Sanctis et al. 2002).

1.2 Current State of Research

Concerning the SGG observations of the GOCE mission, there are basically two methods documented in the literature, in which the covariance matrix is represented by digital filters: the PCGMA algorithm described by Schuh (1996) and the PCCG algorithm described by Klees et al. (2003). Both use a preconditioned conjugate gradient algorithm to determine the least squares solution. Furthermore, both use autoregressive moving-average (ARMA) filters to model the covariance matrix. The point, where the methods differ with respect to the use of digital filters, is the integration of the filtering into the adjustment procedure. Schuh (1996) follows the approach of decorrelating the system of observation equations by filtering. This requires the filtering of the large design matrix in each iteration of the conjugate gradient algorithm. In contrast to that, Klees et al. (2003) avoid filtering the design matrix in each iteration of the conjugate gradient algorithm. Here, the filtering is applied to the product of the design matrix and the parameter vector, which corresponds to the gradient of the residual sum of squares. Because Schuh (1996) filters the design matrix with tens of thousands of columns and Klees et al. (2003) the gradient, which is only a single vector, the filtering is much less computationally demanding for the approach of Klees et al. (2003). However, the approach of Klees et al. (2003) requires the design matrix to be computed twice, which is avoided by Schuh (1996). Which approach is less computationally demanding depends on whether the computation of the design matrix or the filtering of the design matrix needs more calculation time.

The design of decorrelating filters is broadly addressed in the literature. In the context of the GOCE mission Klees and Broersen (2002) introduced suitable methods for estimating the filter coefficients. Further improvements of the filter design are presented by Schuh (2002), who introduced notch filters in order to model peaks in the power spectral density of the measurement noise. However, the latter were not tested by simulations with respect to gravity field recovery. In the fields of time series analysis (with the background of econometrics) and signal processing (with the background of electrical engineering) there exists a great deal of articles and books concerned with the determination of digital filters. The methods can be roughly divided into time domain methods and frequency domain methods. A good overview over frequency methods is given by Pintelon et al. (1994) while time domain methods are discussed for example by Friedlander and Porat (1984) and Schlittgen and Streitberg (2001).

Besides the use of digital filters for the incorporation of the covariance matrix of the SGG observations, the PCCG and PCGMA algorithm are equipped with several features. The PCGMA algorithm for example has the following additional features:

Preconditioning In order to keep the number of iterations of the conjugate gradient algorithm low, preconditioning is required. Boxhammer (2006) developed a tailored numbering scheme for the parameters, the so-called free kite numbering scheme. By means of this numbering scheme it is possible to compute a tailored preconditioner, the so-called kite matrix. The work of Boxhammer (2006) was based on the work of Schuh (1996) on this subject (cf. also Boxhammer and Schuh 2006). Boxhammer (2006) showed that a suitable choice of the kite matrix reduces the necessary number of iterations by a factor of two compared to methods described by Schuh (1996).

Regularization In the context of the GOCE mission, state-of-the-art regularization methods are given by Metzler and Pail (2005) (cf. also Koch and Kusche 2002). Due to the sun-synchronous orbit of the GOCE satellite there are no measurements over the polar regions while the rim of the polar caps is characterized by a maximum data density. For this reason the zonal spherical harmonic coefficients are weakly determined. Metzler and Pail (2005) developed a regularization method tailored for this problem.

Monte Carlo variance component estimation In the case of the GOCE mission SGG and SST observations as well as the regularization information have to be combined in an optimal manner. Especially the relative weighting of the different groups of observations is of importance. Variance component estimation is a technique to determine the optimum relative weights (cf. Koch and Kusche 2002). Alkhatib (2007) showed, how variance component estimation can by integrated into the PCGMA algorithm based on Monte Carlo methods.

Monte Carlo estimation of the covariance matrix Besides the gravity field solution, given in terms of a set of spherical harmonic coefficients, its accuracy is of special interest. The accuracy and correlations of the estimated gravity field parameters are contained within the covariance matrix of the parameters, which is the inverse normal equation matrix of the least squares adjustment. However, the computation and inversion of the normal equation matrix is avoided by the PCGMA method, as it is computationally very demanding. Alkhatib (2007) developed a method for estimating the covariance matrix by means of Monte Carlo methods, which can be integrated into the PCGMA algorithm.

In practice, one special item of concern is the handling of data gaps, which is problematic, because the filtering relies on an uninterrupted observation time series. There are several strategies to solve this problem. One strategy is to simply stop the filtering at each data gap and to restart the filtering after the data gap. However, due to an effect, which is called the filter warm-up (cf. Schuh 2003), the first observations at the beginning of the filtering cannot be used within the least squares adjustment. If there are many data gaps and if the filter warm-up is not short, one loses many observations in this way. Another strategy is therefore not to stop the filtering at each data gap, which then requires to fill-in the data gaps. However, to compute suitable fill-in values for the data gaps is problematic, as shown by Klees and Ditmar (2004). Therefore, Klees and Ditmar (2004) favor another approach. Here, the problem of integrating the covariance matrix into the least squares adjustment is reduced to the solution of a linear equation system with the covariance matrix as equation matrix. The linear equation system is solved using the preconditioned conjugate gradient algorithm, whereas the digital filters are used for preconditioning. However, this strategy introduces much higher computational costs, which is confirmed by Klees and Ditmar (2004).

1.3 The Objectives of this Thesis

This thesis addresses the use of digital filters for decorrelation within large least squares problems. The developed algorithms and methods are described from a general viewpoint rather than from narrower viewpoint of only adjusting the SGG observations. For the sake of generality, moving-average (MA) filters as well as ARMA filters are considered. Furthermore, it is not assumed that a particular adjustment procedure, such as the conjugate gradient algorithm, is used. Only basic matrix operations within least squares adjustment operations are considered. Therefore, a general goal of this thesis is to describe problems and solutions in a way, such that the reader is provided with guidelines, which he may use for different application scenarios. In the chapter on the application scenario of the GOCE mission it is demonstrated, how these guidelines lead to the decision, which of the methods for filtering should be used. Moreover, the following issues are addressed in the chapters to follow.

1. *identification of well-defined conditions for the modeling of the covariance matrix by digital filters*
   To identify conditions, which must be met by the observations in order to be statistically allowed to apply digital filters for decorrelation, is important. Unless such conditions are well-defined, it is not clear, when the methods described in this thesis may be used for some application scenario. Therefore, this point is addressed at the beginning of this thesis in chapter 2.
2. *clear exposure of the interrelation of the covariance matrix and the filter coefficients*
   One question to be answered is: How to design the digital filters? In order to formulate the goal of the filter design, we have to expose the interrelation of the covariance matrix and the filter coefficients. This point is also addressed in chapter 2.
3. *comparison and development of numerically efficient implementations of digital filters*
   Within this thesis we consider both recursive and non-recursive digital filters. Each in both groups may be implemented by different methods. The methods to be compared range from straightforward implementations of the filter equations to implementations exploiting the fact that filtering corresponds to a convolution. Of special interest are block methods which either are well suited to make use of optimized libraries such as the ATLAS library or reduce the computational complexity such as the overlap-add method. The reason for this is that the large design matrix has to be filtered at least once within a least squares adjustment procedure. The implementations are introduced and compared in the first part of chapter 3.
4. *exploration of numerically efficient integration of the filtering into least square adjustment procedures*
   How to implement digital filters is one question, how to integrate them into least squares adjustment procedures is another. There are many different least squares adjustment procedures. To investigate all of them in detail would be very extensive. However, there exists a small set of typical matrix operations, which most of the least squares adjustment procedures make use of. Such a typical matrix operation is for example the computation of the gradient of the residual sum of squares or the normal equation matrix. In the last part of chapter 3 we explore numerically efficient ways of integrating the filtering into least square adjustment procedures based on such typical matrix operations.
5. *accurate investigation of the warm-up effect of digital filters*
   The first filtered values are not well decorrelated and must therefore not be used within the least squares adjustment. This effect is called the filter warm-up. If there are many larger data gaps within the observation time series, the digital filter is stopped at each data gap and restarted behind it. In this case many observations may be lost for the adjustment due to the filter warm-up. For this reason the filter warm-up will be thoroughly investigated in the first part of chapter 4. Furthermore, if it is possible to avoid the filter warm-up, a numerical efficient method will be developed to do so.
6. *development of a numerically efficient algorithm for computing suitable fill-in values for data gaps*
   The computation of suitable fill-in values for data gaps is not as simple as one might guess. We have to take into account that the observations consist of a deterministic part and a stochastic part. A numerically efficient algorithm for computing suitable fill-in values for data gaps is developed in the last part of chapter 4.
7. *selection and compilation of a set of suitable methods for determining the filter coefficients*
   The determination of the filter coefficients is already broadly addressed in the literature. Even in the

context of the GOCE mission there exist some articles on this subject. However, there are still some open questions. For example, by which methods the parameters of notch filter are best estimated. Chapter 5 is dedicated to the design of digital filters.

8. *identification of the best filter implementation and integration for the application scenario*

   The application scenario of this thesis is the adjustment of the SGG observations of the GOCE mission. In chapter 6 it will be described in detail, which of the filter implementations is most useful for this application scenario. Furthermore, it is discussed in detail, which is the best way for integrating the filtering into the least-squares adjustment procedure PCGMA.

9. *comparative study of different filter designs with respect to gravity field recovery from GOCE data*

   As the design of the digital filters is improved within this thesis, the question arises, how big the benefit of the improved design is. In chapter 6 a comparative study of different filter designs with respect to gravity field recovery from GOCE data is performed.

10. *summary of major results and conclusions*

    For the sake of clarity the major results and conclusions are summarized in chapter 7. Within this chapter it is also highlighted, which new algorithms and methods are developed in this thesis.

2. Filtering and Least Squares Methods

In this chapter the basic interconnections between least squares problems and digital filtering are established. We start with the well-known Gauss-Markov model which provides the theoretical basis for least squares problems. After that, we formulate conditions for the observations, which must be met for the applicability of the methods depicted in this thesis. We will focus on large least squares problems, i.e. problems in which the number of observations is large and the number of parameters may be large, too. Therefore, the main challenge is caused by the fact that, if the number of observations is large, their covariance matrix cannot be stored.

Having discussed the Gauss-Markov model and the conditions for the applicability of the methods depicted in this thesis, we proceed with describing three different approaches of representing the covariance matrix of the observations:

- Toeplitz approach
- autoregressive (AR) process approach
- autoregressive moving average (ARMA) process approach

All three approaches are efficient with respect to both storage requirements and computational costs. After that, we focus on the basics of digital signal processing. These basics provide us with the necessary theoretical background for the following chapters. Furthermore, within the last section of this chapter, the link between the coefficients of AR processes and ARMA processes and the covariance matrix of the observations is established. With this knowledge, principles are formulated which must be followed during the design of these filters, i.e. when the filter coefficients are estimated. Last but not least, some restrictions of the filter design are investigated.

2.1 The Gauss-Markov Model

Before we start with the formulation of the Gauss-Markov model, we should clarify that we distinguish between random variables, true values and realizations of random variables.

Random variable Random variables are used to describe the statistical properties of specific variables. They will be denoted in capital letters such as $\mathcal{L}$.

True value Normally, we do not have access to the true values of random variables. Nevertheless we use them for modeling. We will denote true values in Greek lower case letters such as λ.

Realization Observations and parameters, which are derived from the observations, are considered as realizations of random variables. They will be denoted in lower case letters such as l.

Using this notation, we can start formulating the Gauss-Markov model. The main reference for this section is Koch (1997), chapter 3.2. The Gauss-Markov model is defined by

$$\boldsymbol{E}\{\boldsymbol{\mathcal{L}}\} = \boldsymbol{A}\boldsymbol{x}, \quad \boldsymbol{\Sigma}\{\boldsymbol{\mathcal{L}}\} = \boldsymbol{\Sigma}_{\boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{L}}}, \tag{2.1}$$

wherein $\boldsymbol{\mathcal{L}}$ is the vector of observations, $\boldsymbol{A}$ is the so-called design matrix, $\boldsymbol{x}$ is the unknown vector of parameters and $\boldsymbol{\Sigma}_{\boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{L}}}$ is the covariance matrix of the observation vector. The observations are often assumed to follow a multivariate normal distribution (cf. Koch 1997, p. 127)

$$\boldsymbol{\mathcal{L}} \sim N(\boldsymbol{A}\boldsymbol{x}, \boldsymbol{\Sigma}_{\boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{L}}}). \tag{2.2}$$

The number of observations is generally larger than the number of parameters. Because of the unavoidable measurement errors, the system of observation equations

$$\boldsymbol{l} \neq \boldsymbol{A}\boldsymbol{x} \tag{2.3}$$

is then inconsistent. In order to fix this inconsistency, we introduce the error vector $\boldsymbol{e}$ into the observation equations

$$\boldsymbol{l} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e}. \tag{2.4}$$

The measurement errors are then modeled by the random vector $\boldsymbol{\mathcal{E}}$, which stochastically describes the deviation of the observation vector $\boldsymbol{l}$ to its true value $\boldsymbol{\lambda}$ by

$$\boldsymbol{E}\{\boldsymbol{\mathcal{L}}\} = \boldsymbol{E}\{\boldsymbol{A}\boldsymbol{x} + \boldsymbol{\mathcal{E}}\} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{E}\{\boldsymbol{\mathcal{E}}\}. \tag{2.5}$$

Combining (2.5) with (2.1) we find

$$\boldsymbol{E}\{\boldsymbol{\mathcal{E}}\} = \boldsymbol{0}, \quad \boldsymbol{\Sigma}\{\boldsymbol{\mathcal{E}}\} = \boldsymbol{\Sigma}_{\boldsymbol{\mathcal{E}}\boldsymbol{\mathcal{E}}}. \tag{2.6}$$

$\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{E}}$ both describe the stochastic properties of the measurement errors. They only differ from each other in their expectation values. Therefore, their covariance matrices equal each other and we can abbreviatedly write

$$\boldsymbol{\Sigma}_{\boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{L}}} = \boldsymbol{\Sigma}_{\boldsymbol{\mathcal{E}}\boldsymbol{\mathcal{E}}} = \boldsymbol{\Sigma}. \tag{2.7}$$

One problem within this thesis is that the covariance matrix $\boldsymbol{\Sigma}$ is very large. For this reason we will model the covariance by digital filters which mathematically corresponds to replacing it by the product of two filter matrices. Another problem is the computation of the least squares estimates $\widetilde{\boldsymbol{x}}$ for the unknown parameters $\boldsymbol{x}$. This is subject to least squares adjustment procedures. To find numerically efficient ways of connecting the filtering and the least squares adjustment procedures poses a third problem. A direct formula for obtaining the least squares estimates $\widetilde{\boldsymbol{x}}$ is

$$\widetilde{\boldsymbol{x}} = (\boldsymbol{A}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{A})^{-1}\boldsymbol{A}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{l}, \quad \boldsymbol{\Sigma}\{\widetilde{\boldsymbol{\mathcal{X}}}\} = (\boldsymbol{A}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{A})^{-1}. \tag{2.8}$$

Herein, the random variable $\widetilde{\boldsymbol{\mathcal{X}}}$ describes the stochastic properties of the least squares estimates $\widetilde{\boldsymbol{x}}$. The least squares estimates have the property to be the best linear unbiased estimator (BLUE) of the unknown parameters $\boldsymbol{x}$ (cf. Koch 1997, p. 171). With the help of $\widetilde{\boldsymbol{x}}$, we then find that the adjusted observations $\widetilde{\boldsymbol{l}}$ and residuals $\boldsymbol{v}$ are given by

$$\widetilde{\boldsymbol{l}} = \boldsymbol{A}\widetilde{\boldsymbol{x}}, \quad \boldsymbol{\Sigma}\{\widetilde{\boldsymbol{\mathcal{L}}}\} = \boldsymbol{A}(\boldsymbol{A}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{A})^{-1}\boldsymbol{A}^\top \tag{2.9}$$

and

$$\boldsymbol{v} = \widetilde{\boldsymbol{l}} - \boldsymbol{l}, \quad \boldsymbol{\Sigma}\{\boldsymbol{\mathcal{V}}\} = \boldsymbol{\Sigma} - \boldsymbol{A}(\boldsymbol{A}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{A})^{-1}\boldsymbol{A}^\top. \tag{2.10}$$

Note, that the covariance matrix of the measurement errors, which we want to model by digital filters, is the sum of the covariance matrix of the adjusted observations and the residuals, i.e.

$$\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{\widetilde{\boldsymbol{\mathcal{L}}}\widetilde{\boldsymbol{\mathcal{L}}}} + \boldsymbol{\Sigma}_{\boldsymbol{\mathcal{V}}\boldsymbol{\mathcal{V}}} \tag{2.11}$$

Furthermore, it holds that

$$\operatorname{rank}(\boldsymbol{\Sigma}_{\boldsymbol{\mathcal{V}}\boldsymbol{\mathcal{V}}}) = N - M \quad \text{and} \quad \operatorname{rank}(\boldsymbol{\Sigma}_{\widetilde{\boldsymbol{\mathcal{L}}}\widetilde{\boldsymbol{\mathcal{L}}}}) = M, \tag{2.12}$$

whereas M is the number of parameters and N is the number of observations. Formula (2.12) means that both covariance matrices are positive semi-definite (cf. Koch 1997, pp. 174, 177), which turns out to be problematic for the following reason. We do not know the covariance matrix $\boldsymbol{\Sigma}$ of the observations. However, we have to determine the filter coefficients such that they model this covariance matrix. For this reason we typically use the residuals $\boldsymbol{v}$ as an approximation of the measurement errors $\boldsymbol{e}$ and estimate the filter coefficients such that the residuals $\boldsymbol{v}$ are decorrelated. Unfortunately, this corresponds to modeling $\boldsymbol{\Sigma}_{\mathcal{VV}}$ instead of $\boldsymbol{\Sigma}$, so that we systematically underestimate the covariances of the measurement errors. The reason for this is that covariance matrices are positive definite or positive semi-definite matrices, which means for example that the diagonal elements of $\boldsymbol{\Sigma}_{\mathcal{VV}}$ must be smaller or of equal size than the diagonal elements of $\boldsymbol{\Sigma}$ in equation (2.11). However, if we have no a priori information about the covariance matrix $\boldsymbol{\Sigma}$ and no access to the measurement errors $\boldsymbol{e}$, then designing the digital filters such that they decorrelate the residuals $\boldsymbol{v}$ is the best we can do.

In the section about the integration of the filtering into least squares adjustment procedures we will use the term normal equations. It should be mentioned here that the normal equations can be obtained by minimizing the residual sum of squares

$$\boldsymbol{v}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{v} \longrightarrow \text{Min.} \tag{2.13}$$

(cf. Koch 1997, p. 174), which leads to the least squares estimates $\widetilde{\boldsymbol{x}}$ in formula (2.8). The normal equations are

$$\boldsymbol{A}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{A} \boldsymbol{x} = \boldsymbol{A}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{l}, \tag{2.14}$$

wherein

$$\boldsymbol{N} = \boldsymbol{A}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{A} \tag{2.15}$$

is the normal equation matrix and

$$\boldsymbol{n} = \boldsymbol{A}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{l} \tag{2.16}$$

is the right hand side.

2.2 Boundary Conditions for the Observations

As mentioned in the introduction, the preferred method for the integration of the covariance matrix $\boldsymbol{\Sigma}$ into a least squares adjustment procedure is the utilization of decorrelating digital filters. This and the other presented methods in this thesis are not generally applicable to the Gauss-Markov model, which connects the stochastic modeling of the covariance matrix of the observations by digital filters and least squares adjustment procedures. In this section we will introduce the necessary conditions which have to hold for the applicability of these methods. Furthermore, we will discuss additional conditions, which are often met for problems in which these methods are considered. The additional conditions affect certain practical matters like the warm-up length or the implementation of the filters.

Necessary Conditions

The following conditions for the observations must hold for the applicability of the methods within this thesis. Some conditions are formulated for the residuals instead of for the observations. For the sake of a clarity, we will again distinguish between random variables like $\mathcal{L}_n$ and realizations l_n.

Time Series A chronologically sorted series of observations $l_0, l_1, \ldots, l_{N-1}$, gathered in the observation vector $\boldsymbol{l}$, is called a time series (cf. Schlittgen and Streitberg 2001, p. 1). Often, we can imagine that we could have obtained earlier observations $l_{-1}, l_{-2}, \ldots, l_{-\infty}$ or subsequent observations $l_N, l_{N+1}, \ldots, l_\infty$ (cf. Hamilton 1994, p. 25). Not to know the earlier or subsequent observations poses a special problem, which requires an adequate handling. We will discuss this in detail in section 4.1. In many practical circumstances the time interval between two succeeding observations is the same for all observations. That is the case, if t_n is the time of the observation l_n and t_{n+1} is the time of the succeeding observation l_{n+1} and

$$\Delta t = t_{n+1} - t_n \tag{2.17}$$

is a constant value for all n. Though this latter property is not necessary for observations to be a time series, we will require it as well in order to be able to do meaningful spectral analyses easily.

Normal Distribution The observations follow the multivariate normal distribution $N(\boldsymbol{Ax}, \boldsymbol{\Sigma})$. In formula (2.2), we have this condition already assumed to be met. Therefore, the residual time series v_n follows the multivariate normal distribution

$$\boldsymbol{v} \sim N(\boldsymbol{0}, \boldsymbol{\Sigma}), \tag{2.18}$$

as stated in (2.6). A time series, which follows the normal distribution is said to be Gaussian. The following conditions aim at the residual time series v_n rather than at the observations time series l_n. The reason for this is, that the expectation $E\{\mathcal{L}_n\}$ is not independent of the index n, since the observations contain a deterministic trend. However, our goal is to decorrelate the observation time series l_n and therefore only the covariance matrix $\boldsymbol{\Sigma}_{\mathcal{LL}}$ of $\mathcal{L}_n$ is relevant. Thus, we can switch from the observation time series l_n to the residual time series v_n, whose covariance matrix $\boldsymbol{\Sigma}_{\mathcal{VV}}$ equals $\boldsymbol{\Sigma}_{\mathcal{LL}}$ as stated in (2.7).

Stationarity The residual time series v_n is stationary. This means, that the probability distribution of v_{n+m} is independent of the index m for all n (cf. HAMILTON 1994, pp. 45, 46). As a consequence the moments of the probability distribution such as mean, variance, skewness or kurtosis do not chance over time. If only the first and second moments do not change over time, the time series is said to be weakly stationary. This is the case, if

$$E\{\mathcal{V}_n\} = \nu \quad \text{for all } n \tag{2.19}$$

and

$$E\{(\mathcal{V}_n - E\{\mathcal{V}_n\})(\mathcal{V}_{n+k} - E\{\mathcal{V}_{n+k}\})\} = \gamma_k \quad \text{for all } n \text{ and any } k, \tag{2.20}$$

wherein ν is the expectation value and γ_k is the autocovariance function of the time series v_n (cf. HAMILTON 1994, p. 45). As $E\{\mathcal{V}_n\} = 0$ according to (2.6), the condition (2.19) is always met for the residual time series of a Gauss-Markov model. Therefore, equation (2.20) becomes

$$E\{\mathcal{V}_n \mathcal{V}_{n+k}\} = \gamma_k \quad \text{for all } n \text{ and any } k, \tag{2.21}$$

and poses the actual condition. If in addition the time series v_n follows the normal distribution, then weakly stationarity coincides with stationarity (cf. HAMILTON 1994, p. 46). Thus, within this thesis we use the term weakly stationarity tantamount to the term stationary, as we only consider Gaussian time series.

Stationarity is important, because it forms the shape of the covariance matrix $\boldsymbol{\Sigma}$ of the residuals v_n. With equation (2.20) we find that $\boldsymbol{\Sigma}$ has a Toeplitz structure

$$\boldsymbol{\Sigma} = \begin{bmatrix} \gamma_0 & \gamma_1 & \cdots & \gamma_{N-2} & \gamma_{N-1} \\ \gamma_1 & \gamma_0 & \gamma_1 & & \gamma_{N-2} \\ \vdots & \gamma_1 & \gamma_0 & \ddots & \vdots \\ \gamma_{N-2} & & \ddots & \ddots & \gamma_1 \\ \gamma_{N-1} & \gamma_{N-2} & \cdots & \gamma_1 & \gamma_0 \end{bmatrix}. \tag{2.22}$$

This structure permits the application of algorithms based on the fast Fourier transform (FFT) and other fast algorithms.

Ergodicity The observation time series is ergodic. Let us suppose that we possess not a single realization v_n, but an ensemble of realizations $v_n^{(0)}$, $v_n^{(1)}$, …, $v_n^{(J-1)}$. Ergodicity now means that the ensemble averages and the time averages converge against each other (cf. HAMILTON 1994, pp. 46, 47). For example a time series v_n is said to be ergodic for the mean if

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} v_n^{(j)} = \lim_{J\to\infty} \frac{1}{J} \sum_{j=0}^{J-1} v_n^{(j)} = E\{\mathcal{V}_n\}. \tag{2.23}$$

However, we typically possess only one realization v_n. Therefore, we need a criterion which does not depend on ensemble averages. A sufficient criterion is the following: If the time series v_n is Gaussian and its covariance function γ_j is absolutely summable, i.e.

$$\sum_{j=0}^{\infty} |\gamma_j| < \infty, \tag{2.24}$$

then the time series v_n is ergodic (cf. HAMILTON 1994, p. 47). This condition implies that the correlations numerically decay for some index j to zero, because $|\gamma_j| \to 0$ for $j \to \infty$ (cf. FORSTER 1983, p. 38, theorem 2 and 3). Ergodicity is important, because if a time series is not ergodic, we cannot recover the parameters describing the underlying process.

Correlation Length The correlation length of the residual time series is smaller than the length of the residual time series. Within this thesis, the correlation length is defined to be the smallest index, for which the autocovariance function numerically approaches zero. Therefore, the correlation length is the smallest value j for which

$$|\gamma_k| < \varepsilon \quad \text{for all } \ k \geq j \tag{2.25}$$

holds, wherein ε a small positive number representing numerical zero (ε is used in this sense throughout this thesis). Thus, the condition states that $N > \min(j)$, wherein N is the length of the residual time series and $\min(j)$ is the correlation length. It follows, that the covariance matrix $\mathbf{\Sigma}$ of the residuals approximately has a banded structure.

$$\mathbf{\Sigma} \approx \begin{bmatrix} \gamma_0 & \cdots & \gamma_{\min(j)} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \gamma_{\min(j)} & \ddots & \vdots \\ \gamma_{\min(j)} & \cdots & \gamma_0 & & \ddots & 0 \\ 0 & \ddots & & \ddots & & \gamma_{\min(j)} \\ \vdots & \ddots & \gamma_{\min(j)} & \cdots & \gamma_0 & \vdots \\ 0 & \cdots & 0 & \gamma_{\min(j)} & \cdots & \gamma_0 \end{bmatrix} \tag{2.26}$$

Note, that ergodicity already implies that the correlations decay to numerical zero. However, this condition is more restrictive. This condition is also important, because if it is not met, we cannot recover the parameters of the underlying process and the warm-up length of the filter exceeds the length of the time series.

Additional Conditions

Number of Observations and Parameters The methods within this thesis are only considered, when the number of the observations is large. Then, it is not possible to store the covariance matrix $\mathbf{\Sigma}$ of the observations within the memory of a normal computer, even when it has a banded structure. For small problems, where the number of observations is small, $\mathbf{\Sigma}$ can be stored within the working memory and its integration into least squares adjustment procedures can be performed straightforward.

Within adjustment procedures, we need to apply $\boldsymbol{\Sigma}$ at least once to the design matrix $\boldsymbol{A}$, if the normal equation matrix cannot be approximated by means of analytical formulas. If not only the number of observations is large, but also the number of parameters, then the design matrix $\boldsymbol{A}$ is large, too. Therefore, applying $\boldsymbol{\Sigma}$ to $\boldsymbol{A}$ can be a computationally time-consuming task. In this case, it is desirable to use computationally efficient algorithms. Chapter 3 deals with this problem.

High Correlations The observations can be highly correlated. If this is the case, then the entries of the covariance matrix $\boldsymbol{\Sigma}$ of the observations decay only very slowly to zero with increasing distance from the main diagonal. For some methods presented in this thesis, this implies an increased computational effort. Some methods may even be ruled out because of their also increased memory requirements.

2.3 Compressed Representations of the Covariance Matrix

Within this thesis the focus lies on least squares problems in which the number of observations is large. Therefore, the representation of the covariance matrix of the observations in (2.1) must be chosen with care. It has to be efficient with respect to both memory requirements and computational costs. In this section we introduce three efficient approaches of representing the covariance matrix of the observations. These three approaches rely on the conditions formulated in section 2.2. They are described for example by Schuh (1996) (chapter 3) or by Ditmar et al. (2003) or by Klees et al. (2003). In this section we concentrate on the relation of these approaches to the covariance matrix of the observations. How they can be implemented and integrated into the Gauss-Markov model (2.1) will be discussed later on in chapter 3.

Toeplitz Approach

This approach exploits the stationarity of the observations. As already stated in (2.22), the covariance matrix of the observations has a Toeplitz structure, if the observations are stationary. Therefore, the covariance matrix can be simply represented by its first row. This is the most straightforward representation. Another condition in section 2.2 states that the correlation length must be shorter than the number of observations. Hence, it is possible to store only the first non-zero part of the first row of the covariance matrix, whereas the term non-zero is regarded from a numerical point of view.

AR Process Approach

Another way of representing $\boldsymbol{\Sigma}$ is to model the stochastic behavior of the residuals $\boldsymbol{\mathcal{V}}$ in (2.6) by a Gaussian autoregressive (AR) process. Such a process is described by

$$\mathcal{Y}_n = \sum_{k=0}^{P} h_k \mathcal{X}_{n-k} \quad \text{with } \boldsymbol{\mathcal{Y}} \sim N(\mathbf{0}, \boldsymbol{I}), \tag{2.27}$$

wherein $\boldsymbol{I}$ denotes the identity matrix (cf. Hamilton 1994, p. 58). Herein, $\mathcal{X}_n$ is tantamount to $\mathcal{V}_n$. We deliberately do not use $\mathcal{V}_n$ within this equation. The reason for this is derived in the following.

In principle, such a process is defined for all $n \in \mathbb{N}$. However, the observations are only known for $n = 0, 1, \ldots, N-1$. Therefore, we define

$$\boldsymbol{\mathcal{X}} = \left[\mathcal{X}_0 \cdots \mathcal{X}_{N-1}\right]^\top \quad \text{and} \quad \boldsymbol{\mathcal{Y}} = \left[\mathcal{Y}_0 \cdots \mathcal{Y}_{N-1}\right]^\top, \tag{2.28}$$

whereas all $\mathcal{X}_n$ with $n < 0$ in (2.27) are set to zero, which is commonly performed in practice (cf. Schuh 1996, pp. 36,37). Hence, we find

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{H}\boldsymbol{\mathcal{X}}, \tag{2.29}$$

wherein

$$\boldsymbol{H} = \begin{bmatrix} h_0 & 0 & \cdots & 0 \\ h_1 & h_0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ h_{N-1} & h_{N-2} & \cdots & h_0 \end{bmatrix}. \tag{2.30}$$

As matrix $\boldsymbol{H}$ is a lower triangular matrix with Toeplitz structure, it is invertible if $h_0 \neq 0$ (cf. Koch 1997, pp. 23–26). We denote the inverse of $\boldsymbol{H}$ by $\boldsymbol{G}$, i.e.

$$\boldsymbol{H} = \boldsymbol{G}^{-1}. \tag{2.31}$$

According to equation (2.29) we find

$$\boldsymbol{\mathcal{X}} = \boldsymbol{G}\boldsymbol{\mathcal{Y}} \tag{2.32}$$

for the filtering (cf. Burrus 1972), wherein

$$\boldsymbol{G} = \begin{bmatrix} g_0 & 0 & \cdots & 0 \\ g_1 & g_0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ g_{N-1} & g_{N-2} & \cdots & g_0 \end{bmatrix} \tag{2.33}$$

Matrix $\boldsymbol{G}$ is also a lower triangular matrix with Toeplitz structure. How it can be computed without inverting $\boldsymbol{H}$, is shown in section 2.4. Applying variance propagation (cf. Koch 1997, p. 108) we obtain

$$\boldsymbol{\Sigma}\{\boldsymbol{\mathcal{X}}\} = \boldsymbol{G}\boldsymbol{\Sigma}\{\boldsymbol{\mathcal{Y}}\}\boldsymbol{G}^\top = \boldsymbol{G}\boldsymbol{G}^\top, \tag{2.34}$$

because $\boldsymbol{\Sigma}\{\boldsymbol{\mathcal{Y}}\} = \boldsymbol{I}$ according to (2.27). When designing the filter, our goal is to determine the filter coefficients h_k, such that

$$\boldsymbol{\Sigma}\{\boldsymbol{\mathcal{X}}\} = \boldsymbol{G}\boldsymbol{G}^\top \approx \boldsymbol{\Sigma}\{\boldsymbol{\mathcal{V}}\} = \boldsymbol{\Sigma}\{\boldsymbol{\mathcal{L}}\}. \tag{2.35}$$

However, equality cannot be achieved, because $\boldsymbol{G}\boldsymbol{G}^\top$ does not generally posses a Toeplitz structure while $\boldsymbol{\Sigma}$ does. This is the reason why we deliberately did not use $\mathcal{V}_n$ within equation (2.27). The source of this problem is found in setting all $\mathcal{X}_n$ to zero for $n < 0$. A detailed discussion of this problem and the problem's solution is given in chapter 4 of this thesis.

ARMA Process Approach

The third way of representing the covariance matrix of the observations is to model the stochastic behavior of the residuals $\boldsymbol{\mathcal{V}}$ by a Gaussian autoregressive moving-average (ARMA) process. The advantage of this approach is, that this representation is much more compressed and therefore less memory demanding than the Toeplitz approach or the AR process approach. The drawbacks are, that the filter design and the filter implementation are more complicated than the implementation of the other two approaches.

A Gaussian ARMA process is defined by

$$\mathcal{Y}_n = \sum_{k=0}^{P} b_k \mathcal{X}_{n-k} + \sum_{j=1}^{Q} a_j \mathcal{Y}_{n-j} \quad \text{with } \boldsymbol{\mathcal{Y}} \sim N(\boldsymbol{0}, \boldsymbol{I}), \tag{2.36}$$

wherein $\boldsymbol{I}$ denotes the identity matrix (cf. Hamilton 1994, p. 58). As for the AR process approach, the observations are only known for $n = 0, 1, \ldots, N-1$. Thus, we define again

$$\boldsymbol{\mathcal{X}} = \left[\mathcal{X}_0 \cdots \mathcal{X}_{N-1}\right]^\top \quad \text{and} \quad \boldsymbol{\mathcal{Y}} = \left[\mathcal{Y}_0 \cdots \mathcal{Y}_{N-1}\right]^\top, \tag{2.37}$$

whereas all $\mathcal{X}_n$ and $\mathcal{Y}_n$ with $n < 0$ in (2.27) are set to zero. Using these definitions and rearranging (2.36), we find

$$\boldsymbol{A}\boldsymbol{\mathcal{Y}} = \boldsymbol{B}\boldsymbol{\mathcal{X}}, \tag{2.38}$$

wherein

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 & 0 \\ -a_1 & 1 & \ddots & \vdots & & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \cdots & 0 & 0 \\ -a_Q & \cdots & -a_1 & 1 & \ddots & \vdots & \vdots \\ 0 & \ddots & & \ddots & \ddots & 0 & 0 \\ \vdots & \ddots & -a_Q & \cdots & -a_1 & 1 & 0 \\ 0 & \cdots & 0 & -a_Q & \cdots & -a_1 & 1 \end{bmatrix} \quad \text{and} \quad \boldsymbol{B} = \begin{bmatrix} b_0 & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots & & \vdots \\ b_P & \cdots & b_0 & 0 & \cdots & 0 \\ 0 & \ddots & & \ddots & \ddots & \vdots \\ \vdots & \ddots & b_P & \cdots & b_0 & 0 \\ 0 & \cdots & 0 & b_P & \cdots & b_0 \end{bmatrix}. \tag{2.39}$$

As matrix $\boldsymbol{A}$ is lower triangular matrix and its main diagonal is non-zero, it is invertible (cf. KOCH 1997, pp. 23–26). Therefore, we find

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{\mathcal{X}}. \tag{2.40}$$

Apparently, the ARMA process approach is connected to the AR process approach by

$$\boldsymbol{H} = \boldsymbol{A}^{-1}\boldsymbol{B}, \tag{2.41}$$

wherein $\boldsymbol{H}$ is lower triangular matrix with Toeplitz structure as in (2.30). If $b_0 \neq 0$, matrix $\boldsymbol{B}$ is invertible, because it is a lower triangular matrix with Toeplitz structure (cf. KOCH 1997, pp. 23–26). Then, we find

$$\boldsymbol{\mathcal{X}} = \boldsymbol{B}^{-1}\boldsymbol{A}\boldsymbol{\mathcal{Y}} \tag{2.42}$$

and it follows

$$\boldsymbol{G} = \boldsymbol{B}^{-1}\boldsymbol{A}, \tag{2.43}$$

wherein $\boldsymbol{G}$ is lower triangular matrix with Toeplitz structure as in (2.33). If matrix $\boldsymbol{A}$ and matrix $\boldsymbol{B}$ are given, we can compute matrix $\boldsymbol{H}$ and if $b_0 \neq 0$ also matrix $\boldsymbol{G}$, which indicates that we can convert a Gaussian ARMA process into a Gaussian AR process. These two approaches are obviously closely associated with each other. In section 2.4 we will find more details of this interconnection. Within the filter design, our goal is to determine the filter coefficients a_j and b_k such that

$$\boldsymbol{\Sigma}\{\boldsymbol{\mathcal{X}}\} = \boldsymbol{G}\boldsymbol{G}^\top = \boldsymbol{B}^{-1}\boldsymbol{A}\boldsymbol{A}^\top\boldsymbol{B}^{-\top} \approx \boldsymbol{\Sigma}\{\boldsymbol{\mathcal{V}}\} = \boldsymbol{\Sigma}\{\boldsymbol{\mathcal{L}}\}. \tag{2.44}$$

As for the AR process approach, equality can never be achieved, because $\boldsymbol{G}$ is a lower triangular matrix and therefore $\boldsymbol{\Sigma}\{\boldsymbol{\mathcal{X}}\}$ does not posses a Toeplitz structure while $\boldsymbol{\Sigma}\{\boldsymbol{\mathcal{L}}\}$ does. Here, the reason for this is, that $\mathcal{X}_n$ and $\mathcal{Y}_n$ with $n < 0$ in (2.27) are set to zero. Similar to the AR process approach, a detailed discussion of this problem and the problem's solution is given in chapter 4 of this thesis.

2.4 Selected Basics of Digital Signal Processing

This section provides the necessary theory of digital filters which can be found in the standard literature on signal processing and time series analysis like OPPENHEIM and SCHAFER (1999), HAMILTON (1994), BOX and JENKINS (1970) and SCHLITTGEN and STREITBERG (2001). Only those elements are described which are relevant for the application of the Toeplitz approach, AR process approach and ARMA process approach in section 2.3. It constitutes the basis for the implementation of these approaches. Furthermore, this section tries to answer the following questions.

- What is the relationship between the filter coefficients and the covariance matrix of the observations? The answer to this question provides the interconnections between the Toeplitz, AR and ARMA approach. Moreover, the practical computation of the matrices $\boldsymbol{G}$ in (2.33) und $\boldsymbol{H}$ in (2.30) will be described.
- Which criterion is relevant for the design of filters?
The relevant quantity is the power spectral density which is the spectral representation of the autocovariance function in (2.20). Therefore, the criterion is the quality of the adjustment of the power spectral density of the filter to the power spectral density of the residuals $\boldsymbol{\mathcal{V}}$. Furthermore, the process should be stationary, because it models a stationary time series, and the filters should be stable which means that for a bounded filter input the filter output is bounded, too.
- Which restrictions do we have to accept for the design of filters?
A high quality of the adjustment of a filter to the spectral noise characteristics of the residuals comes along with a long warm-up of this filter. Furthermore, a high resolution of the power spectral density of the residuals unavoidably leads to a low resolution in the time domain.

Fourier Transform

The stochastic characteristics of the residuals can be interpreted more easily in the frequency domain than in the time domain. The Fourier transform is therefore a very important tool for analyzing the residuals. Often, only a finite set of values $x_0, x_1, \ldots, x_{N-1}$ of a time series x_n is available. After assuming that this finite set of values is repeated periodically, i.e.

$$\ldots, x_0, \ldots, x_{N-1}, x_0, \ldots, x_{N-1}, \ldots, \tag{2.45}$$

the Fourier transform becomes the discrete Fourier transform (DFT). The fast Fourier transform (FFT) is a very efficient algorithm for evaluating the discrete Fourier transform of such periodic time series. Many algorithms presented in this thesis make use of the FFT. Therefore, we start this section with the Fourier transform of time series and the DFT.

The Fourier transform of a time series x_n is defined by

$$X(\omega) = \mathcal{F}\{x_n\} = \sum_{n=-\infty}^{\infty} x_n e^{-in\omega}, \tag{2.46}$$

wherein ω is the normalized angular frequency (cf. Oppenheim and Schafer 1999, p. 51). This equation is usually evaluated for $\omega \in [-\pi, \pi]$, though it may be evaluated for any interval of width 2π, because $X(\omega)$ is periodical with a period of 2π (cf. Oppenheim et al. 1983, pp. 308, 309). The corresponding inverse Fourier transform (cf. Oppenheim and Schafer 1999, p. 51) is defined by

$$x_n = \mathcal{F}^{-1}\{X(\omega)\} = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\omega) e^{in\omega} d\omega. \tag{2.47}$$

In practice, the time series x_n is only given for $n = 0, 1, \ldots, N-1$. Therefore, we define

$$\boldsymbol{x} = \begin{bmatrix} x_0 & x_1 & \ldots & x_{N-1} \end{bmatrix}^\top. \tag{2.48}$$

Then, the discrete Fourier transform (DFT) of the vector $\boldsymbol{x}$ is denoted by

$$\boldsymbol{y} = \mathcal{F}\{\boldsymbol{x}\}. \tag{2.49}$$

The application of the DFT implies, the elements $\boldsymbol{x}$ are repeated periodically as in (2.45) (cf. Brigham 1988, chapter 6). Therefore it matters, whether we write $\mathcal{F}\{x_n\}$ for time series x_n, which are generally non-periodically, or $\mathcal{F}\{\boldsymbol{x}\}$ for vectors, which implies that the corresponding time series is periodic according to (2.45). Each element of vector $\boldsymbol{y}$ can be computed by

$$\boldsymbol{y}_k = \sum_{n=0}^{N-1} \boldsymbol{x}_n e^{-in\omega_k} \quad \text{for } k = 0, 1, \ldots, N-1, \tag{2.50}$$

wherein $\omega_k = \frac{2\pi k}{N}$ (cf. Oppenheim and Schafer 1999, p. 626). The inverse discrete Fourier transform (IDFT) is denoted by

$$\boldsymbol{y} = \mathcal{F}^{-1}\{\boldsymbol{x}\} \tag{2.51}$$

The elements of vector $\boldsymbol{x}$ can be reconstructed by

$$\boldsymbol{x}_n = \frac{1}{N} \sum_{k=0}^{N-1} \boldsymbol{y}_k e^{in\omega_k} \quad \text{for } n = 0, 1, \ldots, N-1 \tag{2.52}$$

(cf. Oppenheim and Schafer 1999, p. 626). We can compute the DFT and IDFT of vectors very efficiently by FFT algorithms. The complexity of the computation is proportional to $\mathcal{O}(N \ln N)$ (cf. Oppenheim and Schafer 1999, p. 706). If $N = 2^k$, whereas k is a positive integer value, the complexity of the computation is proportional to $\mathcal{O}(N \log_2 N)$ (cf. Oppenheim and Schafer 1999, p. 710). If desired, we can compute the DFT or IDFT in-place (cf. Oppenheim and Schafer 1999, p. 716). It should be mentioned that FFT algorithms can be derived for any basis, though the basis is typically the value 2.

Filter Equation

The basic equation for filtering with an ARMA filter is obtained by replacing the random variables in (2.36) by their realizations (cf. Oppenheim and Schafer 1999, p. 37).

$$y_n = \sum_{k=0}^{P} b_k x_{n-k} + \sum_{j=1}^{Q} a_j y_{n-j}. \tag{2.53}$$

Herein, y_n is the filtered time series and can be understood as the filter output, while x_n is the filter input. The corresponding process is described by equation (2.36), which is repeated here for convenience.

$$\mathcal{Y}_n = \sum_{k=0}^{P} b_k \mathcal{X}_{n-k} + \sum_{j=1}^{Q} a_j \mathcal{Y}_{n-j}. \tag{2.54}$$

If $Q = 0$, then equation (2.53) describes moving-average (MA) filters and equation (2.54) describes AR processes, respectively. If $P = 0$, equation (2.53) describes AR filters and equation (2.54) describes MA processes, respectively. MA and AR filters can be regarded as special cases of ARMA filters. Therefore, we do not need to discuss them separately. For each filter equation there exists a corresponding process equation. Typically, we will use (2.53), when we address the application of the filter while we use (2.54) to describe the stochastic properties of the filter.

Impulse Response

The impulse response h_n of an ARMA filter is a time series which describes certain properties of the filter. It is obtained by filtering an unit impulse d_n which is defined by

$$d_n = \begin{cases} 1, & \text{if } n = 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.55}$$

(cf. Oppenheim and Schafer 1999, p. 12). After filtering this time series by equation (2.53), we get the impulse response h_n of a filter (cf. Oppenheim and Schafer 1999, p. 24).

$$h_n = \sum_{k=0}^{P} b_k d_{n-k} + \sum_{j=1}^{Q} a_j h_{n-j} \tag{2.56}$$

The filters we consider are causal. That is, for calculating the filtered value y_n in (2.53), we do not use values y_n or x_n of the future. We use only values x_k and y_k with $k \leq n$. As a consequence of this, the impulse response h_n is one-sided (cf. Oppenheim and Schafer 1999, p. 251). That means, $h_n = 0$ for $n < 0$.

The time series y_n and x_n are not only connected by equation (2.53), but also by the impulse response.

$$y_n = \sum_{k=-\infty}^{\infty} h_k x_{n-k} = h_n * x_n \tag{2.57}$$

Both the impulse response and the filter equation can be use to determine y_n. The impulse response is therefore an equivalent description to (2.53) of the filter.

Note, that for MA filters $Q = 0$. Therefore, the impulse response of MA filters is given by

$$h_n = \sum_{k=0}^{P} b_k d_{n-k}. \tag{2.58}$$

Because only for $k = n$ the impulse d_{n-k} is non-zero, we find $h_n = b_k$ for MA filters. It is an interesting fact that the impulse response of an MA filter equals its filter coefficients. This also implies that the impulse response of MA filters is finite in contrast to the impulse response of ARMA filters, which is infinite. In order to highlight this difference between MA filters and ARMA filters we write

$$y_n = \sum_{k=0}^{P} h_k x_{n-k} = h_n * x_n \tag{2.59}$$

for the filter equation of MA filters. The term *moving-average filter* is motivated by the fact that each value y_n of the filtered time series is a weighted average of the values $x_n, \ldots, x_{n-P}$. As this average moves along the time series x_n as the index n changes, the corresponding filter is called a moving-average filter. This principle is shown in figure 2.1.

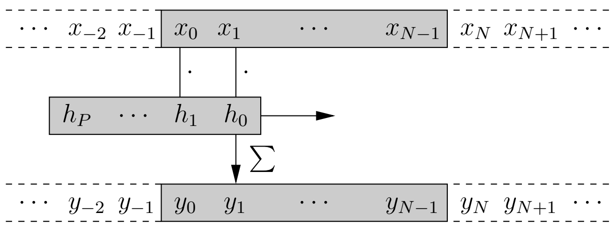

$\cdots$ x_{-2} x_{-1} x_0 x_1 $\cdots$ x_{N-1} x_N x_{N+1} $\cdots$
h_P $\cdots$ h_1 h_0
$\sum$
$\cdots$ y_{-2} y_{-1} y_0 y_1 $\cdots$ y_{N-1} y_N y_{N+1} $\cdots$

Figure 2.1: Each value y_n of the filtered time series is a weighted average of the values $x_n, \ldots, x_{n-P}$. As this average moves along the time series x_n as the index n changes, the corresponding filter is called a moving-average filter. Note, that in practice only values x_n, $n = 0, \ldots, N-1$ are known and the values y_n, $n = 0, \ldots, N-1$ are computed.

The fact that for MA filters the filter coefficients coincide with the impulse response, indicates that ARMA filters can be converted into MA filters by means of the impulse response. However, the order P of the resulting MA filter is then in principle infinite. We will investigate such conversions in more detail later in this section.

z-Transform

The z-transform of a time series is defined as

$$X(z) = \mathcal{Z}\{x_n\} = \sum_{n=-\infty}^{\infty} x_n z^{-n}, \tag{2.60}$$

wherein z is a complex variable (cf. Oppenheim and Schafer 1999, p. 179). It is the discrete counterpart of the Laplace transform. Furthermore, if we substitute $z = e^{i\omega}$, equation (2.60) becomes the Fourier transform in (2.46). The multiplication with z^{-1} can be regarded as time shift, because

$$x_n z^{-k} = x_{n-k} \tag{2.61}$$

holds for integer values of k (cf. Hamilton 1994, p. 26). Therefore, z^{-1} is often called backward shift operator (cf. Box and Jenkins 1970, p. 8), lag operator (cf. Hamilton 1994, p. 26) or delay operator. In order to illustrate the values of z, we plot them in the so-called z-plane. Along the abscissa of the z-plane we plot the real part $\Re(z)$ of the complex variable z and along the ordinate of the z-plane we plot the imaginary part $\Im(z)$ (cf. Oppenheim and Schafer 1999, p. 181).

Transfer function

The transfer function of an ARMA filter is the z-transform of the filter equation, which is given by

$$H(z) = \frac{\sum_{k=0}^{P} b_k z^{-k}}{1 - \sum_{j=1}^{Q} a_j z^{-j}} \tag{2.62}$$

(cf. Oppenheim and Schafer 1999, p. 250). This function is denoted by $H(z)$, because it is closely connected to the impulse response (2.57) through

$$H(z) = \sum_{k=-\infty}^{\infty} h_k z^{-k}. \tag{2.63}$$

It relates the input and output of a filter in terms of the z-transform (cf. Oppenheim and Schafer 1999, p. 245).

$$Y(z) = H(z)X(z) \tag{2.64}$$

Herein, $X(z)$ and $Y(z)$ are the z-transforms of x_n and y_n. By evaluating (2.64) for z on the unit circle in the z-plane, i.e. $z = e^{i\omega}$, we find

$$\mathcal{F}\{y_n\} = H(e^{i\omega})\mathcal{F}\{x_n\}, \tag{2.65}$$

wherein $\mathcal{F}\{x_n\}$ and $\mathcal{F}\{y_n\}$ are the Fourier transforms of x_n and y_n according to (2.46) and ω is the normalized angular frequency (cf. Oppenheim and Schafer 1999, p. 246). Therefore,

$$H(e^{i\omega}) = \frac{\sum_{k=0}^{P} b_k e^{-i\omega}}{1 - \sum_{j=1}^{Q} a_j e^{-i\omega}} \tag{2.66}$$

describes the spectral properties of the filter, which are essential for its decorrelation capability.

Phase Response and Amplitude Response

The Fourier transform of a time series x_n can be expressed in polar form by phase and amplitude (cf. Oppenheim and Schafer 1999, pp. 51, 52).

$$\mathcal{F}\{x_n\} = |\mathcal{F}\{x_n\}| e^{\sphericalangle(\mathcal{F}\{x_n\})} \tag{2.67}$$

The phase is defined as

$$\Psi_x(\omega) = \sphericalangle(\mathcal{F}\{x_n\}) = \arctan \frac{\Im(\mathcal{F}\{x_n\})}{\Re(\mathcal{F}\{x_n\})} \tag{2.68}$$

while the amplitude defined by

$$A_x(\omega) = |\mathcal{F}\{x_n\}|. \tag{2.69}$$

The phase and amplitude of x_n and y_n are connected by the phase response

$$\Psi(\omega) = \arctan \frac{\Im(H(e^{i\omega}))}{\Re(H(e^{i\omega}))}, \tag{2.70}$$

and the amplitude response

$$A(\omega) = |H(e^{i\omega})| \tag{2.71}$$

of the filter. Because

$$\begin{aligned} |\mathcal{F}\{y_n\}| e^{\sphericalangle(\mathcal{F}\{y_n\})} &= |H(e^{i\omega})| e^{\sphericalangle(H(e^{i\omega}))} |\mathcal{F}\{x_n\}| e^{\sphericalangle(\mathcal{F}\{x_n\})} \\ &= |H(e^{i\omega})| |\mathcal{F}\{x_n\}| e^{\sphericalangle(H(e^{i\omega})) + \sphericalangle(\mathcal{F}\{x_n\})} \end{aligned} \tag{2.72}$$

holds, we find

$$A_y(\omega) = |\mathcal{F}\{y_n\}| = |H(e^{i\omega})| |\mathcal{F}\{x_n\}| = A(\omega) A_x(\omega) \tag{2.73}$$

for the amplitude and

$$\Psi_y(\omega) = \sphericalangle(\mathcal{F}\{y_n\}) = \sphericalangle(H(e^{i\omega})) + \sphericalangle(\mathcal{F}\{x_n\}) = \Psi(\omega) + \Psi_x(\omega) \tag{2.74}$$

for the phase of x_n and y_n (cf. Oppenheim and Schafer 1999, p. 246). When designing a filter, there are typically requirements for both phase response and amplitude response. Which one is more important for the design of decorrelating filters, will be clarified in the following.

Power Spectral Density and Autocovariance Function

The power spectral density and the autocovariance function both describe the same properties of a filter or a time series. For the latter, we have to switch from the realization x_n of a time series to its corresponding random variable $\mathcal{X}_n$. The reason for this is that the power spectral density and the autocovariance function are only defined for random variables. Note, that we only consider stationary time series and processes.

The expectation of a time series $\mathcal{X}_n$ is given by

$$\mu_{\mathcal{X}} = E\{\mathcal{X}_n\} \tag{2.75}$$

(cf. Schlittgen and Streitberg 2001, p. 95). Since we require $\mathcal{X}_n$ to be stationary, $\mu_{\mathcal{X}}$ is independent of the index n. The autocovariance function of a real-valued stationary time series $\mathcal{X}_n$ is defined by

$$\gamma_{\mathcal{X}\mathcal{X},k} = E\{(\mathcal{X}_n - \mu_{\mathcal{X}})(\mathcal{X}_{n+k} - \mu_{\mathcal{X}})\}. \tag{2.76}$$

There exist two different definitions of the term autocorrelation function. The first definition

$$\rho_{\mathcal{X}\mathcal{X},k} = \frac{\gamma_{\mathcal{X}\mathcal{X},k}}{\gamma_{\mathcal{X}\mathcal{X},0}} \tag{2.77}$$

corresponds to a normalized version of the autocovariance function. The other definition of the autocorrelation function of a stationary time series $\mathcal{X}_n$ is

$$\phi_{\mathcal{X}\mathcal{X},k} = E\{\mathcal{X}_n \mathcal{X}_{n+k}\} \tag{2.78}$$

(cf. Schlittgen and Streitberg 2001, p. 97). Throughout this thesis, we will use the term autocorrelation function only according to the definition in (2.78). Note, that the autocovariance function is centralized while the autocorrelation function is not. The autocovariance and autocorrelation function are connected by

$$\phi_{\mathcal{X}\mathcal{X},k} = \gamma_{\mathcal{X}\mathcal{X},k} + \mu_{\mathcal{X}}^2. \tag{2.79}$$

Note, that due to the stationarity of $\mathcal{X}_n$ the autocorrelation function ϕ_k and the autocovariance function γ_k are independent of the index n and symmetric around $k = 0$, i.e. $\phi_{\mathcal{X}\mathcal{X},k} = \phi_{\mathcal{X}\mathcal{X},-k}$ and $\gamma_{\mathcal{X}\mathcal{X},k} = \gamma_{\mathcal{X}\mathcal{X},-k}$ (cf. Hamilton 1994, p. 46).

The power spectral density $\Phi_{\mathcal{X}\mathcal{X}}(\omega)$ is the discrete Fourier transform of the autocorrelation function (cf. Oppenheim and Schafer 1999, p. 74).

$$\Phi_{\mathcal{X}\mathcal{X}}(\omega) = \mathcal{F}\{\phi_{\mathcal{X}\mathcal{X},k}\} \tag{2.80}$$

After defining $E\{\mathcal{X}_n^2\} = \phi_{\mathcal{X}\mathcal{X},0}$ to be the power contained in the time series $\mathcal{X}_n$, the power spectral density shows how much power of $\mathcal{X}_n$ belongs to certain parts of frequencies. Therefore, $\frac{1}{2\pi}\Phi_{\mathcal{X}\mathcal{X}}(\omega)d\omega$ is the average contribution of the frequency components of $\mathcal{X}_n$ between the frequencies ω and $\omega + d\omega$ to the total power $E\{\mathcal{X}_n^2\} = \phi_{\mathcal{X}\mathcal{X},0}$ of $\mathcal{X}_n$. Following this interpretation, we find for the total power $E\{\mathcal{X}_n^2\}$ of the time series $\mathcal{X}_n$

$$E\{\mathcal{X}_n^2\} = \phi_{\mathcal{X}\mathcal{X},0} = \frac{1}{2\pi}\int_{-\pi}^{\pi} \Phi_{\mathcal{X}\mathcal{X}}(\omega)d\omega \tag{2.81}$$

(cf. Oppenheim and Schafer 1999, p. 74).

We assume that $\mathcal{Y}_n$ follows the Gaussian distribution $\mathcal{Y}_n \sim N(0,1)$. Then, it holds that $\mu_{\mathcal{X}} = 0$ for processes according to (2.54). In this case $\phi_{\mathcal{X}\mathcal{X},k}$ and $\gamma_{\mathcal{X}\mathcal{X},k}$ are identical.

$$\phi_{\mathcal{X}\mathcal{X},k} = \gamma_{\mathcal{X}\mathcal{X},k}, \quad \text{if } \mu_{\mathcal{X}} = 0 \tag{2.82}$$

Thus, the power spectral density $\Phi_{\mathcal{X}\mathcal{X}}(\omega)$ indicates how the variance $\sigma_{\mathcal{X}}^2 = \gamma_{\mathcal{X}\mathcal{X},0}$ is spread over the frequencies from $-\pi$ to π. Let us consider a time series $\mathcal{W}_n$, having a zero mean and being uncorrelated. For such a time series we find

$$\phi_{\mathcal{W}\mathcal{W},k} = \begin{cases} \sigma_{\mathcal{W}}^2, & \text{if } k = 0 \\ 0, & \text{otherwise} \end{cases}. \tag{2.83}$$

Because $\phi_{\mathcal{W}\mathcal{W},k}$ equals a scaled impulse, its Fourier transform is a constant value for all frequencies.

$$\Phi_{\mathcal{W}\mathcal{W}}(\omega) = \mathcal{F}\{\phi_{\mathcal{W}\mathcal{W},k}\} = \sigma_{\mathcal{W}}^2 \tag{2.84}$$

In physics, one basic property of light is its frequency, perceived by humans (and other living creatures) as the color of the light. If all frequencies contribute the same power, we perceive white light. In analogy to that, the uncorrelated and stationary time series $\mathcal{W}_n$ is referred to as white noise (cf. Hamilton 1994, pp. 47, 48). All time series with a non-constant power spectral density are referred to as colored noise, also in analogy to the colors of light in physics. Other examples of colored noise are pink, brown and black noise. Their power spectral densities are defined as follows (cf. CUDDINGTON and YODZIS 1999).

$$\Phi_{\mathcal{X}\mathcal{X}}(\omega) \sim \omega^{-k}, \begin{cases} k=1 & \text{pink noise} \\ k=2 & \text{brown noise} \\ k=3 & \text{black noise} \end{cases} \tag{2.85}$$

The autocorrelation function of a filter is defined by

$$c_l = \sum_{k=-\infty}^{\infty} h_k h_{k+l} = h_n * h_{-n}, \tag{2.86}$$

which corresponds to correlation of the impulse response of the filter with itself (cf. OPPENHEIM and SCHAFER 1999, p. 73). It connects the autocorrelation function of the time series $\mathcal{X}_n$ and the filtered time series $\mathcal{Y}_n$ in (2.53) by

$$\phi_{\mathcal{Y}\mathcal{Y},k} = \sum_{l=-\infty}^{\infty} \phi_{\mathcal{X}\mathcal{X},k-l} c_l = \phi_{\mathcal{X}\mathcal{X},l} * c_l \tag{2.87}$$

which is a convolution of the autocovariance function of $\mathcal{X}_n$ with the autocorrelation function of the filter. The power spectral density of a filter is defined by

$$P(\omega) = \mathcal{F}\{c_l\} \tag{2.88}$$

(cf. OPPENHEIM and SCHAFER 1999, p. 74). It can also be computed by

$$P(\omega) = |H(e^{i\omega})|^2 = H^*(e^{i\omega})H(e^{i\omega}), \tag{2.89}$$

wherein $H^*(e^{i\omega})$ is the complex conjugate of $H(e^{i\omega})$. It connects the power spectral densities of $\mathcal{X}_n$ and $\mathcal{Y}_n$ by

$$\Phi_{\mathcal{Y}\mathcal{Y}}(\omega) = P(\omega)\Phi_{\mathcal{X}\mathcal{X}}(\omega). \tag{2.90}$$

Our goal is to design a filter, such that the time series $\mathcal{Y}_n$ corresponds to white noise. Therefore, our goal is to chose $P(\omega)$ such that $\Phi_{\mathcal{Y}\mathcal{Y}}(\omega) = 1$ for all ω. Obviously, we have found the best filter, if

$$P(\omega) = \frac{1}{\Phi_{\mathcal{X}\mathcal{X}}(\omega)} \tag{2.91}$$

holds.

We will now establish the link between the filter and the covariance matrix $\mathbf{\Sigma}$ in (2.8). Because the residual time series $\mathcal{V}_n$ is regarded as a stationary time series, the covariance matrix $\mathbf{\Sigma}_{\boldsymbol{\mathcal{V}}\boldsymbol{\mathcal{V}}}$ of the residual vector

$$\boldsymbol{\mathcal{V}} = \begin{bmatrix} \mathcal{V}_0 & \mathcal{V}_1 & \cdots & \mathcal{V}_{N-1} \end{bmatrix}^\top \tag{2.92}$$

has a Toeplitz structure (cf. section 2.2). Furthermore, the time series $\mathcal{V}_n$ has a zero mean, i.e. $E\{\mathcal{V}_n\} = 0$ (cf. section 2.1). Therefore, the covariances

$$\sigma_{\mathcal{V}_n\mathcal{V}_{n+k}} = E\{(\mathcal{V}_n - E\{\mathcal{V}_n\})(\mathcal{V}_{n+k} - E\{\mathcal{V}_{n+k}\})\} = \gamma_{\mathcal{V}\mathcal{V},k} = E\{\mathcal{V}_n\mathcal{V}_{n+k}\} = \phi_{\mathcal{V}\mathcal{V},k} \tag{2.93}$$

are equal to the autocorrelation function of $\mathcal{V}_n$. Thus, the elements of $\mathbf{\Sigma}_{\boldsymbol{\mathcal{V}}\boldsymbol{\mathcal{V}}}$ are given by

$$\mathbf{\Sigma} = \mathbf{\Sigma}_{\boldsymbol{\mathcal{V}}\boldsymbol{\mathcal{V}}} = \begin{bmatrix} \phi_{\mathcal{V}\mathcal{V},0} & \phi_{\mathcal{V}\mathcal{V},1} & \cdots & \phi_{\mathcal{V}\mathcal{V},N-1} \\ \phi_{\mathcal{V}\mathcal{V},1} & \phi_{\mathcal{V}\mathcal{V},0} & \cdots & \phi_{\mathcal{V}\mathcal{V},N-2} \\ \vdots & \vdots & & \vdots \\ \phi_{\mathcal{V}\mathcal{V},N-1} & \phi_{\mathcal{V}\mathcal{V},N-2} & \cdots & \phi_{\mathcal{V}\mathcal{V},0} \end{bmatrix}. \tag{2.94}$$

Now, the link is given by the design goal for the filter in (2.91). We have $\Phi_{\mathcal{V}\mathcal{V}}(\omega) = P^{-1}(\omega)$ and

$$\sigma_{\mathcal{V}_n \mathcal{V}_{n+k}} = \phi_{\mathcal{V}\mathcal{V},k} = \mathcal{F}^{-1}\{\Phi_{\mathcal{V}\mathcal{V}}(\omega)\} = \mathcal{F}^{-1}\{P^{-1}(\omega)\}. \tag{2.95}$$

This equation is the link between the filter and the covariance matrix $\mathbf{\Sigma}$ in (2.8). Since the power spectral density $P(\omega) = |H(e^{i\omega})|^2$ of the filter depends only on its amplitude response $A(\omega) = |H(e^{i\omega})|$, the phase response $\Phi(\omega)$ of the filter is irrelevant within the design of decorrelating filters.

In practical situations, we need to estimate the autocovariance function and the power spectral density from a given time series

$$x_n = \begin{cases} x_n, & \text{for } n = 0, \ldots, N-1 \\ 0, & \text{otherwise} \end{cases}. \tag{2.96}$$

A commonly used estimator for the autocovariance function $\gamma_{\mathcal{X}\mathcal{X},k}$ is

$$c_{xx,k} = \frac{1}{N} \sum_{n=0}^{N-k-1} (x_n - m_x)(x_{n+k} - m_x) = \frac{1}{N}(x_n - m_x) * (x_{-n} - m_x), \tag{2.97}$$

wherein m_x is the mean of x_n (cf. Box and Jenkins 1970, p. 32). Note, that this estimation is biased towards zero.

$$m_x = \frac{1}{N} \sum_{n=0}^{N-1} x_n \tag{2.98}$$

We can estimate the autocorrelation function $\phi_{\mathcal{X}\mathcal{X},k}$ by

$$f_{xx,k} = \frac{1}{N} \sum_{n=0}^{N-k-1} x_n x_{n+k} = \frac{1}{N} x_n * x_{-n}. \tag{2.99}$$

An efficient computation of $f_{xx,k}$ is given via the FFT and IFFT by

$$f_{xx,k} = \frac{1}{N} \mathcal{F}^{-1}\{\mathcal{F}\{x_n\}\mathcal{F}^*\{x_n\}\} \tag{2.100}$$

(cf. Oppenheim and Schafer 1999, p. 65). The power spectral density $\Phi_{\mathcal{X}\mathcal{X}}(\omega)$ can be estimated by

$$P_{xx}(\omega) = \mathcal{F}\{f_{xx,k}\} = \frac{1}{N}\mathcal{F}\{x_n\}\mathcal{F}^*\{x_n\} = \frac{1}{N}|\mathcal{F}\{x_n\}|^2 \tag{2.101}$$

(cf. Oppenheim and Schafer 1999, p. 888). An estimation of the power spectral density is referred to as a periodogram. Note, that $NP_{xx}(\omega)$ is equal to the squared amplitude response $A_x^2(\omega)$. Of course, there are many other estimation methods for the autocovariance function and the power spectral density. For example we could use a window function to reduce the spectral leakage (cf. Brigham 1988, pp. 178–188) combined with Welch's method (cf. Welch 1967) in order to obtain a smoother estimate of $\Phi_{\mathcal{X}\mathcal{X}}(\omega)$.

Zeros and Poles

The transfer function $H(z)$ in (2.62) can also be expressed by

$$H(z) = \frac{\sum_{k=0}^{P} b_k z^{-k}}{1 - \sum_{j=1}^{Q} a_j z^{-j}} = b_0 \frac{\prod_{k=1}^{P}(1 - \beta_k z^{-1})}{\prod_{j=1}^{Q}(1 - \alpha_j z^{-1})}, \tag{2.102}$$

as any polynomial $p(z) = a_0 + a_1 z + a_2 z^2 + \ldots$ can be factorized into factors $p(z) = (r_1 - z)(r_2 - z)\ldots$, whereas the r_k are the roots of the polynomial (cf. Oppenheim and Schafer 1999, p. 250). Herein β_k are the zeros and α_j are the poles off $H(z)$. The position of zeros and poles play an important role for the design of highpass, lowpass and notch filters. In section 3 it is shown that even the realization of an ARMA filter is affected by the position of zeros and poles. For now, we will use the position of the zeros and poles only as a tool to determine, whether a filter is stable or not.

Stability and Stationarity

A filter is called stable, if the poles α_j of the transfer function $H(z)$ lie inside the unit circle of the z-plane, i.e. $|\alpha_j| < 1$. A stable filter has the property, that for a bounded time series x_n, the filtered time series y_n is bounded, too. Furthermore, stable filters have a minimum-phase transfer function (cf. Oppenheim and Schafer 1999, p. 255). This means, if $h_n^{(\min)}$ is the impulse response of the minimum-phase transfer function and h_n is the impulse response of any other non-minimum-phase transfer function which has the same power spectral density, then

$$\sum_{m=0}^{n} |h_m|^2 \leq \sum_{m=0}^{n} |h_m^{(\min)}|^2 \tag{2.103}$$

holds (cf. Oppenheim and Schafer 1999, p. 298). Hence, the energy of the impulse response of a minimum-phase transfer function has the highest concentration near to $n = 0$ in comparison to any other transfer function. Therefore, stable filters automatically have the best possible localization property in the time domain. Another property of a stable filter is, that its impulse response h_n decays to zero, as n approaches infinity.

$$\lim_{n\to\infty} h_n = 0 \quad \text{for stable filters} \tag{2.104}$$

This follows from the fact, that

$$\sum_{n=-\infty}^{\infty} |h_n| < \infty \tag{2.105}$$

holds for stable filters (cf. Oppenheim and Schafer 1999, p. 251 and Forster 1983, p. 38, theorem 2 and 3). We will make use of this property in many of the algorithms within this thesis.

A process is stable, if the zeros β_k of the transfer function lie inside the unit circle of the z-plane, i.e. $|\beta_k| < 1$. Stable processes are also stationary. After defining the inverse transfer function

$$G(z) = \frac{1}{H(z)} \tag{2.106}$$

and the process corresponding to the transfer function $H(z)$ is stable, then the impulse response g_n of $G(z)$ decays to zero, as n approaches infinity.

$$\lim_{n\to\infty} g_n = 0 \quad \text{for stable processes} \tag{2.107}$$

Analog to stable filters, this follows from the fact, that

$$\sum_{n=-\infty}^{\infty} |g_n| < \infty \tag{2.108}$$

holds for stable processes (cf. Oppenheim and Schafer 1999, p. 251 and Forster 1983, p. 38, theorem 2 and 3).

If a zero β_k or a pole α_j of a filter lies outside the unit circle, we may stabilize the filter by changing the transfer function $H(z)$ (cf. Oppenheim and Schafer 1999, p. 288). We start by expressing the pole or zero in polar form.

$$\alpha_j = r_{\alpha_j} e^{\phi_{\alpha_j}} \quad \text{or} \quad \beta_k = r_{\beta_k} e^{\phi_{\beta_k}} \tag{2.109}$$

Then, we substitute the radius of the pole or the zero for its reciprocal value.

$$\alpha_j = \frac{1}{r_{\alpha_j}} e^{\phi_{\alpha_j}} \quad \text{or} \quad \beta_k = \frac{1}{r_{\beta_k}} e^{\phi_{\beta_k}} \tag{2.110}$$

The power spectral density of the stabelized filter will remain the same as the power spectral density of the original filter. Thus, the decorrelation capability of the filter remains unaffected, too. Only if poles or zeros lie on the unit circle, we cannot stabilize the transfer function without altering the corresponding filter's decorrelation capabilities. However, for some types of highpass filters, such as Butterworth filters, the zeros of the filter lie on the unit circle. Though the filter is still stable, the corresponding process is not stationary.

Localization Features of Filters

In many applications it is desirable for a filter to have good localization features in the time domain. In our case, this means, that local phenomena such as outliers remain local after filtering. For a given filter characteristic, a stable filter provides the best possible localization features. Furthermore, it is desirable to keep the impulse response h_n of the stable filter as short as possible in order to obtain even better localization features. However, Fourier's uncertainty relation states, that the product of the resolution in the time domain and the resolution in the frequency domain must exceed a certain constant. From this it follows, that increasing the localization features of the filter results in decreasing its decorrelating capabilities. Here, we have two contradicting requirements. Because the decorrelating capabilities of the filter affect the quality of the solution (2.8) of the least squares problem, they are by far more important than the localization features of the filter.

From this discussion, we can conclude that the localization features of the filter in the time domain mainly depend on the noise characteristics of the residuals in the Gauss-Markov model in (2.1). If the power spectral density of the residuals contains local phenomena with respect to frequency, then the impulse response of the filter will be long. For example, such local phenomena can be peaks in the power spectral density.

Conversions between AR, MA and ARMA Processes

Every filter is uniquely characterized by the impulse response of its transfer function. As the impulse response of AR filters equals the filter coefficients, it is possible to convert AR and ARMA filters into MA filters only by computing their impulse response h_n and then using these values as filter coefficients. We can obtain the impulse response by

$$h_n = \sum_{k=0}^{P} b_k d_{n-k} + \sum_{j=1}^{Q} a_j h_{n-j}, \tag{2.111}$$

wherein d_n is the unit impulse as defined in (2.55). The drawback is, that the impulse response of AR and ARMA filters is in principle infinitely long. However, the impulse response h_n of stable filters will decay to zero as n approaches infinity as stated in equation (2.104). Thus, we can approximate the AR and ARMA filters by MA filters with arbitrary precision by choosing the order R of the MA filter, whereas R equals the index n at which we truncate the impulse response h_n.

In the same way we can convert MA and ARMA filters into AR filters. The only difference is, that we use the inverse transfer function $G(z)$ defined in (2.106). With $G(z)$ we can compute the impulse response g_n by

$$g_n = \frac{1}{b_0}(d_n - \sum_{j=1}^{Q} a_j d_{n-j} - \sum_{k=1}^{P} b_k g_{n-k}), \tag{2.112}$$

wherein d_n is the unit impulse as defined in (2.55). If the process is stable, then the impulse response g_n will decay to zero as n approaches infinity as stated in (2.107). Thus, we can approximate the MA and ARMA filters by AR filters with arbitrary precision by choosing the order R of the AR filter, whereas R equals the index n at which we truncate the impulse response g_n.

Practical Use

We will now demonstrate, how these equations can be evaluated in practice. The question we want to answer in this subsection is: How are x_0, x_1, ..., x_{N-1} and y_0, y_1, ..., y_{N-1} related to each other, if x_n is a time series and y_n is the corresponding filtered time series? This question arises from the fact, that the observations l_n are given for $n = 0, 1, \ldots, N-1$ and we minimize the sum of squares of the filtered residuals also for $n = 0, 1, \ldots, N-1$ and not for $-\infty < n < \infty$. In order to keep the example simple, we will consider MA filters for the filtering of x_n. The mathematical background of the following is very well depicted in FARHANG-BOROUJENY (1998), pp. 251–255.

MA filters are defined by equation (2.59). We can compute the time series y_n by a convolution, as equation (2.59) indicates. But a convolution evaluated in the time domain is usually a computationally intensive task, if the filter order P is not a small value. Because x_n is in practice only given for $n = 0, \dots, N-1$ (cf. section 2.3), we set x_n to zero for all other n, i.e.

$$x_n = \begin{cases} x_n, & \text{for } n = 0, 1, \dots, N-1 \\ 0, & \text{for all other } n \end{cases} \tag{2.113}$$

In this case y_n is non-zero for $n = 0, \dots, N-1+P$, which follows from equation (2.59).

$$y_n = \begin{cases} y_n, & \text{for } n = 0, 1, \dots, N-1+P \\ 0, & \text{for all other } n \end{cases} \tag{2.114}$$

With the goal to reduce the computational effort of the convolution in (2.59), we express it as a circulant convolution. This is possible due to (2.113) and (2.114). The circulant convolution is defined as follows. If

$$\boldsymbol{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{N-1} \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{N-1} \end{bmatrix}, \quad \boldsymbol{c} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{N-1} \end{bmatrix} \quad \text{and} \quad \boldsymbol{C} = \begin{bmatrix} c_0 & c_{N-1} & \cdots & c_2 \\ c_1 & c_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & c_{N-1} \\ c_{N-1} & \cdots & c_1 & c_0 \end{bmatrix}, \tag{2.115}$$

then

$$\boldsymbol{b} = \boldsymbol{C}\boldsymbol{a} = \boldsymbol{c} \circledast \boldsymbol{a} = \mathcal{F}^{-1}\{\mathcal{F}\{\boldsymbol{c}\} \circ \mathcal{F}\{\boldsymbol{a}\}\} \tag{2.116}$$

is the circulant convolution of $\boldsymbol{c}$ and $\boldsymbol{a}$. Herein, $\circ$ denotes an element-wise multiplication. In order to make use of this, we define the following vectors of length $N + P$.

$$\boldsymbol{x} = \begin{bmatrix} x_0 & \cdots & x_{N-1} & 0 & \cdots & 0 \end{bmatrix}^\top \tag{2.117}$$

$$\boldsymbol{h} = \begin{bmatrix} h_0 & \cdots & h_P & 0 & \cdots & 0 \end{bmatrix}^\top \tag{2.118}$$

$$\boldsymbol{y} = \begin{bmatrix} y_0 & \cdots & y_{N-1+P} \end{bmatrix}^\top \tag{2.119}$$

In vector $\boldsymbol{x}$, the time series x_n is padded by P zeros, while in vector $\boldsymbol{h}$, the filter coefficients h_k are padded by $N-1$ zeros. This technique is called zero-padding. Any convolution of finite length can by computed by a circulant convolution, if zero-padding is applied. With these zero-padded vectors we find

$$\boldsymbol{y} = \boldsymbol{h} \circledast \boldsymbol{x} = \mathcal{F}^{-1}\{\mathcal{F}\{\boldsymbol{h}\} \circ \mathcal{F}\{\boldsymbol{x}\}\}. \tag{2.120}$$

In our application, we are only interested in the values $y_0, \dots, y_{N-1}$. For this reason we need to set the values $y_N, \dots, y_{N-1+P}$ to zero. This corresponds to an element-wise multiplication with the rectangular window w_n, which is defined by

$$w_n = \begin{cases} 1, & \text{if } 0 \le n \le N-1 \\ 0, & \text{otherwise} \end{cases}. \tag{2.121}$$

In order to integrate this window in equation (2.120), we define

$$\boldsymbol{w} = \begin{bmatrix} w_0 & \cdots & w_{N-1+P} \end{bmatrix}^\top. \tag{2.122}$$

Multiplying equation (2.120) by the window vector $\boldsymbol{w}$ yields

$$\boldsymbol{w} \circ \boldsymbol{y} = \boldsymbol{w} \circ (\boldsymbol{h} \circledast \boldsymbol{x}) = \boldsymbol{w} \circ \mathcal{F}^{-1}\{\mathcal{F}\{\boldsymbol{h}\} \circ \mathcal{F}\{\boldsymbol{x}\}\} = \frac{1}{N+P}\mathcal{F}^{-1}\{\mathcal{F}\{\boldsymbol{w}\} \circledast (\mathcal{F}\{\boldsymbol{h}\} \circ \mathcal{F}\{\boldsymbol{x}\})\}. \tag{2.123}$$

If we want to relate the spectral properties of $y_0, \ldots, y_{N-1}$ to the spectral properties of $x_0, \ldots, x_{N-1}$ by means of the filter coefficients $h_0, \ldots, h_P$, we must use equation (2.123) instead of (2.120). This demonstrates that windowing effect have to be taken into account in practice.

This example shows also that the filter equation (2.59) for MA filter can be implemented very fast by means of FFT techniques. Which other techniques lead to fast filter implementations, is investigated in the next chapter.

3. Implementation of Filters

In this chapter different methods for filtering with MA and ARMA filters are described. For the sake of generality the methods are formulated for the application to a time series which is stored in a vector. The methods range from straightforward implementation of the filter equation (2.53) to an implementation which performs the filtering by a multiplication in the frequency domain. Special emphasis is put on the block methods which perform better concerning the calculation time. We will discuss the following four implementations.

- time domain straightforward implementation (filter a single element at a time)
- time domain block implementation (filter a block of elements at a time)
- frequency domain block implementation (filter a block of elements at a time)
- frequency domain implementation (filter all elements at once)

After that we discuss how to integrate these methods into least-squares adjustment procedures. It will be shown that there are many possibilities of integrating the filter methods into different adjustment procedures. Because each of the methods has its pros and cons, we will highlight under which circumstance which methods should be preferred. Within this context we also describe the integration of the Toeplitz approach.

3.1 MA Filters

The impulse response h_n of MA filters is finite while the impulse response of ARMA filters is infinite. For this reason, we can describe the following algorithms easier for the MA filters. As the results can be partially transferred to the case of ARMA filters, we can regard this section also as a preparation for the section on the implementation of ARMA filters.

Block Method (Time Domain Block Implementation)

We can implement MA filters straightforward by equation (2.59) in the time domain. Within this straightforward implementation only vector-vector products occur and any one value y_n of the filter output is calculated at a time. In order to increase the calculation speed, we can compute not any single value y_n, but a whole block

$$\boldsymbol{y}^{(k)} = \begin{bmatrix} y_{kL} & y_{kL+1} & \cdots & y_{(k+1)L-1} \end{bmatrix}^\top \tag{3.1}$$

of the filter output at a time. Herein, we call L the block size of the filter output or simply block size while k is the index of the block. The increase of calculation speed is achieved by switching from vector-vector products to matrix-vector products. In numerical libraries like BLAS/ATLAS matrix-vector products are often more effective compared to vector-vector products (cf. ANDERSON et al. 1999. pp. 56, 57). Therefore, even if the number of arithmetic operations increases, applying the filter block-wise will be faster.

We define the filter matrix $\boldsymbol{F}$ as a $L \times L + P$ matrix by

$$\boldsymbol{F} = \begin{bmatrix} h_P & h_{P-1} & \cdots & h_0 & 0 & \cdots & 0 \\ 0 & h_P & \ddots & & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & h_{P-1} & \cdots & h_0 & 0 \\ 0 & \cdots & 0 & h_P & \cdots & h_1 & h_0 \end{bmatrix}, \tag{3.2}$$

whereas P is the order of the MA filter. The time series x_n is divided into blocks $\boldsymbol{x}^{(k)}$ of size $L + P$, which overlap each other by P elements.

$$\boldsymbol{x}^{(k)} = \begin{bmatrix} x_{kL-P} & x_{kL-P+1} & \cdots & x_{(k+1)L-1} \end{bmatrix}^\top \quad \text{for } k = 0, 1, \ldots, K-1 \tag{3.3}$$

Figure 3.1 illustrates the partitioning of the time series x_n and y_n.

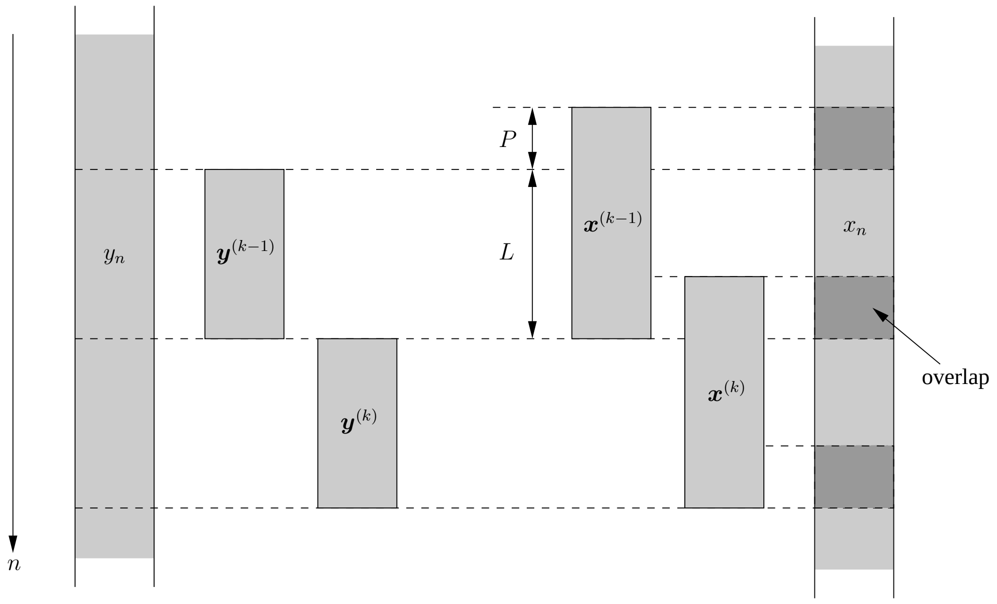


Figure 3.1: Partitioning of the time series x_n and y_n for the MA block filter method. The block size is equal to L. The blocks $\boldsymbol{x}^{(k)}$ overlap by P values while the blocks $\boldsymbol{y}^{(k)}$ do not overlap. The overlapping values of x_n are marked by darker gray.

We compute the filtered blocks $\boldsymbol{y}^{(k)}$ by

$$\boldsymbol{y}^{(k)} = \boldsymbol{F}\boldsymbol{x}^{(k)} \quad \text{for } k = 0, 1, \ldots, K-1. \tag{3.4}$$

Herein, the blocks $\boldsymbol{y}^{(k)}$ are defined by (3.1) and do not overlap each other.

As the time series x_n is only known for $n = 0, 1, \ldots, N-1$, the number of blocks is $K = \operatorname{ceil}\left(\frac{N}{L}\right)$. At least within the first block $\boldsymbol{x}^{(0)}$ there occur values x_n with $n < 0$. These values are simply set to zero, which is commonly performed in practice (cf. Schuh 1996, pp. 36,37). Within the last block $\boldsymbol{x}^{(K-1)}$ there may occur values x_n with $n > N-1$. We can leave these values undefined, as they only affect output values y_n with $n < N-1$ which are not used for further processing.

The number of multiplications for filtering N values of the time series x_n is $N(L+P)$. Thus, we obtain the minimum number of arithmetic operations for $L = 1$ which is the smallest possible block size and corresponds to the straightforward implementation of MA filters mentioned at the beginning of this subsection. However, the optimum block size is typically $L > 1$ and we can determine it best by measuring the calculation time of test computations for different block sizes L.

Within adjustment procedures for least squares problems, for example the conjugate gradient algorithm (cf. Schuh 1996), we need to filter the design matrix $\boldsymbol{A}$ at least once, if the normal equation matrix cannot be approximated by means of analytical formulas. In this case, we replace the input blocks $\boldsymbol{x}^{(k)}$ by blocks $\boldsymbol{A}^{(k)}$. Then, the matrix-vector products within the filtering procedure become matrix-matrix products. From the point of view of calculation speed, this is a best case scenario as matrix-matrix products are highly optimized operations within numerical libraries like BLAS/ATLAS (cf. Anderson et al. 1999. pp. 56, 57).

Overlap-Add Method (Frequency Domain Block Implementation)

There are two methods which make use of partintioning the input time series x_n in order to decrease the computational costs of filtering in the frequency domain: the overlap-save method and the overlap-add method. Both methods are described in Brigham (1988), chapter 10. As these methods are almost equivalent concerning their computational costs, we consider here only the overlap-add method. Within this method the input time series is divided into blocks similarly to the block implementation of the filtering in the time domain.

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}^{(0)} \\ \boldsymbol{x}^{(1)} \\ \vdots \\ \boldsymbol{x}^{(K-1)} \end{bmatrix} \tag{3.5}$$

Herein, $K = \operatorname{ceil} \frac{N}{L}$ is the number of blocks, L is the block size and

$$\boldsymbol{x}^{(k)} = \begin{bmatrix} x_{kL} & x_{kL+1} & \cdots & x_{(k+1)L-1} \end{bmatrix}^\top \tag{3.6}$$

is one of the blocks. Within the last block $\boldsymbol{x}^{(K-1)}$, there may occur values x_n with $n > N-1$. We can leave these values undefined, as they only affect the output values y_n with $n > N-1$ which are not used within the further processing. This partition of the input time series into blocks corresponds formally to the identity

$$x_n = \sum_{k=0}^{K-1} x^{(k)}_{n-kL}, \quad \text{wherein } x^{(k)}_n = \begin{cases} x_{n+kL}, & \text{for } n = 0, 1, \dots, L-1 \\ 0, & \text{otherwise} \end{cases}. \tag{3.7}$$

The time series x_n, which we assume to be non-zero only for $n = 0, 1, \dots, N-1$, is split into K time series $x^{(k)}_n$ which are in principle defined for $-\infty < n < \infty$. However, for each of these time series $x^{(k)}_n$ only the first L values $x^{(k)}_0, \dots, x^{(k)}_{L-1}$ are non-zero. The filtering by MA filters is according to equation (2.59) equivalent to a convolution in the time domain. Therefore, we find

$$h_n * x_n = \sum_{k=0}^{K-1} h_n * x^{(k)}_{n-kL} = \sum_{k=0}^{K-1} y^{(k)}_{n-kL}, \tag{3.8}$$

wherein $y^{(k)}_n$ is defined analog to $x^{(k)}_n$.

$$y_n = \sum_{k=0}^{K-1} y^{(k)}_{n-kL}, \quad \text{wherein } y^{(k)}_n = \begin{cases} y_{n+kL}, & \text{for } n = 0, 1, \dots, L-1 \\ 0, & \text{otherwise} \end{cases} \tag{3.9}$$

According to equation (3.8), we can perform the convolution of x_n and h_n in small parts as the time series $x^{(k)}_n$ has less non-zero values than the time series x_n. After the convolution, we only have to rearrange and add the resulting time series $y^{(k)}_n$. The only thing we have to be aware of is, that $L + P$ values of $y^{(k)}_{n-kL}$ are non-zero and thus the non-zero elements of $y^{(k)}_{n-kL}$ overlap by P values. This explains the name of this method.

Each of the convolutions in (3.8) can by performed as an element-wise multiplication in the time domain according to (2.116) and (2.120). After defining

$$\boldsymbol{x}^{(k)} = \begin{bmatrix} x^{(k)}_0 & x^{(k)}_1 & \cdots & x^{(k)}_{L-1} & 0 & \cdots & 0 \end{bmatrix}^\top \tag{3.10}$$

and

$$\boldsymbol{h} = \begin{bmatrix} h_0 & h_1 & \cdots & h_P & 0 & \cdots & 0 \end{bmatrix}^\top, \tag{3.11}$$

whereas both vectors are padded with zeros, such that their length is equal to $N_{\text{FFT}} = 2^{\operatorname{ceil}(\log_2(L+P))}$, we obtain

$$\boldsymbol{y}^{(k)} = \mathcal{F}^{-1}\{\mathcal{F}\{\boldsymbol{h}\} \circ \mathcal{F}\{\boldsymbol{x}^{(k)}\}\}, \tag{3.12}$$

wherein

$$\boldsymbol{y}^{(k)} = \begin{bmatrix} y_0^{(k)} & y_1^{(k)} & \cdots & y_{L+P-1}^{(k)} \end{bmatrix}^\top . \tag{3.13}$$

Figure 3.2 illustrates the partitioning of the time series x_n and y_n.

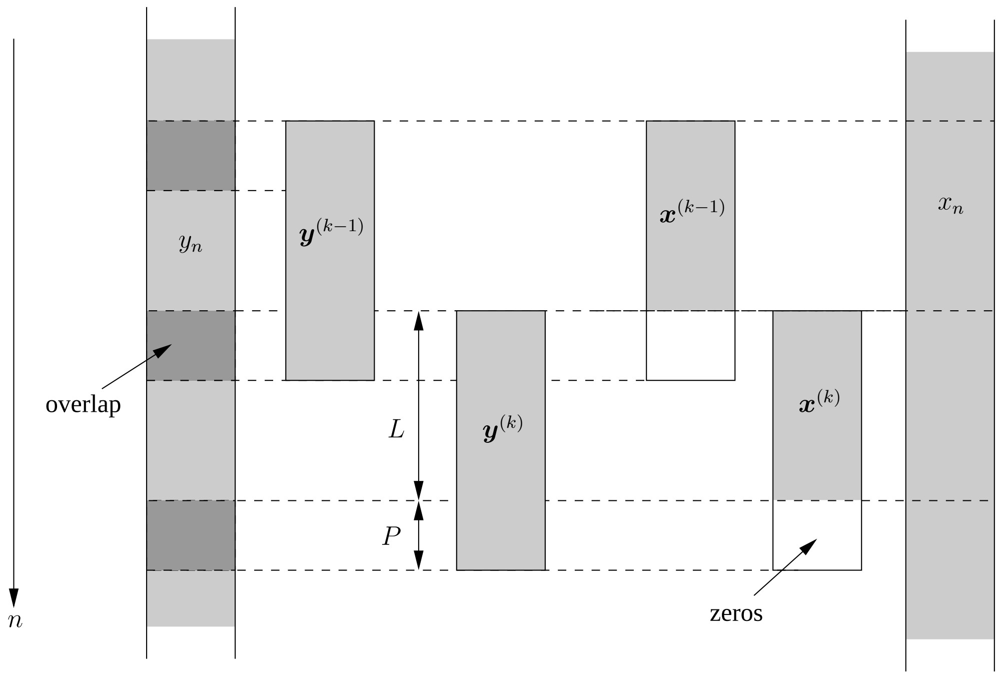


Figure 3.2: Partitioning of the time series x_n and y_n for the overlap add method for MA filters. The block size is equal to L, which should be chosen such that $L + P = 2^k$, whereas k is an integer value. The blocks $\boldsymbol{y}^{(k)}$ overlap by P values while the blocks $\boldsymbol{x}^{(k)}$ do not overlap. The overlapping values of y_n are marked by darker gray. The last P values of $\boldsymbol{x}^{(k)}$ are equal to zero.

The choice of N_{FFT} to be equal to 2^m with m being a positive integer value is due to the fact, that FFT algorithms work fastest for this length. In order to filter a maximum number of elements within each convolution in (3.8), we choose the block size L such that $L + P = N_{\text{FFT}}$.

FFT Method (Frequency Domain Implementation)

As the filtering by MA filters corresponds according to (2.59) to a convolution in the time domain, we can perform this convolution by an element-wise multiplication in the frequency domain according to (2.116) and (2.120). After defining

$$\boldsymbol{x} = \begin{bmatrix} x_0 & x_1 & \cdots & x_{N-1} & 0 & \cdots & 0 \end{bmatrix} \tag{3.14}$$

and

$$\boldsymbol{h} = \begin{bmatrix} h_0 & h_1 & \cdots & h_P & 0 & \cdots & 0 \end{bmatrix}, \tag{3.15}$$

whereas both vectors are padded with zeros, such that they are of length $N_{\text{FFT}} = 2^{\text{ceil}\,(\log_2(N+P))}$, we find

$$\boldsymbol{y} = \mathcal{F}^{-1}\{\mathcal{F}\{\boldsymbol{h}\} \circ \mathcal{F}\{\boldsymbol{x}\}\}. \tag{3.16}$$

The first N elements of this vector contain

$$\boldsymbol{y}_{1:N} = \begin{bmatrix} y_0 & y_1 & \cdots & y_{N-1} \end{bmatrix}^\top \tag{3.17}$$

which is the desired filtered output. As for the overlap add method, the choice of N_{FFT} to be equal to 2^m with m being a positive integer value is due to the fact, that FFT algorithms work the fastest for this length. We can regard the FFT method as a special case of the overlap-add method. This special case is obtained, if the block size L is chosen large enough such that $K = 1$.

3.2 ARMA Filters

In this section we describe the same methods for ARMA filters as in section 3.1 for MA filters. These methods are more complicated to develop and to apply because of two major differences between MA and ARMA filters: The first difference is the recursion for y_n in equation (2.53). While for MA filters the derivation of the block method is straightforward as there is no recursion within the filter equation (2.59), the block method for ARMA filters needs to resolve this recursion at least for the block size. The second difference is, that the impulse response of MA filters is of finite length while the impulse response of ARMA filters is infinitely long. This affects the methods based on FFT techniques, i.e. the overlap-add method and the FFT method. As we design stable filters, the values h_n of the impulse response decay to zero as n increases (cf. section 2.4). The workaround is to truncate the infinite impulse response of ARMA filters such that the truncation error is smaller than roundoff errors and therefore negligible.

Block Method (Time Domain Block Implementation)

The motivation to use a block implementation of ARMA filters is the same as for MA filters: Within the computation the vector-vector products of a straightforward implementation of equation (2.53) are substituted by matrix-vector products of a block implementation of equation (2.53), which can be performed more efficiently by numerical libraries such as the BLAS/ATLAS library. Depending on the filter design, roundoff noise in equation (2.53) can become a serious problem. In order to reduce the roundoff noise to an acceptable level, we subdivide the ARMA filter into sections, whereas each section is again an ARMA filter. These filter sections are then applied in a serial sequence. It should be mentioned that filter sections are also called filter cascades. Within this thesis, the terms section and cascade are equivalent.

Formation of Filter Sections

First of all, we show how an ARMA filter can be subdivided into filter sections. After that, we investigate under which circumstances roundoff noise becomes a problem. Finally, we realize that the sectioning of the filter should not be performed arbitrarily, but depending on rules given by Dehner (2003).

We can subdivide the transfer function $H(z)$ of an ARMA filter into filter sections $H_s(z)$ by

$$H(z) = \prod_{s=0}^{S-1} H_s(z). \tag{3.18}$$

Herein, each filter section

$$H_s(z) = \frac{\sum_{k=0}^{P_s} b_k^{(s)} z^{-k}}{1 - \sum_{j=1}^{Q_s} a_j^{(s)} z^{-j}} \tag{3.19}$$

is again an ARMA filter, whereas its coefficients are denoted by $a_j^{(s)}$ and $b_k^{(s)}$. For this subdivision, the filter sections are connected in series, which also means that they have to be applied serially to the time series. Typically, the subdivision is performed such that the filter sections are of order $P_s = Q_s = 1$ or $P_s = Q_s = 2$. We call such sections first order sections or second order sections, respectively. According to equation (2.102) the transfer function of an ARMA filter can be expressed by its poles and zeros.

$$H(z) = b_0 \frac{\prod_{k=1}^{P_1}(1-\gamma_k z^{-1}) \prod_{k=1}^{P_2}(1-\eta_k z^{-1})(1-\eta_k^* z^{-1})}{\prod_{j=1}^{Q_1}(1-\kappa_j z^{-1}) \prod_{j=1}^{Q_2}(1-\lambda_j z^{-1})(1-\lambda_j^* z^{-1})} \tag{3.20}$$

Herein, γ_k are real-valued zeros, η_k are complex zeros, κ_j are real-valued poles and λ_j are complex poles. Note, that the poles are the roots of the denominator polynomial of the transfer function while the zeros are the roots of the numerator polynomial of the transfer function. First order sections are commonly set up for real poles and zeros while second order sections are used for pairs of complex poles and zeros. For the complex case, those poles are paired, which are complex conjugate to each other and those zeros are paired, which are complex conjugate to each other, respectively. Therefore, we find for first order sections

$$H(z) = b_0 \frac{1-\gamma z^{-1}}{1-\kappa z^{-1}} = \frac{b_0 + b_1 z^{-1}}{1-a_1 z^{-1}} \tag{3.21}$$

and for second order sections

$$H(z) = b_0 \frac{(1-\eta z^{-1})(1-\eta^* z^{-1})}{(1-\lambda z^{-1})(1-\lambda^* z^{-1})} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1-a_1 z^{-1} - a_2 z^{-2}}. \tag{3.22}$$

Reasons for Forming Filter Sections

The main reason for forming filter sections is the quantization noise of the filter coefficients. Quantization noise is the error, which occurs, when number are stored with finite precision, for example double precision. In some cases, the poles and zeros of the transfer function in (2.102) are located very close to the unit circle in the z-plane. In such cases, small changes in the location of a pole or zero can result in large changes of the power spectral density (2.89) and thus may destroy the decorrelation capability of a filter. Therefore, we will now investigate under which circumstances the quantization noise of the filter coefficients affects the location of the filter's poles and zeros. The following derivations are taken from Oppenheim and Schafer (1999), pp. 402–411.

The filter coefficients and are typically stored with double precision. This means that we introduce a small quantization error for the filter coefficients. Therefore, the actual filter coefficients a_j and b_k in (2.62) differ from the stored filter coefficients, which we denote by $a_j^{(q)}$ and $b_k^{(q)}$. The superscript (q) indicates that these coefficients are quantized, i.e. rounded. We find

$$a_j^{(q)} = a_j + \Delta a_j \quad \text{and} \quad b_k^{(q)} = b_k + \Delta b_k, \tag{3.23}$$

wherein Δa_j und Δb_k are the quantization errors of the filter coefficients. Thus, the implemented transfer function reads

$$H^{(q)} = \frac{\sum_{k=0}^{P} b_k^{(q)} z^{-k}}{1 - \sum_{j=1}^{Q} a_j^{(q)} z^{-j}}. \tag{3.24}$$

The poles and zeros of this transfer function are not the same as the poles and zeros of the transfer function (2.62). Therefore, we denote the poles and zeros of the transfer function $H^{(q)}$ by $\alpha_j^{(q)}$ and $\beta_k^{(q)}$. They are related to the filter coefficients of the implemented filter by

$$\sum_{k=0}^{P} b_k^{(q)} z^{-k} = b_0^{(q)} \prod_{k=1}^{P}(1-\beta_k^{(q)} z^{-1}) \quad \text{and} \quad 1 - \sum_{j=1}^{Q} a_j^{(q)} z^{-j} = \prod_{j=1}^{Q}(1-\alpha_j^{(q)} z^{-1}). \tag{3.25}$$

For the following derivations, we will understand the quantized poles and zeros as a function of the quantized coefficients, i.e.

$$\alpha_j^{(q)} = f_{\alpha_j}(a_1^{(q)}, \ldots, a_Q^{(q)}) \quad \text{and} \quad \beta_k^{(q)} = f_{\beta_k}(b_0^{(q)}, \ldots, b_P^{(q)}). \tag{3.26}$$

Our goal is now to express the quantization errors of the poles and zeros

$$\Delta\alpha_j = \alpha_j^{(q)} - \alpha_j \quad \text{and} \quad \Delta\beta_k = \beta_k^{(q)} - \beta_k \tag{3.27}$$

in terms of the quantization errors Δa_j and Δb_k of the filter coefficients. We begin with a single pole α_i, which we extract from the denominator polynomial of the transfer function (3.26).

$$1 - \sum_{j=1}^{Q} a_j^{(q)} z^{-j} = \prod_{j=1}^{Q} \frac{z - \alpha_j^{(q)}}{z} = z^{-Q} \prod_{j=1}^{Q} (z - \alpha_j^{(q)}) = z^{-Q}(z - \alpha_i^{(q)}) \prod_{j=1, j\neq i}^{Q} (z - \alpha_j^{(q)}) \tag{3.28}$$

After solving this equation for α_i, we get

$$\alpha_i^{(q)} = z - z^Q \Big(1 - \sum_{j=1}^{Q} a_j^{(q)} z^{-j}\Big) \prod_{j=1, j\neq i}^{Q} (z - \alpha_j^{(q)})^{-1} = f_{\alpha_i}(a_1^{(q)}, \ldots, a_Q^{(q)}). \tag{3.29}$$

Now we have found the function f_{α_j} in (3.26). In order to avoid a recursion, we assume that the quantized poles $\alpha_j^{(q)}$ are constant values. In the next step, we linearize this function by expanding it in a Taylor series. After choosing the actual filter coefficients a_j as the Taylor point, we find for $\alpha_i^{(q)}$ in (3.29)

$$\begin{aligned} f_{\alpha_i}(a_1^{(q)}, \ldots, a_Q^{(q)}) &= f_{\alpha_i}(a_1 + \Delta a_1, \ldots, a_Q + \Delta a_Q) \\ &= f_{\alpha_i}(a_1, \ldots, a_Q) + \sum_{j=1}^{Q} \frac{\partial f_{\alpha_i}}{\partial a_j^{(q)}} \bigg|_{a_1^{(q)}=a_1, \ldots, a_Q^{(q)}=a_Q} \Delta a_j. \end{aligned} \tag{3.30}$$

Equality holds, because (3.29) is linear for the quantized filter coefficients $a_j^{(q)}$ and thus higher derivatives of f_{α_i} are zero. As $f_{\alpha_i}(a_1^{(q)}, \ldots, a_Q^{(q)}) = \alpha_i^{(q)}$ according to (3.26) and $f_{\alpha_i}(a_1, \ldots, a_Q) = \alpha_i$ analogously, (3.30) becomes

$$\alpha_i^{(q)} = \alpha_i + \sum_{j=1}^{Q} \frac{\partial f_{\alpha_i}}{\partial a_j^{(q)}} \bigg|_{a_1^{(q)}=a_1, \ldots, a_Q^{(q)}=a_Q} \Delta a_j. \tag{3.31}$$

With (3.27) we find

$$\Delta\alpha_i = \sum_{j=1}^{Q} \frac{\partial f_{\alpha_i}}{\partial a_j^{(q)}} \bigg|_{a_1^{(q)}=a_1, \ldots, a_Q^{(q)}=a_Q} \Delta a_j. \tag{3.32}$$

Building the partial derivatives yields

$$\Delta\alpha_i = \sum_{j=1}^{Q} z^{Q-j} \prod_{l=1, l\neq i}^{Q} (z - \alpha_l^{(q)})^{-1} \Delta a_j = \sum_{j=1}^{Q} z^{Q-j} \Delta a_j \prod_{l=1, l\neq i}^{Q} (z - \alpha_l^{(q)})^{-1}. \tag{3.33}$$

This equation relates the quantization errors Δa_j of the filter coefficients to the resulting quantization errors of the poles $\Delta\alpha_i$.

In order to investigate how the decorrelating capabilities of filters are affected by the quantization noise of the filter coefficients, we look at their power spectral density

$$|H(e^{i\omega})|^2 = b_0^2 \frac{\prod_{k=1}^{P} |1 - \beta_k e^{-i\omega}|^2}{\prod_{j=1}^{Q} |1 - \alpha_j e^{-i\omega}|^2} = b_0^2 \frac{\prod_{k=1}^{P} |\frac{e^{i\omega} - \beta_k}{e^{i\omega}}|^2}{\prod_{j=1}^{Q} |\frac{e^{i\omega} - \alpha_j}{e^{i\omega}}|^2} = b_0^2 \frac{\prod_{k=1}^{P} |e^{i\omega} - \beta_k|^2}{\prod_{j=1}^{Q} |e^{i\omega} - \alpha_j|^2}, \tag{3.34}$$

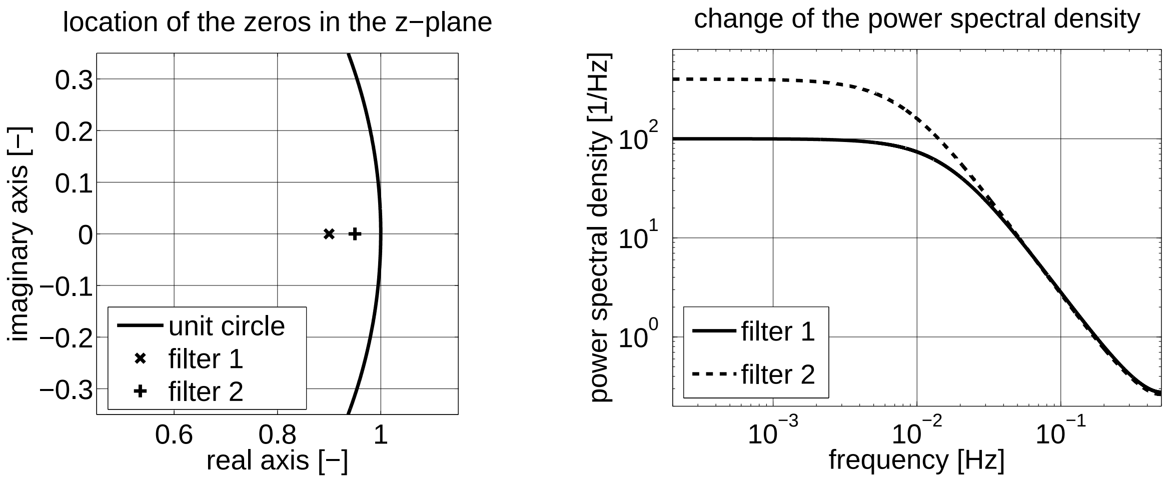


Figure 3.3: Effect of a small change in the location of a zero β on the decorrelating capabilities of an MA filter of order one. The left panel shows the location of the zero of two filters in the z-plane. The two zeros lie both close to each other and close to the unit circle. The effect of the small difference in the location of the zeros on the decorrelating capabilities of the filters is shown in the right panel. The right panel illustrates that the power spectral density of the two filters differs by a factor of four around the frequency $\omega = \sphericalangle(\beta) = 0$.

which is here expressed by poles and zeros (cf. equation (2.102)). The value $e^{i\omega}$ describes the unit circle in the z-plane. If a pole α_j or a zero β_k is located close to the unit circle in the z-plane, small changes in the location of the pole or zero cause large changes in the power spectral density $|H(e^{i\omega})|^2$ around $\omega \approx \sphericalangle(\alpha_j)$ or $\omega \approx \sphericalangle(\beta_k)$, respectively. Figure 3.3 illustrates the effect of a small change of the location of a zero.

Therefore, we analyze the effect of the quantization errors Δa_j of the filter coefficients on the quantization error $\Delta\alpha_i$ of a pole by substituting $z = e^{i\omega}$ in (3.33), which yields

$$\Delta\alpha_i = \sum_{j=1}^{Q} e^{i(Q-j)\omega} \Delta a_j \prod_{l=1, l\neq i}^{Q} (e^{i\omega} - \alpha_l^{(q)})^{-1}. \tag{3.35}$$

We can neglect the term $e^{i(Q-j)\omega}$ within our considerations, because its magnitude is equal to 1. The term $(e^{i\omega} - \alpha_l^{(q)})^{-1}$ is more interesting, because it becomes large for $\omega \approx \sphericalangle(\alpha_l^{(q)})$, if $\alpha_l^{(q)}$ is located close to the unit circle in the z-plane, i.e. $|\alpha_l^{(q)}| \approx 1$. Then, the quantization errors Δa_j are multiplied by a large value and the quantization error $\Delta\alpha_i$ of the pole α_i increases. However, if only a single pole $\alpha_l^{(q)}$ is close to the unit circle, the effect will still be small, because the quantization error Δa_j is typically much smaller than the large value $(e^{i\omega} - \alpha_l^{(q)})^{-1}$. Nevertheless, the situation gets much worse, if not only any single pole is close to the unit circle, but a whole series of poles are close to the unit circle and close to each other. Then the large values $(e^{i\omega} - \alpha_l^{(q)})^{-1}$ are multiplied by each other in (3.35) and their product $\prod_{l=1,l\neq i}^{Q}(e^{i\omega} - \alpha_l^{(q)})^{-1}$ can become dramatically large. In this case, we have to subdivide the filter into first or second order sections. For first order sections, this product vanishes completely, because then $Q = 1$ and we find

$$\Delta\alpha_1 = \sum_{j=1}^{1} e^{i(1-j)\omega} \Delta a_1 \prod_{l=1, l\neq 1}^{1} (e^{i\omega} - \alpha_l^{(q)})^{-1} = \Delta a_1. \tag{3.36}$$

For second order sections the product reduces to a single multiplier $e^{i\omega} - \alpha_l^{(q)}$, because then $Q = 2$ and we find

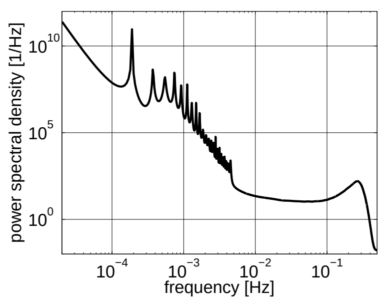
$$\Delta\alpha_1 = \sum_{j=1}^{2} e^{i(2-j)\omega} \Delta a_j \prod_{l=1, l\neq 1}^{2} (e^{i\omega} - \alpha_l^{(q)})^{-1} = \frac{e^{i\omega}\Delta a_1 + \Delta a_2}{e^{i\omega} - \alpha_2^{(q)}} \tag{3.37}$$

and

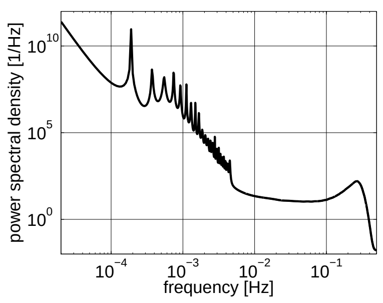
$$\Delta\alpha_2 = \sum_{j=1}^{2} e^{i(2-j)\omega}\Delta a_j \prod_{l=1,l\neq 2}^{2} (e^{i\omega} - \alpha_l^{(q)})^{-1} = \frac{e^{i\omega}\Delta a_1 + \Delta a_2}{e^{i\omega} - \alpha_1^{(q)}}. \tag{3.38}$$

The same discussion takes place for the quantization errors $\Delta\beta_i$ of the zeros. Therefore we can state the following rule:

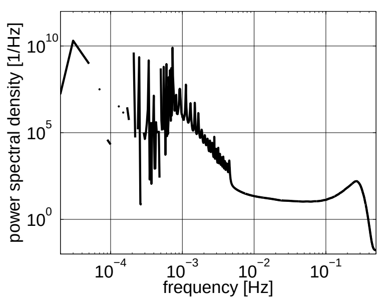
If a series of poles or a series of zeros is located close to the unit circle and close to each other, then first and second order sections should be used to implement the filter. In practice, this rule is followed already in the design stage of decorrelating filters. Figure 3.4 shows the effect of disregarding this instruction on the decorrelating capabilities of filter at the example of the aggressive filter described in section 6.4.



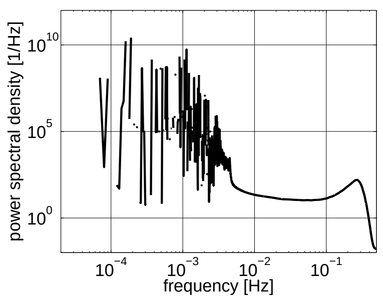
(a) 27 sections, no sections merged



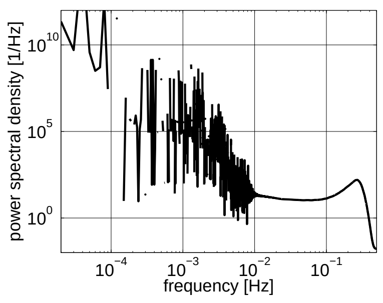
(b) 26 sections, 2 sections merged



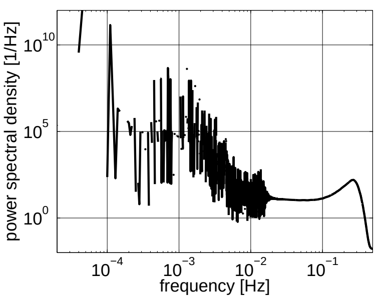
(c) 25 sections, 3 sections merged



(d) 24 sections, 4 sections merged



(e) 23 sections, 5 sections merged



(f) 22 sections, 6 sections merged

Figure 3.4: If a series of poles or a series of zeros is located close to the unit circle and close to each other, then first and second order sections should be used to implement the filter. If this instruction is disregarded, the decorrelating capabilities of a filter can be destroyed, which is demonstrated in this figure. Missing values in this figure are explained by the fact that negative values cannot be displayed in logarithmic scaled plots. The figures 3.4(a) to 3.4(f) show the power spectral density of the aggressive ARMA filter described in section 6.4 for the case that critical filter sections are successively merged to one section. In figure 3.4(b) two critical sections are merged, which does not affect the decorrelating capabilities of the filter. However, if three critical sections are merged, which is shown in figure 3.4(c), the power spectral density is altered and even contains values equal to infinity.

Allocation of Poles and Zeros in Filter Sections

In the preceding subsections we have seen how to construct first and second order sections of ARMA filters. Furthermore, we have discussed under which circumstances this is necessary. If it is necessary, we can use equation (3.20) to factorize the transfer function $H(z)$ into poles and zeros. After that, we can use the equations (3.21) and (3.22) in order to group poles and zeros into first and second order sections. The question arises, if this grouping can be performed arbitrarily. As discussed by DEHNER (2003), the grouping of poles and zeros should not be performed arbitrarily, because otherwise the magnitude of quantization noise increases significantly. To find the best possible combination of poles and zeros grouped in first and second order sections is computational very demanding task as the number of combinations increases proportional to the factorial of the filter order. However, Dehner (2003) gives three heuristic rules for the allocation of poles and zeros in first and second order sections, which are repeated here for convenience:

Rule 1 Starting with the poles closest to the unit circle, in a first step those poles and zeros are combined which are situated most closely to each other. Then without taking into account already combined pairs, the procedure is repeated in the following steps.

Rule 2 Having assigned pairs of poles and zeros, the pairs have to be allocated in the section in a way that both the corresponding numerator and denominator have an effect on one and the same quantization point.

Rule 3 Poles are sorted according to their distance from the unit circle and consequently to their resonance behavior (cf. Dehner 2003). Within the section the pairs of poles and zeros, assembled according to the first rule, are either put in an ascendant ("up-ordering") or in a descendant order ("down-ordering" regarding their resonance behavior).

Note, that rule 1 just gives the instruction which poles and zeros should be combined to pairs, but not what to do with these pairs. Rule 2 then states that each of the pairs of poles and zeros resulting from rule 1 should form one filter section. How to order these filter sections within the whole filter is then defined by rule 3. The effect of disregarding these rules is illustrates in figure 3.5.

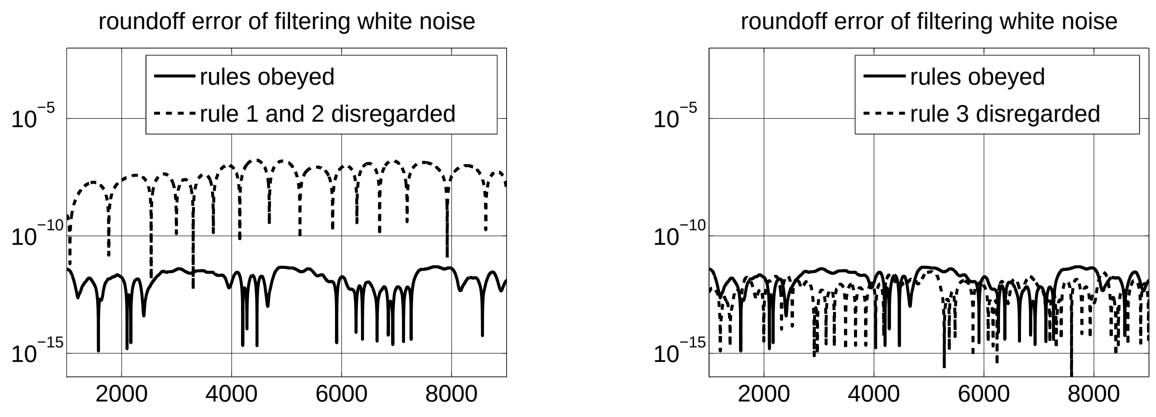


Figure 3.5: Effect of disregarding the rules given by Dehner (2003) for forming filter sections. The simple averaging ARMA filter described in section 6.4 has been used to create this figure. The standard deviation of the filtered time series is normalized to one, such that this figure directly indicates the numerical precision of the filtering. The reference time series has been computed with a precision of 40 digits using ASCII arithmetic, which is superior to double precision, which corresponds to approximately 15 digits. The left panel shows the effect of disregarding rule 1 and 2, which give advice for the pairing and allocation of zeros and poles. The standard deviation of the roundoff noise for the case the these rules are obeyed is equal to $2.0 \cdot 10^{-12}$. The standard deviation of the roundoff noise for the case that these rules are disregarded is equal to $1.1 \cdot 10^{-7}$. Apparently, disregarding these rules has a large effect on the roundoff noise, which is increased by a factor of $5.5 \cdot 10^{4}$. The right panel shows the effect of disregarding rule 3, which gives advice for the ordering of the filter sections. The standard deviation of the roundoff noise is equal to $1.6 \cdot 10^{-12}$, which is even smaller than the standard deviation of the reference time series. This indicates that obeying rule 3 is not very important in this example.

Block Formulation of the Filter Equation

The goal of this section is to derive a block implementation of ARMA filters. At the beginning of this section we have discussed, that there are cases, where we have to split the filter into sections, which are themselves full-fledged filters. For the moment, we neglect this fact for the sake of simplicity. Later on, when we derive the filter algorithm, we will take this fact back into consideration. The derivation of the block implementation of ARMA filters is not as straightforward as it is for MA filters. The reason for this lies within the recursion for y_n in the filter equation (2.53). We have to resolve this recursion at least for the desired block size of the block implementation. In order to do so, we start by formulating the filter equation (2.53) as a state-space model in the following.

In the literature exists a large number of different state-space models for ARMA filters (cf. for example Hamilton 1994, chapter 13 or Schlittgen and Streitberg 2001, pp. 144–151). Therefore, our first task is to choose the best one for our purpose, which is to derive a fast block implementation of ARMA filters with little memory requirements. The state-space formulation of our choice is the following. First, we define

$$\boldsymbol{y}_n = \begin{bmatrix} y_{n-Q+1} & y_{n-Q+2} & \cdots & y_n \end{bmatrix}^\top \tag{3.39}$$

and

$$\boldsymbol{z}_n = \begin{bmatrix} 0 & \cdots & 0 & z_n \end{bmatrix}^\top, \tag{3.40}$$

wherein

$$z_n = \sum_{k=0}^{P} b_k x_{n-k}. \tag{3.41}$$

If now

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ a_Q & a_{Q-1} & \cdots & a_2 & a_1 \end{bmatrix}, \tag{3.42}$$

then

$$\boldsymbol{y}_n = \boldsymbol{A}\boldsymbol{y}_{n-1} + \boldsymbol{z}_n \tag{3.43}$$

holds. The last row of the equations in (3.43) contains the filter equation (2.53) while all other rows just state that $y_k = y_k$ for $k = n - Q + 1, \ldots, n - 1$, which is apparently true. Within this thesis, we refer to equation (3.43) as the state-space equation.

With the state-space equation (3.43) as a starting point, we now derive a block formulation of the filter equation (2.53). The procedure is in principle the same as the one of Sung and Mitra (1986). However, we use a different state-space equation. Shifting the index in equation (3.43) yields

$$\boldsymbol{y}_{n-1} = \boldsymbol{A}\boldsymbol{y}_{n-2} + \boldsymbol{z}_{n-1}. \tag{3.44}$$

With this equation, we can substitute $\boldsymbol{y}_{n-1}$ in the state space equation (3.43) and we find

$$\boldsymbol{y}_n = \boldsymbol{A}\boldsymbol{y}_{n-1} + \boldsymbol{z}_n = \boldsymbol{A}(\boldsymbol{A}\boldsymbol{y}_{n-2} + \boldsymbol{z}_{n-1}) + \boldsymbol{z}_n = \boldsymbol{A}^2\boldsymbol{y}_{n-2} + \boldsymbol{A}\boldsymbol{z}_{n-1} + \boldsymbol{z}_n. \tag{3.45}$$

After repeating such substitutions $k - 2$ times, we obtain

$$\boldsymbol{y}_n = \boldsymbol{A}^k\boldsymbol{y}_{n-k} + \sum_{l=0}^{k-1} \boldsymbol{A}^l \boldsymbol{z}_{n-l}. \tag{3.46}$$

Herein, $\boldsymbol{A}^0 = \boldsymbol{I}$.

We now verify equation (3.46) by induction. The basis of the induction is already given by equation (3.45). Furthermore, we substitute $\boldsymbol{y}_{n-k}$ according to the state-space equation (3.43) and find

$$\begin{aligned}\boldsymbol{y}_n &= \boldsymbol{A}^k \boldsymbol{y}_{n-k} + \sum_{l=0}^{k-1} \boldsymbol{A}^l \boldsymbol{z}_{n-l} \\ &= \boldsymbol{A}^k (\boldsymbol{A}\boldsymbol{y}_{n-k-1} + \boldsymbol{z}_{n-k}) + \sum_{l=0}^{k-1} \boldsymbol{A}^l \boldsymbol{z}_{n-l} \\ &= \boldsymbol{A}^{k+1} \boldsymbol{y}_{n-k-1} + \boldsymbol{A}^k \boldsymbol{z}_{n-k} + \sum_{l=0}^{k-1} \boldsymbol{A}^l \boldsymbol{z}_{n-l} \\ &= \boldsymbol{A}^{k+1} \boldsymbol{y}_{n-k-1} + \sum_{l=0}^{k} \boldsymbol{A}^l \boldsymbol{z}_{n-l}, \end{aligned} \tag{3.47}$$

which proves equation (3.46).

Similarly as for the block implementation of MA filters, we define L to be the block size, i.e. in each filter step we gain L new values of y_n. After setting $k = L$ in equation (3.46), we find

$$\boldsymbol{y}_n = \boldsymbol{A}^L \boldsymbol{y}_{n-L} + \sum_{l=0}^{L-1} \boldsymbol{A}^l \boldsymbol{z}_{n-l}. \tag{3.48}$$

The last L elements of $\boldsymbol{y}_n$ contain the L new values of y_n. After choosing $L > Q$, then $\boldsymbol{y}_n$ consists of too few elements. Therefore, we assume $L \le Q$ for the moment and discuss the necessary changes later. We can simplify the computation, if we take into account, that only the last element of $\boldsymbol{z}_{n-l}$ is non-zero. After defining

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{A}^{L-1}_{:,Q} & \boldsymbol{A}^{L-2}_{:,Q} & \cdots & \boldsymbol{A}_{:,Q} & \boldsymbol{I}_{:,Q} \end{bmatrix}, \tag{3.49}$$

we can write

$$\boldsymbol{y}_n = \boldsymbol{A}^L \boldsymbol{y}_{n-L} + \sum_{l=0}^{L-1} \boldsymbol{A}^l \boldsymbol{z}_{n-l} = \boldsymbol{A}^L \boldsymbol{y}_{n-L} + \sum_{l=0}^{L-1} \boldsymbol{A}^l_{:,Q} z_{n-l} = \boldsymbol{A}^L \boldsymbol{y}_{n-L} + \boldsymbol{D} \begin{bmatrix} z_{n-L+1} \\ \vdots \\ z_n \end{bmatrix}. \tag{3.50}$$

Herein, $\boldsymbol{A}^l_{:,Q}$ is the last column of $\boldsymbol{A}^l$. The next step within the derivation is to substitute z_n for x_n. In order to do so, we define

$$\boldsymbol{B} = \begin{bmatrix} b_P & b_{P-1} & \cdots & b_0 & 0 & \cdots & 0 \\ 0 & b_P & \ddots & & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & b_{P-1} & \cdots & b_0 & 0 \\ 0 & \cdots & 0 & b_P & \cdots & b_1 & b_0 \end{bmatrix}. \tag{3.51}$$

According to equation (3.41), we find

$$\begin{bmatrix} z_{n-L+1} \\ \vdots \\ z_n \end{bmatrix} = \boldsymbol{B} \begin{bmatrix} x_{n-L-P+1} \\ \vdots \\ x_n \end{bmatrix} = \boldsymbol{B}_{:,1:P} \begin{bmatrix} x_{n-L-P+1} \\ \vdots \\ x_{n-L} \end{bmatrix} + \boldsymbol{B}_{:,P+1:P+L} \begin{bmatrix} x_{n-L+1} \\ \vdots \\ x_n \end{bmatrix}, \tag{3.52}$$

where

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{B}_{:,1:P} \boldsymbol{B}_{:,P+1:P+L} \end{bmatrix} \tag{3.53}$$

The reason for splitting matrix $\boldsymbol{B}$ is to separate the old values x_n, which were used in the previous filter step, from the new values x_n, which were not used in the previous filter step. After combining equation (3.50) and equation (3.52), we obtain

$$\boldsymbol{y}_n = \boldsymbol{A}^L \boldsymbol{y}_{n-L} + \boldsymbol{D}\boldsymbol{B}_{:,1:P} \begin{bmatrix} x_{n-L-P+1} \\ \vdots \\ x_{n-L} \end{bmatrix} + \boldsymbol{D}\boldsymbol{B}_{:,P+1:P+L} \begin{bmatrix} x_{n-L+1} \\ \vdots \\ x_n \end{bmatrix}, \tag{3.54}$$

which is the desired block formulation of the filter equation (2.53). If $L < Q$, then only the last L elements of vector $\boldsymbol{y}_n$ are new. The other elements of this vector are already known from the previous filter step and are contained in vector $\boldsymbol{y}_{n-L}$. We can therefore reduce the computational effort by computing only the last L elements of $\boldsymbol{y}_n$. Thus, we need only the last L rows of the matrices $\boldsymbol{A}^L$, $\boldsymbol{D}\boldsymbol{B}_{:,1:P}$ and $\boldsymbol{D}\boldsymbol{B}_{:,P+1:P+L}$. So far, we omitted the case $L > Q$. For this case we simply redefine vector $\boldsymbol{y}_n$ such that it contains more elements, i.e. L elements instead of Q elements.

$$\boldsymbol{y_n} = \begin{bmatrix} y_{n-L+1} & y_{n-L+2} & \cdots & y_{n-1} & y_n \end{bmatrix}^\top \tag{3.55}$$

We also need to redefine matrix $\boldsymbol{A}$ such that the matrix dimensions of equation (3.54) fit each other.

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ a_L & a_{L-1} & \cdots & a_2 & a_1 \end{bmatrix} \tag{3.56}$$

Herein, the additional coefficients a_j are zero, i.e. $a_{Q+1} = a_{Q+2} = \cdots = a_L = 0$. The other coefficients $a_1, \ldots, a_Q$ remain unchanged. In all cases, which are $L < Q$, $L = Q$ and $L > Q$, only the last L rows of the matrices $\boldsymbol{A}^L$, $\boldsymbol{D}\boldsymbol{B}_{:,1:P}$ and $\boldsymbol{D}\boldsymbol{B}_{:,P+1:P+L}$ are needed. For $L \geq Q$ the last L rows contain the whole matrices, but this does not change the statement.

Equation (3.54) is the same block formulation, as proposed by Burrus (1972), formula (17). However, Burrus (1972) aims in his further derivations at the application of FFT techniques, which we do not. For us, the goal is to use optimized libraries for matrix multiplications such as the BLAS/ATLAS library in order to reduce the computing time. We also chose a different derivation of the block formulation than Burrus (1972). The reason for this is, that our derivation provides more insight which we can exploit for the computation of the matrices $\boldsymbol{A}^L$ and $\boldsymbol{D}$. Furthermore, we are concerned about the memory requirements which become important, if we filter not a single vector but a whole matrix. The reason for this is, that in least squares adjustment procedures, we typically need to filter the design matrix at least once.

Fast Computation of the Filter Matrices

Before we proceed with the derivation of the actual filter algorithm, we investigate the computation of the matrices $\boldsymbol{A}^L$ and $\boldsymbol{D}$. A straightforward computation of $\boldsymbol{A}^L$ is to start with the identity matrix $\boldsymbol{I}$ and multiply it L times by matrix $\boldsymbol{A}$. According to equation (3.49) each intermediate product $\boldsymbol{A}^l$, where $l = 0, \ldots, L-1$, contributes one column of matrix $\boldsymbol{D}$. But if the block size L is large, both the number of products and the size of matrix $\boldsymbol{A}$ increase and the computations become expensive. However, there exists a very efficient way of computing $\boldsymbol{A}^L$ in place, which costs no more than one multiplication and one addition of two matrices of the size of matrix $\boldsymbol{A}$. Furthermore, we will see, that we can set up matrix $\boldsymbol{D}$ by simply copying when $\boldsymbol{A}^L$ is already computed. Thus the computational effort for the computation of $\boldsymbol{A}^L$ and $\boldsymbol{D}$ is very low and typically negligible in comparison to the computational costs of the actual filtering.

As already mentioned, we can obtain matrix $\boldsymbol{A}^L$, if we start with the identity matrix $\boldsymbol{I}$ and multiply it L times by $\boldsymbol{A}$. After defining

$$\boldsymbol{a} = \begin{bmatrix} a_U & \cdots & a_1 \end{bmatrix} \tag{3.57}$$

wherein

$$U = \max(Q, L), \tag{3.58}$$

we can split matrix $\boldsymbol{A}$ into two matrices, whereas each is subdivided into sub-matrices.

$$\boldsymbol{A} = \begin{bmatrix} \underset{U-1\times 1}{\mathbf{0}} & \underset{U-1\times U-1}{\boldsymbol{I}} \\ \underset{1\times 1}{\mathbf{0}} & \underset{1\times U-1}{\mathbf{0}} \end{bmatrix} + \begin{bmatrix} \underset{U-1\times U}{\mathbf{0}} \\ \underset{1\times U}{\boldsymbol{a}} \end{bmatrix} \tag{3.59}$$

The left matrix on the right side of the equation is called a shift matrix. This definition for $\boldsymbol{A}$ is general in the sense that it holds for both cases $L \leq Q$ and $L > Q$, which we have treated separate so far. After multiplying an arbitrary matrix $\boldsymbol{M}$ of size $U \times U$ by matrix $\boldsymbol{A}$, we obtain

$$\boldsymbol{AM} = \begin{bmatrix} \boldsymbol{M}_{2:U,:} \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \boldsymbol{aM} \end{bmatrix} = \begin{bmatrix} \boldsymbol{M}_{2:U,:} \\ \boldsymbol{aM} \end{bmatrix} \tag{3.60}$$

Obviously, matrix $\boldsymbol{M}$ is shifted upwards by one row and therefore we need to compute only the last row of the product $\boldsymbol{AM}$. Thus, the computation of $\boldsymbol{A}^L$ reduces to L computations of the last row of the intermediate products. In order to avoid the shifting, we have a closer look at these intermediate products. For the first three intermediate products we find

$$\boldsymbol{A} = \boldsymbol{AI} = \begin{bmatrix} \boldsymbol{I}_{2:U,:} \\ \boldsymbol{a} \end{bmatrix}, \quad \boldsymbol{A}^2 = \boldsymbol{AA} = \begin{bmatrix} \boldsymbol{A}_{2:U,:} \\ \boldsymbol{aA} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_{3:U,:} \\ \boldsymbol{a} \\ \boldsymbol{aA} \end{bmatrix}, \quad \boldsymbol{A}^3 = \boldsymbol{AA}^2 = \begin{bmatrix} \boldsymbol{A}^2_{2:U,:} \\ \boldsymbol{aA}^2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_{4:U,:} \\ \boldsymbol{a} \\ \boldsymbol{aA} \\ \boldsymbol{aA}^2 \end{bmatrix}. \tag{3.61}$$

After continuing this series, we obtain for the k-th intermediate product

$$\boldsymbol{A}^k = \begin{bmatrix} \boldsymbol{A}^{k-1}_{2:U,:} \\ \boldsymbol{aA}^{k-1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_{k+1:U,:} \\ \boldsymbol{a} \\ \boldsymbol{aA} \\ \vdots \\ \boldsymbol{aA}^{k-1} \end{bmatrix}. \tag{3.62}$$

In the previous section we discussed, that we only need the last L rows of matrix $\boldsymbol{A}^L$. For these rows we find

$$\boldsymbol{A}^L_{U-L+1:L,:} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{aA} \\ \vdots \\ \boldsymbol{aA}^{L-1} \end{bmatrix}. \tag{3.63}$$

Hence, starting with the first row of $\boldsymbol{A}^L_{U-L+1:L,:}$, we can compute all other rows by

$$\boldsymbol{aA}^k = \boldsymbol{aA}^{k-1}\boldsymbol{A} \tag{3.64}$$

for $k = 1, \ldots, L-1$. For this reason we investigate the multiplication of an arbitrary vector $\boldsymbol{m}$ by matrix $\boldsymbol{A}$.

$$\boldsymbol{mA} = \boldsymbol{m} \begin{bmatrix} \boldsymbol{I}_{2:U,:} \\ \boldsymbol{a} \end{bmatrix} = \begin{bmatrix} \boldsymbol{m}_{1:U-1} & \boldsymbol{m}_U \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{2:U,:} \\ \boldsymbol{a} \end{bmatrix} = \begin{bmatrix} 0 & \boldsymbol{m}_{1:U-1} \end{bmatrix} + \boldsymbol{m}_U \boldsymbol{a} \tag{3.65}$$

When we apply this formula to successively compute the rows of $\boldsymbol{A}^L_{U-L+1:L,:}$, vector $\boldsymbol{m}$ is equal to vector $\boldsymbol{a}$ in the first step. Because the first $U - Q$ elements of vector $\boldsymbol{a}$ are zero, according to equation (3.65) the first $U - Q$ elements of the product $\boldsymbol{aA}$ are zero, too. In the second step vector $\boldsymbol{m}$ equals $\boldsymbol{aA}$. Because the first $U - Q$ elements of $\boldsymbol{m} = \boldsymbol{aA}$ are zero, according to equation (3.65) the first $U - Q$ elements of the product $\boldsymbol{aA}^2$ are zero, too. After proceeding in this way, we see that the first $U - Q$ columns of matrix $\boldsymbol{A}^L_{U-L+1:L,:}$

are equal to zero while the last Q columns are non-zero. Therefore, we need the last L rows and the last Q columns of matrix $\boldsymbol{A}^L$. For simplicity we define

$$\boldsymbol{F}^{(y)} = \boldsymbol{A}^L_{U-L+1:L,U-Q+1:U}. \tag{3.66}$$

The size of this matrix is $L \times Q$. When we compute $\boldsymbol{F}^{(y)}$ row by row, we can leave out the first $U-Q$ elements. Equation (3.65) then becomes

$$\boldsymbol{m}_{U-Q+1:U}\boldsymbol{A}_{U-Q+1:U,U-Q+1:U} = \begin{bmatrix}0 & \boldsymbol{m}_{U-Q+1:U-1}\end{bmatrix} + m_U \boldsymbol{a}_{U-Q+1:U}. \tag{3.67}$$

For the computation of the k-th row of $\boldsymbol{F}^{(y)}$ we use $\boldsymbol{m} = \boldsymbol{a}\boldsymbol{A}^{k-2}$. The first row of $\boldsymbol{F}^{(y)}$ contains vector $\boldsymbol{a}$. This leads us directly to algorithm 3.1 for the in-place computation of $\boldsymbol{F}^{(y)}$.

input	$a_1, \ldots, a_Q$	filter coefficients
output	$\boldsymbol{F}^{(y)}$	filter matrix of dimension $L \times Q$

```
initialization
$\boldsymbol{F}^{(y)}_{1,:} = \begin{bmatrix} a_Q & \cdots & a_1 \end{bmatrix}$

loop
for $k = 1, 2, \ldots, L-1$
    $\boldsymbol{F}^{(y)}_{k+1,:} = \boldsymbol{F}^{(y)}_{k,Q}\boldsymbol{F}^{(y)}_{1,:}$
    $\boldsymbol{F}^{(y)}_{k+1,2:Q} = \boldsymbol{F}^{(y)}_{k+1,2:Q} + \boldsymbol{F}^{(y)}_{k,1:Q-1}$
end
```

Algorithm 3.1: In-place computation of the filter matrix $\boldsymbol{F}^{(y)}$.

It should be mentioned that $\boldsymbol{A}$ is a companion matrix (cf. Meyer 2000, p. 648). There are also other ways of efficiently computing powers of companion matrices, cf. for example Kaufman (1969). However, algorithm 3.1 is easy to implement and works very fast.

In equation (3.59) matrix $\boldsymbol{A}$ is defined to be of size $U \times U$, where $U = \max(Q, L)$. This definition affects the definition of matrix $\boldsymbol{D}$ in (3.49), which now becomes

$$\boldsymbol{D} = \begin{bmatrix}\boldsymbol{A}^{L-1}_{:,U} & \boldsymbol{A}^{L-2}_{:,U} & \cdots & \boldsymbol{A}_{:,U} & \boldsymbol{I}_{:,U}\end{bmatrix}. \tag{3.68}$$

From equation (3.60) we know that multiplying an arbitrary matrix $\boldsymbol{M}$ by matrix $\boldsymbol{A}$ results in shifting that matrix $\boldsymbol{M}$ one row upwards. Therefore, matrix $\boldsymbol{D}$ has a Toeplitz structure, as its columns are obtained by successive multiplications by matrix $\boldsymbol{A}$. The size of matrix $\boldsymbol{D}$ is $U \times L$. Like for matrix $\boldsymbol{A}^L$ only the last L rows of matrix $\boldsymbol{D}$, i.e. $\boldsymbol{D}_{U-L+1:U,:}$, are needed to obtain L new values of y_n. Thus, we only need to set up $\boldsymbol{D}_{U-L+1:U,:}$ which is of size $L \times L$. Because the last column of $\boldsymbol{D}_{U-L+1:U,:}$ equals the last column of an identity matrix and because of the Toeplitz structure of $\boldsymbol{D}$, the upper triangular part of $\boldsymbol{D}_{U-L+1:U,:}$ is equal to the upper triangular part of an identity matrix. The lower triangular part of $\boldsymbol{D}_{U-L+1:U,:}$ without its main diagonal is a shifted version of the last column of matrix $\boldsymbol{A}^L$. Hence, we find

$$\boldsymbol{D}_{U-L+1:U,:} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ \boldsymbol{A}^L_{1,U} & 1 & \ddots & \vdots & \vdots \\ \boldsymbol{A}^L_{2,U} & \boldsymbol{A}^L_{1,U} & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & 1 & 0 \\ \boldsymbol{A}^L_{U-1,U} & \boldsymbol{A}^L_{U-2,U} & \cdots & \boldsymbol{A}^L_{1,U} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ \boldsymbol{F}^{(y)}_{1,Q} & 1 & \ddots & \vdots & \vdots \\ \boldsymbol{F}^{(y)}_{2,Q} & \boldsymbol{F}^{(y)}_{1,Q} & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & 1 & 0 \\ \boldsymbol{F}^{(y)}_{L-1,Q} & \boldsymbol{F}^{(y)}_{L-2,Q} & \cdots & \boldsymbol{F}^{(y)}_{1,Q} & 1 \end{bmatrix}. \tag{3.69}$$

Obviously, we can set up matrix $\boldsymbol{D}_{U-L+1:U,:}$ by simple copying, if matrix $\boldsymbol{F}^{(y)}$ is already computed.

Filter Algorithm

Equation (3.54) is the block formulation of the filter equation (2.53). It is the starting point for the derivation of the filter algorithm. The computation of the matrices $\boldsymbol{A}^L$ and $\boldsymbol{D}$ in the block formulation (3.54) can be expensive, if it is performed in a straightforward manner and L is not a small value. Therefore we described in the previous subsection, how these matrices can be computed very efficiently. However, we left out the possibility that the filter can consist of more than one section. At the end of the following derivations, we take the filter sections back into consideration.

For the implementation of the block-wise filtering, we need to modify the block formulation of the filter in equation (3.54). First we define

$$\boldsymbol{F}^{(x)} = \boldsymbol{D}_{U-L+1:U,:}\boldsymbol{B}_{:,1:P} \quad \text{and} \quad \boldsymbol{F} = \boldsymbol{D}_{U-L+1:U,:}\boldsymbol{B}_{:,P+1:P+L}. \tag{3.70}$$

In each filter step we compute L new values of y_n. Therefore, we only need to compute the last L rows of equation (3.54), which are

$$\begin{bmatrix} y_{n-L+1} \\ \vdots \\ y_n \end{bmatrix} = \boldsymbol{F}^{(y)} \begin{bmatrix} y_{n-L-Q+1} \\ \vdots \\ y_{n-L} \end{bmatrix} + \boldsymbol{F}^{(x)} \begin{bmatrix} x_{n-L-P+1} \\ \vdots \\ x_{n-L} \end{bmatrix} + \boldsymbol{F} \begin{bmatrix} x_{n-L+1} \\ \vdots \\ x_n \end{bmatrix}. \tag{3.71}$$

Herein, matrix $\boldsymbol{F}$ is the actual filter matrix. It connects the current block $x_{n-L+1}, \dots, x_n$ of size L of the time series with the current block $y_{n-L+1}, \dots, y_n$ of the filtered time series. Therefore, it connects the input and output of the filter. The matrices $\boldsymbol{F}^{(x)}$ and $\boldsymbol{F}^{(y)}$ are associated with values of x_n and y_n, which were already used in the preceding filter steps. We could say that the filter has to remember these values for the current filter step. Thus, we call $y_{n-L-Q+1}, \dots, y_{n-L}$ and $x_{n-L-P+1}, \dots, x_{n-L}$ the memory of the filter and $\boldsymbol{F}^{(x)}$ and $\boldsymbol{F}^{(y)}$ memory matrices.

For the sake of simplicity in the following derivations, we define

$$\boldsymbol{m}^{(x)} = \begin{bmatrix} x_{n-L-P+1} \\ \vdots \\ x_{n-L} \end{bmatrix}, \quad \boldsymbol{m}^{(y)} = \begin{bmatrix} y_{n-L-Q+1} \\ \vdots \\ y_{n-L} \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_{n-L+1} \\ \vdots \\ x_n \end{bmatrix} \quad \text{and} \quad \boldsymbol{y} = \begin{bmatrix} y_{n-L+1} \\ \vdots \\ y_n \end{bmatrix}. \tag{3.72}$$

One major goal of this section is to derive a fast block implementation of ARMA filters. Therefore, we use the BLAS/ATLAS library for matrix multiplications. Because the BLAS routines work more efficiently for bigger matrices, we combine

$$\boldsymbol{m} = \begin{bmatrix} \boldsymbol{m}^{(x)} \\ \boldsymbol{m}^{(y)} \end{bmatrix} \quad \text{and} \quad \boldsymbol{E} = \begin{bmatrix} \boldsymbol{F}^{(x)} & \boldsymbol{F}^{(y)} \end{bmatrix}. \tag{3.73}$$

With these definitions, the last L rows of the block formulation (3.54) become

$$\boldsymbol{y} = \boldsymbol{E}\boldsymbol{m} + \boldsymbol{F}\boldsymbol{x}. \tag{3.74}$$

In order to optimize the computation of (3.74), we investigate now the structure of the involved matrices. Matrix $\boldsymbol{F}$ is obtained as the product of two lower triangular matrices and thus is itself a triangular matrix. Matrix $\boldsymbol{F}^{(x)}$ is obtained as the product of a lower triangular matrix and an upper trapezoidal matrix while matrix $\boldsymbol{F}^{(y)}$ is general matrix. Thus, matrix $\boldsymbol{E}$ is a general matrix from the point of view of matrix multiplications. Figure 3.6 illustrates the structure of the matrices.

Therefore, we can exploit only the triangular structure of matrix $\boldsymbol{F}$. The multiplication $\boldsymbol{F}\boldsymbol{x}$ can be performed approximately twice as fast and in-place. This accommodates the second major goal of this section, which

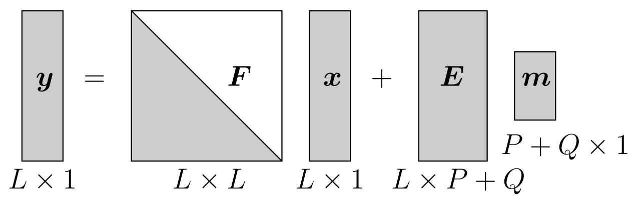


Figure 3.6: Structure of the filter matrices of the ARMA block filter method. The filter matrix $\boldsymbol{F}$ has a lower triangular structure, which can be exploited by the BLAS routines. The filter matrix $\boldsymbol{E}$ is a general matrix.

is to derive a block implementation with low memory requirements, because we do not need memory for $\boldsymbol{y}$. We only need to store the values $x_{n-P+1}, \ldots, x_n$ before the multiplication $\boldsymbol{Fx}$, because we need them for the memory $\boldsymbol{m}$ of the next filter step and they will be lost due to the in-place computation. For the storage of these values we introduce the vector

$$\boldsymbol{n} = \begin{bmatrix} x_{n-P+1} \\ \vdots \\ x_n \end{bmatrix}. \tag{3.75}$$

Now, the computations for one filter step are performed in the following order. For simplicity, we assume that $L > P, Q$.

$$\boldsymbol{n} = \boldsymbol{x}_{L-P+1:L} \tag{3.76}$$

$$\boldsymbol{x} = \boldsymbol{Fx} \tag{3.77}$$

$$\boldsymbol{x} = \boldsymbol{Em} + \boldsymbol{x} \tag{3.78}$$

$$\boldsymbol{m}_{1:P} = \boldsymbol{n} \tag{3.79}$$

$$\boldsymbol{m}_{P+1:P+Q} = \boldsymbol{x}_{L-Q+1:L} \tag{3.80}$$

At the end of these computation, vector $\boldsymbol{x}$ contains the block of filtered values y_n, i.e.

$$\boldsymbol{x} = \begin{bmatrix} y_{n-L+1} \\ \vdots \\ y_n \end{bmatrix}. \tag{3.81}$$

If $L < P$, we must replace equation (3.76), because then the filter memory $\boldsymbol{m}^{(x)}$, which is stored in $\boldsymbol{n}$ for then next filter step, has more rows than $\boldsymbol{x}$. Analogously, we must replace equation (3.80), if $L < Q$, because then the filter memory $\boldsymbol{m}^{(y)}$ has more rows than $\boldsymbol{x}$. In both cases we first copy the lower part of $\boldsymbol{m}^{(x)}$ or $\boldsymbol{m}^{(y)}$ to the upper part before we copy the new values from $\boldsymbol{x}$ to the lower part. The detailed indexing is given in algorithm 3.2.

Now, we take the filter sections back into consideration. This means, that the filter consists of several sections, that are themselves filters, which have to be applied serially. There are two principle ways of implementing the filter sections. The first one is, that we filter the whole time series x_n by the first section and then proceed with the other sections. The second one is, that we filter the first block of x_n serially by all sections and then proceed with the next blocks of x_n. We favor the second way for the following reason. Within adjustment procedures we need to filter the design matrix at least once, if the normal equation matrix cannot be approximated by means of analytical formulas. For the applications we consider, the design matrix is typically that large, that we can have only a very small portion of it in the working memory. If we would choose the first way, we would need to save the design matrix after filtering with the first section on some big storage medium, so that we can load it again for the next filter section. This saving and loading is very time consuming and needs lots of storage space, which rules out this way of implementing the filter sections. The second way of implementing the filter requires only memory capacity for one block of the design matrix plus some memory capacity for the filter memories. Hence, the second way allows sequentially computing and filtering the design matrix and performing further processing block by block.

If the filter consists of more than one section, we have to compute the filter matrix $\boldsymbol{F}$ and the memory matrix $\boldsymbol{E}$ for each filter section, whereas we denote by the superscript s to which section the filter matrix or memory matrix belongs, i.e. $\boldsymbol{F}^{(s)}$ and $\boldsymbol{E}^{(s)}$. Furthermore, each filter section has its own filter memory $\boldsymbol{m}^{(s)}$ and $\boldsymbol{n}^{(s)}$, which we also denote by the superscript s. Using this notation, we can formulate the filter algorithm 3.2, which is based on the equations (3.76) to (3.80).

Finally, we should mention a simple trick, which can make the filter algorithm slightly faster. This trick only applies to the situation, where we have to filter a matrix instead of a vector. However, filtering a matrix is typically more time consuming than filtering a vector. Furthermore, we assume a column-major storage scheme for matrices. If the storage scheme is row-major, then the trick is already at hand. For column-major storage schemes of matrices, the trick is to filter not the matrix, but its transpose. Of course, this implies that the whole equation (3.74) is transposed. Within the filter algorithm there are several formulas, which correspond to copying. This can be performed then all at once, because the data to be copied lies contiguously in the memory. In some programming languages there exist special functions for such copying, for example in the programming language c this is the function memcopy.

Filtering Speed

The purpose of this subsection is to emphasize the gain of calculation speed due to the block formulation of the filter equation and the accompanied better performance of the BLAS routines of the ATLAS library. The comparison begins with the number of arithmetic operations which are given in table 3.1. The copying operations are left out of consideration, because they are much faster than arithmetic operations and therefore do not change the overall calculation time significantly.

	straightforward implementation	block implementation
multiplications	$P+Q+1$	$\frac{L+1}{2}+P+Q$
additions	$P+Q$	$\frac{L-1}{2}+P+Q$
total	$2(P+Q)+1$	$L+2(P+Q)$

Table 3.1: Comparison of the number of arithmetic operations of the straightforward implementation according to equation (2.53) and the block implementation according to algorithm 3.2.

According to table 3.1 the block implementation requires more arithmetic operations than the straightforward implementation. In fact, the number of arithmetic operations increases linearly with respect to the block size. However, the block implementation is much faster than the straightforward implementation, if we choose the block size well. The reason for this is, that the multiplications are carried out by the BLAS routines of the ATLAS library. These routines perform better for bigger matrices, because then the risk of cache-misses is lower than for smaller matrices. Nevertheless, this effect is countervailed by the fact that the number of arithmetic operations increases linearly with respect to the block size. Thus there is an optimum block size for which the block implementation works fastest. The best way to determine this optimum block size is to measure the calculation time for different block sizes, i.e. empirically. Figure 3.7 depicts the calculation time for different block sizes for two of the filters, which were used for the test data in chapter 6.

Quantization Noise

One property of the block implementation is that the roundoff noise of the filtering may increase linearly with respect to the block size. This is caused by the magnitude of the elements of matrix $\boldsymbol{F}^{(y)}$. If the elements of $\boldsymbol{F}^{(y)}$ increase e.g. by a factor 100, then the roundoff noise of the corresponding matrix multiplication increases also by a factor of 100. In the following, we show one case for which this effect arises.

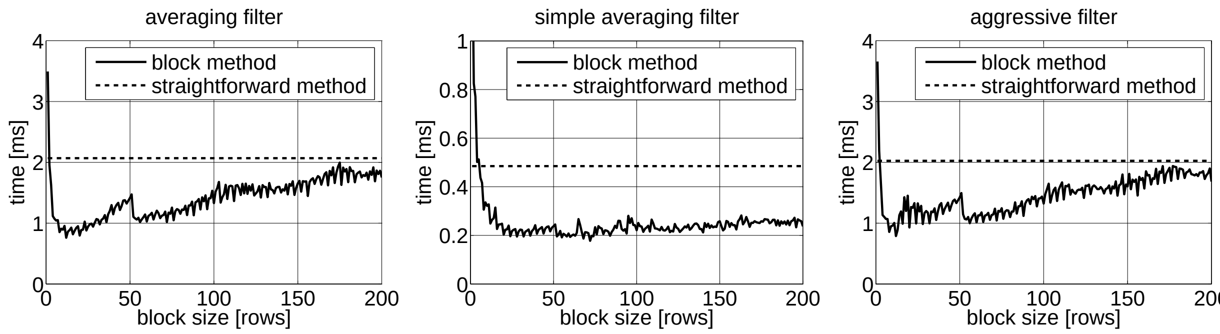


Figure 3.7: Comparison of the filtering speed of the ARMA block method and the straightforward implementation. The used filters are described in section 6.4. The figures show the calculation time of filtering one row of a matrix with 1000 columns. The calculation time for the straightforward method has been measured under ideal conditions, which means that no copy operations are included in the calculation time. Therefore, the dotted line is a theoretical value, which cannot be reached by any variant of a straightforward implementation in practice. Thus, the speed-up factors presented in the following are lower than in practice. If the time for copying operations would be included, then the straightforward implementation is equivalent to the block method with block size $L = 1$. The measured calculation time for the block method does include all copy operations. The optimum block size for the averaging filter is $L = 12$, for which the filtering speed is increased by a factor of 2.7. The optimum block size for the simple averaging filter is $L = 72$, for which the filtering speed is increased by a factor of 2.7. Finally, the optimum block size for the aggressive filter is $L = 12$, for which the filtering speed is increased by a factor of 2.6. If we choose a block size close to the optimum block size, the calculation time would not change much. However, above certain block size the increase of calculation time is approximately linear. It should be mentioned that this figure concerns only the calculation time for filtering. In practice, there may be also other influencing factors for the overall calculation time of some application.

Let us assume the following values for the filter coefficients.

$$a_1 = 2, \quad a_2 = -1 \tag{3.82}$$

According to algorithm (3.1) we then find for the first row of $\boldsymbol{F}^{(y)}$

$$\boldsymbol{F}^{(y)}_{1,:} = \begin{bmatrix} -1 & 2 \end{bmatrix}. \tag{3.83}$$

Computing matrix $\boldsymbol{F}^{(y)}$ for e.g. the block size $L = 4$ yields

$$\boldsymbol{F}^{(y)} = \begin{bmatrix} -1 & 2 \\ -2 & 3 \\ -3 & 4 \\ -4 & 5 \end{bmatrix}. \tag{3.84}$$

This leads us to the assumption, that

$$\boldsymbol{F}^{(y)}_{k,:} = \begin{bmatrix} -k & k+1 \end{bmatrix}. \tag{3.85}$$

We can show that this equation is true by induction. The basis for the induction is that

$$\boldsymbol{F}^{(y)}_{1,:} = \begin{bmatrix} a_2 & a_1 \end{bmatrix} = \begin{bmatrix} -1 & 2 \end{bmatrix} \tag{3.86}$$

holds according to algorithm (3.1). Now we perform the inductive step by showing that if $\boldsymbol{F}^{(y)}_{k,:} = \begin{bmatrix} -k & k+1 \end{bmatrix}$ holds, then $\boldsymbol{F}^{(y)}_{k+1,:} = \begin{bmatrix} -(k+1) & k+2 \end{bmatrix}$ is true, too. By formula (3.67) we find

$$\boldsymbol{F}^{(y)}_{k+1,:} = \begin{bmatrix} 0 & -k \end{bmatrix} + (k+1) \begin{bmatrix} a_2 & a_1 \end{bmatrix} = \begin{bmatrix} 0 & -k \end{bmatrix} + (k+1) \begin{bmatrix} -1 & 2 \end{bmatrix} = \begin{bmatrix} -(k+1) & k+2 \end{bmatrix}, \tag{3.87}$$

which proves our assumption. Thus, we find

$$\boldsymbol{F}^{(y)} = \begin{bmatrix} -1 & 2 \\ -2 & 3 \\ \vdots & \vdots \\ -L & L+1 \end{bmatrix}, \tag{3.88}$$

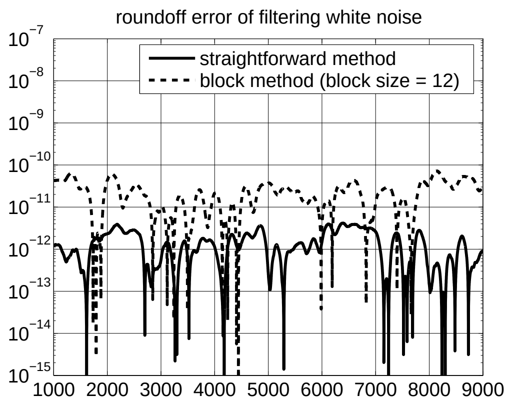
if $a_1 = 2$ and $a_2 = -1$. However, for $a_1 = 2$ and $a_2 = -1$ the corresponding pair of poles is $\alpha_1 = \alpha_2 = 1$. As these poles lay on the unit circle in the z-plane, the corresponding filter is unstable. Therefore, equation (3.88) will not occur in practical cases. But if, $a_1 \approx 2$ and $a_2 \approx -1$, then $\boldsymbol{F}^{(y)}$, then we find

$$\boldsymbol{F}^{(y)} \approx \begin{bmatrix} -1 & 2 \\ -2 & 3 \\ \vdots & \vdots \\ -L & L+1 \end{bmatrix}. \tag{3.89}$$

The question arises, which constellation of a complex conjugate pair of poles leads to the coefficients $a_1 \approx 2$ and $a_2 \approx -1$. From equation (2.102) we find

$$\begin{aligned}(1-\alpha z^{-1})(1-\alpha^* z^{-1}) &= 1-(\alpha+\alpha^*)z^{-1}+\alpha\alpha^* z^{-2} \\ &= 1-2\alpha z^{-1}+|\alpha|^2 z^{-2} \\ &= 1-2r\cos\phi z^{-1}+r^2 z^{-2},\end{aligned} \tag{3.90}$$

wherein $\alpha = re^{i\phi}$ is the polar representation of the the pole. Thus, the effect arises if $r \approx 1$ and $\phi \approx 0$. This is the case for so-called notch filters which are described in chapter 5. Figure 3.8 depicts the roundoff noise for one of the filters which was used for test data in chapter 6.



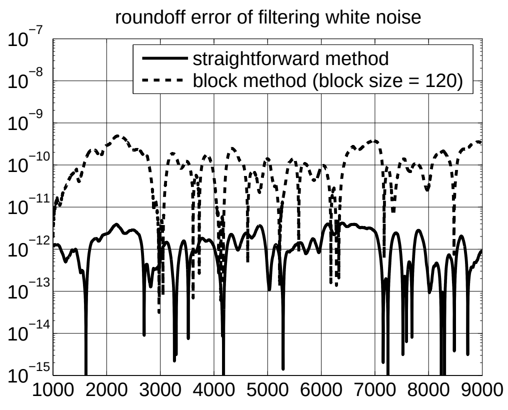


Figure 3.8: Connection of the magnitude of roundoff noise and the block size. A white noise time series has been filtered with the aggressive filter described in section 6.4 by three different methods: The first method uses a precision of 40 digits, which is superior to double precision, corresponding approximately to 15 digits. The second method is the straightforward implementation of ARMA filters, which is the reference for the comparison. The third method is the block method for ARMA filters. The comparison is based on the difference of the latter two methods to the first method. The filtered time series is normalized such that its standard deviation is equal to one. Therefore this figure directly indicates the numerical precision of the filtering. The left panel shows the roundoff noise of the straightforward implementation, which has a standard deviation equal to $2.0 \cdot 10^{-12}$, and the roundoff noise of the block method for the block size $L = 12$, which has a standard deviation equal to $4.0 \cdot 10^{-11}$. The right panel shows also the roundoff noise of the straightforward implementation and the roundoff noise of the block method for the block size $L = 120$, which has a standard deviation equal to $2.8 \cdot 10^{-10}$. Apparently, the roundoff noise increases linearly with the block size.

Summary of ARMA block method

As the derivation of the ARMA block method is not as short as the derivations of the other methods, a short summary is presented here.

Formation of Filter Sections The reason for forming filter sections is the quantization noise of the filter coefficients. The quantization noise becomes large, if series of poles and zeros are close to each other and close to the unit circle in the z-plane. In this case the three rules of thumb given in Dehner (2003) should be applied.

Block Formulation The difficulty of deriving a block formulation for ARMA filters is caused by the recursion within equation (2.53). The recursion can be overcome by means of a state-space formulation of the filter equation. This formulation also leads to very fast algorithms for computing the filter matrices. The algorithm for filtering is based on matrix multiplications, which makes it very fast if optimized numerical libraries such as the BLAS library are used.

Numerical Aspects The optimum block size of the filter algorithm should be determined empirically. This means that we should filter a dummy matrix using different block sizes and measure the time for each block size. It should also be mentioned, that increasing the block size results in increased roundoff noise.

Overlap-Add Method (Frequency Domain Block Implementation)

The overlap-add method is based on a block-wise convolution of a time series x_n by the impulse response h_n of a filter. The length of the impulse response has to be considerably smaller than the length of the time series. For ARMA filters the impulse response is in principle of infinite length. Strictly speaking, we are thus not allowed to apply the overlap-add method for ARMA filters. Nevertheless, the filters we consider are stable (cf. section 2.4). This means, that their impulse response h_n decays to zero for $n \to \infty$ according to equation (2.104) which is repeated here for convenience.

$$\lim_{n\to\infty} h_n = 0 \quad \text{for stable filters} \tag{3.91}$$

Therefore, there exists a J for which $h_n < \varepsilon$ for all $n > J$ holds, whereas ε represents some small positive number. If we choose ε to represent numerical zero and if then J is considerably smaller than the length of the time series, we still may apply the overlap-add method for ARMA filters. We do this by treating all values h_n for $n > J$ as zeros. The error in the filtering, which we then introduce, will be of the same magnitude as the roundoff noise and thus negligible.

For the application of the overlap-add method for ARMA filters, we first have to determine J. How we can do this, is described in detail in section 4.1, where it is more appropriate. For now we accept, that there is a way to compute this value. After that, we compute the values

$$h_0,\, h_1,\, \ldots,\, h_J \tag{3.92}$$

of the impulse response h_n by equation (2.56). Using these values to represent the impulse response of the ARMA filter can be interpreted as approximating the ARMA filter by an MA filter. Thus, it is not very remarkable, that the whole rest of the computations of the overlap-add method for ARMA filters is exactly the same as for the overlap-add method for MA filters. The only difference is, that we have to replace P by J in the formulas (3.5) to (3.13). Therefore, we do not need to repeat these formulas here.

FFT Method (Frequency Domain Implementation)

The FFT method for ARMA filters is based on equation (2.57), which is repeated here for convenience.

$$y_n = \sum_{k=-\infty}^{\infty} h_k x_{n-k} = h_n * x_n \tag{3.93}$$

It states that the filtering of a time series x_n by an ARMA filter corresponds to a convolution of that time series by the filter's impulse response h_n. In practice, we only know the values x_0, x_1, ..., x_{N-1} and we set all others values of x_n to zero (cf. SCHUH 1996, pp. 36,37). For this case, formula (3.93) is equivalent to

$$y_n = \sum_{k=0}^{n} h_k x_{n-k} \quad \text{for } n = 0, 1, \ldots, N-1 \tag{3.94}$$

Though the impulse response of an ARMA filter is in principle of infinite length, we therefore only need the first N values of the impulse response h_n to correctly apply formula (3.93). After defining

$$\boldsymbol{h} = \begin{bmatrix} h_0 & \cdots & h_{N-1} & 0 & \cdots & 0 \end{bmatrix}^\top \tag{3.95}$$

and

$$\boldsymbol{x} = \begin{bmatrix} x_0 & \cdots & x_{N-1} & 0 & \cdots & 0 \end{bmatrix}^\top, \tag{3.96}$$

whereas both vectors are padded by zeros to length $N_{\text{FFT}} = 2^{\text{ceil}\,(\log_2(2N-1))}$, we can compute equation (3.94) by an element-wise multiplication in the frequency domain (cf. section 2.4).

$$\boldsymbol{y} = \mathcal{F}^{-1}\{\mathcal{F}\{\boldsymbol{h}\} \circ \mathcal{F}\{\boldsymbol{x}\}\}. \tag{3.97}$$

To choose N_{FFT} equal to 2^m, whereas m is an integer value, is justified by the fact that FFT algorithms work faster for this choice. The values y_0, y_1, ..., y_{N-1} of the filtered time series y_n are contained in the first N elements of vector $\boldsymbol{y}$, i.e.

$$\boldsymbol{y}_{1:N} = \begin{bmatrix} y_0 & y_1 & \cdots & y_{N-1} \end{bmatrix}^\top. \tag{3.98}$$

Summary of Filter Implementations

At this point we have discussed the principle ways of implementing MA and ARMA filters. Of course, there exist many more ways of implementing digital filters in the literature. For example DEHNER (2003) introduces a state-space implementation, which is very stable with respect to roundoff noise. However, we have only need for the presented implementations as these are the fastest with respect to filtering speed. Table 3.2 gives an overview of the pros and cons of the filter implementations introduced in sections 3.1 and 3.2. The memory requirements within the table are only relevant for the design matrix, which has to be filtered at least once within an adjustment procedure, if the normal equation matrix cannot be approximated by means of analytical formulas.

Within table 3.2 the term short filter means that the order P and Q of the filter in equation (2.53) is low while the term long filter means that the order P and Q of the filter in equation (2.53) is high. For example, $P = Q = 2$ is a low filter order, whereas $P = Q = 10000$ is a high filter order. Note, that there is no exact limit for low and high filter orders.

Table 3.2 indicates that the time and frequency domain block implementations are superior to the other implementations with respect to the filtering speed. If the time and frequency domain block implementations are compared with each other, then the time domain block implementation is faster, if the filter is short. Consequently, if the filter is long, then the frequency domain block implementation is faster than the time domain block implementation. As the filter order depends in the end on the correlations of the measurement

implementation	MA filter	ARMA filter
time domain straightforward	+ lowest memory requirements − slower than block method	+ lowest memory requirements − slower than block method
time domain block	+ low memory requirements + fast for short filters	+ low memory requirements + fast for short filters
frequency domain block	− high memory requirements + fast for longer filters	− high memory requirements + fast for longer filters − truncated impulse response
frequency domain	− highest memory requirements − slower than block method	− highest memory requirements − slower than block method

Table 3.2: Comparison of the pros and cons of the introduced filter implementations.

errors, we only have to choose between MA and ARMA filters. Because of the recursion within the filter equation ARMA filters have lower filter orders than MA filters with the same decorrelating capabilities. This results from the fact that ARMA filters can be converted into MA filters with almost the same decorrelating capabilities by approximating the impulse response of the ARMA filter, which is generally longer than the filter order (cf. section 2.4). For this reason the ARMA filters should be preferred for the time domain block implementation. For the frequency domain block implementation ARMA filters have the drawback that we have to truncate their impulse response. Therefore, it is better to directly use MA filters.

For these reasons we should recommend to use the time domain block implementation in combination with ARMA filters or to use the frequency domain block implementation in combination with MA filters. Which one should be preferred depends on the memory requirements and on the correlations of the measurement errors.

Implementing the filters is one problem, integrating the filtering into adjustment procedures is another. The first problem is now solved. Thus, we proceed with the next problem of integrating the filtering into adjustment procedures in the next section.

3.3 Integration of Filters in Adjustment Procedures

In section 3.1 and 3.2 we discussed different numerical methods for filtering by MA and ARMA filters. Now we discuss how to integrate these methods into least-squares adjustment procedures which are used to compute the solution (2.8) of the Gauss-Markov model depicted in section 2.1. There exist many different least-squares adjustment procedures and we cannot describe them all in detail. However, nearly all of these adjustment procedure make use of small set of operations, for example the computation of the normal equation matrix $\boldsymbol{A}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{A}$.

The filtering methods of section 3.1 and 3.2 can be regarded as tools for integrating the covariance matrix $\boldsymbol{\Sigma}$ in the computations. Their integration into least-squares adjustment procedure is typically affected by some basic conditions. One condition which often occurs in the problems we consider is that the design matrix is very large and does not fit in the working memory. Typically, we then have either a row-wise or column-wise access to the design matrix. Another condition is, that we need to integrate the filter methods either into the computation of the normal equation matrix or the computation of the gradient of the residual sum of squares in (2.13). This results in either applying the filtering methods to a matrix or to a vector.

In section (2.3), we have introduced three representations of the covariance matrix: Convolution, AR process and ARMA process approach. For each of the process approaches, we discussed three different methods of implementing the corresponding filter. Furthermore, the access to the design matrix may be either row-wise or column-wise. All these possibilities may occur in any combination. However, not any combination does make sense. Therefore, in section we order the possible combinations and highlight under which circumstances which combination is most useful.

Principle Ways of Integrating the Covariance Matrix

First of all, we analyze the principle ways of integrating the covariance matrix into the Gauss-Markov model described in section 2.1. In this context we will discuss the Toeplitz approach which allows only one way of integration. After that, we investigate two ways of integrating the AR and ARMA process approaches. The first one is based on the idea of decorrelating the observation equations while the second is similar to integration of the Toeplitz approach.

Toeplitz Approach

For the integration of the covariance matrix $\mathbf{\Sigma}$ of the observations into adjustment procedures via the Toeplitz approach, we discuss how to apply $\mathbf{\Sigma}^{-1}$ to a vector $\boldsymbol{y}$, i.e.

$$\boldsymbol{x} = \mathbf{\Sigma}^{-1}\boldsymbol{y}, \tag{3.99}$$

where $\boldsymbol{y}$ is known and $\boldsymbol{x}$ is unknown. Apparently, vector $\boldsymbol{y}$ follows from

$$\boldsymbol{y} = \mathbf{\Sigma}\boldsymbol{x}. \tag{3.100}$$

According to the conditions for the observations described in section 2.2, we can assume that $\mathbf{\Sigma}$ has a Toeplitz structure. In addition to that, the covariance matrix $\mathbf{\Sigma}$ is symmetric. Thus, the covariance matrix has the following structure

$$\mathbf{\Sigma} = \begin{bmatrix} \gamma_0 & \gamma_1 & \cdots & \gamma_{N-2} & \gamma_{N-1} \\ \gamma_1 & \gamma_0 & \gamma_1 & & \gamma_{N-2} \\ \vdots & \gamma_1 & \gamma_0 & \ddots & \vdots \\ \gamma_{N-2} & & \ddots & \ddots & \gamma_1 \\ \gamma_{N-1} & \gamma_{N-2} & \cdots & \gamma_1 & \gamma_0 \end{bmatrix}. \tag{3.101}$$

To efficiently solve the symmetric Toeplitz system in (3.100) is subject to the extended Levinson-Durbin algorithm (cf. Levinson 1947), which is shown in algorithm 3.3. The algorithm is also described in detail by Farhang-Boroujeny (1998), chapter 11. Its computational complexity is of order $\mathcal{O}(N^2)$ (cf. Schuh 1996, p. 127) and it needs only very few memory.

In section 2.2 we assumed that the residual time series v_n of the Gauss-Markov model is ergodic. For ergodic time series the autocovariance function is absolutely summable (cf. section 2.2), i.e.

$$\sum_{j=0}^{\infty} |\gamma_j| < \infty. \tag{3.102}$$

This implies that

$$\lim_{j\to\infty} \gamma_j = 0 \tag{3.103}$$

(cf. Forster 1983, p. 38, theorem 2 and 3). Thus, there exists a value J for which $|\gamma_j| < \varepsilon$ for all $j \geq J$, whereas ε is some small positive number. After choosing ε to represent numerical zero, and if then $J < N$, we can set all γ_j with $j \geq J$ to zero and approximate $\mathbf{\Sigma}$ by a band matrix. The approximation error is then of the magnitude of roundoff noise.

If the covariance matrix in (3.100) has a banded structure, we can take advantage of this by utilizing the algorithm proposed by Schuh (1996), pp. 125–132. The idea is to extend the covariance matrix such that the extended matrix is a circulant matrix. In this way, it is possible to exploit that the multiplication by a circulant matrix is equivalent to the circulant convolution, which can be computed very efficiently by means of FFT techniques. Let the vectors $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ and the matrix $\boldsymbol{C}$ be defined as in (2.115), i.e.

$$\boldsymbol{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{N-1} \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{N-1} \end{bmatrix}, \quad \boldsymbol{c} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{N-1} \end{bmatrix} \quad \text{and} \quad \boldsymbol{C} = \begin{bmatrix} c_0 & c_{N-1} & \cdots & c_2 \\ c_1 & c_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & c_{N-1} \\ c_{N-1} & \cdots & c_1 & c_0 \end{bmatrix}. \tag{3.104}$$

Then

$$\boldsymbol{b} = \boldsymbol{C}\boldsymbol{a} = \boldsymbol{c} \circledast \boldsymbol{a} = \mathcal{F}^{-1}\{\mathcal{F}\{\boldsymbol{c}\} \circ \mathcal{F}\{\boldsymbol{a}\}\} \tag{3.105}$$

is the circulant convolution of $\boldsymbol{c}$ and $\boldsymbol{a}$. Within the algorithm proposed by Schuh (1996) matrix $\boldsymbol{C}$, vector $\boldsymbol{a}$ and vector $\boldsymbol{b}$ are subdivided into blocks in the following manner.

$$\boldsymbol{a} = \begin{bmatrix} \boldsymbol{a}^{(1)} \\ \boldsymbol{a}^{(2)} \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} \boldsymbol{C}^{(11)} & \boldsymbol{C}^{(12)} \\ \boldsymbol{C}^{(21)} & \boldsymbol{C}^{(22)} \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}^{(1)} \\ \boldsymbol{b}^{(2)} \end{bmatrix} \tag{3.106}$$

As $\boldsymbol{C}$ is a circular matrix, we can compute the first line of its inverse matrix by

$$\boldsymbol{D}_{1,:} = \mathcal{F}^{-1}\{(\mathcal{F}\{\boldsymbol{C}_{1,:}\})^{-1}\}, \tag{3.107}$$

wherein $\boldsymbol{D}$ denotes the inverse matrix of $\boldsymbol{C}$, i.e. $\boldsymbol{D} = \boldsymbol{C}^{-1}$. Herein, the term $(\mathcal{F}\{\boldsymbol{C}_{1,:}\})^{-1}$ stands for computing the reciprocal values of $\mathcal{F}\{\boldsymbol{C}_{1,:}\}$ element-by-element. Because the inverse matrix of a circular matrix is again a circular matrix, equation (3.107) describes how to compute $\boldsymbol{C}^{-1}$. We subdivide matrix $\boldsymbol{D}$ in the same way as $\boldsymbol{C}$.

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{D}^{(11)} & \boldsymbol{D}^{(12)} \\ \boldsymbol{D}^{(21)} & \boldsymbol{D}^{(22)} \end{bmatrix} \tag{3.108}$$

Each block $\boldsymbol{C}_{ij}$ and $\boldsymbol{D}_{ij}$ has a Toeplitz structure. Furthermore, as $\boldsymbol{C}$ and $\boldsymbol{D}$ are both circular matrices, we find

$$\begin{bmatrix} \boldsymbol{C}^{(11)} & \boldsymbol{C}^{(12)} \\ \boldsymbol{C}^{(21)} & \boldsymbol{C}^{(22)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{C}^{(22)} & \boldsymbol{C}^{(21)} \\ \boldsymbol{C}^{(12)} & \boldsymbol{C}^{(11)} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \boldsymbol{D}^{(11)} & \boldsymbol{D}^{(12)} \\ \boldsymbol{D}^{(21)} & \boldsymbol{D}^{(22)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{D}^{(22)} & \boldsymbol{D}^{(21)} \\ \boldsymbol{D}^{(12)} & \boldsymbol{D}^{(11)} \end{bmatrix}. \tag{3.109}$$

From the matrix identity $\boldsymbol{C}\boldsymbol{D} = \boldsymbol{C}\boldsymbol{C}^{-1} = \boldsymbol{I}$ and (3.109) it follows that

$$(\boldsymbol{C}^{(11)})^{-1} = \boldsymbol{D}^{(11)} - \boldsymbol{D}^{(12)}(\boldsymbol{D}^{(22)})^{-1}\boldsymbol{D}^{(21)} \tag{3.110}$$

(cf. Koch 1997, pp. 36, 37). With this knowledge, we can solve (3.100). For this, we set

$$\boldsymbol{a}^{(1)} = \boldsymbol{y}, \quad \boldsymbol{a}^{(2)} = \boldsymbol{0} \quad \text{and} \quad \boldsymbol{C}^{(11)} = \boldsymbol{\Sigma}. \tag{3.111}$$

The other blocks of matrix $\boldsymbol{C}$ in (3.106) are filled up such that $\boldsymbol{C}$ is a circular matrix. For convenience, we specify the blocks in more detail.

$$\underset{N \times N}{\boldsymbol{C}^{(11)}} = \begin{bmatrix} \gamma_0 & \cdots & \gamma_{J-1} & 0 & \cdots & 0 \\ \vdots & \ddots & & \ddots & \ddots & \vdots \\ \gamma_{J-1} & & \gamma_0 & & \gamma_{J-1} & 0 \\ 0 & \gamma_{J-1} & & \gamma_0 & & \gamma_{J-1} \\ \vdots & \ddots & \ddots & & \ddots & \vdots \\ 0 & \cdots & 0 & \gamma_{J-1} & \cdots & \gamma_0 \end{bmatrix} \tag{3.112}$$

$$\underset{J \times N}{\boldsymbol{C}^{(21)}} = \begin{bmatrix} 0 & 0 & \cdots & 0 & \cdots & 0 & \gamma_{J-1} & \cdots & \gamma_1 \\ \gamma_{J-1} & 0 & \cdots & 0 & \cdots & 0 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots & & \vdots & \vdots & \ddots & \gamma_{J-1} \\ \gamma_1 & \cdots & \gamma_{J-1} & 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{3.113}$$

$$\underset{N\times J}{\boldsymbol{C}^{(12)}} = (\boldsymbol{C}^{(21)})^\top \tag{3.114}$$

$$\underset{J\times J}{\boldsymbol{C}^{(22)}} = \begin{bmatrix} \gamma_0 & \cdots & \gamma_{J-1} \\ \vdots & \ddots & \vdots \\ \gamma_{J-1} & \cdots & \gamma_0 \end{bmatrix} \tag{3.115}$$

Therefore, the vectors $\boldsymbol{a}^{(1)}$ and $\boldsymbol{b}^{(1)}$ are of length N while the vectors $\boldsymbol{a}^{(2)}$ and $\boldsymbol{b}^{(2)}$ are of length J. By equation (3.110) we find

$$\boldsymbol{x} = \boldsymbol{\Sigma}^{-1}\boldsymbol{y} = (\boldsymbol{C}^{(11)})^{-1}\boldsymbol{y} = \boldsymbol{D}^{(11)}\boldsymbol{y} - \boldsymbol{D}^{(12)}(\boldsymbol{D}^{(22)})^{-1}\boldsymbol{D}^{(21)}\boldsymbol{y}. \tag{3.116}$$

By solving (3.105) for $\boldsymbol{b}$, we obtain

$$\boldsymbol{b} = \boldsymbol{C}^{-1}\boldsymbol{a} = \boldsymbol{D}\boldsymbol{a}. \tag{3.117}$$

Using the block partitioning introduced in (3.106) and (3.108), we find

$$\begin{bmatrix} \boldsymbol{b}^{(1)} \\ \boldsymbol{b}^{(2)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{D}^{(11)}\boldsymbol{a}^{(1)} + \boldsymbol{D}^{(12)}\boldsymbol{a}^{(2)} \\ \boldsymbol{D}^{(21)}\boldsymbol{a}^{(1)} + \boldsymbol{D}^{(22)}\boldsymbol{a}^{(2)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{D}^{(11)}\boldsymbol{y} \\ \boldsymbol{D}^{(21)}\boldsymbol{y} \end{bmatrix}, \tag{3.118}$$

whereas also (3.111) was used. Inserting (3.118) in (3.116) yields

$$\boldsymbol{x} = \boldsymbol{b}^{(1)} - \boldsymbol{D}^{(12)}(\boldsymbol{D}^{(22)})^{-1}\boldsymbol{b}^{(2)}. \tag{3.119}$$

Now, we indicate how to compute vector $\boldsymbol{b}$ efficiently. For this we define

$$\boldsymbol{d} = \begin{bmatrix} \boldsymbol{D}_{1,1} & \cdots & \boldsymbol{D}_{N+J,1} & 0 & \cdots & 0 & \boldsymbol{D}_{1,N} & \boldsymbol{D}_{1,N-1} & \cdots & \boldsymbol{D}_{1,2} \end{bmatrix}^\top \tag{3.120}$$

and

$$\boldsymbol{e} = \begin{bmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_N & 0 & \cdots & 0 \end{bmatrix}^\top. \tag{3.121}$$

In both vectors we have inserted zeros such that their length is equal to $N_{\text{FFT}} = 2^{\text{ceil}\,(\log_2(3N+J-2))}$. The product of the circular matrix $\boldsymbol{D}$ and the vector $\boldsymbol{a}$ can be computed by the convolution

$$\boldsymbol{f} = \boldsymbol{d} * \boldsymbol{e} = \mathcal{F}^{-1}\{\mathcal{F}\{\boldsymbol{d}\} \circ \mathcal{F}\{\boldsymbol{e}\}\}. \tag{3.122}$$

Vector $\boldsymbol{b}$ is contained in the first $N+J$ elements of vector $\boldsymbol{f}$.

$$\boldsymbol{f}_{1:N+J} = \boldsymbol{D}\boldsymbol{a} = \boldsymbol{b}. \tag{3.123}$$

In the next step, we set

$$\boldsymbol{h} = (\boldsymbol{D}^{(22)})^{-1}\boldsymbol{b}^{(2)}. \tag{3.124}$$

Because vector $\boldsymbol{h}$ is the solution of $\boldsymbol{D}^{(22)}\boldsymbol{h} = \boldsymbol{b}^{(2)}$ and $\boldsymbol{D}^{(22)}$ is a symmetric Toeplitz matrix, we can use algorithm 3.3 to solve

$$\boldsymbol{D}^{(22)}\boldsymbol{h} = \boldsymbol{b}^{(2)} \tag{3.125}$$

for $\boldsymbol{h}$. Now that we know how to compute $\boldsymbol{h}$, we substitute (3.124) in (3.119) and find

$$\boldsymbol{x} = \boldsymbol{b}^{(1)} - \boldsymbol{D}^{(12)}\boldsymbol{h}. \tag{3.126}$$

As we also know how to compute vector $\boldsymbol{b}^{(1)}$, we only have to look at the product $\boldsymbol{D}^{(12)}\boldsymbol{h}$. Because $\boldsymbol{D}^{(12)}$ is Toeplitz matrix, it corresponds to a convolution. We redefine the vectors $\boldsymbol{d}$ and $\boldsymbol{e}$ to be

$$\begin{aligned} \boldsymbol{d} &= \begin{bmatrix} \boldsymbol{D}^{(12)}_{1,1} & \cdots & \boldsymbol{D}^{(12)}_{N,1} & 0 & \cdots & 0 & \boldsymbol{D}^{(12)}_{1,J} & \boldsymbol{D}^{(12)}_{1,J-1} & \cdots & \boldsymbol{D}^{(12)}_{1,2} \end{bmatrix}^\top \\ &= \begin{bmatrix} \boldsymbol{D}_{1,N+1} & \cdots & \boldsymbol{D}_{N,N+1} & 0 & \cdots & 0 & \boldsymbol{D}_{1,N+J} & \boldsymbol{D}_{1,N+J-1} & \cdots & \boldsymbol{D}_{1,N+2} \end{bmatrix}^\top \end{aligned} \tag{3.127}$$

and

$$\boldsymbol{e} = \begin{bmatrix}\boldsymbol{h}^\top & 0 & \cdots & 0\end{bmatrix}^\top, \tag{3.128}$$

whereas in both vector we insert zeros such that their length is equal to $N_{\text{FFT}} = 2^{\operatorname{ceil}(\log_2(2N-2+J))}$. Then we can perform the convolution as an element-wise multiplication in the frequency domain.

$$\boldsymbol{f} = \boldsymbol{d} * \boldsymbol{e} = \mathcal{F}^{-1}\{\mathcal{F}\{\boldsymbol{d}\} \circ \mathcal{F}\{\boldsymbol{e}\}\}. \tag{3.129}$$

The result of the convolution is contained in the first N elements.

$$\boldsymbol{f}_{1:N} = \boldsymbol{D}^{(12)}\boldsymbol{h}. \tag{3.130}$$

Finally, we can evaluate equation (3.126) and thus compute $\boldsymbol{x} = \boldsymbol{\Sigma}^{-1}\boldsymbol{y}$. The efficiency of the algorithm depends on the bandwidth J of the covariance matrix $\boldsymbol{\Sigma}$. If it is considerably smaller than N, then the computational costs of the algorithm are also considerably smaller than the computational costs of algorithm 3.3.

AR and ARMA Process Approaches

In this subsection we investigate the consequences of modeling the covariance matrix $\boldsymbol{\Sigma}$ in (2.8) by MA or ARMA filters. This investigation can be regarded as a preliminary stage to the actual integration of the MA or ARMA filters into least-squares adjustment procedure. We will see that there are two principle ways of interpreting the filtering. The first one states, that the filters are directly used to multiply the inverse covariance matrix $\boldsymbol{\Sigma}^{-1}$ by the observation vector $\boldsymbol{l}$ and by the design matrix $\boldsymbol{A}$. The second way of interpreting the filtering is that the observation equations are filtered in order to decorrelate the observations. Thus, we call the first way the direct approach and the second way decorrelation approach. It should be emphasized, that these two ways differ only in their interpretation. The least-squares estimation of the parameters in (2.8) remains unaffected.

We start with the decorrelation approach. Here, the entire observation equations are filtered. From (2.40) and (2.41) we know, that the filtering corresponds to a multiplication by matrix $\boldsymbol{H}$.

$$\boldsymbol{H}\boldsymbol{l} + \boldsymbol{H}\boldsymbol{v} = \boldsymbol{H}\boldsymbol{A}\boldsymbol{x} \tag{3.131}$$

We design the filter to ensure that

$$\boldsymbol{\Sigma}\{\boldsymbol{H}\boldsymbol{\mathcal{L}}\} = \boldsymbol{H}\boldsymbol{\Sigma}\boldsymbol{H}^\top \approx \boldsymbol{I} \tag{3.132}$$

holds. As the observations time series l_n begins with the $n = 0$, the filtering also starts at $n = 0$. According to equation (2.53) we need for $n = 0$ observations l_{-1}, l_{-2}, ..., which are not given. These observations are commonly set to zero (cf. Schuh 1996, pp. 36,37). As we manipulated these observations by setting them to zero, we introduce an error at the beginning of the filtering. Because of the recursion of the filter equation (2.53) this effect does not vanish after $n = \max(P, Q)$. Nevertheless, the effect will decrease as the index n increases, if the filter is stable. The index for which the effect decreases sufficiently to be neglected is $n = R$. How we can determine R, is discussed in detail in section 4.1. For the moment we accept that the filtered observations for $n = 0, \ldots, R-1$ should not be used within the least-squares adjustment procedure. The same problem arises not only for the observations but also for the filtering of the design matrix $\boldsymbol{A}$. That we should not use the first R observations within the least-squares adjustment procedures can also be seen from their covariance matrix $\boldsymbol{\Sigma}\{\boldsymbol{H}\boldsymbol{\mathcal{L}}\}$. Only the lower right part is equal to the identity matrix. The other part differ from the identity matrix which we denote by an asterisk $*$ for simplicity.

$$\boldsymbol{\Sigma}\{\boldsymbol{H}\boldsymbol{\mathcal{L}}\} = \begin{bmatrix} * & * \\ * & \boldsymbol{I}\end{bmatrix} \tag{3.133}$$

Note, that this is the reason for using the approximation sign instead of the equality sign in (3.132). A common practice is to leave the first R observations in equation (3.131) out after filtering (cf. SCHUH 1996, pp. 36, 37). This can be achieved, if we define the windowing matrix

$$\boldsymbol{S} = \begin{bmatrix} \underset{R\times R}{\boldsymbol{0}} & \underset{R\times N-R}{\boldsymbol{0}} \\ \underset{N-R\times R}{\boldsymbol{0}} & \underset{N-R\times N-R}{\boldsymbol{I}} \end{bmatrix} \tag{3.134}$$

and multiply the filtered observation equations (3.131) by this matrix.

$$\boldsymbol{SHl} + \boldsymbol{SHv} = \boldsymbol{SHAx} \tag{3.135}$$

By applying variance propagation (cf. KOCH 1997, p. 108) we then obtain

$$\boldsymbol{\Sigma}\{\boldsymbol{SH\mathcal{L}}\} = \boldsymbol{SH\Sigma H}^\top\boldsymbol{S}^\top = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{H}_{R+1:N,:}\boldsymbol{\Sigma H}^\top_{R+1:N,:} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}. \tag{3.136}$$

Thus, leaving the first R filtered observations out corresponds to removing those parts of the covariance matrix of the filtered observations in (3.133), which differ from the structure of an identity matrix. Note, that we could start the filtering at $n = R$ instead of $n = 0$. However, this is only true for MA filters. For ARMA filters, this is not possible because of the recursion in equation (2.53).

For the windowed and filtered observation equations (3.135) the least-squares estimation (2.8) reads

$$\widetilde{\boldsymbol{x}} = ((\boldsymbol{SHA})^\top(\boldsymbol{SHA}))^{-1}(\boldsymbol{SHA})^\top(\boldsymbol{SHl}). \tag{3.137}$$

Because the windowing matrix $\boldsymbol{S}$ has the property, that

$$\boldsymbol{S}^\top\boldsymbol{S} = \boldsymbol{S}, \tag{3.138}$$

we can simplify (3.137) to

$$\widetilde{\boldsymbol{x}} = (\boldsymbol{A}^\top\boldsymbol{H}^\top\boldsymbol{S}^\top\boldsymbol{SHA})^{-1}\boldsymbol{A}^\top\boldsymbol{H}^\top\boldsymbol{S}^\top\boldsymbol{SHl} = (\boldsymbol{A}^\top\boldsymbol{H}^\top\boldsymbol{SHA})^{-1}\boldsymbol{A}^\top\boldsymbol{H}^\top\boldsymbol{SHl}. \tag{3.139}$$

According to equation (2.30) matrix $\boldsymbol{H}$ is a Toeplitz matrix. Therefore, it holds that

$$\boldsymbol{H}^\top = \boldsymbol{PHP}, \tag{3.140}$$

where matrix $\boldsymbol{P}$ is permutation matrix, which turns a matrix or vector upside down, i.e.

$$\boldsymbol{P} = \begin{bmatrix} 0 & \cdots & 0 & 1 \\ \vdots & \iddots & 1 & 0 \\ 0 & \iddots & \iddots & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix}. \tag{3.141}$$

By substituting (3.140) into (3.139), we obtain

$$\widetilde{\boldsymbol{x}} = (\boldsymbol{A}^\top\boldsymbol{PHPSHA})^{-1}\boldsymbol{A}^\top\boldsymbol{PHPSHl}. \tag{3.142}$$

We can interpret the matrix product $\boldsymbol{PHPSH}$ as series of instructions which are performed on the design matrix $\boldsymbol{A}$ and the observation vector $\boldsymbol{l}$. The instructions are evaluated from the right to the left. For example $\boldsymbol{PHPSHl}$ means:

1. Filter the observation vector: $\boldsymbol{Hl}$.
2. Apply the window to the vector: $\boldsymbol{SHl}$.
3. Flip the vector: $\boldsymbol{PSHl}$.

4. Filter the vector: $\boldsymbol{HPSHl}$.

5. Flip the vector: $\boldsymbol{PHPSHl}$.

Before and after the filtering in step 4, the vector is turned upside down. Therefore, we may interpret the last three steps as backward filtering while the first step corresponds to forward filtering.

If we model the covariance matrix $\boldsymbol{\Sigma}$ by MA or ARMA filters, we can compute the least-squares estimation (2.8) either by (3.142) or by (3.137). We have obtained equation (3.137) by decorrelating the observation equations. Thus, we call equation (3.137) the decorrelation approach. Equation (3.142) is based on a series of instructions which all together correspond to applying the covariance matrix $\boldsymbol{\Sigma}$ to a matrix or vector. Thus, we call equation (3.142) the direct approach.

Typical Operations within Adjustment Procedures

There exist many different adjustment procedures to compute the least-squares estimates $\widetilde{\boldsymbol{x}}$ in equation (2.8). As we are concerned about the integration of the filtering into these procedures, we do not need to describe the procedures itself. Formula (3.139) indicates, that we can replace the inverse covariance matrix $\boldsymbol{\Sigma}^{-1}$ by $\boldsymbol{H}^\top \boldsymbol{SH}$, whereas the multiplication by $\boldsymbol{H}$ corresponds to filtering and the multiplication by $\boldsymbol{S}$ corresponds to a windowing operation. Therefore, it is sufficient to look at the typical operations within the procedures which incorporate the covariance matrix $\boldsymbol{\Sigma}$. One of these operations is the computation of the normal equation matrix

$$\boldsymbol{N} = \boldsymbol{A}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{A}, \quad \boldsymbol{n} = \boldsymbol{A}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{l}. \tag{3.143}$$

The computation of the normal equation matrix is often required in direct procedures. For example, we could compute the normal equation, perform a Cholesky decomposition of the normal equation matrix and solve the normal equations by forward and backward substitution (cf. for example Plank 2004, pp. 40–42). In iterative procedures, the computation of the normal equations is often partly performed in order to compute a suitable preconditioner (cf. for example Boxhammer 2006, pp. 60, 61 or Schuh 1996, pp. 82–85).

Instead of computing the normal equation matrix, some procedures make use of the QR decomposition of the design matrix. For these procedures, the observation equations are homogenized before the QR decomposition is performed. This can be achieved by the decorrelation approach in equation (3.131). The QR decomposition then reads

$$\boldsymbol{QR} = \boldsymbol{HA}, \tag{3.144}$$

wherein $\boldsymbol{Q}$ is an orthonormal matrix, $\boldsymbol{R}$ is an upper triangular matrix, $\boldsymbol{H}$ symbolizes the filtering and $\boldsymbol{A}$ is the unfiltered design matrix. However, for the problems we concern the number of observations and parameters is typically large. The fact, that the QR decomposition of a fully populated design matrix is in this case a computationally very demanding task, often rules out the application of this method.

Iterative adjustment procedures often require the computation of the gradient of the square sum of residuals in some form, i.e.

$$\boldsymbol{A}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{Ax} \quad \text{or} \quad \boldsymbol{A}^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{Ax} - \boldsymbol{l}). \tag{3.145}$$

As these operations are performed within the iterations, the filters have to be applied more than once. Though the filtering within the iterations can be applied to a vector which is computationally inexpensive, iterative methods typically require a suitable preconditioner. For this reason, even for iterative methods, the filtering has to be applied to at least once to a matrix, if the normal equation matrix cannot be approximated by means of analytical formulas.

Access to the Design Matrix

If the design matrix $\boldsymbol{A}$ is too big to fit in the working memory, the design matrix is typically subdivided horizontally or vertically into blocks. Which subdivision is favored, depends on the application. If the design matrix nonetheless fits in the memory all filter methods are applicable and no special treatment is necessary.

If the design matrix is vertically subdivided into blocks, we have a column-wise access to the design matrix. In this case all filter methods are applicable. The reason for this is, that the filtering corresponds to a multiplication by matrix $\boldsymbol{H}$ from the left side. We simply have to perform this multiplication for each vertical block separately. If we want to apply $\boldsymbol{\Sigma}^{-1}$, then this corresponds to solving a system of equations, whereas the vertical blocks of the design matrix correspond to multiple right hand sides. Thus, the subdivision of the design matrix into vertical blocks poses no problem.

If the design matrix is subdivided into horizontally blocks, we have a row-wise access to the design matrix. In this case, not every method is applicable. For example, if we want to apply $\boldsymbol{\Sigma}^{-1}$, which corresponds to solving a system of equations, the right hand sides of the equations are not simultaneously accessible in the working memory. In this case the application of algorithm 3.3 to this problem is still possible, but becomes computationally very demanding, as the design matrix needs to be set up many times for one computation of $\boldsymbol{\Sigma}^{-1}\boldsymbol{A}$. Another example is the overlap-add method. Its application is only reasonable, if the number of rows in each horizontal block is not too low.

However, there is one exception. Iterative adjustment procedures often require the computation of (3.145). Then, the filtering can be applied to the vector $\boldsymbol{Ax}$ or $\boldsymbol{Ax} - \boldsymbol{l}$ instead of the design matrix $\boldsymbol{A}$. As vectors do not require much memory, we assume that we are able to have these vectors in the working memory. We can compute for each block of the design matrix the product of the block by the parameter vector and then append the result to the previous results. When the whole vector $\boldsymbol{Ax}$ or $\boldsymbol{Ax} - \boldsymbol{l}$ is computed, all filter methods are applicable again. Table 3.3 gives an overview of the applicability of the filter methods for the different access types to the design matrix for the case that the design matrix has to be filtered. If the process approaches are favored, then table 3.2 should be conferred for recommendations on the choice of the filter method.

approach	row-wise access	column-wise access
convolution		extended Levinson-Durbin algorithm
		Schuh's algorithm for banded covariance matrices
AR process	block method	block method
	overlap-add method	overlap-add method
		FFT method
ARMA process	block method	block method
	overlap-add method	overlap-add method
		FFT method

Table 3.3: Overview of the applicable methods to compute either $\boldsymbol{A}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{A}$ or $\boldsymbol{A}^\top\boldsymbol{H}^\top\boldsymbol{SHA}$.

Synchronous Forward and Backward Filtering

If the design matrix is too big to fit into the working memory, we typically have either row-wise or column-wise access to the design matrix. Furthermore, we can use the direct approach in (3.142) or the decorrelation approach in (3.137). If we use the direct approach in combination with column-wise access to the design matrix, we can filter each column forward and backward before any further processing is done. However, if we use the direct approach in combination with row-wise access to the design matrix, we may not be able to perform the forward and backward filtering at the same time. This means that we have to compute the design matrix twice instead of once. The reason for this is that the number of rows of the design matrix in the working memory is restricted by the memory space.

In this subsection we discuss the case that we use the direct approach in combination with row-wise access to the design matrix, whereas the number of rows of the design matrix in the working memory is sufficient for synchronous forward and backward filtering. Then, for synchronous forward and backward filtering we can use the block method or the overlap-add method. We show how these methods are applied in the case of synchronous forward and backward filtering for MA filters. Due to the recursion within equation (2.53), the extension to ARMA filters is not possible. However, in most cases we can be approximate the ARMA filter by an MA filter. We can utilize this MA filter for synchronous forward and backward filtering.

Synchronous Block Method

In equation (3.142) we have to apply $\boldsymbol{H}^\top \boldsymbol{S}\boldsymbol{H} = \boldsymbol{P}\boldsymbol{H}\boldsymbol{P}\boldsymbol{S}\boldsymbol{H}$ to the design matrix $\boldsymbol{A}$. In order to describe the synchronous block method more general, we assume that we have to compute

$$\boldsymbol{z} = \boldsymbol{H}^\top \boldsymbol{S}\boldsymbol{H}\boldsymbol{x}. \tag{3.146}$$

Matrix $\boldsymbol{H}$ contains the filter coefficients according to equation (2.30) and matrix $\boldsymbol{S}$ is a windowing matrix according to equation (3.134). Within the computation the forward filtering

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} \tag{3.147}$$

is an intermediate result. We define

$$\boldsymbol{x} = \begin{bmatrix} x_0 & \cdots & x_{N-1} \end{bmatrix}^\top, \quad \boldsymbol{y} = \begin{bmatrix} y_0 & \cdots & y_{N-1} \end{bmatrix}^\top \quad \text{and} \quad \boldsymbol{z} = \begin{bmatrix} z_0 & \cdots & z_{N-1} \end{bmatrix}^\top. \tag{3.148}$$

Furthermore, we define $\boldsymbol{s}$ to be the diagonal of matrix $\boldsymbol{S}$.

$$\boldsymbol{s} = \operatorname{diag}(\boldsymbol{S}) = \begin{bmatrix} s_0 & \cdots & s_{N-1} \end{bmatrix}^\top \tag{3.149}$$

Each element y_n of the intermediate step $\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x}$ is obtained by the filter equation (2.59) for MA filters which is repeated here for convenience.

$$y_n = \sum_{k=0}^{P} h_k x_{n-k} \tag{3.150}$$

Analogously, vector $\boldsymbol{z}$ is obtained by $\boldsymbol{z} = \boldsymbol{H}^\top \boldsymbol{S}\boldsymbol{y} = \boldsymbol{H}^\top(\boldsymbol{s} \circ \boldsymbol{y})$, which corresponds to

$$z_n = \sum_{k=0}^{P} h_k s_{n+k} y_{n+k}, \tag{3.151}$$

As before, the values x_n are only known for $n = 0, 1, \ldots, N-1$. Therefore, we set all x_n, y_n and z_n with $n < 0$ or $n > N-1$ to zero. With these formulas, we can now construct the synchronous block method. For this, we define the filter matrices

$$\boldsymbol{F}^{(1)} = \begin{bmatrix} h_P & h_{P-1} & \cdots & h_0 & 0 & \cdots & 0 \\ 0 & h_P & \ddots & & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & h_{P-1} & \cdots & h_0 & 0 \\ 0 & \cdots & 0 & h_P & \cdots & h_1 & h_0 \end{bmatrix} \tag{3.152}$$

and

$$\boldsymbol{F}^{(2)} = \begin{bmatrix} h_0 & h_1 & \cdots & h_P & 0 & \cdots & 0 \\ 0 & h_0 & \ddots & & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & h_1 & \cdots & h_P & 0 \\ 0 & \cdots & 0 & h_0 & \cdots & h_{P-1} & h_P \end{bmatrix}. \tag{3.153}$$

The block partition of the involved time series is defined by

$$\boldsymbol{x}^{(k)} = \begin{bmatrix} x_{kL-P} & x_{kL-P+1} & \cdots & x_{(k+1)L-1} \end{bmatrix}^\top, \tag{3.154}$$

$$\boldsymbol{y}^{(k)} = \begin{bmatrix} y_{kL-P} & y_{kL-P+1} & \cdots & y_{(k+1)L-1} \end{bmatrix}^\top, \tag{3.155}$$

$$\boldsymbol{z}^{(k)} = \begin{bmatrix} z_{kL-P} & z_{kL-P+1} & \cdots & z_{(k+1)L-P-1} \end{bmatrix}^\top \tag{3.156}$$

and

$$\boldsymbol{s}^{(k)} = \begin{bmatrix} s_{kL-P} & s_{kL-P+1} & \cdots & s_{(k+1)L-1} \end{bmatrix}^\top \tag{3.157}$$

for $k = 0, 1, \ldots, K-1$, whereas L is the block size and $K = \operatorname{ceil}\left(\frac{N}{L}\right)$ is the number of blocks. Note, that the first P elements of these vectors for $k = 0$ are zero. Figure 3.9 illustrates the partitioning of the time series x_n, y_n and z_n.

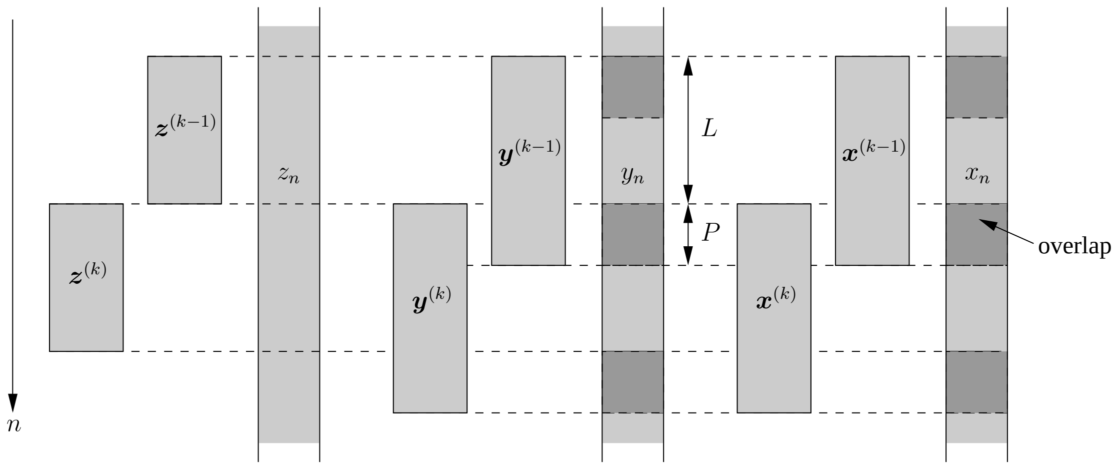


Figure 3.9: Partitioning of the time series x_n, y_n and z_n for the synchronous block method for MA filters. The blocks $\boldsymbol{x}^{(k)}$ overlap by P values of x_n. The same holds for the blocks $\boldsymbol{y}^{(k)}$. The overlapping values are marked by darker gray. Note, the blocks $\boldsymbol{z}^{(k)}$ do not overlap.

The first step of the synchronous block method is the forward filtering

$$\boldsymbol{F}^{(1)}\boldsymbol{x}^{(k)} = \begin{bmatrix} y_{kL} & y_{kL+1} & \cdots & y_{(k+1)L-1} \end{bmatrix}^\top = \boldsymbol{y}^{(k)}_{P+1:P+L} \tag{3.158}$$

Herein, $\boldsymbol{y}^{(k)}_{P+1:P+L}$ contains the first forward-filtered block. Because for $k = 0$ the the first P elements of $\boldsymbol{y}^{(k)}$ are zero, the computation of $\boldsymbol{y}^{(0)}$ is complete. For all $k > 0$ we must copy the last P elements of $\boldsymbol{y}^{(k)}$ to its front before overwriting its last L elements by the product $\boldsymbol{F}^{(1)}\boldsymbol{x}^{(k)}$.

$$\boldsymbol{y}^{(k)}_{1:P} = \boldsymbol{y}^{(k)}_{L+1:P+L} \tag{3.159}$$

The second step consists of the windowing and the backward filtering.

$$\boldsymbol{F}^{(2)}(\boldsymbol{s}^{(k)} \circ \boldsymbol{y}^{(k)}) = \boldsymbol{z}^{(k)} \tag{3.160}$$

For the application of the synchronous block method the working memory must be large enough to contain the vectors $\boldsymbol{x}^{(k)}$, $\boldsymbol{y}^{(k)}$ and $\boldsymbol{z}^{(k)}$. At the first glance, this poses no problem as vectors typically require not much memory. However, we may use this method for the computation of $\boldsymbol{A}^\top\boldsymbol{H}^\top\boldsymbol{SHA}$ or $\boldsymbol{A}^\top\boldsymbol{H}^\top\boldsymbol{SHAx}$. In this case the vectors $\boldsymbol{x}^{(k)}$, $\boldsymbol{y}^{(k)}$ and $\boldsymbol{z}^{(k)}$ are replaced by matrices. Thus, the working memory must be large enough to contain at least two matrices of the size $L + P \times M$ and one matrix of the size $L \times M$, whereas

M is the number of columns of the design matrix, i.e. the number of parameters within the Gauss-Markov model in formula (2.1).

For the case of the computation of $\boldsymbol{A}^\top \boldsymbol{H}^\top \boldsymbol{SHA}$ vector $\boldsymbol{x}$ is replaced by the design matrix $\boldsymbol{A}$. The partitioning is performed in the following manner.

$$\boldsymbol{x}^{(k)} \longrightarrow \boldsymbol{A}^{(k)} = \boldsymbol{A}_{kL-P+1:(k+1)L,:} \quad \text{for } k = 0,\, 1,\, \ldots,\, K-1 \tag{3.161}$$

Herein, L is the block size and $K = \operatorname{ceil}\left(\frac{N}{L}\right)$ is the number of blocks. Note, that vector $\boldsymbol{x}^{(k)}$ has become a matrix. With this definition of $\boldsymbol{x}^{(k)}$ we find for normal equation matrix

$$\boldsymbol{A}^\top \boldsymbol{H}^\top \boldsymbol{SHA} = \sum_{k=0}^{K-1} (\boldsymbol{A}^{(k)}_{1:L})^\top \boldsymbol{z}^{(k)}. \tag{3.162}$$

Note, that in this case not only $\boldsymbol{x}^{(k)}$ is replaced by a matrix, but also $\boldsymbol{y}^{(k)}$ and $\boldsymbol{z}^{(k)}$ are matrices.

For the case of the computation of $\boldsymbol{A}^\top \boldsymbol{H}^\top \boldsymbol{SHAx}$, whereas in this formula $\boldsymbol{x}$ denotes the parameter vector of the Gauss-Markov model in (2.1), we define

$$\boldsymbol{A}^{(k)} = \boldsymbol{A}_{kL-P+1:(k+1)L,:} \quad \text{for } k = 0,\, 1,\, \ldots,\, K-1. \tag{3.163}$$

Furthermore, we define

$$\boldsymbol{x}^{(k)} \longrightarrow \boldsymbol{A}^{(k)}\boldsymbol{x}, \tag{3.164}$$

whereas $\boldsymbol{x}^{(k)}$ denotes the time series according to equation (3.154) and $\boldsymbol{x}$ denotes the parameter vector of the Gauss-Markov model in (2.1). With these definitions we find

$$\boldsymbol{A}^\top \boldsymbol{H}^\top \boldsymbol{SHAx} = \sum_{k=0}^{K-1} (\boldsymbol{A}^{(k)}_{1:L})^\top \boldsymbol{z}^{(k)} \tag{3.165}$$

Synchronous Overlap-Add Method

The synchronous overlap-add method is very similar to the synchronous block method. The main difference is that the filtering is carried out by a multiplication in the frequency domain. The synchronous overlap-add method typically requires more memory than the synchronous block method. This is caused by the fact that the block size of the synchronous overlap-add method needs to exceed a certain value for efficiency reasons. If we have to apply the synchronous block method to the design matrix $\boldsymbol{A}$, this can lead to memory problems.

The time series x_n is partitioned into blocks of size L. These blocks are contained in the vectors

$$\boldsymbol{x}^{(k)} = \begin{bmatrix} x_{kL} & \cdots & x_{(k+1)L-1} & 0 & \cdots & 0 \end{bmatrix}^\top \quad \text{for } k = 0,\, 1,\, \ldots,\, K-1. \tag{3.166}$$

Herein, L is the block size and $K = \operatorname{ceil}\left(\frac{N}{L}\right)$ is the number of blocks. The best efficiency is achieved, if the number of zeros at the end of the vector is equal to P and if we choose the length of this vector to be equal to $N_{\text{FFT}} = 2^m$, whereas m is a positive integer number. Then, we find for the best block size $L = N_{\text{FFT}} - P$. As before, the values x_n are only known for $n = 0,\, 1,\, \ldots,\, N-1$. Therefore, we set all x_n, y_n and z_n with $n < 0$ or $n > N-1$ to zero.

Furthermore we define

$$\boldsymbol{h}^{(1)} = \begin{bmatrix} h_0 & h_1 & \cdots & h_P & 0 & \cdots & 0 \end{bmatrix}^\top \tag{3.167}$$

and

$$\boldsymbol{h}^{(2)} = \begin{bmatrix} h_P & h_{P-1} & \cdots & h_0 & 0 & \cdots & 0 \end{bmatrix}^\top, \tag{3.168}$$

which are both padded by $L-1$ zeros, such that their length is equal to N_{FFT}. The elements of vector $\boldsymbol{h}^{(2)}$ are the same as the elements of vector $\boldsymbol{h}^{(1)}$, but in reverse order. We could have perform this reversion in the frequency domain by conjugating the Fourier transform of vector $\boldsymbol{h}^{(1)}$. However, in this case the result of the convolution would have been in an unfavorable order.

The forward filtering in (3.150) is performed by

$$\boldsymbol{y}^{(k)} = \mathcal{F}^{-1}\{\mathcal{F}\{\boldsymbol{h}^{(1)}\} \circ \mathcal{F}\{\boldsymbol{x}^{(k)}\}\} \tag{3.169}$$

and

$$\boldsymbol{y}^{(k)}_{1:P} = \boldsymbol{y}^{(k)}_{1:P} + \boldsymbol{y}^{(k-1)}_{L+1:L+P}, \tag{3.170}$$

whereas $\boldsymbol{y}^{(k)}$ is defined to be

$$\boldsymbol{y}^{(k)} = \begin{bmatrix} y_{kL} & y_{kL+1} & \cdots & y_{(k+1)L+P-1} \end{bmatrix}^\top . \tag{3.171}$$

Note, that the elements of the vectors $\boldsymbol{y}^{(k-1)}$ and $\boldsymbol{y}^{(k-1)}$ are not the same. It should be also mentioned, that only the computation of the first L elements of $\boldsymbol{y}^{(k)}$ is complete. The computation of the last P elements of this vector is not finished, because these elements are the overlap.

Then we define

$$\boldsymbol{w}^{(k)} = \boldsymbol{y}^{(k)} \circ \begin{bmatrix} s_{kL} & \cdots & s_{(k+1)L-1} & 0 & \cdots & 0 \end{bmatrix}^\top , \tag{3.172}$$

whereas this vector is padded by $L-1$ zeros such that its length is equal to N_{FFT}. The elements s_n are the same as in (3.149). Therefore, the definition of vector $\boldsymbol{w}^{(k)}$ includes the windowing. With this definition we can perform the backward filtering in (3.151) by

$$\boldsymbol{z}^{(k)} = \mathcal{F}^{-1}\{\mathcal{F}\{\boldsymbol{h}^{(2)}\} \circ \mathcal{F}\{\boldsymbol{w}^{(k)}\}\} \tag{3.173}$$

and

$$\boldsymbol{z}^{(k)}_{1:P} = \boldsymbol{z}^{(k)}_{1:P} + \boldsymbol{z}^{(k-1)}_{L+1:L+P}, \tag{3.174}$$

whereas $\boldsymbol{z}^{(k)}$ is defined to be

$$\boldsymbol{z}^{(k)} = \begin{bmatrix} z_{kL-P} & z_{kL+1-P} & \cdots & z_{(k+1)L-1} \end{bmatrix}^\top . \tag{3.175}$$

It should be mentioned that in analogy to vector $\boldsymbol{y}^{(k)}$ only the computation of the first L elements of $\boldsymbol{z}^{(k)}$ is complete. The computation of the last P elements of this vector is not finished, because these elements represent the overlap. Figure 3.10 illustrates the partitioning of the time series x_n, y_n and z_n.

If we want to apply this method to the computation of $\boldsymbol{A}^\top\boldsymbol{H}^\top\boldsymbol{SHA}$, we define

$$\boldsymbol{A}^{(k)} = \boldsymbol{A}_{(k-1)L+1-P:kL,:}. \tag{3.176}$$

Vector $\boldsymbol{x}^{(k)}$ is again replaced by a matrix with M columns, whereas M is the number of columns of the design matrix $\boldsymbol{A}$, i.e. the number of parameter in the Gauss-Markov model (2.1). Then, the first L rows of $\boldsymbol{x}^{(k)}$ are equal to $\boldsymbol{A}^{(k)}$.

$$\boldsymbol{x}^{(k)}_{1:L,:} \longrightarrow \boldsymbol{A}^{(k)} \tag{3.177}$$

Within this context, the Fourier transform of a matrix is equivalent to the Fourier transform of each column. As the other vectors are also replaced by matrices, we need memory for three matrices of size $N_{\mathrm{FFT}} \times M$. For the computation of the normal equation matrix $\boldsymbol{A}^\top\boldsymbol{H}^\top\boldsymbol{SHA}$ we obtain

$$\boldsymbol{A}^\top\boldsymbol{H}^\top\boldsymbol{SHA} = \sum_{k=0}^{K-1} (\boldsymbol{A}^{(k)}_{1:L})^\top \boldsymbol{z}^{(k)}_{1:L}. \tag{3.178}$$

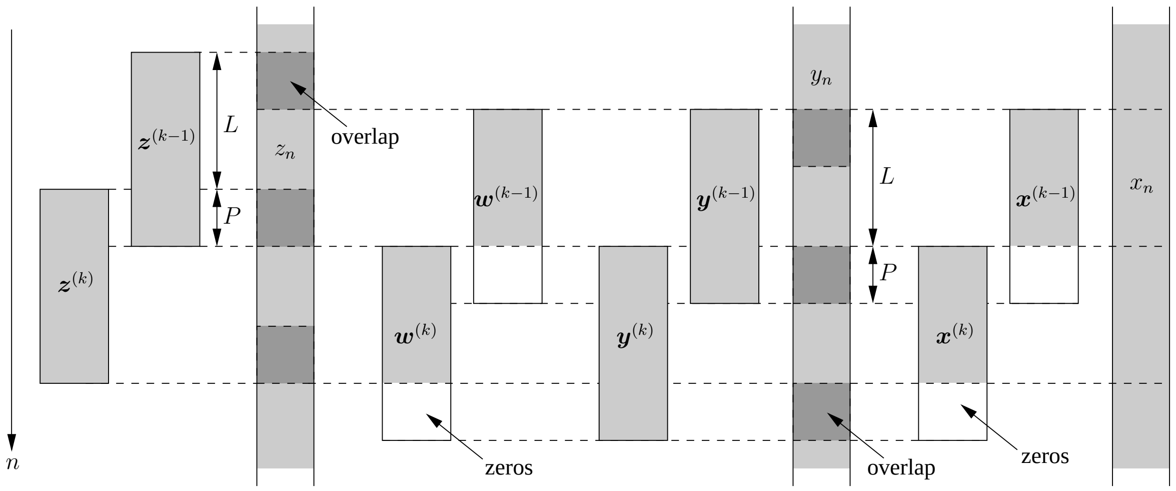


Figure 3.10: Partitioning of the time series x_n, y_n and z_n for the synchronous overlap-add method for MA filters. The blocks $\boldsymbol{x}^{(k)}$ do not overlap, but are padded by P zeros. The blocks $\boldsymbol{y}^{(k)}$ and $\boldsymbol{z}^{(k)}$ each overlap by P values, which are marked by darker gray. The blocks $\boldsymbol{w}^{(k)}$ are needed as intermediate results within the computations. Note, that the blocks $\boldsymbol{z}^{(k)}$ are shifted in comparison the blocks $\boldsymbol{x}^{(k)}$ and $\boldsymbol{y}^{(k)}$.

Note, that in this case not only $\boldsymbol{x}^{(k)}$ is replaced by a matrix, but also $\boldsymbol{y}^{(k)}$ and $\boldsymbol{z}^{(k)}$ are matrices.

For the computation of $\boldsymbol{A}^\top\boldsymbol{H}^\top\boldsymbol{SHAx}$ we use the same definition (3.176). But then we define

$$\boldsymbol{x}_{1:L}^{(k)} \longrightarrow \boldsymbol{A}^{(k)}\boldsymbol{x}, \tag{3.179}$$

whereas $\boldsymbol{x}$ is the parameter vector of the Gauss-Markov model in (2.1). Therefore, $\boldsymbol{x}^{(k)}$ remains to be a vector. For the computation of $\boldsymbol{A}^\top\boldsymbol{H}^\top\boldsymbol{SHAx}$ we find

$$\boldsymbol{A}^\top\boldsymbol{H}^\top\boldsymbol{SHAx} = \sum_{k=0}^{K-1}(\boldsymbol{A}_{1:L}^{(k)})^\top\boldsymbol{z}_{1:L}^{(k)}. \tag{3.180}$$

input	$a_0^{(s)}, \ldots, a_P^{(s)}$	filter coefficients of sections $s = 1, \ldots, S$
	$b_0^{(s)}, \ldots, b_P^{(s)}$	filter coefficients of sections $s = 1, \ldots, S$
	$\boldsymbol{x}^{(k)} = \begin{bmatrix} x_{kL} & \cdots & x_{(k+1)L-1} \end{bmatrix}^\top$	blocks $k = 0, \ldots, K-1$ of the time series x_n
output	$\boldsymbol{y}^{(k)} = \begin{bmatrix} y_{kL} & \cdots & y_{(k+1)L-1} \end{bmatrix}^\top$	blocks $k = 0, \ldots, K-1$ of the time series y_n

construction of filter matrices and initialization of filter memories

```
$\boldsymbol{m}^{(s)} = 0$   for $s = 1, \ldots, S$
$\boldsymbol{n}^{(s)} = 0$   for $s = 1, \ldots, S$
compute $\boldsymbol{F}^{(s)}$ for $s = 1, \ldots, S$ according to equation (3.70)
compute $\boldsymbol{E}^{(s)}$ for $s = 1, \ldots, S$ according to equation (3.73) and algorithm 3.1
```

filtering

```
for $k = 0, 1, \ldots, K-1$
    for $s = 0, 1, \ldots, S-1$
        if $L > P$
            $\boldsymbol{n}^{(s)} = \boldsymbol{x}^{(k)}_{L-P+1:L}$
        else
            $\boldsymbol{n}^{(s)}_{1:P-L} = \boldsymbol{n}^{(s)}_{L+1:P}$
            $\boldsymbol{n}^{(s)}_{P-L+1:P} = \boldsymbol{x}^{(k)}$
        end
        $\boldsymbol{x}^{(k)} = \boldsymbol{F}^{(s)}\boldsymbol{x}^{(k)}$   by BLAS routine dtrmm
        $\boldsymbol{x}^{(k)} = \boldsymbol{E}^{(s)}\boldsymbol{m}^{(s)} + \boldsymbol{x}^{(k)}$   by BLAS routine dgemm
        $\boldsymbol{m}^{(s)}_{1:P} = \boldsymbol{n}^{(s)}$
        if $L > Q$
            $\boldsymbol{m}^{(s)}_{P+1:P+Q} = \boldsymbol{x}^{(k)}_{L-Q+1:L}$
        else
            $\boldsymbol{m}^{(s)}_{P+1:P+Q-L} = \boldsymbol{m}^{(s)}_{P+L+1:P+Q}$
            $\boldsymbol{m}^{(s)}_{P+Q-L+1:P+Q} = \boldsymbol{x}^{(k)}$
        end
    end
end
```

Algorithm 3.2: Block-wise filtering of a time series in-place using ARMA filters.

input	$\gamma_0,\, \gamma_1,\, \ldots,\, \gamma_{N-1}$	autocovariance secquence / first row of $\mathbf{\Sigma}$
	$y_0,\, y_1,\, \ldots,\, y_{N-1}$	right hand side of the Toeplitz system
output	$x_0^{(N-1)},\, x_1^{(N-1)},\, \ldots,\, x_{N-1}^{(N-1)}$	solution of the Toeplitz system

```
initialization

$\sigma_0^2 = \gamma_0$
$c_0 = \frac{y_0}{\sigma_0^2}$
$x_0^{(0)} = c_0$
$k_1 = \frac{r_1}{\sigma_0^2}$
$b_1^{(1)} = -k_1$
$\sigma_1^2 = (1 - k_1^2)\sigma_0^2$
$c_1 = (y_1 + b_1^{(1)} y_0)\sigma_1^{-2}$
$x_0^{(1)} = c_0 + b_1^{(1)} c_1$
$x_1^{(1)} = c_1$

loop

for $n = 1, 2, \ldots, N-2$
    $k_{n+1} = (\gamma_{n+1} + \sum_{i=1}^{n} b_i^{(n)} \gamma_{n+1-i})\sigma_n^{-2}$
    $b_i^{(n+1)} = b_i^{(n)} - k_{n+1} b_{n+1-i}^{(n)}$   for $i = 1, 2, \ldots, n$
    $b_{n+1}^{(n+1)} = -k_{n+1}$
    $\sigma_{n+1}^2 = (1 - k_{n+1}^2)\sigma_n^2$
    $c_{n+1} = (y_{n+1} + \sum_{i=0}^{n} b_{n+1-i}^{(n+1)} y_i)\sigma_{n+1}^{-2}$
    $x_i^{(n+1)} = x_i^{(n)} + b_{n+1-i}^{(n+1)} c_{n+1}$   for $i = 0, 1, \ldots, n$
    $x_{n+1}^{(n+1)} = c_{n+1}$
end
```

Algorithm 3.3: The extended Levinson-Durbin algorithm for solving symmetric Toeplitz systems $\boldsymbol{y} = \mathbf{\Sigma}\boldsymbol{x}$.

4. Practical Issues

So far we have discussed how to represent the covariance matrix if the number of observations is large. For this purpose we introduced three approaches: the convolution, the AR process and ARMA process approach. Furthermore, we described in detail how to integrate these approaches into least-squares adjustment procedure. In this context we found out, that MA and ARMA filters do not produce usable values at the start of the filtering. The reason for this is that the filters need some time to warm up, when they are started. Therefore, we call this effect the warm-up of the filters (cf. SCHUH 2003).

The first part of this chapter is concerned with the determination of the warm-up length. This problem is not trivial, because the warm-up length depends on unknown values of the input time series as well as on the filter characteristics. We will see that the determination of the warm-up length is only possible, if the input time series is stationary and if we know its statistics.

Knowing the length of the warm-up does not solve the problem, that there are unusable values at the start of the filtering. However, if we have millions of observations and the warm-up length includes only a few thousand observations, then the effect of leaving these values out of the least-squares estimation in (2.8) may be negligible. Nevertheless, there may be many larger data gaps in the input time series, each requiring a restart of the filtering after the gap. In this case we lose a few thousand observations times the number of gaps. Thus, the loss of observations in the case of large data gaps may have an effect on the least-squares estimation in (2.8). For this reason we develop an algorithm for the filtering of the values at the start of the time series.

If a filter is restarted after a data gap, all correlations between the values before and after the gap are neglected. If the data gap is larger, this may be reasonable. But for short data gaps and highly correlated observations this introduces an error in the stochastic part of the Gauss-Markov model in (2.1). Therefore, we need to find a different algorithm for short data gaps. KLEES and DITMAR (2004) propose to restructure the problem such that the filtering is only used for preconditioning when integrating the inverse covariance matrix $\mathbf{\Sigma}^{-1}$. Though their algorithm is from the theoretical point of view the best solution, the calculation time of the least-squares adjustment increases dramatically. Therefore, we develop a different solution which is based on computing fill-in values for the short data gaps. These fill-in values can be regarded as a bridge for the filtering, which then can run over the gap and thus no correlations are neglected. The algorithm for computing the fill-in values for the short data gaps is designed to compute these values such that the least-squares solution in (2.8) is not affected.

4.1 Warm-up of Filters

Consider the MA filter in (2.59) which is repeated here for convenience.

$$y_n = \sum_{k=0}^{P} h_k x_{n-k} \tag{4.1}$$

Herein, the values x_n with $n < 0$ are unknown. A common practice is to set these values to zero (cf. SCHUH 1996, pp. 36,37), which is equivalent to truncating the filter. For example, the second output y_1 of the filter is obtained by

$$y_1 = h_0 x_1 + h_1 x_0 \tag{4.2}$$

instead of

$$y_1 = h_0 x_1 + h_1 x_0 + h_2 x_{-1} + \cdots + h_P x_{1-P}. \tag{4.3}$$

Truncating the filter can change its characteristics dramatically. In this spirit, the warm-up length is defined to be the smallest value of n, for which the truncation results in an acceptable change of the filter characteristics.

For MA filters this effect completely vanishes for $n \geq P$. Thus, only the values $y_0, \ldots, y_{P-1}$ are affected and the warm-up of an MA filter is simply defined to be of length P. If the filter order P is relatively small compared to the length N of the observation time series, just leaving out the first P observations in the adjustment process does not perceptibly affect the solution of the Gauss-Markov model in (2.1).

We can make use of this discussion about MA filters, when considering ARMA filters. Every ARMA filter can be converted to an MA filter of infinite order (cf. section 2.4). If the ARMA filter is stable, the filter coefficients h_k of the converted filter converge to zero as k increases. Therefore, we can approximate this infinite MA filter with a finite MA filter of order R by setting all h_k with $k > R$ to zero. The warm-up of an ARMA filter is then defined to be the smallest value R, for which the difference of the filter characteristics between ARMA filter and finite MA filter is acceptably small.

For the practical determination of the warm-up length, we need to define the meaning of *acceptably small* in some way. Let us assume for a moment that the unknown values x_{-1}, x_{-2}, ... were given, such that the time series x_n begins at $n = -\infty$ and is therefore infinitely long. With these values, we can express the difference Δy_n between the filtered finite and infinite filtered time series y_n introduced by setting x_{-1}, x_{-2}, ... to zero. We define *acceptably small* such that this difference is at least two orders of magnitude smaller than the values y_n of the filtered infinite time series.

However, we do not know the residuals x_{-1}, x_{-2}, ... and we therefore need to find a workaround. In order to derive the workaround step by step, we start by expressing the difference Δx_n between the finite and infinite time series in vector form.

$$\begin{bmatrix} \Delta x_{-\infty} \\ \vdots \\ \Delta x_{-1} \\ \Delta x_0 \\ \vdots \\ \Delta x_{N-1} \end{bmatrix} = \begin{bmatrix} x_{-\infty} \\ \vdots \\ x_{-1} \\ x_0 \\ \vdots \\ x_{N-1} \end{bmatrix} - \begin{bmatrix} 0 \\ \vdots \\ 0 \\ x_0 \\ \vdots \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} x_{-\infty} \\ \vdots \\ x_{-1} \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{4.4}$$

The difference between the filtered infinite and finite time series is therefore the impulse response of the filter to the values x_{-1}, x_{-2}, ... (cf. section 2.4). It should be mentioned, that using the impulse response as criterion makes the analysis easy, even for ARMA filters consisting of several sections. After combining equation (4.4) with (2.57), we find

$$\begin{bmatrix} \Delta y_0 \\ \Delta y_1 \\ \vdots \\ \Delta y_{N-1} \end{bmatrix} = x_{-1} \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_N \end{bmatrix} + x_{-2} \begin{bmatrix} h_2 \\ h_3 \\ \vdots \\ h_{N+1} \end{bmatrix} + x_{-3} \begin{bmatrix} h_3 \\ h_4 \\ \vdots \\ h_{N+2} \end{bmatrix} + \cdots \tag{4.5}$$

for the last N values of this difference. For stable filters the h_k will converge to zero as k increases (cf. section 2.4). This has two consequences. The first consequence is, that the difference will be dominated by x_k with k close to zero, while the x_k with big negative k have practically no influence. The second consequence is, that only the first elements of Δy_n of this vector differ distinctly from zero.

The next step in the workaround is the index k for which the h_k have converged to numerically zero. This means that we need to find the index W where

$$|h_k| < \varepsilon \quad \text{for all } k > W. \tag{4.6}$$

Herein ϵ denotes the computational accuracy which is equal to 2^{-52} for double precision numbers. For the determination of an upper bound $W^{(u)}$ for W we use the state-space equation (3.43) in combination with the impulse response h_n. According to (2.56), the impulse response h_n is obtained by filtering the unit impulse

d_n as defined in (2.55). Because then $x_n = d_n \neq 0$ only for $n = 0$, it follows from equation (3.41) that $z_n = 0$ for $n > P$. Thus, if we define

$$\boldsymbol{h}_n = \begin{bmatrix} h_{n-Q+1} & \cdots & h_{n-1} & h_n \end{bmatrix}^\top, \tag{4.7}$$

we find

$$\boldsymbol{h}_{n+1} = \boldsymbol{A}\boldsymbol{h}_n \quad \text{for } \ n > P-1, \tag{4.8}$$

whereas $\boldsymbol{A}$ is the companion matrix, which is defined in equation (3.42). By equation (3.48) we obtain

$$\boldsymbol{h}_{n+L} = \boldsymbol{A}^L\boldsymbol{h}_n \quad \text{for } \ n > P-1. \tag{4.9}$$

Performing a spectral decomposition for matrix $\boldsymbol{A}$ yields

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^{-1} \quad \Longrightarrow \quad \boldsymbol{A}^2 = \boldsymbol{U}\boldsymbol{D}^2\boldsymbol{U}^{-1} \quad \Longrightarrow \quad \cdots \quad \Longrightarrow \quad \boldsymbol{A}^L = \boldsymbol{U}\boldsymbol{D}^L\boldsymbol{U}^{-1}, \tag{4.10}$$

whereas $\boldsymbol{U}$ is a matrix containing the eigenvectors of $\boldsymbol{A}$ in its columns, and $\boldsymbol{D}$ is diagonal matrix containing the eigenvalues of $\boldsymbol{A}$ on its diagonal. By substituting (4.10) into (4.9) we find

$$\boldsymbol{h}_{n+L} = \boldsymbol{U}\boldsymbol{D}^L\boldsymbol{U}^{-1}\boldsymbol{h}_n \quad \text{for } \ n > P-1. \tag{4.11}$$

As matrix $\boldsymbol{A}$ as defined in (3.42) is the companion matrix of the polynomial $1 - a_1z^{-1} - \cdots - a_qz^{-Q}$, the eigenvalues of $\boldsymbol{D}$ are the roots of the polynomial and thus the poles α_j of the transfer function in equation (2.102) (cf. Meyer 2000, p. 648, 649), i.e.

$$\boldsymbol{D} = \begin{bmatrix} \alpha_1 & 0 & \cdots & 0 \\ 0 & \alpha_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \alpha_Q \end{bmatrix} \tag{4.12}$$

Furthermore, equation (4.11) is linear with respect to the poles α_j. Thus, we can write the last row of the equations in (4.11) in the form

$$h_{n+L} = v_1\alpha_1^L + v_2\alpha_2^L + \cdots + v_Q\alpha_Q^L \tag{4.13}$$

The coefficients $v_1, \ldots, v_Q$ are obtained by

$$h_{n+L} = \boldsymbol{U}_{Q,:}\boldsymbol{D}^L\boldsymbol{U}^{-1}\boldsymbol{h}_n = \boldsymbol{U}_{Q,:}(\operatorname{diag}(\boldsymbol{D}^L) \circ \boldsymbol{U}^{-1}\boldsymbol{h}_n) = \boldsymbol{U}_{Q,:}(\boldsymbol{U}^{-1}\boldsymbol{h}_n \circ \operatorname{diag}(\boldsymbol{D}^L)), \tag{4.14}$$

whereas $v_i = \boldsymbol{U}_{Q,i}\boldsymbol{U}^{-1}_{i,:}\boldsymbol{h}_n$. By the triangle inequality we find

$$|h_{n+L}| = |v_1\alpha_1^L + v_2\alpha_2^L + \cdots + v_Q\alpha_Q^L| \leq |v_1\alpha_1^L| + |v_2\alpha_2^L| + \cdots + |v_Q\alpha_Q^L| \leq Qv_{\max}\alpha_{\max}^L, \tag{4.15}$$

whereas

$$\alpha_{\max} = \max(|\alpha_1|, \ldots, |\alpha_Q|) \quad \text{and} \quad v_{\max} = \max(|v_1|, \ldots, |v_Q|). \tag{4.16}$$

With equation (4.6) we can now write

$$|h_{n+L}| \leq Qv_{\max}\alpha_{\max}^L < \varepsilon. \tag{4.17}$$

For stable filters, it holds that $\alpha_{\max} < 1$. The logarithm of $\alpha_{\max}$ is thus negative. After taking this into account, we find by rearranging (4.17)

$$L > \frac{1}{\ln \alpha_{\max}} \ln \frac{\varepsilon}{Qv_{\max}}. \tag{4.18}$$

We choose the smallest integer value for L, for which (4.18) holds, i.e. we round up the right hand side of (4.18). The upper bound $W^{(u)}$ for W in equation (4.6) is then

$$W^{(u)} = L + P. \tag{4.19}$$

The order P has to be added, because the value L results only from the recursive part of the filter equation (2.53). In the case that the ARMA filter consists of more than one section, we can compute this value for each filter section which we then denote by $W_k^{(u)}$. We can imagine that for each filter section the unit impulse takes place after the effect of the unit impulse of the preceding filter section has converged to numerical zero which is represented by ε. If we add the unit impulse at the point, where the impulse response of the preceding filter section has converged to numerical zero, the result is exactly equal to the unit impulse, because the impulse response is truncated in the addition due to the limited numerical accuracy. The upper bound $W^{(u)}$ for the whole filter is then obtained by

$$W^{(u)} = \sum_{k=0}^{K-1} W_k^{(u)}, \tag{4.20}$$

whereas K is the number of sections. Now, we simply compute the impulse response h_n for $n = 0, \ldots, W^{(u)}$. In order to determine the smallest possible W according to equation (4.6), we only have to search the values $h_0, \ldots, h_{W^{(u)}}$.

Now that we know how to determine W, we set all $h_k = 0$ with $k > W$. This truncation of the impulse response practically has no effect on the filter, because the magnitude of the truncation error is maximum of the size of the unavoidable roundoff noise. Therefore equation (4.5) yields

$$\begin{bmatrix} \Delta y_0 \\ \Delta y_1 \\ \vdots \\ \Delta y_{W-1} \end{bmatrix} = \begin{bmatrix} h_W & h_{W-1} & \cdots & h_1 \\ 0 & h_W & \cdots & h_2 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & h_W \end{bmatrix} \begin{bmatrix} x_{-W} \\ x_{-W+1} \\ \vdots \\ x_{-1} \end{bmatrix}. \tag{4.21}$$

Herein we assume that W is smaller than N. By variance propagation we obtain

$$\mathbf{\Sigma}\{\begin{bmatrix} \Delta y_0 \\ \Delta y_1 \\ \vdots \\ \Delta y_{W-1} \end{bmatrix}\} = \begin{bmatrix} h_W & h_{W-1} & \cdots & h_1 \\ 0 & h_W & \cdots & h_2 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & h_W \end{bmatrix} \mathbf{\Sigma}\{\begin{bmatrix} x_{-W} \\ x_{-W+1} \\ \vdots \\ x_{-1} \end{bmatrix}\} \begin{bmatrix} h_W & 0 & \cdots & 0 \\ h_{W-1} & h_W & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ h_1 & h_2 & \cdots & h_W \end{bmatrix}. \tag{4.22}$$

In section 2.2 we discussed that one necessary condition for the residuals is, that they correspond to a stationary time series. For this reason, the covariance matrix $\mathbf{\Sigma}$ within the Gauss-Markov model in (2.1) has a symmetric Toeplitz structure. Because the ARMA filters are used to filter the residual time series v_n, we find

$$\mathbf{\Sigma}\{\begin{bmatrix} v_{-W} \\ v_{-W+1} \\ \vdots \\ v_{-1} \end{bmatrix}\} = \mathbf{\Sigma}\{\begin{bmatrix} x_{-W} \\ x_{-W+1} \\ \vdots \\ x_{-1} \end{bmatrix}\} = \begin{bmatrix} \gamma_0 & \gamma_1 & \cdots & \gamma_{W-1} \\ \gamma_1 & \gamma_0 & \cdots & \gamma_{W-2} \\ \vdots & \vdots & & \vdots \\ \gamma_{W-1} & \gamma_{W-2} & \cdots & \gamma_0 \end{bmatrix}. \tag{4.23}$$

In order to determine the magnitude of the values $\Delta y_0, \ldots, \Delta y_{W-1}$ we compute only their variances $\sigma^2_{\Delta y_0}, \ldots, \sigma^2_{\Delta y_{W-1}}$, i.e. the diagonal of the covariance matrix in (4.22). We find for the variances

$$\sigma^2_{\Delta y_n} = \begin{bmatrix} h_W & h_{W-1} & \cdots & h_n \end{bmatrix} \begin{bmatrix} \gamma_0 & \gamma_1 & \cdots & \gamma_{W-n} \\ \gamma_1 & \gamma_0 & \cdots & \gamma_{W-n-1} \\ \vdots & \vdots & & \vdots \\ \gamma_{W-n} & \gamma_{W-n-1} & \cdots & \gamma_0 \end{bmatrix} \begin{bmatrix} h_W \\ h_{W-1} \\ \vdots \\ h_n \end{bmatrix}. \tag{4.24}$$

Because of

$$\begin{aligned}
\sigma^2_{\Delta\mathcal{Y}_{n+1}} &= \begin{bmatrix} h_W & \cdots & h_{n+1} \end{bmatrix} \begin{bmatrix} \gamma_0 & \cdots & \gamma_{W-n-1} \\ \vdots & & \vdots \\ \gamma_{W-n-1} & \cdots & \gamma_0 \end{bmatrix} \begin{bmatrix} h_W \\ \vdots \\ h_{n+1} \end{bmatrix} \\
&= \begin{bmatrix} h_W & \cdots & h_{n+1} \end{bmatrix} \left(\begin{bmatrix} \gamma_0 & \cdots & \gamma_{W-n-1} & \gamma_{W-n} \\ \vdots & & \vdots & \vdots \\ \gamma_{W-n-1} & \cdots & \gamma_0 & \gamma_1 \end{bmatrix} \begin{bmatrix} h_W \\ \vdots \\ h_n \end{bmatrix} - h_n \begin{bmatrix} \gamma_{W-n} \\ \vdots \\ \gamma_1 \end{bmatrix} \right) \\
&= \begin{bmatrix} h_W & \cdots & h_n \end{bmatrix} \begin{bmatrix} \gamma_0 & \cdots & \gamma_{W-n} \\ \vdots & & \vdots \\ \gamma_{W-n} & \cdots & \gamma_0 \end{bmatrix} \begin{bmatrix} h_W \\ \vdots \\ h_n \end{bmatrix} - 2h_n \begin{bmatrix} h_W & \cdots & h_n \end{bmatrix} \begin{bmatrix} \gamma_{W-n} \\ \vdots \\ \gamma_1 \end{bmatrix} - h_n^2 \gamma_0 \\
&= \sigma^2_{\Delta\mathcal{Y}_n} - 2h_n \begin{bmatrix} h_W & \cdots & h_n \end{bmatrix} \begin{bmatrix} \gamma_{W-n} \\ \vdots \\ \gamma_1 \end{bmatrix} - h_n^2 \gamma_0 \qquad (4.25)
\end{aligned}$$

we can compute the variances $\sigma^2_{\Delta\mathcal{Y}_n}$ recursively for $n = W-1,\, W-2,\, \dots,\, 0$. Now, we only have to find the index R, for which the standard deviations $\sigma_{\Delta\mathcal{Y}_n}$ are at least two orders of magnitude smaller than y_n. If we designed the ARMA filter such that the variance of the filtered time series is equal to one, then this means that all $\sigma_{\Delta\mathcal{Y}_n} < 10^{-2}$ for all $n > R$.

In practice we filter the observation time series l_n, whereas the unknown observations $l_{-1}, l_{-2}, \dots$ are set to zero. This is equivalent to assuming that $E\{l_n\} = 0$ for $n < 0$, which we must take into account for the estimation of the autocovariances γ_j. Therefore, we can use for example

$$r_k = \frac{1}{N-k} \sum_{n=0}^{N-k-1} l_k l_{n+k}. \qquad (4.26)$$

as an estimation for the autocovariances γ_j. It should be noted, that the assumption $E\{l_n\} = 0$ for $n < 0$ has the potential to violate the Toeplitz structure of the covariance matrix for the observations $l_{-1}, l_{-2}, \dots$.

Finally, we should mention that the position of the poles has a large influence of the length of the warm-up. The closer a pole α_j lies to the unit circle in the z-plane, the greater is its reciprocal logarithm $\ln^{-1} \alpha_j$ in equation (4.18). Therefore, it would make sense to design filters with poles, which are not close to th unit cirlce in the z-plane. However, the position of the poles depends mainly on the correlations of the residual time series v_n while the design is secondary in this context. Algorithm 4.1 summarizes the necessary steps for the determination of the warm-up length.

input	$a_0^{(s)}, \ldots, a_P^{(s)}$	filter coefficients of sections $s = 1, \ldots, S$
	$b_0^{(s)}, \ldots, b_P^{(s)}$	filter coefficients of sections $s = 1, \ldots, S$
	ε	small constant defining numerical zero
	l_n	observation time series
output	R	warm-up length

```
upper bound for warm-up length

$W^{(u)} = 0$
for $s = 0, 1, \ldots, S-1$
    set up the companion matrix $\boldsymbol{A}^{(s)}$
    compute the poles $\alpha_j^{(s)}$ for $j = 1, \ldots, Q^{(s)}$ as eigenvalues of the companion matrix $\boldsymbol{A}^{(s)}$
    $W^{(u)} = W^{(u)} + P^{(s)} + \text{ceil}\left(\frac{1}{\ln \alpha_{\max}^{(s)}} \ln \frac{\varepsilon}{Q^{(s)} v_{\max}^{(s)}}\right)$
end

impulse response of the filter and autocovariance function of the observations

compute the impulse response $h_n$ for $n = 0, \ldots, W^{(u)}$
set $W$ equal to the smallest $k$ for which $|h_k| < \varepsilon$ for all $k$ holds
compute the autocovariance function $r_k$ for $k = 0, \ldots, W-1$ of the observations according to (4.26)

variance propagation

compute $\sigma^2_{\Delta\mathcal{y}_W} = \begin{bmatrix} h_W & \cdots & h_n \end{bmatrix} \begin{bmatrix} r_0 & \cdots & r_{W-n} \\ \vdots & & \vdots \\ r_{W-n} & \cdots & r_0 \end{bmatrix} \begin{bmatrix} h_W \\ \vdots \\ h_n \end{bmatrix}$ applying FFT techniques
for $n = W-1, W-2, \ldots, 1$
    $\sigma^2_{\Delta\mathcal{y}_n} = \sigma^2_{\Delta\mathcal{y}_{n+1}} + 2h_n \begin{bmatrix} h_W & \cdots & h_n \end{bmatrix} \begin{bmatrix} r_{W-n} \\ \vdots \\ r_1 \end{bmatrix} + h_n^2 r_0$
end

warm-up length

set $R$ equal to the smallest $k$ for which $\sigma_{\Delta\mathcal{y}_k} < 10^{-2}$ for all $k$ holds
```

Algorithm 4.1: Computation of the warm-up length of ARMA filters.

4.2 Long Data Gaps

In practice we often have to deal with data gaps within the observation time series. Whenever we use the AR or ARMA process approach to represent the covariance matrix of the observations, we have two strategies for the handling of data gaps. The first strategy is tailored for long data gaps while the second is tailored for short data gaps. The main difference between these two strategies is, that the strategy for long data gaps neglects all correlations between the observations before and after the data gap, while the strategy for short data gaps allows for correlations. Therefore, whether a data gap is long or short depends on the correlations between the observations, i.e. the autocovariance function of the observations.

In this section we describe the strategy for long data gaps. The strategy is to stop the filter before the data gap and restart it after the data gap. This directly leads us to the problem of the filter warm-up. Without any special handling we loose R observations after the data gap, whereas R is the warm-up length of the filter. If there exist many data gaps which are classified as long data gaps, this procedure leads to an enormous loss of observations which affects the least-squares solution in (2.8). Therefore, we need to find a procedure of starting the filter without this loss of observations. This procedure can then be used for the start at the beginning of the observation time series and every restart of the filter after a data gap.

In section 4.1 we already discussed, that the loss of observations is due to the truncation of the filter at the start of the filtering. In order to go into more detail we look at equation (2.40) in combination with (2.41), which states that

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{H}\boldsymbol{\mathcal{X}}. \tag{4.27}$$

The entries of matrix $\boldsymbol{H}$ correspond to the impulse response of the filter. In section 2.4 we discussed that for stable filters the impulse response converges to zero. This fact was used in section 4.1 to define the warm-up length of ARMA filters, which was defined to be equal to the order R of an MA filter which approximates the ARMA filter such that the approximation error is negligible. Typically, this order R is much smaller than the length N of the observation time series. In this case matrix $\boldsymbol{H}$ can be approximated by a banded lower triangular Toeplitz matrix.

$$\boldsymbol{H} = \begin{bmatrix} h_0 & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & h_0 & \ddots & \vdots & & \vdots \\ h_R & \vdots & \ddots & 0 & \cdots & 0 \\ h_{R+1} & h_R & & h_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots & \ddots & 0 \\ h_N & \cdots & h_{R+1} & h_R & \cdots & h_0 \end{bmatrix} \approx \begin{bmatrix} h_0 & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & h_0 & \ddots & \vdots & & \vdots \\ h_R & \vdots & \ddots & 0 & \cdots & 0 \\ 0 & h_R & & h_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots & \ddots & 0 \\ 0 & \cdots & 0 & h_R & \cdots & h_0 \end{bmatrix} \tag{4.28}$$

We can regard each row of this matrix an MA filter. Take for example the second row of this matrix. It corresponds to the truncated filter in equation (4.2). In fact, not only the second row, but the first R rows of this matrix correspond to truncated versions of the MA filter which approximates the ARMA filter. For the last $N - R$ rows of this matrix, the truncation error is negligible. Thus, the upper left part

$$\boldsymbol{H}_{1:R,1:R} = \begin{bmatrix} h_0 & 0 & \cdots & 0 \\ h_1 & h_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ h_{R-1} & \cdots & h_1 & h_0 \end{bmatrix} \tag{4.29}$$

is corrupted by the truncation of the impulse response due to the warm-up of the filter. In order to analyze the effect on the decorrelation of the observations in equation (3.131), we subdivide matrix $\boldsymbol{H}$ into an upper and a lower part.

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H}_{1:R,:} \\ \boldsymbol{H}_{R+1:N,:} \end{bmatrix} \tag{4.30}$$

In equation (3.131) we multiply the observation equations by matrix $\boldsymbol{H}$. From the preceding discussion we can conclude that the decorrelation is is only successful, where the upper part $\boldsymbol{H}_{1:R,:}$ has no influence. This is for the last $N-R$ rows of the observation equations. By variance propagation we find

$$\boldsymbol{\Sigma}\{\boldsymbol{H}\boldsymbol{\mathcal{L}}\} = \begin{bmatrix} \boldsymbol{H}_{1:R,:} \\ \boldsymbol{H}_{R+1:N,:} \end{bmatrix} \boldsymbol{\Sigma} \begin{bmatrix} \boldsymbol{H}_{1:R,:}^{\top} & \boldsymbol{H}_{R+1:N,:}^{\top} \end{bmatrix} = \begin{bmatrix} \boldsymbol{H}_{1:R,:}\boldsymbol{\Sigma}\boldsymbol{H}_{1:R,:}^{\top} & \boldsymbol{H}_{1:R,:}\boldsymbol{\Sigma}\boldsymbol{H}_{R+1:N,:}^{\top} \\ \boldsymbol{H}_{R+1:N,:}\boldsymbol{\Sigma}\boldsymbol{H}_{1:R,:}^{\top} & \boldsymbol{H}_{R+1:N,:}\boldsymbol{\Sigma}\boldsymbol{H}_{R+1:N,:}^{\top} \end{bmatrix}. \quad (4.31)$$

for the covariance matrix of the decorrelated observations. Because only the lower right block of this matrix is not affected by $\boldsymbol{H}_{1:R,:}$, the covariance matrix $\boldsymbol{\Sigma}\{\boldsymbol{H}\boldsymbol{\mathcal{L}}\}$ has the structure given in equation (3.133)

$$\boldsymbol{\Sigma}\{\boldsymbol{H}\boldsymbol{\mathcal{L}}\} = \begin{bmatrix} * & * \\ * & \boldsymbol{I} \end{bmatrix}, \quad (4.32)$$

whereas the asterisk denotes that the block differs from the corresponding block of an identity matrix. In section 3.3 we circumvented this problem by setting the first R filtered observation equations to zero which corresponds to setting $\boldsymbol{H}_{1:R,:}$ to zero. However, this leads to a loss of observations which we want to avoid here.

The idea for the solution of this problem is quite simple: If the problem is caused by the truncation of the MA filters in the first R rows of matrix $\boldsymbol{H}$, then replace the MA filters in these rows by MA filters which are not truncated and approximate the decorrelating capabilities of the ARMA filter as good as possible. Figure 4.1 illustrates the truncation of the filter coefficients as well as the solution of this problem.

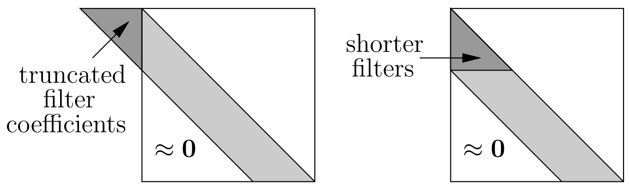


Figure 4.1: The filter warm-up is caused by the truncation of the filter coefficients in the first rows of matrix $\boldsymbol{H}$ in equation (4.27). The solution of this problem is to replace the truncated filters in the first lines of matrix $\boldsymbol{H}$ by shorter filters. The shorter filters should approximate the decorrelating capabilities of the original filter.

Therefore, the task is to compute MA filters from order 1 to order $R-1$ which best approximate either the autocovariance function of the ARMA filter or the power spectral density of the ARMA filter. The Levinson-Durbin algorithm is best dedicated for this task, because it determines the required MA filters recursively with low computational costs. Within the original algorithm, described for example by FARHANG-BOROUJENY (1998), pp. 357–377, we only have to add the derivation of the filter coefficients. Furthermore, we can directly integrate the filtering of the first R elements of a time series which then replace the warm-up elements of the filtered time series. Thus, the procedure for the filtering of the warm-up is now at hand. Algorithm 4.2 shows the Levinson-Durbin algorithm extended for the filtering of the warm-up.

At this point we should repeat that the MA filters approximate the autocovariance function of the ARMA filter. As the autocovariance function is the time domain counterpart of the power spectral density, the phase of the ARMA filter is irrelevant for the approximation. We could say that we buy a good quality approximation of the power spectral density at the cost of the phase. However, the phase of a filter is irrelevant for the filters decorrelating capabilities as discussed in section 2.4. Figure 4.2 shows the effect of the new algorithm to prevent the filter warm-up on the covariance matrix of the observations, which is derived by $\boldsymbol{H}$.

Integration into the Computation of the Normal Equation Matrix

In least-squares adjustment procedures, we have to filter the design matrix $\boldsymbol{A}$ at least once, if the normal equation matrix cannot be approximated by means of analytical formulas. Therefore, we need to investigate

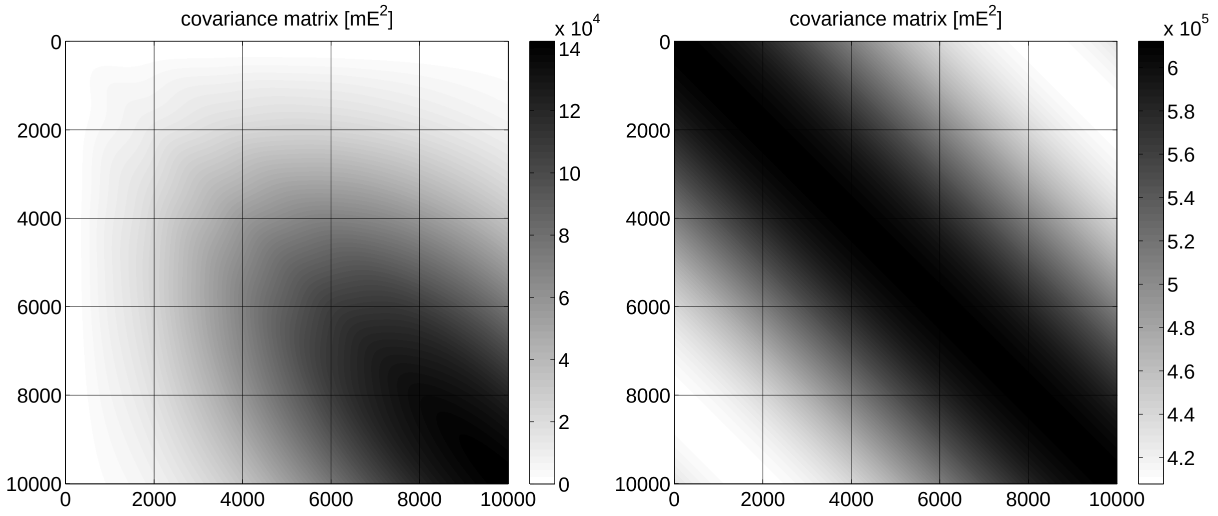


Figure 4.2: Effect of the new algorithm to prevent the filter warm-up on the covariance matrix of the observations. The basis for the computations is the simple averaging filter described in section 6.4. The covariance matrix has been computed by formula (2.34) and (2.31). The left panel shows the covariance matrix computed with the filter matrix $\boldsymbol{H}$ according to equation (2.41). The structure of the covariance matrix apparently deviates from the assumed Toeplitz structure. The right panel shows the covariance matrix computed with the filter matrix according to equation (4.35), which corresponds to the application of algorithm 4.2. Here, the structure of the covariance matrix comes very close to the structure of a Toeplitz matrix.

how to integrate the filtering of the warm-up, if we have to filter the design matrix. For column-wise access to the design matrix we can directly use algorithm 4.2 for each column. For row-wise access to the design matrix the integration is more complicated. Here, we need to have R rows of the design matrix in the working memory at a time for algorithm 4.2. If this is possible, then the problem is solved easily. If this is not possible, a strict integration is not possible. However, if the design matrix is filtered for the computation of a preconditioner within iterative least-squares adjustment procedures, then we are able to find a workaround which does provide an approximate integration. Because only the preconditioner will be affected by this, the least-squares estimates in (2.8) are in principle not affected. Unfortunately, for direct least-squares adjustment procedures which require the computation of the normal equation matrix, the least-squares estimates in (2.8) may be affected by the approximate integration of the filtering of the warm-up.

If we use algorithm 4.2 for the filtering of the first R values of y_n, then the filter equation for these values changes from (2.53) to

$$y_n = \sum_{k=0}^{n} h_k^{(n)} x_{n-k} \quad \text{for } n = 0, \ldots, R-1. \tag{4.33}$$

Herein, the $h_k^{(n)}$ are the filter coefficients of the MA filters of algorithm 4.2. After writing this in matrix-vector form, we find

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{R-1} \end{bmatrix} = \boldsymbol{W} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{R-1} \end{bmatrix}, \tag{4.34}$$

where

$$\boldsymbol{W} = \begin{bmatrix} h_0^{(0)} & 0 & \cdots & 0 \\ h_1^{(1)} & h_0^{(1)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ h_{R-1}^{(R-1)} & \cdots & h_1^{(R-1)} & h_0^{(R-1)} \end{bmatrix}. \tag{4.35}$$

Formally, replacing the filtered values y_n of equation (2.53) by the filtered values y_n of algorithm 4.2 corresponds to replacing the upper part $\boldsymbol{H}_{1:R,1:R}$ of matrix $\boldsymbol{H}$ in (4.27) by this matrix $\boldsymbol{W}$ which contains the MA filters estimated by the Levinson-Durbin algorithm.

$$\begin{bmatrix} \boldsymbol{W} & \boldsymbol{0} \\ \boldsymbol{H}_{R+1:N,1:R} & \boldsymbol{H}_{R+1:N,R+1:N} \end{bmatrix} = \boldsymbol{SH} + \begin{bmatrix} \boldsymbol{W} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \tag{4.36}$$

Herein, matrix $\boldsymbol{S}$ the same windowing matrix as in equation (3.134) which we used to set the first R rows of the observation equations to zero in section 3.3.

For the filtering of the design matrix we find

$$(\boldsymbol{SH} + \begin{bmatrix} \boldsymbol{W} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix})\boldsymbol{A} = \boldsymbol{SHA} + \begin{bmatrix} \boldsymbol{WA}_{1:R,:} \\ \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{WA}_{1:R,:} \\ \boldsymbol{H}_{R+1:N,:}\boldsymbol{A} \end{bmatrix} \tag{4.37}$$

which replaces $\boldsymbol{SHA}$ in equation (3.135). For the normal equation matrix we obtain

$$\boldsymbol{N} = (\boldsymbol{SHA} + \begin{bmatrix} \boldsymbol{WA}_{1:R,:} \\ \boldsymbol{0} \end{bmatrix})^\top (\boldsymbol{SHA} + \begin{bmatrix} \boldsymbol{WA}_{1:R,:} \\ \boldsymbol{0} \end{bmatrix}). \tag{4.38}$$

Alternatively we can write

$$\begin{aligned} \boldsymbol{N} &= \begin{bmatrix} \boldsymbol{WA}_{1:R,:} \\ \boldsymbol{H}_{R+1:N,:}\boldsymbol{A} \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{WA}_{1:R,:} \\ \boldsymbol{H}_{R+1:N,:}\boldsymbol{A} \end{bmatrix} \\ &= (\boldsymbol{H}_{R+1:N,:}\boldsymbol{A})^\top \boldsymbol{H}_{R+1:N,:}\boldsymbol{A} + (\boldsymbol{WA}_{1:R,:})^\top \boldsymbol{WA}_{1:R,:} \\ &= \boldsymbol{A}^\top \boldsymbol{H}^\top \boldsymbol{SHA} + \boldsymbol{A}_{1:R,:}^\top \boldsymbol{W}^\top \boldsymbol{WA}_{1:R,:} \\ &= \boldsymbol{A}^\top \boldsymbol{PHPSHA} + \boldsymbol{A}_{1:R,:}^\top \boldsymbol{W}^\top \boldsymbol{WA}_{1:R,:}. \end{aligned} \tag{4.39}$$

Equation (4.38) corresponds to the integration into the decorrelation approach in equation (3.137). Here, we filter the design matrix as before and then overwrite the first R rows by the result of algorithm 4.2. Equation (4.39) describes the integration into the direct approach in equation (3.142). Here, too, we perform the filtering as before. Then we add an additional matrix to the normal equation matrix. Thus, for both approaches the integration of the filtering of the warm-up turns out not to change the computation procedure, but to be an additional part of it.

As matrix $\boldsymbol{W}$ is a lower triangular matrix, the product $\boldsymbol{WA}_{1:R,:}$ can be computed in-place. Thus, the computation for this step requires memory for matrices of size $R \times R$ and $R \times M$, whereas M is the number of parameters in the Gauss-Markov model in (2.1). If the warm-up length R and the number of parameters M are both not small, then it is likely these matrices do not fit into the working memory. In this case we could divide both matrices into blocks and compute the product block by block. However, this block by block computation typically turns out to be very time consuming. For direct least-squares adjustment procedures, which make use of the normal equation matrix, this time consuming computation cannot be avoided, if we do not want to loose the first R observations. Within iterative least-squares adjustment procedures the normal equation matrix is typically partly computed in order to obtain a suitable preconditioner. Here, the least-squares estimates in (2.8) in principle do not depend on the preconditioner, which only influences the convergence speed of the iterative procedure. Thus, the least-squares estimates in (2.8) do not rely on the exact computation of $\boldsymbol{W}\boldsymbol{A}_{1:R,:}$. In order to reduce the computational costs, we redefine

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{W}^{(0)} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \boldsymbol{W}^{(1)} & \boldsymbol{W}^{(2)} & \mathbf{0} & \ddots & \vdots \\ \mathbf{0} & \ddots & \ddots & \ddots & \mathbf{0} \\ \vdots & \ddots & \boldsymbol{W}^{(1)} & \boldsymbol{W}^{(2)} & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{W}^{(1)} & \boldsymbol{W}^{(2)} \end{bmatrix} \tag{4.40}$$

where

$$\boldsymbol{W}^{(0)} = \begin{bmatrix} h_0^{(0)} & 0 & \cdots & 0 \\ h_1^{(1)} & h_0^{(1)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ h_{V-1}^{(V-1)} & \cdots & h_1^{(V-1)} & h_0^{(V-1)} \end{bmatrix}, \quad \boldsymbol{W}^{(1)} = \begin{bmatrix} h_V^{(V)} & h_{V-1}^{(V)} & \cdots & h_1^{(V)} \\ 0 & h_V^{(V)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & h_{V-1}^{(V)} \\ 0 & \cdots & 0 & h_V^{(V)} \end{bmatrix} \tag{4.41}$$

and

$$\boldsymbol{W}^{(2)} = \begin{bmatrix} h_0^{(V)} & 0 & \cdots & 0 \\ h_1^{(V)} & h_0^{(V)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ h_{V-1}^{(V)} & \cdots & h_1^{(V)} & h_0^{(V)} \end{bmatrix}. \tag{4.42}$$

From row $V+1$ to the last row matrix $\boldsymbol{W}$ has a Toeplitz structure, because in each row is the same filter. We can interpret this definition of matrix $\boldsymbol{W}$ as an approximation of the previous definition. The computation of the blocks $\boldsymbol{W}^{(0)}$, $\boldsymbol{W}^{(1)}$ and $\boldsymbol{W}^{(2)}$ is described in algorithm 4.3.

According to the partitioning of matrix $\boldsymbol{W}$ in (4.37) matrix $\boldsymbol{A}_{1:R,:}$ is also subdivided into blocks.

$$\boldsymbol{A}_{1:R,:} = \begin{bmatrix} \boldsymbol{A}_{1:V,:} \\ \boldsymbol{A}_{V+1:2V,:} \\ \vdots \\ \boldsymbol{A}_{U(V-1)+1:UV,:} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}^{(0)} \\ \boldsymbol{A}^{(1)} \\ \vdots \\ \boldsymbol{A}^{(U-1)}, \end{bmatrix} \tag{4.43}$$

where the number of blocks is

$$U = \operatorname{ceil}(R/V). \tag{4.44}$$

In the case of $UV > R$ the last block $\boldsymbol{A}^{(U)}$ is $UV - R$ rows longer than the warm-up length of the filter. These rows are eliminated in the computation by setting them to zero after the filtering. Therefore, the contents of these rows before the filtering may be arbitrary. With this partitioning the product $\boldsymbol{W}\boldsymbol{A}_{1:R,:}$ reduces to the computation of

$$\boldsymbol{W}^{(0)}\boldsymbol{A}^{(1)} \tag{4.45}$$

and

$$\boldsymbol{W}^{(1)}\boldsymbol{A}^{(k-1)} + \boldsymbol{W}^{(2)}\boldsymbol{A}^{(k)}, \quad k = 1, 3, \ldots, U-1. \tag{4.46}$$

We should not forget, that besides the computation of $\boldsymbol{W}\boldsymbol{A}_{1:R,:}$ we have to filter the design matrix $\boldsymbol{A}$ from the row first on as before. As the block size V for the warm-up procedure and the block-size L for the tering differ, we need two matrices $\boldsymbol{M}^{(1)}$ and $\boldsymbol{M}^{(2)}$ to store copies of the blocks of the design matrix. This is necessary, because we exploit the fact that the matrices $\boldsymbol{W}^{(0)}$, $\boldsymbol{W}^{(1)}$ and $\boldsymbol{W}^{(2)}$ are triangular matrices, which has the side effect that the computations of the products $\boldsymbol{W}^{(i)}\boldsymbol{M}^{(j)}$ are computed in-place. Algorithm 4.4 describes the sequence of computations.

In practice there are two special cases which require special handling. The first case concerns filters with a high gain in the power spectral density, for example high-pass, low-pass or band-pass filters. Such filters can only be approximated by MA filters of order $P = 1$ and higher. The reason for this is, that the power spectral density of an MA filter of order $P = 0$ is constant, i.e.

$$P(\omega) = |H(e^{i\omega})|^2 = b_0^2 \tag{4.47}$$

according to equation (2.53) and (2.89). Because the first row of the filtered design matrix is obtained by using an MA filter of order $P = 0$, it should therefore not be used within the further processing. Thus, if the designed filter has a gain in the power spectral density, we set the first row after the filtering to zero.

The second case, which requires special treatment, arises, if the filter has one or more zeros β_k on the unit circle in the z-plane, i.e. $|\beta_k| = 1$. The algorithm 4.4 is based on the Levinson-Durbin algorithm which in turn uses the autocovariance function of the inverse filter. If the filter has a zero on the unit circle in the z-plane, the inverse filter is not stable. In this case we first manipulate the filter such that the inverse filter becomes stable. For the manipulation we compute the power spectral density $P(\omega)$ of the filter (cf. equation (2.89) in section 2.4). The power spectral density is exactly zero for the frequencies $\omega = \sphericalangle(\beta_k)$. Thus, the power spectral density of the inverse filter is infinitely large for these frequencies as it is simply the reciprocal of the power spectral density of the filter. In order to avoid this, we set the power spectral density $P(\omega)$ for $\omega = \sphericalangle(\beta_k)$ with $|\beta_k| = 1$ to a small positive value. Then, the reciprocal power spectral density is finite and the manipulated inverse filter is stable. The decorrelation capabilities of the manipulated inverse filter are then almost the same as those of the inverse filter.

Iterations

Within iterative least-squares adjustment procedures we typically have to compute

$$\boldsymbol{A}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{A}\boldsymbol{x} \quad \text{or} \quad \boldsymbol{A}^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{l}) \tag{4.48}$$

according to equation (3.145). For simplicity we set

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \quad \text{or} \quad \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}-\boldsymbol{l} \tag{4.49}$$

such that equation (4.48) becomes

$$\boldsymbol{A}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{y}. \tag{4.50}$$

If we use ARMA filters to integrate the inverse covariance matrix $\boldsymbol{\Sigma}^{-1}$, then we replace it by $\boldsymbol{H}^\top\boldsymbol{S}^\top\boldsymbol{S}\boldsymbol{H}$ according to equation (3.139). This leads to

$$\boldsymbol{A}^\top\boldsymbol{H}^\top\boldsymbol{S}^\top\boldsymbol{S}\boldsymbol{H}\boldsymbol{y}. \tag{4.51}$$

As in equation (4.37) we now replace $\boldsymbol{S}\boldsymbol{H}$ by

$$\boldsymbol{S}\boldsymbol{H} + \begin{bmatrix}\boldsymbol{W} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix} \tag{4.52}$$

in order to integrate the filtering of the warm-up, which yields

$$\boldsymbol{A}^\top(\boldsymbol{S}\boldsymbol{H} + \begin{bmatrix}\boldsymbol{W} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix})^\top(\boldsymbol{S}\boldsymbol{H} + \begin{bmatrix}\boldsymbol{W} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix})\boldsymbol{y} = \boldsymbol{A}^\top\boldsymbol{H}^\top\boldsymbol{S}\boldsymbol{H}\boldsymbol{y} + \boldsymbol{A}_{1:R,:}^\top\boldsymbol{W}^\top\boldsymbol{W}\boldsymbol{y}_{1:R,:}. \tag{4.53}$$

Herein, some terms vanish because of

$$\boldsymbol{S}\begin{bmatrix}\boldsymbol{W} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix} = \begin{bmatrix}\boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{W} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix} = \boldsymbol{0} \tag{4.54}$$

due to the definition of $\boldsymbol{S}$ in (3.134).

Equation (4.53) indicates, that the integration of the inverse covariance matrix $\boldsymbol{\Sigma}^{-1}$ can be performed in two steps. The first step consists of the computation of $\boldsymbol{A}^\top\boldsymbol{H}^\top\boldsymbol{S}\boldsymbol{H}\boldsymbol{y}$. Here, the integration is the same as before. All of the in section 3 introduced filter methods are applicable. The second step consists of the computation of $\boldsymbol{A}^\top_{1:R,:}\boldsymbol{W}^\top\boldsymbol{W}\boldsymbol{y}_{1:R,:}$ which belongs to the filtering of the warm-up. It is remarkable, that though we replace the upper part of the filter matrix $\boldsymbol{H}$ in equation (4.36), the implementation of the new method for the warm-up results in adding a vector to the first elements of the original filter method.

Now we investigate how to compute $\boldsymbol{A}^\top_{1:R,:}\boldsymbol{W}^\top\boldsymbol{W}\boldsymbol{y}_{1:R,:}$. There are two possibilities. The first corresponds to the decorrelation approach in equation (3.137), which directly results from the decorrelation of the observation equations. For this approach the design matrix is filtered. In the previous subsection about the computation of the normal equation matrix we already discussed how to do this. The second possibility corresponds to the direct approach in equation (3.142). For this approach all operations which concern the filtering are carried out from the right to left. This means, that the filtering is here applied to the vector $\boldsymbol{y}$, such that the filtering is computationally cheap in this case. Thus, for the implementation of the filtering of the warm-up we need to analyze how to compute $\boldsymbol{W}^\top\boldsymbol{W}\boldsymbol{y}_{1:R,:}$. The product $\boldsymbol{W}\boldsymbol{y}_{1:R,:}$ can be computed by algorithm 4.2. Therefore, we only have to extend this algorithm for the additional multiplcation by $\boldsymbol{W}^\top$. For the derivation we recall that within algorithm 4.2 the lines of matrix $\boldsymbol{W}$ are computed recursively. For this reason we formulate the product $\boldsymbol{W}^\top\boldsymbol{W}\boldsymbol{y}_{1:R,:}$ in terms of dyadic products of the lines of $\boldsymbol{W}$. We define

$$\boldsymbol{z} = \boldsymbol{W}^\top\boldsymbol{W}\boldsymbol{y}_{1:R} = \sum_{r=1}^{R}\boldsymbol{W}^\top_{r,1:r}\boldsymbol{W}_{r,1:r}\boldsymbol{y}_{1:r} = \sum_{r=1}^{R}\boldsymbol{W}^\top_{r,1:r}\boldsymbol{g}_r, \quad \text{whereas } \boldsymbol{g} = \boldsymbol{W}\boldsymbol{y}. \tag{4.55}$$

With this equation we can now formulate algorithm (4.5) for the computation of $\boldsymbol{z}$.

For convenience we summarize the filtering in algorithm 4.6. There we use the block method for ARMA filters for the normal filtering, which could be replaced by any of the other filter methods described in chapter 3. If there are one or more long data gaps, algorithm 4.6 is applied for each observation segment between the data gaps. Because we neglect all correlations between the observation segment before and the observation segment after a long data gap, we can process each observation segment independently of the other observation segments.

4.3 Short Data Gaps

In contrast to long data gaps the filtering process is not stopped at short data gaps. In this way, correlations between observations before and after a data gap are taken into account. However, we need to compute fill-in values for the data gaps. This is not only necessary for the observations, but also for the design matrix. The reason for this is, that not the observations are filtered, but the observations equations according to equation (3.131). The observation can be split into two parts: the deterministic and the stochastic part. Within the Gauss-Markov model in (2.1) the deterministic part of the observations corresponds to the adjusted observations $\widetilde{\boldsymbol{l}}$ in equation (2.9) while the stochastic part corresponds to the residuals $\boldsymbol{v}$ in equation (2.10).

$$\boldsymbol{l} = \widetilde{\boldsymbol{l}} - \boldsymbol{v} \tag{4.56}$$

For the observations it is important to compute fill-in values for both parts. Because the focus of this thesis lies on the stochastic part, we do not investigate how to compute fill-in values for the deterministic part in detail. It should be mentioned, that we can use either interpolation techniques or a priori knowledge about the deterministic model $\boldsymbol{A}\boldsymbol{x}$ of the Gauss-Markov model in (2.1). Note, that this knowledge is not only needed for the model parameters $\boldsymbol{x}$ but also for the design matrix $\boldsymbol{A}$, which has to be computed for the data gaps. Another reason for not going into detail is, that the best way of computing fill-in values for the deterministic part depends on the application.

More complex than the deterministic part is the stochastic part represented by the residuals $\boldsymbol{v}$. Though the expectation of the residuals is zero according to (2.6), it is not reasonable to simply set this part to zero. The fill-in values of the stochastic part have to fit to the values before and after the data gaps, because otherwise the filters would get a large impulse contaminating the observations after the data gap. Furthermore, the fill-in values need to accommodate the decorrelating capabilities of the filter. These two aspects are most important for highly correlated observations. The procedure for the computation of the fill-in values for the stochastic part of the observations starts with a simple interpolation of the residuals in (2.10). If the observations are highly correlated, this step is important because of the low frequency components of the power spectral density, which in this case possesses a high gain. In the second step the filled residual time series is used to compute a first filter model. From then on we alternately compute fill-in values and estimate a filter model until the decorrelation capabilities of the estimated filters converge. As the decorrelating capabilities are described by the power spectral density, convergence is reached, if the power spectral density of the estimated filters do not change from one iteration to the next. Figure 4.3 illustrates this procedure.

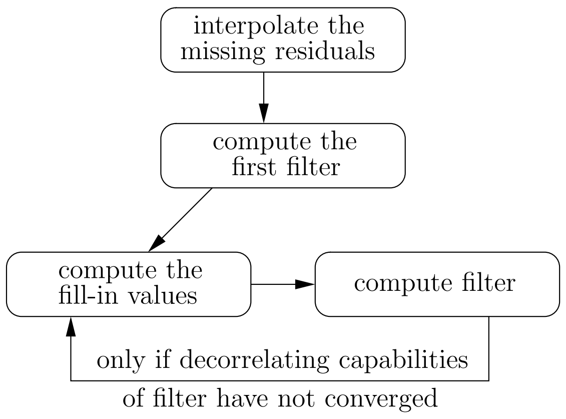

interpolate the missing residuals
compute the first filter
compute the fill-in values
compute filter
only if decorrelating capabilities of filter have not converged

Figure 4.3: Procedure for the computation of the fill-in values for short data gaps.

The methods which are used here for the estimation of ARMA filters are described in chapter 5. If the ARMA filters are stable, we can convert them into MA filters (cf. section 2.4). When we do this using also the method for the warm-up described in section 4.1, we find for the relation between the time series x_n and the filtered time series y_n

$$\boldsymbol{y} = \boldsymbol{F}\boldsymbol{x}, \tag{4.57}$$

where

$$\boldsymbol{x} = \begin{bmatrix} x_0 & \cdots & x_{N-1} \end{bmatrix}^\top, \quad \boldsymbol{y} = \begin{bmatrix} y_0 & \cdots & y_{N-1} \end{bmatrix}^\top \tag{4.58}$$

and

$$\boldsymbol{F} = \boldsymbol{S}\boldsymbol{H} + \begin{bmatrix} \boldsymbol{W} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \approx \begin{bmatrix} h_0^{(0)} & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ h_1^{(1)} & h_0^{(1)} & \ddots & \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \ddots & 0 & 0 & 0 & \cdots & 0 \\ h_{R-1}^{(R-1)} & h_{R-2}^{(R-1)} & \cdots & h_0^{(R-1)} & 0 & 0 & \cdots & 0 \\ h_R^{(R)} & h_{R-1}^{(R)} & \cdots & h_1^{(R)} & h_0^{(R)} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & h_R^{(R)} & h_{R-1}^{(R)} & \cdots & h_1^{(R)} & h_0^{(R)} & 0 \\ 0 & \cdots & 0 & h_R^{(R)} & \cdots & h_2^{(R)} & h_1^{(R)} & h_0^{(R)} \end{bmatrix} \tag{4.59}$$

is a band matrix, which has a Toeplitz structure below row R. The assumption that $R < N$ does not restrict this approach, as it is only used for the purpose of deriving an efficient computation procedure. For the computation of the fill-in values, the missing values x_n are regarded as parameters. We can estimate them by minimizing the residual sum of squares

$$\sum_{n=0}^{N-1} y_n^2 = \boldsymbol{y}^\top \boldsymbol{y} = \boldsymbol{x}^\top \boldsymbol{F}^\top \boldsymbol{F}\boldsymbol{x} \tag{4.60}$$

within a least-square adjustment. Note, that the filter coefficients are determined such that $\boldsymbol{\Sigma}\{\boldsymbol{y}\} = \boldsymbol{I}$, which accommodates the minimization of the residual sum of squares in (4.60). Moreover, if we estimate the missing values x_n this way, the fill-in values will have the smallest possible effect on the residual sum of squares in (2.13) of the Gauss-Markov model, because the time series y_n corresponds to the residual time series v_n.

In order to simplify the handling of indices we define $\boldsymbol{d}$ as the vector which contains the indices of the data gaps plus one. If for example the values x_4 and x_8 were data gaps, then the vector would be $\boldsymbol{d} = \begin{bmatrix}\boldsymbol{d}_1 & \boldsymbol{d}_2\end{bmatrix} = \begin{bmatrix}5 & 9\end{bmatrix}$. The length of $\boldsymbol{d}$, which is equal to the number of missing values, is denoted by D. Furthermore, we define

$$z_n = \begin{cases} x_n, & \text{if } n \notin \{\boldsymbol{d}_1, \dots, \boldsymbol{d}_D\} \\ 0, & \text{if } n \in \{\boldsymbol{d}_1, \dots, \boldsymbol{d}_D\} \end{cases}. \tag{4.61}$$

The time series z_n is the same as the time series x_n except for the data gaps, where it is zero. Analogously to $\boldsymbol{x}$ and $\boldsymbol{y}$ we define

$$\boldsymbol{z} = \begin{bmatrix} z_0 & \cdots & z_{N-1}\end{bmatrix}^\top . \tag{4.62}$$

With these definitions we can extract the missing values x_n in equation (4.57).

$$\boldsymbol{y} = \boldsymbol{F}\boldsymbol{x} = \begin{bmatrix}\boldsymbol{F}_{:,\boldsymbol{d}_1} & \cdots & \boldsymbol{F}_{:,\boldsymbol{d}_D}\end{bmatrix} \begin{bmatrix}\boldsymbol{x}_{\boldsymbol{d}_1} \\ \vdots \\ \boldsymbol{x}_{\boldsymbol{d}_D}\end{bmatrix} + \boldsymbol{F}\boldsymbol{z} \tag{4.63}$$

Note, that for example $\boldsymbol{x}_{\boldsymbol{d}_1}$ is a single missing value of the time series x_n. The missing values $\boldsymbol{x}_{\boldsymbol{d}_1}, \dots, \boldsymbol{x}_{\boldsymbol{d}_D}$ are the parameters for minimizing the residual sum of squares in equation (4.60). The least-squares estimates for these missing values are

$$\begin{bmatrix}\boldsymbol{x}_{\boldsymbol{d}_1} \\ \vdots \\ \boldsymbol{x}_{\boldsymbol{d}_D}\end{bmatrix} = -\left(\begin{bmatrix}\boldsymbol{F}_{:,\boldsymbol{d}_1} & \cdots & \boldsymbol{F}_{:,\boldsymbol{d}_D}\end{bmatrix}^\top \begin{bmatrix}\boldsymbol{F}_{:,\boldsymbol{d}_1} & \cdots & \boldsymbol{F}_{:,\boldsymbol{d}_D}\end{bmatrix}\right)^{-1} \begin{bmatrix}\boldsymbol{F}_{:,\boldsymbol{d}_1} & \cdots & \boldsymbol{F}_{:,\boldsymbol{d}_D}\end{bmatrix}^\top \boldsymbol{F}\boldsymbol{z}. \tag{4.64}$$

For the sake of clarity, we consider the following example. Let

$$\boldsymbol{F} = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 1 & 2 \end{bmatrix} \quad \text{and} \quad \boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ ? \\ 2 \\ -2 \end{bmatrix} \tag{4.65}$$

be given. Herein, x_1 is the missing value. Form this it follows that

$$\boldsymbol{d} = \begin{bmatrix}\boldsymbol{d}_1\end{bmatrix} = \begin{bmatrix}2\end{bmatrix}, \quad \boldsymbol{z} = \begin{bmatrix} 3 \\ 0 \\ 2 \\ -2 \end{bmatrix} \quad \text{and} \quad \boldsymbol{F}_{:,\boldsymbol{d}_1} = \boldsymbol{F}_{:,2} = \begin{bmatrix} 0 \\ 2 \\ 1 \\ 0 \end{bmatrix}. \tag{4.66}$$

Then, the least squares estimate of the missing value x_1 is

$$x_1 = \begin{bmatrix}\boldsymbol{x}_{\boldsymbol{d}_1}\end{bmatrix} = -\left(\boldsymbol{F}_{:,\boldsymbol{d}_1}^\top \boldsymbol{F}_{:,\boldsymbol{d}_1}\right)^{-1} \boldsymbol{F}_{:,\boldsymbol{d}_1}^\top \boldsymbol{F}\boldsymbol{z} = -2. \tag{4.67}$$

Because the number N of observations is large for the problems we consider, the number D of missing values x_n may be large, too. In this case matrix $\begin{bmatrix}\boldsymbol{F}_{:,\boldsymbol{d}_1} & \cdots & \boldsymbol{F}_{:,\boldsymbol{d}_D}\end{bmatrix}$ may be too big to fit into the working memory. Thus, we need to find a tailored strategy to evaluate equation (4.64).

First we investigate the normal equation matrix

$$\boldsymbol{N} = \begin{bmatrix}\boldsymbol{F}_{:,\boldsymbol{d}_1} & \cdots & \boldsymbol{F}_{:,\boldsymbol{d}_D}\end{bmatrix}^\top \begin{bmatrix}\boldsymbol{F}_{:,\boldsymbol{d}_1} & \cdots & \boldsymbol{F}_{:,\boldsymbol{d}_D}\end{bmatrix}. \tag{4.68}$$

The elements of $\boldsymbol{N}$ are all contained in the matrix $\boldsymbol{F}^\top\boldsymbol{F}$, which can be efficiently computed because of its banded Toeplitz structure below row R. Therefore, we subdivide $\boldsymbol{F}$ into the upper part $\boldsymbol{F}_{1:R,:}$ and the lower part $\boldsymbol{F}_{R+1:N,:}$ and find

$$\boldsymbol{F}^\top\boldsymbol{F} = \begin{bmatrix}\boldsymbol{F}_{1:R,:}\\ \boldsymbol{F}_{R+1:N,:}\end{bmatrix}^\top \begin{bmatrix}\boldsymbol{F}_{1:R,:}\\ \boldsymbol{F}_{R+1:N,:}\end{bmatrix} = \boldsymbol{F}_{1:R,:}^\top\boldsymbol{F}_{1:R,:} + \boldsymbol{F}_{R+1:N,:}^\top\boldsymbol{F}_{R+1:N,:}. \tag{4.69}$$

Now we can fully exploit the banded Toeplitz structure of the lower part $\boldsymbol{F}_{R+1:N,:}$. We can express the normal equation matrix $\boldsymbol{N}$ as the sum of two matrices

$$\boldsymbol{N} = \boldsymbol{N}^{(1)} + \boldsymbol{N}^{(2)} \tag{4.70}$$

whereas

$$\boldsymbol{N}^{(1)} = \begin{bmatrix}\boldsymbol{F}_{1:R,\boldsymbol{d}_1} & \cdots & \boldsymbol{F}_{1:R,\boldsymbol{d}_D}\end{bmatrix}^\top \begin{bmatrix}\boldsymbol{F}_{1:R,\boldsymbol{d}_1} & \cdots & \boldsymbol{F}_{1:R,\boldsymbol{d}_D}\end{bmatrix} \tag{4.71}$$

and

$$\boldsymbol{N}^{(2)} = \begin{bmatrix}\boldsymbol{F}_{R+1:N,\boldsymbol{d}_1} & \cdots & \boldsymbol{F}_{R+1:N,\boldsymbol{d}_D}\end{bmatrix}^\top \begin{bmatrix}\boldsymbol{F}_{R+1:N,\boldsymbol{d}_1} & \cdots & \boldsymbol{F}_{R+1:N,\boldsymbol{d}_D}\end{bmatrix}. \tag{4.72}$$

Within the computation of $\boldsymbol{N}^{(1)}$ we can only exploit the triangular structure of $\boldsymbol{F}_{1:R,:}$ while for the computation of $\boldsymbol{N}^{(2)}$ we benefit much from the Toeplitz structure of $\boldsymbol{F}_{R+1:N,:}$. Furthermore, the normal equation matrix $\boldsymbol{N}$ is likely to be sparse, if the number of small data gaps is small in comparison to the length of the observation time series and if the warm-up length of the filter is not too long. In order to exploit this, we remark here that for each element $\boldsymbol{N}_{i,j}$ of the normal equation matrix it holds that $\boldsymbol{d}_i \leq \boldsymbol{d}_j$, if the vector $\boldsymbol{d}$ is arranged in ascending order and $i \leq j$. We make use of this later on. Now we investigate the product $\boldsymbol{F}_{1:R,:}^\top\boldsymbol{F}_{1:R,:}$.

With the Levinson-Durbin algorithm we can compute all non-zero elements of $\boldsymbol{F}_{1:R,:}$ row by row. In each row $\boldsymbol{F}_{k,:}$ only the first k elements are non-zero, because $\boldsymbol{F}_{1:R,:}$ is a triangular matrix. Because of the row-wise access to matrix $\boldsymbol{F}_{1:R,:}$ we compute the first part $\boldsymbol{N}^{(1)}$ of the normal equation matrix by summation of dyadic products. For these dyadic products we only need to incorporate those columns, which belong to the data gaps. Thus, the dyadic products are generally not computed for the whole row. For simplicity we first copy each row $\boldsymbol{F}_{k,1:k}$ into a vector, before we choose the needed elements of this row with the help of vector $\boldsymbol{d}$.

$$\boldsymbol{h} = \boldsymbol{F}_{k,1:k} = \begin{bmatrix}h_{k-1}^{(k-1)} & h_{k-2}^{(k-1)} & \cdots & h_0^{(k-1)}\end{bmatrix}. \tag{4.73}$$

After that, we copy the elements of the row which belong to short data gaps to the front of vector $\boldsymbol{h}$.

$$\boldsymbol{h}_i = \boldsymbol{h}_{\boldsymbol{d}_i}, \quad \text{for } i = 1, \ldots, j \tag{4.74}$$

Herein, j is the number of missing values x_n up to the index $n = k$. Then, we initialize

$$\boldsymbol{N}^{(1)} = \boldsymbol{0}. \tag{4.75}$$

and proceed with the summation of the dyadic products

$$\boldsymbol{N}_{1:j,1:j}^{(1)} = \boldsymbol{N}_{1:j,1:j}^{(1)} + \boldsymbol{h}_{1:j}^\top\boldsymbol{h}_{1:j}, \tag{4.76}$$

whereas $\boldsymbol{h}_{1:j}$ corresponds to the row $\boldsymbol{F}_{k,1:k}$ as defined in (4.73) and (4.74). In algorithm 4.7 for the computation of $\boldsymbol{N}^{(1)}$ formula (4.76) is integrated for $k = 1, \ldots, R$ into the Levinson-Durbin algorithm.

Now we investigate the product $\boldsymbol{F}_{R+1:N,:}^{\top}\boldsymbol{F}_{R+1:N,:}$, which is only calculated if $N > R$. We can compute the elements of this product by dot products of the columns of $\boldsymbol{F}_{R+1:N,:}$. Because matrix $\boldsymbol{F}_{R+1:N,:}$ has a banded Toeplitz structure, these dot products can be obtained by a correlation, which in turn can be efficiently computed by an element-wise multiplication in the frequency domain. Furthermore, only $R + 1$ elements in each column are non-zero. Thus, we define for the correlation

$$\boldsymbol{h} = \begin{bmatrix} h_0^{(R)} & \cdots & h_R^{(R)} & 0 & \cdots & 0 \end{bmatrix}^{\top}. \tag{4.77}$$

Vector $\boldsymbol{h}$ is padded by zeros such that its length is equal to $N_{\text{FFT}} = 2^{\text{ceil}\,(\log_2(2R+1))}$. With this definition the correlation can be computed by

$$\boldsymbol{c} = \mathcal{F}^{-1}\{\mathcal{F}\{\boldsymbol{h}\} \circ \mathcal{F}^{*}\{\boldsymbol{h}\}\}. \tag{4.78}$$

The elements of vector $\boldsymbol{c}$ now correspond to the dot products of the columns of matrix $\boldsymbol{F}_{R+1:N,:}$ in the following manner.

$$\boldsymbol{c}_1 = \boldsymbol{h}_{1:R+1}^{\top}\boldsymbol{h}_{1:R+1}, \quad \boldsymbol{c}_2 = \boldsymbol{h}_{2:R+1}^{\top}\boldsymbol{h}_{1:R}, \quad \ldots, \quad \boldsymbol{c}_{R+1} = \boldsymbol{h}_{R+1}^{\top}\boldsymbol{h}_1 \tag{4.79}$$

Thus, all products $\boldsymbol{F}_{R+1:N,\boldsymbol{d}_i}^{\top}\boldsymbol{F}_{R+1:N,\boldsymbol{d}_j}$ for which $\boldsymbol{d}_i \leq N - R$ and $\boldsymbol{d}_j > R$ are elements of $\boldsymbol{c}$.

$$\boldsymbol{N}_{i,j}^{(2)} = \boldsymbol{F}_{R+1:N,\boldsymbol{d}_i}^{\top}\boldsymbol{F}_{R+1:N,\boldsymbol{d}_j} = \begin{cases} \boldsymbol{c}_{\boldsymbol{d}_j-\boldsymbol{d}_i+1}, & \text{if } \boldsymbol{d}_i + R \leq \boldsymbol{d}_j \\ 0, & \text{otherwise} \end{cases} \tag{4.80}$$

The remaining elements of $\boldsymbol{N}_{i,j}^{(2)}$, these are the elements for which either $\boldsymbol{d}_i > N - R$ or $\boldsymbol{d}_j \leq R$, are computed by dot products of the corresponding columns. The detailed indexing for these dot products is depicted in algorithm 4.8 for the computation of $\boldsymbol{N}^{(2)}$.

Putting algorithm 4.7 and 4.8 together, we know how to compute the normal equation matrix $\boldsymbol{N}$. In order to complete the computation of the normal equations we now investigate the right hand side

$$\boldsymbol{n} = \begin{bmatrix} \boldsymbol{F}_{:,\boldsymbol{d}_1} & \cdots & \boldsymbol{F}_{:,\boldsymbol{d}_D} \end{bmatrix}^{\top} \boldsymbol{F}\boldsymbol{z} \tag{4.81}$$

of the normal equations. All elements of $\boldsymbol{n}$ are contained in the vector

$$\boldsymbol{u} = \boldsymbol{F}^{\top}\boldsymbol{F}\boldsymbol{z}. \tag{4.82}$$

As the multiplication by matrix $\boldsymbol{F}$ corresponds to filtering according to equation (4.59), the computation of $\boldsymbol{F}^{\top}\boldsymbol{F}\boldsymbol{z}$ via forward and backward filtering can be performed as described in section 4.1. Once vector $\boldsymbol{u}$ is computed, we only have to copy the required elements.

$$\boldsymbol{n}_i = \boldsymbol{u}_{\boldsymbol{d}_i} \quad \text{for } i = 1, 2, \ldots, D \tag{4.83}$$

At this point we know how to compute the normal equations, but not how to solve them efficiently. As indicated before, the normal equation may become large. Thus, to store the full normal equation matrix may not be possible. However, in many cases the normal equation matrix will be sparse. In the following we describe the outlines of a sparse storage scheme for the normal equations. Because normal equation matrices are symmetric, we only need to compute their upper triangular part. Furthermore, we introduce the algorithms 4.9 for the Cholesky decomposition and algorithm 4.10 for forward and backward substitution, which are both suited to the sparse storage scheme.

Because matrix $\boldsymbol{F}$ has a banded structure according to equation (4.59), in each column starting at the diagonal there are only $R + 1$ non-zero elements. Therefore, it holds that

$$\boldsymbol{N}_{i,j} = 0, \quad \text{if } R > |\boldsymbol{d}_j - \boldsymbol{d}_i|. \tag{4.84}$$

R corresponds to the warm-up length of the filter, which is typically small in comparison to the length N of the time series. Thus, it is likely that many elements $\boldsymbol{N}_{i,j}$ of the normal equation matrix are zero, if the short data gaps are distributed over the whole time series x_n. For this reason we use a sparse storage scheme, which we call the shell-oriented storage scheme, for the normal equation matrix. This storage scheme is best suited in this case, because it only stores the non-zero elements of the normal equation matrix. Furthermore, it allows the in-place solution of the normal equation matrix, which is also much faster than the conventional solution, if many elements $\boldsymbol{N}_{i,j}$ are zero.

The shell-oriented storage scheme is designed to store all elements of the columns of the normal equation matrix, starting at the first non-zero element in each column and ending at the main diagonal element. Because these elements are stored contiguously in vector, the indices of the main diagonal elements are stored in an additional index vector $\boldsymbol{i}$. With this index vector $\boldsymbol{i}$, we can navigate through the sparse stored normal equation matrix. The first entry in the index vector $\boldsymbol{i}$ references a virtual column $j = 0$, which does not exist. However, this allows for a simpler navigation. For convenience, we look at the following example.

$$\boldsymbol{N} = \begin{bmatrix} 1 & 2 & 0 & 0 \\ 2 & 3 & 0 & 5 \\ 0 & 0 & 4 & 6 \\ 0 & 5 & 6 & 7 \end{bmatrix} \quad \Longrightarrow \quad \begin{aligned} \boldsymbol{N}^{(\text{shell})} &= \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{bmatrix}, \\ \boldsymbol{i} &= \begin{bmatrix} 0 & 1 & 3 & 4 & 7 \end{bmatrix} \end{aligned}$$

The normal equations are then solved by a Cholesky decomposition depicted in algorithm 4.9 and a subsequent forward and backward substitution depicted in algorithm 4.10. Both algorithms are carried out in-place, such that no additional working memory is needed.

input	$c_0, c_1, \ldots, c_{R-1}$	autocovariance secquence of the process and the inverse filter, respectively
	$x_0, x_1, \ldots, x_{R-1}$	time series
output	$y_0, y_1, \ldots, y_{R-1}$	filtered time series

```
initialization

$\sigma_0^2 = c_0$
$h_0^{(0)} = \sigma_0^{-1}$
$y_0 = h_0^{(0)} x_0$
$k_1 = c_1 \sigma_0^{-2}$
$u_1^{(1)} = -k_1$
$\sigma_1^2 = (1 - k_1^2)\sigma_0^2$
$h_0^{(1)} = \sigma_1^{-1}$
$h_1^{(1)} = u_1^{(1)} \sigma_1^{-1}$
$y_1 = h_0^{(1)} x_1 + h_1^{(1)} x_0$

loop

for $r = 1, 2, \ldots, R-2$
    $k_{r+1} = (c_{r+1} + \sum_{k=1}^{r} u_k^{(r)} c_{r+1-k})\sigma_r^{-2}$
    $u_k^{(r+1)} = u_k^{(r)} - k_{r+1} u_{r+1-k}^{(r)}$   for $k = 1, 2, \ldots, r$
    $u_{r+1}^{(r+1)} = -k_{r+1}$
    $\sigma_{r+1}^2 = (1 - k_{r+1}^2)\sigma_r^2$
    $h_0^{(r+1)} = \sigma_{r+1}^{-1}$
    $h_k^{(r+1)} = u_k^{(r+1)} \sigma_{r+1}^{-1}$   for $k = 1, 2, \ldots, r+1$
    $y_{r+1} = \sum_{k=0}^{r+1} h_k^{(r+1)} x_{r+1-k}$
end
```

Algorithm 4.2: The Levinson-Durbin Algorithm extended for the filtering of the warm-up.

input	$c_0,\, c_1,\, \ldots,\, c_R$	autocovariance sequence of the process and the inverse filter, respectively
	V	maximum filter order for approximation
output	$\boldsymbol{W}^{(0)},\, \boldsymbol{W}^{(1)},\, \boldsymbol{W}^{(2)}$	filter matrices for warm-up

```
initialization
$\sigma_0^2 = c_0$
$\boldsymbol{W}^{(0)}_{1,1} = \sigma_0^{-1}$
$k_1 = c_1\sigma_0^{-2}$
$u_1^{(1)} = -k_1$
$\sigma_1^2 = (1-k_1^2)\sigma_0^2$
$\boldsymbol{W}^{(0)}_{2,2} = \sigma_1^{-1}$
$\boldsymbol{W}^{(0)}_{2,1} = u_1^{(1)}\sigma_1^{-1}$

loop
for $r = 1, 2, \ldots, V-2$
    $k_{r+1} = (c_{r+1} + \sum_{k=1}^{r} u_k^{(r)} c_{r+1-k})\sigma_r^{-2}$
    $u_k^{(r+1)} = u_k^{(r)} - k_{r+1}u_{r+1-k}^{(r)}$   for  $k = 1, 2, \ldots, r$
    $u_{r+1}^{(r+1)} = -k_{r+1}$
    $\sigma_{r+1}^2 = (1-k_{r+1}^2)\sigma_r^2$
    $\boldsymbol{W}^{(0)}_{r+2,r+2} = \sigma_{r+1}^{-1}$
    $\boldsymbol{W}^{(0)}_{r+2,r+2-k} = u_k^{(r+1)}\sigma_{r+1}^{-1}$   for  $k = 1, 2, \ldots, r+1$
end
$k_V = (c_V + \sum_{k=1}^{V-1} u_k^{(V-1)} c_{V-k})\sigma_{V-1}^{-2}$
$u_k^{(V)} = u_k^{(V-1)} - k_V u_{V-k}^{(V-1)}$   for  $k = 1, 2, \ldots, V-1$
$u_V^{(V)} = -k_V$
$\sigma_V^2 = (1-k_V^2)\sigma_{V-1}^2$
$h_0^{(V)} = \sigma_V^{-1}$
$h_k^{(V)} = u_k^{(V)}\sigma_V^{-1}$   for  $k = 1, 2, \ldots, V$
construct matrix $\boldsymbol{W}^{(1)}$ and $\boldsymbol{W}^{(2)}$ using $h_k^{(V)}$, $k = 0, 1, \ldots, V$
```

Algorithm 4.3: Computation of the filter matrices for the filtering of the warm-up of the design matrix.

input	$\boldsymbol{W}^{(0)}, \boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}$	filter matrices for the warm-up
	$\boldsymbol{A}^{(0)}, \boldsymbol{A}^{(0)}, \ldots, \boldsymbol{A}^{(U-1)}$	blocks of the designmatrix
output	$\boldsymbol{N}^{(w)}$	warm-up part of normal equation matrix

initialization

```
$\boldsymbol{M}^{(1)} = \boldsymbol{A}^{(0)}$
$\boldsymbol{M}^{(2)} = \boldsymbol{A}^{(0)}$
$\boldsymbol{M}^{(1)} = \boldsymbol{W}^{(0)}\boldsymbol{M}^{(1)}$    with BLAS routine dtrmm
$\boldsymbol{N} = (\boldsymbol{M}^{(1)})^\top \boldsymbol{M}^{(1)}$
```

loop

```
for $k = 0, 1, \ldots, U-2$
    if  mod $(k, 2) = 0$
        $\boldsymbol{M}^{(1)} = \boldsymbol{A}^{(k)}$
        $\boldsymbol{M}^{(1)} = \boldsymbol{W}^{(2)}\boldsymbol{M}^{(1)}$    with BLAS routine dtrmm
        $\boldsymbol{M}^{(2)} = \boldsymbol{W}^{(1)}\boldsymbol{M}^{(2)}$    with BLAS routine dtrmm
        $\boldsymbol{M}^{(2)} = \boldsymbol{M}^{(1)} + \boldsymbol{M}^{(2)}$
        if $k = U-2$ and $UV \neq R$, then set $\boldsymbol{M}^{(2)}_{V-UV+R+1:V} = \boldsymbol{0}$
        $\boldsymbol{N}^{(w)} = \boldsymbol{N}^{(w)} + (\boldsymbol{M}^{(2)})^\top \boldsymbol{M}^{(2)}$
    else
        $\boldsymbol{M}^{(2)} = \boldsymbol{A}^{(k)}$
        $\boldsymbol{M}^{(1)} = \boldsymbol{W}^{(1)}\boldsymbol{M}^{(1)}$    with BLAS routine dtrmm
        $\boldsymbol{M}^{(2)} = \boldsymbol{W}^{(2)}\boldsymbol{M}^{(2)}$    with BLAS routine dtrmm
        $\boldsymbol{M}^{(1)} = \boldsymbol{M}^{(1)} + \boldsymbol{M}^{(2)}$
        if $k = U-2$ and $UV \neq R$, then set $\boldsymbol{M}^{(1)}_{V-UV+R+1:V} = \boldsymbol{0}$
        $\boldsymbol{N}^{(w)} = \boldsymbol{N}^{(w)} + (\boldsymbol{M}^{(1)})^\top \boldsymbol{M}^{(1)}$
    end
end
```

Algorithm 4.4: Filtering of the warm-up of the design matrix for one data segment.

input	$c_0, c_1, \ldots, c_R$	autocovariance secquence of the process and the inverse filter, respectively
	$\boldsymbol{y}_{1:R}$	first R elements of vector $\boldsymbol{y}$ according to equation (4.49)
output	$\boldsymbol{z}$	vector according to equation (4.55)
	$\boldsymbol{g}$	vector according to equation (4.55)

```
initialization

$\boldsymbol{z} = \mathbf{0}$
$\sigma_0^2 = c_0$
$h_0^{(0)} = \sigma_0^{-1}$
$\boldsymbol{g}_1 = h_0^{(0)} \boldsymbol{y}_1$
$\boldsymbol{z}_1 = h_0^{(0)} \boldsymbol{g}_1$
$k_1 = c_1 \sigma_0^{-2}$
$u_1^{(1)} = -k_1$
$\sigma_1^2 = (1 - k_1^2)\sigma_0^2$
$h_0^{(1)} = \sigma_1^{-1}$
$h_1^{(1)} = u_1^{(1)} \sigma_1^{-1}$
$\boldsymbol{g}_2 = h_0^{(1)} \boldsymbol{y}_2 + h_1^{(1)} \boldsymbol{y}_1$
$\boldsymbol{z}_{1:2} = \boldsymbol{z}_{1:2} + \boldsymbol{g}_2 \begin{bmatrix} h_1^{(1)} & h_0^{(1)} \end{bmatrix}^\top$

loop

for $r = 1, 2, \ldots, R-2$
    $k_{r+1} = (c_{r+1} + \sum_{k=1}^{r} u_k^{(r)} c_{r+1-k}) \sigma_r^{-2}$
    $u_k^{(r+1)} = u_k^{(r)} - k_{r+1} u_{r+1-k}^{(r)}$    for $k = 1, 2, \ldots, r$
    $u_{r+1}^{(r+1)} = -k_{r+1}$
    $\sigma_{r+1}^2 = (1 - k_{r+1}^2)\sigma_r^2$
    $h_0^{(r+1)} = \sigma_{r+1}^{-1}$
    $h_k^{(r+1)} = u_k^{(1)} \sigma_{r+1}^{-1}$    for $k = 1, 2, \ldots, r+1$
    $\boldsymbol{g}_{r+2} = \sum_{k=0}^{r+1} h_k^{(r+1)} \boldsymbol{y}_{r+2-k}$
    $\boldsymbol{z}_{1:r+2} = \boldsymbol{z}_{1:r+2} + \boldsymbol{g}_{r+2} \begin{bmatrix} h_{r+1}^{(r+1)} & h_r^{(r+1)} & \ldots & h_0^{(r+1)} \end{bmatrix}^\top$
end
```

Algorithm 4.5: The filtering of the warm-up within iterative least-squares adjustment procedures using the direct approach in equation (3.142).

input	$a_0^{(s)}, \ldots, a_P^{(s)}$	filter coefficients of sections $s = 1, \ldots, S$
	$b_0^{(s)}, \ldots, b_P^{(s)}$	filter coefficients of sections $s = 1, \ldots, S$
	$c_0, c_1, \ldots, c_R$	autocovariance secquence of the process and the inverse filter, respectively
	$\boldsymbol{y}$	vector according to equation (4.49)
	R	length of warm-up
	L	block size for algorithm 3.2
output	$\boldsymbol{y}$	filtered vector according to equation (4.49)
	$\boldsymbol{g}$	decorrelated vector

```
filtering
$\boldsymbol{y}^{(\text{copy})} = \boldsymbol{y}_{1:R}$
execute algorithm 4.5, whereas $\boldsymbol{y}^{(\text{copy})}$ is the input and $\boldsymbol{z}^{(\text{copy})}$, $\boldsymbol{g}^{(\text{copy})}$ are the output
$\boldsymbol{g}_{1:R} = \boldsymbol{g}^{(\text{copy})}$
execute algorithm 3.2 to filter $\boldsymbol{y}$ in-place
$\boldsymbol{g}_{R+1:N} = \boldsymbol{y}_{R+1:N}$
$\boldsymbol{y}_{1:R} = \mathbf{0}$    apply matrix $\boldsymbol{S}$
$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_N & \boldsymbol{y}_{N-1} & \cdots & \boldsymbol{y}_1 \end{bmatrix}^\top$    flip vector $\boldsymbol{y}$
execute algorithm 3.2 to filter $\boldsymbol{y}$ in-place
$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_N & \boldsymbol{y}_{N-1} & \cdots & \boldsymbol{y}_1 \end{bmatrix}^\top$    flip vector $\boldsymbol{y}$
$\boldsymbol{y}_{1:R} = \boldsymbol{y}_{1:R} + \boldsymbol{y}^{(\text{copy})}$
```

Algorithm 4.6: The filtering within iterative least-squares adjustment procedures applying algorithm 3.2 for the normal filtering.

input	$c_0,\, c_1,\, \ldots,\, c_R$	ACF of the filter
output	$\boldsymbol{N}^{(1)}$	first part of the normal equation matrix

initialization

```
$\sigma_0^2 = c_0$
$k_1 = c_1 \sigma_0^{-2}$
$u_1^{(1)} = -k_1$
$\sigma_1^2 = (1 - k_1^2)\sigma_0^2$
if $\boldsymbol{d}_k = 2$
    $\boldsymbol{N}^{(1)}_{1,1} = \sigma_1^{-2}$
end
```

loop

```
for $r = 1,\, 2,\, \ldots,\, R-2$
    $k_{r+1} = (c_{r+1} + \sum_{k=1}^{r} u_k^{(r)} c_{r+1-k})\sigma_r^{-2}$
    $u_k^{(r+1)} = u_k^{(r)} - k_{r+1} u_{r+1-k}^{(r)}$   for $k = 1,\, 2,\, \ldots,\, r$
    $u_{r+1}^{(r+1)} = -k_{r+1}$
    $\sigma_{r+1}^2 = (1 - k_{r+1}^2)\sigma_r^2$
    $\boldsymbol{h} = \sigma_{r+1}^{-1} \begin{bmatrix} u_{r+1}^{(r+1)} & \cdots & u_1^{(r+1)} & 1 \end{bmatrix}^\top$
    for $k = 1,\, 2,\, \ldots,\, r+2$
        if $\boldsymbol{d}_k > r+2$
            break
        end
        $\boldsymbol{h}_k = \boldsymbol{h}_{\boldsymbol{d}_k}$
    end
    $\boldsymbol{N}^{(1)}_{1:k-1,1:k-1} = \boldsymbol{N}^{(1)}_{1:k-1,1:k-1} + \boldsymbol{h}_{1:k-1}\boldsymbol{h}_{1:k-1}^\top$
end
```

Algorithm 4.7: The Levinson-Durbin algorithm extended to compute the normal equation matrix for short data gaps.

input	$\boldsymbol{h} = \begin{bmatrix} h_0^{(R)} & \cdots & h_R^{(R)} \end{bmatrix}^\top$	MA filter coefficients of order R
	$\boldsymbol{c} = \begin{bmatrix} c_0 & \cdots & c_R \end{bmatrix}^\top$	ACF of the MA filter according to equation (4.78)
output	$\boldsymbol{N}^{(2)}$	second part of the normal equation matrix (shell storage sheme)

Computation of the elements of the normal equations - second part

```
$k = 0$
for $j = 1, 2, \ldots, D$
    for $i = 1, 2, \ldots, j$
        if $\boldsymbol{d}_j \leq R$
            $k = k + 1$
            if $\boldsymbol{d}_i \leq N - R$
                $\boldsymbol{N}_k^{(2)} = \boldsymbol{h}_{R-\boldsymbol{d}_i+2:R+1}^\top \boldsymbol{h}_{R-\boldsymbol{d}_j+2:R-\boldsymbol{d}_j+\boldsymbol{d}_i+1}$
            else
                $\boldsymbol{N}_k^{(2)} = \boldsymbol{h}_{R-\boldsymbol{d}_i+2:N-\boldsymbol{d}_i+1}^\top \boldsymbol{h}_{R-\boldsymbol{d}_j+2:N-\boldsymbol{d}_j+1}$
            end
        else
            if $\boldsymbol{d}_i \leq N - R$
                if $\boldsymbol{d}_i + R \leq \boldsymbol{d}_j$
                    $k = k + 1$
                    $\boldsymbol{N}_k^{(2)} = \boldsymbol{c}_{\boldsymbol{d}_j-\boldsymbol{d}_i+1}$
                end
            else
                $k = k + 1$
                if $N - R < R$
                    $\boldsymbol{N}_k^{(2)} = \boldsymbol{h}_{1+\boldsymbol{d}_j-\boldsymbol{d}_i:2N-2R-\boldsymbol{d}_i+1}^\top \boldsymbol{h}_{1:2N-2R-\boldsymbol{d}_j+1}$
                else
                    $\boldsymbol{N}_k^{(2)} = \boldsymbol{h}_{1+\boldsymbol{d}_j-\boldsymbol{d}_i:N-\boldsymbol{d}_i+1}^\top \boldsymbol{h}_{1:N-\boldsymbol{d}_j+1}$
                end
            end
        end
    end
```

Algorithm 4.8: Computation of the normal equation matrix for the fill-in elements of the short data gaps.

input	$\boldsymbol{N}$	normal equation matrix stored in the shell oriented scheme (i.e. $\boldsymbol{N}$ is a vector)
	$\boldsymbol{i}$	index vector of the shell storage scheme
output	$\boldsymbol{N}$	upper triangular matrix of Cholesky decomposition stored in the shell oriented scheme

Cholesky decomposition

```
for $j = 1, 2, \ldots, M$
    $j^{(\text{first})} = j - \boldsymbol{i}_{j+1} + \boldsymbol{i}_j + 1$
    if $j^{(\text{first})} < j$
        $k = \boldsymbol{i}_j + 1$
        $\boldsymbol{N}_k = \boldsymbol{N}_k \,/\, \boldsymbol{N}_{\boldsymbol{i}_{j^{(\text{first})}+1}}$
        for $i = j^{(\text{first})} + 1, j^{(\text{first})} + 2, \ldots, j - 1$
            $k = k + 1$
            $i^{(\text{first})} = i - \boldsymbol{i}_{i+1} + \boldsymbol{i}_i + 1$
            if $i^{(\text{first})} < i$
                if $i^{(\text{first})} < j^{(\text{first})}$
                    $k^{(\text{start})} = \boldsymbol{i}_{i+1} - i + j^{(\text{first})}$
                    $k^{(\text{diff})} = i - j^{(\text{first})}$
                else
                    $k^{(\text{start})} = \boldsymbol{i}_{i+1} - i + i^{(\text{first})}$
                    $k^{(\text{diff})} = i - i^{(\text{first})}$
                end
                $\boldsymbol{N}_k = \boldsymbol{N}_k - \boldsymbol{N}^\top_{k^{(\text{start})}:k^{(\text{start})}+k^{(\text{diff})}-1} \boldsymbol{N}_{k-k^{(\text{diff})}:k-1}$
            end
            $\boldsymbol{N}_k = \boldsymbol{N}_k \,/\, \boldsymbol{N}_{\boldsymbol{i}_{i+1}}$
        end
        $k = k + 1$
        $k^{(\text{diff})} = j - j^{(\text{first})}$
        $\boldsymbol{N}_k = \boldsymbol{N}_k - \boldsymbol{N}^\top_{k-k^{(\text{diff})}:k-1} \boldsymbol{N}_{k-k^{(\text{diff})}:k-1}$
        if $\boldsymbol{N}_k < \varepsilon$, then abort algorithm (matrix is not positiv definite)
        $\boldsymbol{N}_k = \sqrt{\boldsymbol{N}_k}$
    else
        $k = \boldsymbol{i}_{j+1}$
        if $\boldsymbol{N}_k < \varepsilon$, then abort algorithm (matrix is not positiv definite)
        $\boldsymbol{N}_k = \sqrt{\boldsymbol{N}_k}$
    end
end
```

Algorithm 4.9: Solving normal equations which are stored in the shell oriented scheme (first part).

input	$\boldsymbol{N}$	Cholesky decomposed normal equation matrix (shell storage scheme)
	$\boldsymbol{n}$	right hand side of the normal equations
	$\boldsymbol{i}$	index vector of the shell storage scheme
output	$\boldsymbol{n}$	solution of the normal equations

Forward substitution

```
for $i = 1, 2, \ldots, M$
    if $\boldsymbol{i}_i + 1 < \boldsymbol{i}_{i+1}$
        $\boldsymbol{n}_i = \boldsymbol{n}_i - \boldsymbol{N}^\top_{\boldsymbol{i}_i+1:\boldsymbol{i}_{i+1}-1} \boldsymbol{n}_{i-\boldsymbol{i}_{i+1}+\boldsymbol{i}_i+1:i-1}$
    end
    $\boldsymbol{n}_i = \boldsymbol{n}_i \,/\, \boldsymbol{N}_{\boldsymbol{i}_{i+1}}$
end
```

Backward substitution

```
for $i = M, M-1, \ldots, 1$
    $\boldsymbol{n}_i = \boldsymbol{n}_i \,/\, \boldsymbol{N}_{\boldsymbol{i}_{i+1}}$
    if $\boldsymbol{i}_i + 1 < \boldsymbol{i}_{i+1}$
        $k^{(diff)} = \boldsymbol{i}_{i+1} - \boldsymbol{i}_i - 1$
        $\boldsymbol{n}_{i-k^{(diff)}:i-1} = \boldsymbol{n}_{i-k^{(diff)}:i-1} - \boldsymbol{n}_i$
        $\boldsymbol{N}_{\boldsymbol{i}_i+1:\boldsymbol{i}_i+k^{(diff)}}$
    end
end
```

Algorithm 4.10: Solving normal equations which are stored in the shell oriented scheme (second part).

5. Design of Decorrelating Filters

So far we discussed in chapter 2 how the decorrelating filters are related to the Gauss-Markov model and in chapter 3 how they can be integrated into least-squares adjustment procedures. Within these discussions we assumed that the filter coefficients were given. In this chapter we introduce methods for the determination of the filter coefficients.

The methods for the determination of the filter coefficients can be divided into methods, which work in the time domain and methods, which work in the frequency domain. A very good overview over time frequency methods is given by Pintelon et al. (1994). Time domain methods are described for example by Friedlander and Porat (1984), Schlittgen and Streitberg (2001) and Klees and Broersen (2002). Especially Klees and Broersen (2002) are concerned with the same problems as we are in this thesis. There exist too many methods to introduce all of them in this chapter. Therefore, we describe only a set of methods, which has proven to be useful for the application scenario in chapter 6.

One main feature of the problems we consider is, that the number of observations is large. Otherwise there would be no need for the use of decorrelating filters to represent the covariance matrix of the observation. For this reason we do not have to consider small sample size effects. Thus, many estimation methods turn out to perform equally well, because they have the same asymptotic properties. A special part of the estimation of the filter coefficients is the order selection of MA and ARMA filters.

In the following sections the random variables $\mathcal{X}_n$ are assumed to be correlated while the random variables $\mathcal{Y}_n$ and $\mathcal{Z}_n$ are uncorrelated. As the goal is to design decorrelating filters in the context of least-squares adjustments, $\mathcal{X}_n$ models the stochastic properties of the error vector $\boldsymbol{e}$ in the observation equations (2.4) of the Gauss-Markov model. In practice, the error vector $\boldsymbol{e}$ is not available and the residual vector $\boldsymbol{v}$ is used instead. In this case, the random variables $\mathcal{Y}_n$ model the stochastic properties of the filtered residual vector in the filtered observation equations (3.131), which is verified by equation (3.132).

5.1 Features of the Colored Noise of the GOCE Gradiometer

In this chapter we begin to focus on the application scenario. Though the methods for the estimation of the filter coefficients are described generally, the choice of the methods, which are described in this chapter, is tailored for the application scenario. However, these methods may be applied to any other application scenario as well. In order to select a set of methods we need to investigate the correlation characteristics of the observations. Within the application scenario in chapter 6 the ARMA filters are used to represent the covariance matrix of the satellite gravity gradient (SGG) observations. Because the observations time series l_n is simulated, we have precise knowledge about the deterministic part of the observations. By subtracting the deterministic part, we obtain the noise time series e_n, which represents the stochastic part of the observations l_n. In this sense, the noise time series e_n is also simulated. Figure 5.1 shows the power spectral density of the noise time series e_n.

From 0.005 Hz to 0.1 Hz the power spectral density of the residuals time series is very low. This frequency band is called the measurement bandwidth of the observations. Below and above the measurement bandwidth the noise power increases. In the frequency band from 0 Hz to 0.005 Hz the gain is proportional to $\frac{1}{f^k}$, where $k \approx 2$, which corresponds to brown noise according to (2.85). This noise characteristic is due to the fact that the gradiometer consists of six accelerometers, and that accelerometers typically insensitive to low frequency information. Furthermore, at certain frequencies sharp peaks occur in the power spectral density. From these features we can conclude, that we need the following types of filters:

- high-pass filters to model the gain below the measurement bandwidth,
- notch-filter to model the sharp peaks in the power spectral density and
- general ARMA filters for the remaining overall curve progression of the power spectral density.

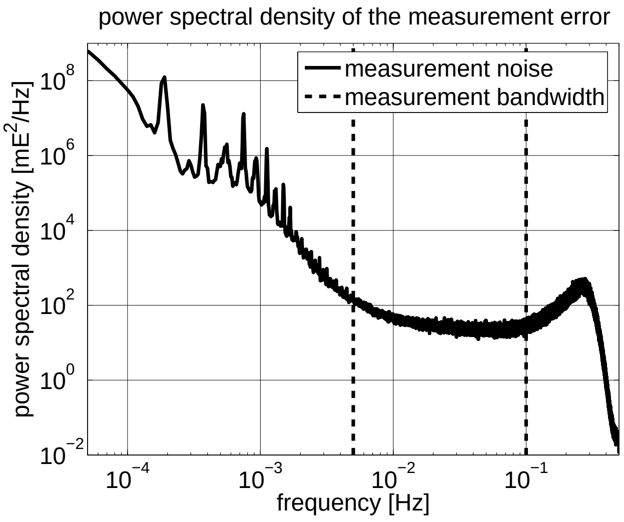


Figure 5.1: Power spectral density of the noise time series of the ZZ component of the SGG observations estimated by Welch's method (cf. WELCH 1967). The noise time series results from the simulated observation time series described in chapter 6 by subtracting the know deterministic part of the observations. The power spectral density is characterized by its measurement bandwidth from 0.005 Hz to 0.1 Hz, wherein the power is low. Below the measurement bandwidth the power increases inverse proportional to frequency. The power spectral density also contains sharp peaks below the measurement bandwidth. Note, that both axes are scaled logarithmically. If the axis were scaled linearly, then there would be a sharp edge at 0.005 Hz.

5.2 Estimation of a General MA Filter

For the estimation of a general MA filter, we consider the corresponding AR process

$$\mathcal{Z}_n = \mathcal{X}_n + \sum_{k=1}^{R} u_k \mathcal{X}_{n-k} \quad \text{with } \boldsymbol{\mathcal{Z}} \sim N(\mathbf{0}, \sigma^2 \boldsymbol{I}). \tag{5.1}$$

After defining

$$\boldsymbol{u} = \begin{bmatrix} u_1 & u_2 & \cdots & u_R \end{bmatrix}^\top \tag{5.2}$$

and

$$\boldsymbol{\mathcal{X}}_{n-1} = \begin{bmatrix} \mathcal{X}_{n-1} & \mathcal{X}_{n-2} & \cdots & \mathcal{X}_{n-R} \end{bmatrix}^\top, \tag{5.3}$$

equation (5.1) becomes

$$\mathcal{Z}_n = \mathcal{X}_n + \boldsymbol{u}^\top \boldsymbol{\mathcal{X}}_{n-1} \quad \text{with } \boldsymbol{\mathcal{Z}} \sim N(\mathbf{0}, \sigma^2 \boldsymbol{I}). \tag{5.4}$$

The parameters $u_1, \ldots, u_R$ of the AR process are determined by minimizing the variance $\Sigma\{\mathcal{Z}_n\}$ of $\mathcal{Z}_n$. Because the $E\{\mathcal{Z}_n\} = 0$ according to (5.1), we find for the variance

$$\Sigma\{\mathcal{Z}_n\} = E\{(\mathcal{Z}_n - E\{\mathcal{Z}_n\})^2\} = E\{\mathcal{Z}_n^2\}. \tag{5.5}$$

Therefore, we have to minimize $E\{\mathcal{Z}_n^2\}$. Substituting (5.4) for $\mathcal{Z}_n$ zields

$$E\{\mathcal{Z}_n^2\} = E\{(\mathcal{X}_n + \boldsymbol{u}^\top \boldsymbol{\mathcal{X}}_{n-1})^2\} = E\{\mathcal{X}_n^2\} + 2\boldsymbol{u}^\top \boldsymbol{E}\{\mathcal{X}_n \boldsymbol{\mathcal{X}}_{n-1}\} + \boldsymbol{u}^\top \boldsymbol{E}\{\boldsymbol{\mathcal{X}}_{n-1} \boldsymbol{\mathcal{X}}_{n-1}^\top\} \boldsymbol{u} \tag{5.6}$$

The minimum of $E\{\mathcal{Z}_n^2\}$ is found by setting the derivative to zero.

$$\frac{\partial E\{\mathcal{Z}_n^2\}}{\partial \boldsymbol{u}} = 2\boldsymbol{E}\{\mathcal{X}_n \boldsymbol{\mathcal{X}}_{n-1}\} + 2\boldsymbol{E}\{\boldsymbol{\mathcal{X}}_{n-1} \boldsymbol{\mathcal{X}}_{n-1}^\top\} \boldsymbol{u} = 0 \tag{5.7}$$

Because we assume that $\mathcal{X}_n$ is stationary, it follows that $E\{\mathcal{X}_n\}$ has the same value for all n. We also know that $E\{\mathcal{Z}_n\} = 0$ according to equation (5.1). Then it follows from equation (5.1) that $E\{\mathcal{X}_n\} = 0$. Therefore $E\{(\mathcal{X}_n - E\{x\}_n)(\mathcal{X}_{n+k} - E\{\mathcal{X}_{n+k}\})\} = E\{\mathcal{X}_n\mathcal{X}_{n+k}\} = \gamma_k$ is the autocovariance function of $\mathcal{X}_n$. Thus, we find

$$\boldsymbol{E}\{\boldsymbol{\mathcal{X}}_{n-1}\boldsymbol{\mathcal{X}}_{n-1}^\top\} = \begin{bmatrix} E\{\mathcal{X}_{n-1}^2\} & \cdots & E\{\mathcal{X}_{n-1}\mathcal{X}_{n-R}\} \\ \vdots & & \vdots \\ E\{\mathcal{X}_{n-R}\mathcal{X}_{n-1}\} & \cdots & E\{\mathcal{X}_{n-R}^2\} \end{bmatrix} = \begin{bmatrix} \gamma_0 & \cdots & \gamma_{R-1} \\ \vdots & & \vdots \\ \gamma_{R-1} & \cdots & \gamma_0 \end{bmatrix} \tag{5.8}$$

and

$$\boldsymbol{E}\{\mathcal{X}_n\boldsymbol{\mathcal{X}}_{n-1}\} = \begin{bmatrix} E\{\mathcal{X}_n\mathcal{X}_{n-1}\} \\ \vdots \\ E\{\mathcal{X}_n\mathcal{X}_{n-R}\} \end{bmatrix} = \begin{bmatrix} \gamma_1 \\ \vdots \\ \gamma_R \end{bmatrix}. \tag{5.9}$$

In practice we can use

$$c_k = \frac{1}{N-k}\sum_{n=k}^{N-1} x_n x_{n-k} \tag{5.10}$$

as an estimation for γ_k. Note, that the true mean of $\mathcal{X}_n$ is zero and therefore known, which affects the denominator of the fraction. Because $\sum_{n=k}^{N-1} x_n x_{n-k}$ corresponds to a correlation of the time series x_n with itself, we can compute the sum for $k = 0, \ldots, R$ efficiently applying FFT techniques (cf. Brigham 1988, pp. 65–69). Of course, we could use other estimators for the autocovariance function. We define

$$\boldsymbol{C} = \begin{bmatrix} c_0 & \cdots & c_R \\ \vdots & & \vdots \\ c_R & \cdots & c_0 \end{bmatrix} \quad \text{and} \quad \boldsymbol{c} = \begin{bmatrix} c_1 \\ \vdots \\ c_R \end{bmatrix}. \tag{5.11}$$

After substituting $\boldsymbol{C}$ and $\boldsymbol{c}$ into (5.7), we find

$$2\boldsymbol{C}\boldsymbol{u} + 2\boldsymbol{c} = 0 \quad \Longleftrightarrow \boldsymbol{C}\boldsymbol{u} = -\boldsymbol{c}. \tag{5.12}$$

This system of equations can be solved recursively by the Levinson-Durbin algorithm 5.1 (cf. Farhang-Boroujeny 1998, pp.357–377), because matrix $\boldsymbol{C}$ has a Toeplitz structure. The algorithm also provides the variance of $\mathcal{Z}_n$

$$\sigma_R^2 = E\{\mathcal{Z}_n^2\} = E\{(\mathcal{X}_n + \sum_{k=1}^{R} u_k\mathcal{X}_{n-k})^2\}, \tag{5.13}$$

which depends on the order R of the AR process. The recursive computation of the Levinson-Durbin algorithm 5.1 starts with the order $r = 0$ and ends with the order $r = R$. The variances $\sigma^{2^{(r)}}$ for $r = 0, 1, \ldots, R$ are also computed within algorithm and can be used for order selection. Order selection means to choose the order R of the AR process as small as possible under the condition that the time series $\mathcal{X}_n$ is correctly modeled by the AR process. For this task we could use information criteria such as the Akaike criterion or the Hannan-Quinn criterion (cf. Stier 2001 and Ng and Perron 2003). However, the order selection based on information criteria should only be used as an indication for the correct order. The reason for this is, that we need a very good approximation of the power spectral density, where it is small, while in regions, where the power spectral density is high, the approximation is often not that much important. This is at least the case for the application scenario of this thesis. Therefore, we should check the power spectral density of the AR process manually, whether the order of the AR process was chosen correctly.

After choosing the order R of the filter, we can convert the process parameters $b_1, \ldots, b_R$ into the filter coefficients $h_0, \ldots, h_R$ of the MA filter in equation (2.59). This is motivated by equation (5.1), which states that $\Sigma\{\mathcal{Z}_n\} = \sigma^2$. If R is chosen correctly, then $\sigma_R \approx \sigma$. After dividing (5.1) by σ_R, we find

$$\mathcal{Y}_n = \frac{\mathcal{Z}_n}{\sigma_R}, \tag{5.14}$$

where $\Sigma\{\mathcal{Y}_n\} = 1$. This division leads to the filter coefficients

$$h_k = \begin{cases} \frac{1}{\sigma_R}, & \text{if } k = 0 \\ \frac{b_k}{\sigma_R}, & \text{if } k > 0 \end{cases}. \tag{5.15}$$

This way the filter produces a normalized output such that the variance of the filtered time series is equal to one.

input	$c_0, c_1, \ldots, c_R$	autocovariance secquence
output	$u_1^{(R)}, u_2^{(R)}, \ldots, u_R^{(R)}$	MA filter coefficients
	$\sigma_0^2, \sigma_1^2, \ldots, \sigma_R^2$	variances of the MA filters of order 0, 1, …, R

```
initialization

$\sigma_0^2 = c_0$
$k_1 = c_1\sigma_0^{-2}$
$u_1^{(1)} = -k_1$
$\sigma_1^2 = (1 - k_1^2)\sigma_0^2$

loop

for $r = 1, 2, \ldots, R-1$
    $k_{r+1} = (c_{r+1} + \sum_{k=1}^{r} u_k^{(r)} c_{r+1-k})\sigma_r^{-2}$
    $u_k^{(r+1)} = u_k^{(r)} - k_{r+1}u_{r+1-k}^{(r)}$   for $k = 1, 2, \ldots, r$
    $u_{r+1}^{(r+1)} = -k_{r+1}$
    $\sigma_{r+1}^2 = (1 - k_{r+1}^2)\sigma_r^2$
end
```

Algorithm 5.1: The Levinson-Durbin algorithm.

5.3 Estimation of a General ARMA Filter

The estimation method described by Durbin (1960) is originally designed to obtain starting values for a subsequent better estimation method. However, due to the very long length of the time series the coefficients estimated by this method are already very good. For this reason we describe a very simple estimation method, which performs well for the application scenario in chapter 6. The basis for the estimation method is the ARMA process equation

$$\mathcal{Z}_n = \mathcal{X}_n + \sum_{k=1}^{P} c_k \mathcal{X}_{n-k} + \sum_{j=1}^{Q} a_j \mathcal{Z}_{n-j}. \tag{5.16}$$

In order to estimate the a_j and c_k we insert the time series x_n and z_n for the random variables $\mathcal{X}_n$ and $\mathcal{Z}_n$.

$$z_n = x_n + \sum_{k=0}^{P} c_k x_{n-k} + \sum_{j=1}^{Q} a_j z_{n-j} \tag{5.17}$$

Because the time series z_n can only be calculated if the filter coefficients a_j and c_k are given, we do not know the values z_{n-j} on the right hand side. To overcome this problem we make use of the fact, that stationary ARMA processes can be approximated by long AR processes. We estimate an MA filter according to the Levinson-Durbin algorithm 5.1 and then filter the time series x_n to obtain the time series $\widehat{z}_n$ as an approximation of the time series z_n.

$$z_n = x_n + \sum_{k=1}^{\infty} u_k x_{n-k} \approx x_n + \sum_{k=1}^{R} u_k x_{n-k} = \widehat{z}_n \tag{5.18}$$

Then, we substitute the time series $\widehat{z}_{n-j}$ in equation (5.17) for z_{n-j} and find

$$z_n = x_n + \sum_{k=0}^{P} c_k x_{n-k} + \sum_{j=1}^{Q} a_j \widehat{z}_{n-j}. \tag{5.19}$$

The parameters a_j and c_k are estimated by minimizing

$$\sum_{n=0}^{N-1} z_n^2, \tag{5.20}$$

which corresponds to a least squares adjustment. Note, that the $\mathcal{Z}_n$ are assumed to be uncorrelated with equal variance. Within the least squares adjustment we find for the parameter vector, the observation vector and the design matrix

$$\boldsymbol{x} = \begin{bmatrix} c_1 \\ \vdots \\ c_Q \\ a_1 \\ \vdots \\ a_P \end{bmatrix}, \quad \boldsymbol{l} = \begin{bmatrix} -x_1 \\ -x_2 \\ \vdots \\ -x_N \end{bmatrix} \quad \text{and} \quad \boldsymbol{A} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ x_1 & 0 & \cdots & 0 & \widehat{z}_1 & 0 & \cdots & 0 \\ x_2 & x_1 & \ddots & \vdots & \widehat{z}_2 & \widehat{z}_1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 & \vdots & \vdots & \ddots & 0 \\ x_P & x_{P-1} & \cdots & x_1 & \widehat{z}_Q & \widehat{z}_{Q-1} & \cdots & \widehat{z}_1 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ x_{N-1} & x_{N-2} & \cdots & x_{N-P} & \widehat{z}_{N-1} & \widehat{z}_{N-2} & \cdots & \widehat{z}_{N-Q} \end{bmatrix}. \tag{5.21}$$

Because of the truncation of the filter coefficients at the beginning of the filtering, the first R rows of the design matrix $\boldsymbol{A}$ should be set to zero, whereas R is the order of the long AR process. The easiest way to compute the least squares estimates is to directly compute the normal equation matrix $\boldsymbol{N} = \boldsymbol{A}^\top \boldsymbol{A}$ and the right hand side of the normal equations $\boldsymbol{n} = \boldsymbol{A}^\top \boldsymbol{l}$. After computing the least squares estimates, we only have to convert the the filter coefficients c_k to obtain the filter according to equation (2.53). The first step of the conversion is to determine the variance of the time series z_n.

$$\sigma^2 = \frac{1}{N-R} \sum_{n=R}^{N} z_n^2 \tag{5.22}$$

The variance σ^2 is then used for a normalization, such that the variance of the filtered time series y_n in (2.53) is equal to one. Thus, the coefficients b_k are related to the coefficients c_k by

$$b_0 = \frac{1}{\sigma} \quad \text{and} \quad b_k = \frac{c_k}{\sigma} \quad \text{for } k = 1, \ldots, P. \tag{5.23}$$

The coefficients a_j in (2.53) and (5.17) are the same. The time series z_n is therefore a scaled version of the time series y_n.

Like for the MA filters we could perform order selection for ARMA filters based on information criteria. However, a better criterion is the quality of the decorrelating capabilities of the filter, which can be checked by investigating the power spectral density of the filter.

5.4 Design of Highpass Filters

There exist many different kinds of highpass filters. The most useful one for the application scenario depicted in chapter 6 is the Butterworth filter, because its power spectral density is monotonic in the stop- and passband (cf. Oppenheim and Schafer 1999, p. 1030). Its transfer function is defined by the power spectral density

$$|H_c(s)|^2 = H_c(s)H_c(-s) = \frac{1}{1+(\frac{s}{i\Omega_c})^{2N}} \tag{5.24}$$

of the filter, whereas s is a complex variable, N is the filter order which controls the steepness of the power spectral density $|H_c(i\Omega)|^2$ in the transition range from the stopband to the passband and Ω_c is the stop frequency, defining the half power point of the power spectral density. The subscript c indicates, that $H_c(s)$ is the transfer function of the continuous filter, which we have to convert into a digital filter. The parameters N and Ω_c can be chosen by trial and error. Good approximate values are identified, if the power spectral density of the Butterworth filter fits the power spectral density of the colored noise, which we want to decorrelate. There is no need to estimate the parameters N and Ω_c, because the insufficiencies of the highpass filter design are compensated by the estimation of the general ARMA filter. The poles of the transfer function are

$$s_k = (-1)^{\frac{1}{2N}}(i\Omega_c) = \Omega_c e^{\frac{i\pi}{2N}(2k+N-1)}, \quad k = 0, 1, \ldots, 2N-1. \tag{5.25}$$

The poles are lie equidistant on a circle in the s-plane with radius Ω_c and are located symmetrically to the imaginary axis of the s-plane (cf. Oppenheim and Schafer 1999, pp. 508, 1031).

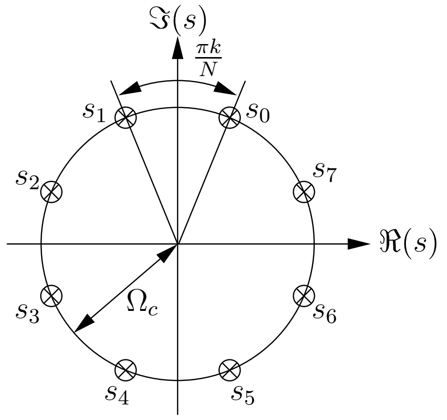


Figure 5.2: Poles s_k of the power spectral density of a Butterworth filter. The order of the filter is $N = 4$. The poles are located equidistant on a circle with radius Ω_c and lie always symmetrically to the imaginary axis of the s-plane. In order to obtain a stable filter, we have to select the poles on the left side of the imaginary axis, i.e. s_1, s_2, s_3 and s_4.

From the location of the poles it follows, that there are either N complex conjugate pairs of poles, if N is even, or $N-1$ complex conjugate poles and two real poles, if N is odd. In order to obtain a stable filter, we need to choose those poles, which are located in the left half of the s-plane. The reason for this is, that the left half of the s-plane is mapped to the inside of the unit circle in the z-plane. Thus, the transfer function of the Butterworth filter reads

$$H_c(s) = \frac{1}{\prod\limits_{k=1}^{N}(1-s_k s^{-1})} = \prod_{k=1}^{N}(1-s_k s^{-1})^{-1}. \tag{5.26}$$

After combining the complex conjugate pairs of poles, we find

$$(1-s_k s^{-1})(1-s_k^* s^{-1}) = 1-(s_k+s_k^*)s^{-1} + s_k s_k^* s^{-2} = 1-2\Re(s_k)s^{-1} + |s_k|^2 s^{-2}. \tag{5.27}$$

Substituting (5.27) into (5.26) yields

$$H_c(s) = \begin{cases} \prod_{k=1}^{N/2} (1 - 2\Re(s_k)s^{-1} + |s_k|^2 s^{-2})^{-1}, & \text{if } N \text{ even} \\ (1 - \Re(s_k)s^{-1})^{-1} \prod_{k=1}^{(N-1)/2} (1 - 2\Re(s_k)s^{-1} + |s_k|^2 s^{-2})^{-1}, & \text{if } N \text{ odd} \end{cases}, \tag{5.28}$$

where

$$\Re(s_k) = \Omega_c \cos(\frac{\pi}{2N}(2k + N - 1)) \quad \text{and} \quad |s_k|^2 = \Omega_c^2. \tag{5.29}$$

With the help of the bilinear transform (cf. Oppenheim and Schafer 1999)

$$s = 2\frac{1 - z^{-1}}{1 + z^{-1}}, \tag{5.30}$$

we can convert the continuous filter into a digital filter. The transfer function of the digital filter and the continuous filter are related by

$$H(z) = H_c(s) = H_c(2\frac{1 - z^{-1}}{1 + z^{-1}}). \tag{5.31}$$

(cf. Oppenheim and Schafer 1999, p. 501). The poles s_k in (5.26) are also translated by means of the bilinear transform.

$$s_k = 2\frac{1 - z_k^{-1}}{1 + z_k^{-1}} \tag{5.32}$$

In order to derive the filter coefficients in dependence of the parameters of the Butterworth filter, we factorize the filter into first and section order sections (cf. section 3.2). This is already performed in equation (5.28). For first order sections we find

$$H_k(z) = \frac{1}{1 - \Re(s_k)\frac{1}{2}\frac{1+z^{-1}}{1-z^{-1}}} = \frac{2(1 - z^{-1})}{2(1 - z^{-1}) - \Re(s_k)(1 + z^{-1})} = \frac{2 - 2z^{-1}}{(2 - \Re(s_k)) - (2 + \Re(s_k))z^{-1}}. \tag{5.33}$$

By comparing this formula with the transfer function in equation (2.62) we obtain the filter coefficients

$$b_0 = \frac{2}{2 - \Re(s_k)}, \quad b_1 = \frac{-2}{2 - \Re(s_k)} \quad \text{and} \quad a_1 = \frac{2 + \Re(s_k)}{2 - \Re(s_k)} \tag{5.34}$$

of the first order section of the digital filter. For second order sections we find

$$H_k(z) = \frac{1}{1 - 2\Re(s_k)\frac{1}{2}\frac{1+z^{-1}}{1-z^{-1}} + |s_k|^2\frac{1}{4}\frac{(1+z^{-1})^2}{(1-z^{-1})^2}} \tag{5.35}$$

$$= \frac{4(1 - z^{-1})^2}{4(1 - z^{-1})^2 - 4\Re(s_k)(1 + z^{-1})(1 - z^{-1}) + |s_k|^2(1 + z^{-1})^2} \tag{5.36}$$

$$= \frac{4 - 8z^{-1} + 4z^{-2}}{(4 - 4\Re(s_k) + |s_k|^2) + (-8 + 2|s_k|^2)z^{-1} + (4 + 4\Re(s_k) + |s_k|^2)z^{-2}} \tag{5.37}$$

Here too, by comparing this formula with the transfer function in equation (2.62) we obtain the filter coefficients

$$b_0 = \frac{4}{4 - 4\Re(s_k) + |s_k|^2}, \quad b_1 = \frac{-8}{4 - 4\Re(s_k) + |s_k|^2}, \quad b_2 = \frac{4}{4 - 4\Re(s_k) + |s_k|^2},$$
$$a_1 = \frac{8 - 2|s_k|^2}{4 - 4\Re(s_k) + |s_k|^2} \quad \text{and} \quad a_2 = -\frac{4 + 4\Re(s_k) + |s_k|^2}{4 - 4\Re(s_k) + |s_k|^2} \tag{5.38}$$

of the second order section of the digital filter.

5.5 Design and Estimation of Notch Filters

The transfer function $H(z)$ of a filter can be factorized according to equation (2.102) with respect to its poles and zeros. For better understanding of the design of notch filters we first investigate the influence of the position of the poles and zeros on the power spectral density of filters. We investigate exemplarily the influence of a single zero β. The corresponding factor of the transfer function is then

$$1 - \beta z^{-1} = \frac{z - \beta}{z}. \tag{5.39}$$

The power spectral density $|H(e^{i\omega})|^2$ is obtained by substituting $e^{i\omega}$ for z and squaring each factor of the transfer function. Therefore, we find for the factor of the single zero

$$\left| \frac{e^{i\omega} - \beta}{e^{i\omega}} \right|^2 = |e^{i\omega} - \beta|^2. \tag{5.40}$$

Note, that this is in principle the squared distance of a vector in the z-plane from the zero β to a point $e^{i\omega}$ on the unit circle.

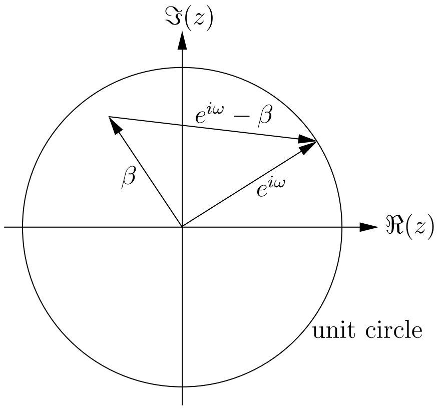


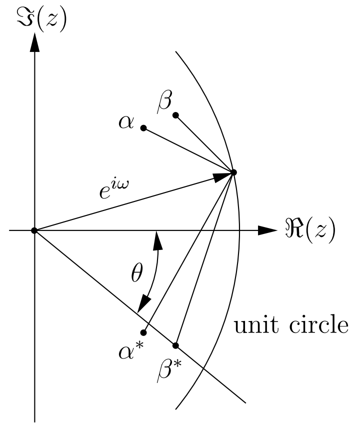
Figure 5.3: Geometrical interpretation of the factors of the transfer function according to equation (2.102). Here, the factor $e^{i\omega} - \beta$ of a zero β is illustrated. The power spectral density of a filter can be expressed by the squared distances of the vectors $e^{i\omega} - \beta_k$ for zeros and $e^{i\omega} - \alpha_k$ for poles, if the equations (2.102) and (2.89) are combined.

After expressing each pole and zero of the transfer function this way, we find for the power spectral density of a filter

$$|H(e^{i\omega})|^2 = b_0^2 \frac{\prod_{k=1}^{P} |e^{i\omega} - \beta_k|^2}{\prod_{j=1}^{Q} |e^{i\omega} - \alpha_j|^2}. \tag{5.41}$$

Thus, the power spectral density of filters is nothing but the ratio of products of squared distances of vectors from the poles and zeros to a point on the unit circle in the z-plane.

With this in mind it is easy to understand the design of notch filters. The name notch filters refers to the shape of their power spectral density. A notch in the power spectral density can be achieved by setting a zero on or close to the unit circle in the z-plane, and placing a pole close to the zero a little nearer to the center of the z-plane. The filter is then complemented by a zero and a pole which are complex conjugates to the first zero and pole. The ratio of the vectors from the zeros and poles will then be approximately equal to one except for the angle $\omega = \sphericalangle(\beta_k)$ of the zero, where a notch in the power spectral density can be found. Therefore, the angle of the zero is called the notch frequency.



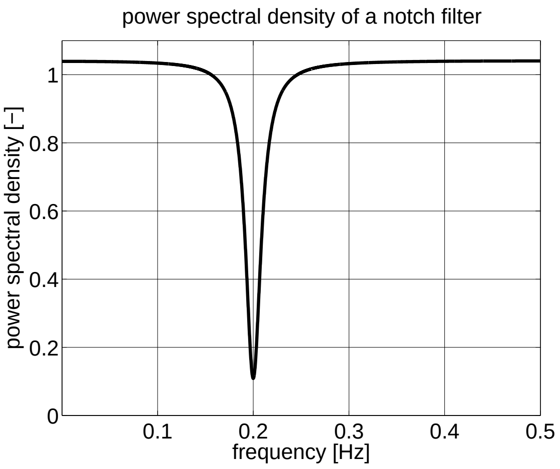


Figure 5.4: Principle design of notch filters. The left panel shows, that the zeros are placed on or close to the unit circle in the z-plane. The poles are located close to the zeros, but a little bit closer to the origin of the z-plane. The notch frequency θ is equal to the angle of the zero β. The right panel shows the power spectral density of a notch filter. The transfer function of the notch filter has been chosen according to equation (5.46). The notch frequency is $\theta = 0.2$ Hz and the bandwidth is $\delta = 0.04$ Hz. The scaling factor m was chosen such that the power spectral density at the notch frequency is equal to 0.1, i.e. $P(\theta) = 0.1$. Note, that θ and δ have to be expressed in radians in equation (5.46), i.e. $\theta = 2\pi \cdot 0.2 = 0.4\pi$ and $\delta = 2\pi \cdot 0.04 = 0.08\pi$.

Such a design is given by Widrow et al. (1975), who propose the following parameterization of the transfer function

$$H(z) = \frac{1 - 2z^{-1}\cos\theta + z^{-2}}{1 - 2(1-\delta)z^{-1}\cos\theta + (1-2\delta)z^{-2}} \tag{5.42}$$

Herein, θ is the notch frequency and δ is band width of the notch, defined by the half power points $|H(e^{i(\theta\pm\delta)})|^2 \approx \frac{1}{2}$. For the poles and zeros of the transfer function we find

$$\alpha = \sqrt{1-2\delta}\cos\theta + i\sqrt{(1-2\delta) - (1-\delta)^2\cos^2\theta}, \quad \beta = e^{i\theta}, \tag{5.43}$$

$$\alpha^* = \sqrt{1-2\delta}\cos\theta - i\sqrt{(1-2\delta) - (1-\delta)^2\cos^2\theta} \quad \text{and} \quad \beta^* = e^{-i\theta}. \tag{5.44}$$

The transfer function itself can then be expressed by its poles and zeros.

$$H(z) = \frac{(1-\beta z^{-1})(1-\beta^* z^{-1})}{(1-\alpha z^{-1})(1-\alpha^* z^{-1})} \tag{5.45}$$

Because the zeros β and β^* in (5.45) lie on the unit circle of the z-plane, the corresponding process of the notch filter is not stationary. Therefore, we introduce an additional parameter m, which simply scales the poles and zeros, such that both lie inside the unit circle. The transfer function $H(z)$ in (5.45) then becomes

$$\begin{aligned} H(z) &= \frac{(1-m\beta z^{-1})(1-m\beta^* z^{-1})}{(1-m\alpha z^{-1})(1-m\alpha^* z^{-1})} \\ &= \frac{1 - m(\beta+\beta^*)z^{-1} + m^2\beta\beta^* z^{-2}}{1 - m(\alpha+\alpha^*)z^{-1} + m^2\alpha\alpha^* z^{-2}} \\ &= \frac{1 - 2m\cos\theta z^{-1} + m^2 z^{-2}}{1 - 2m(1-\delta)\cos\theta z^{-1} + m^2(1-2\delta)z^{-2}}, \end{aligned} \tag{5.46}$$

whereas $m < 1$. Furthermore, we normalize the transfer function such that $H(1) = 1$. This introduces an additional factor in front the transfer function in (5.46).

$$H(z) = \frac{1 + 2m(1-\delta)\cos\theta + m^2(1-2\delta)}{1 + 2m\cos\theta + m^2} \cdot \frac{1 - 2m\cos\theta z^{-1} + m^2 z^{-2}}{1 - 2m(1-\delta)\cos\theta z^{-1} + m^2(1-2\delta)z^{-2}} \tag{5.47}$$

Now that we have identified the parameterization of the transfer function, we have to select a suitable estimation method for the parameters θ, δ and m of the notch filter. The estimation method which best suits the problem of estimating these parameters is described by SIDMAN et al. (1991). Here, the power spectral density $P(\omega) = |H(e^{i\omega})|^2$ of the filter is fitted to the power spectral density $\Phi_{\mathcal{X}\mathcal{X}}(\omega)$ of a correlated noise time series. In a first step, we derive the target function, which is minimized, and highlight its advantage. Then we discuss how to minimize the target function.

First, we consider that the filter can consist of more than one filter section (cf. section 3.2). Combining equation (2.90) and (3.18) yields

$$\Phi_{\mathcal{Y}\mathcal{Y}}(\omega) = |H(e^{i\omega})|^2 \Phi_{\mathcal{X}\mathcal{X}}(\omega) = \Phi_{\mathcal{X}\mathcal{X}}(\omega) \prod_{j=1}^{J} |H_j(e^{i\omega})|^2 \tag{5.48}$$

By extracting the root, logarithmizing and using the process instead of the filter, i.e. $H^{-1}(z)$ instead of $H(z)$, we find

$$\ln \sqrt{\Phi_{\mathcal{Y}\mathcal{Y}}(\omega)} = \ln \sqrt{\Phi_{\mathcal{X}\mathcal{X}}(\omega)} - \sum_{j=1}^{J} \ln |H_j^{-1}(e^{i\omega})| \tag{5.49}$$

For the distribution of $\ln \sqrt{\Phi_{\mathcal{Y}\mathcal{Y}}(\omega_k)}$ we assume

$$\ln \sqrt{\Phi_{\mathcal{Y}\mathcal{Y}}(\omega_k)} \sim N(\mathbf{0}, \boldsymbol{I}), \tag{5.50}$$

which is in a strict sense statistically not correct (cf. SCHLITTGEN and STREITBERG 2001, p. 364). This leads to minimizing the residual sum of squares

$$\sum_{k=1}^{K} \big| \ln \sqrt{\Phi_{\mathcal{Y}\mathcal{Y}}(\omega_k)} \big|^2 \quad \rightarrow \quad \text{Min.} \tag{5.51}$$

The advantage of this procedure is, that not the differences but the logarithmic ratio

$$\big| \ln \sqrt{\Phi_{\mathcal{X}\mathcal{X}}(\omega_k)} - \ln |H(e^{i\omega_k})^{-1}| \big|^2 = \big| \ln \frac{\sqrt{\Phi_{\mathcal{X}\mathcal{X}}(\omega_k)}}{|H^{-1}(e^{i\omega_k})|} \big|^2 \tag{5.52}$$

is minimized. The advantage results from the fact that minimizing the logarithmic ratio provides a good fit of the power spectral density of the filter to the power spectral density of the colored noise, where the power spectral density of the colored noise is low. This means that the power spectral density of the colored noise will be modeled more accurately by the filter for frequencies, which contribute less to the variance in the sense of equation (2.81). It should also be mentioned that for example, if $\Phi_{\mathcal{X}\mathcal{X}}(\omega_k)$ is ten times smaller than $|H^{-1}(e^{i\omega_k})|^2$, this has the same influence as if $\Phi_{\mathcal{X}\mathcal{X}}(\omega_k)$ is ten times greater than $|H^{-1}(e^{i\omega_k})|^2$.

The observation vector and the residual vector of the Gauss-Markov model read

$$\boldsymbol{l} = \begin{bmatrix} \sqrt{\Phi_{\mathcal{X}\mathcal{X}}(\omega_1)} \\ \sqrt{\Phi_{\mathcal{X}\mathcal{X}}(\omega_2)} \\ \vdots \\ \sqrt{\Phi_{\mathcal{X}\mathcal{X}}(\omega_K)} \end{bmatrix} \quad \text{and} \quad \boldsymbol{v} = \begin{bmatrix} -\sqrt{\Phi_{\mathcal{Y}\mathcal{Y}}(\omega_1)} \\ -\sqrt{\Phi_{\mathcal{Y}\mathcal{Y}}(\omega_2)} \\ \vdots \\ -\sqrt{\Phi_{\mathcal{Y}\mathcal{Y}}(\omega_K)} \end{bmatrix}. \tag{5.53}$$

The parameter vector is

$$\boldsymbol{x} = \begin{bmatrix} \theta_1 & \delta_1 & m_1 & \theta_2 & \delta_2 & m_2 & \cdots & \theta_J & \delta_J & m_J \end{bmatrix}^\top. \tag{5.54}$$

Because the functional connection between the observations and the parameters

$$f_k(\boldsymbol{x}) = \ln |H^{-1}(e^{i\omega_k})| = \prod_{j=1}^{J} \ln |H_j^{-1}(e^{i\omega})| \tag{5.55}$$

is not linear, we need to linearize it. This is typically performed by a Taylor series expansion

$$f_k(\boldsymbol{x}) \approx f_k(\boldsymbol{x}^{(0)}) + \nabla^\top f_k(\boldsymbol{x}^{(0)})\Delta\boldsymbol{x}, \tag{5.56}$$

which is truncated after the linear term. Herein, $\Delta\boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}^{(0)}$ is the difference between the approximate values $\boldsymbol{x}^{(0)}$ for the parameters and the parameters $\boldsymbol{x}$. For the reduced observation vector $\Delta\boldsymbol{l}$ and the design matrix we find

$$\Delta\boldsymbol{l} = \begin{bmatrix} \sqrt{\Phi_{\mathcal{XX}}(\omega_1)} - f_1(\boldsymbol{x}^{(0)}) \\ \sqrt{\Phi_{\mathcal{XX}}(\omega_2)} - f_2(\boldsymbol{x}^{(0)}) \\ \vdots \\ \sqrt{\Phi_{\mathcal{XX}}(\omega_K)} - f_K(\boldsymbol{x}^{(0)}) \end{bmatrix} \quad \text{and} \quad \boldsymbol{A} = \begin{bmatrix} \nabla^\top f_1(\boldsymbol{x}^{(0)}) \\ \nabla^\top f_2(\boldsymbol{x}^{(0)}) \\ \vdots \\ \nabla^\top f_K(\boldsymbol{x}^{(0)}) \end{bmatrix}. \tag{5.57}$$

The observation equations then read

$$\Delta\boldsymbol{l} = \boldsymbol{A}\Delta\boldsymbol{x}. \tag{5.58}$$

Since they are often numerically instable, we use the approximate values for the parameters for regularization in order to stabilize observation equations.

$$\boldsymbol{\mathcal{X}} \sim N(\boldsymbol{x}^{(0)}, \boldsymbol{D}_{\boldsymbol{\mathcal{XX}}}) \quad \Longrightarrow \quad \Delta\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{X}} - \boldsymbol{x}^{(0)} \sim N(\boldsymbol{0}, \boldsymbol{D}_{\boldsymbol{\mathcal{XX}}}) \tag{5.59}$$

Herein, $\boldsymbol{D}_{\boldsymbol{\mathcal{XX}}}$ is a diagonal matrix, which we have to choose appropriately. This regularization introduces additional observations equations

$$\boldsymbol{0} = \boldsymbol{I}\Delta\boldsymbol{x}. \tag{5.60}$$

The whole observation equations are

$$\begin{bmatrix} \Delta\boldsymbol{l} \\ \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{I} \end{bmatrix} \Delta\boldsymbol{x}. \tag{5.61}$$

For the stochastic part of the Gauss-Markov model we find

$$\boldsymbol{\Sigma}\{\begin{bmatrix} \Delta\boldsymbol{\mathcal{L}} \\ \Delta\boldsymbol{\mathcal{X}} \end{bmatrix}\} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}_{\boldsymbol{\mathcal{XX}}} \end{bmatrix}. \tag{5.62}$$

Because the solution of this least squares problem tends to be instable, we first homogenize the observation equations (5.61) (cf. Koch 1997, p. 168) and then compute the least squares solution by utilizing the QR decomposition. For the homogenization of the observation equations, we factorize the covariance matrix in (5.62)

$$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}_{\boldsymbol{\mathcal{XX}}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}^{\frac{1}{2}}_{\boldsymbol{\mathcal{XX}}} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}^{\frac{1}{2}}_{\boldsymbol{\mathcal{XX}}} \end{bmatrix} \tag{5.63}$$

and multiply the observation equations (5.61) with one of the factors.

$$\begin{bmatrix} \Delta\boldsymbol{l} \\ \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}^{\frac{1}{2}}_{\boldsymbol{\mathcal{XX}}} \end{bmatrix} \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{I} \end{bmatrix} \Delta\boldsymbol{x} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{D}^{\frac{1}{2}}_{\boldsymbol{\mathcal{XX}}} \end{bmatrix} \Delta\boldsymbol{x}. \tag{5.64}$$

Then we compute the QR decomposition

$$\begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{D}^{\frac{1}{2}}_{\boldsymbol{\mathcal{XX}}} \end{bmatrix} = \boldsymbol{Q}\boldsymbol{R}, \tag{5.65}$$

whereas $\boldsymbol{Q}$ is an orthonormal matrix and $\boldsymbol{R}$ upper triangular matrix. The least squares solution is finally obtained by solving

$$\boldsymbol{R}\widetilde{\Delta\boldsymbol{x}} = \boldsymbol{Q}^\top\Delta\boldsymbol{l} \tag{5.66}$$

for $\widetilde{\Delta \boldsymbol{x}}$.

In some cases we want to design or estimate notch filters, which are very aggressive. This means that the notch filters adjust to the highest values of the power spectral density instead of all values. We can achieve this by introducing weights for the observations. The covariance matrix of the observations becomes

$$\boldsymbol{\Sigma}\{\boldsymbol{\mathcal{L}}\} = \boldsymbol{\Sigma}\{\Delta\boldsymbol{\mathcal{L}}\} = \boldsymbol{D}_{\boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{L}}} = \begin{bmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & d_n \end{bmatrix}, \tag{5.67}$$

whereas d_1, d_2, …, d_n are the inverse weights. A good choice of the inverse weights is for example

$$d_i = e^{v_i/\sigma_v} \quad \text{with} \quad \sigma_v^2 = \frac{1}{n}\sum_{i=1}^{n} v_i^2. \tag{5.68}$$

Herein, the v_i are the residuals of the observations in the Gauss-Markov model. The standard deviation σ_v is used to normalize the inverse weights. The stochastic part of the Gauss-Markov model then becomes

$$\boldsymbol{\Sigma}\{\begin{bmatrix} \Delta\boldsymbol{l} \\ \boldsymbol{0} \end{bmatrix}\} = \begin{bmatrix} \boldsymbol{D}_{\boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{L}}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}_{\boldsymbol{\mathcal{X}}\boldsymbol{\mathcal{X}}} \end{bmatrix}. \tag{5.69}$$

Again, the covariance matrix is factorized into

$$\begin{bmatrix} \boldsymbol{D}_{\boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{L}}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}_{\boldsymbol{\mathcal{X}}\boldsymbol{\mathcal{X}}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{D}^{\frac{1}{2}}_{\boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{L}}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}^{\frac{1}{2}}_{\boldsymbol{\mathcal{X}}\boldsymbol{\mathcal{X}}} \end{bmatrix} \begin{bmatrix} \boldsymbol{D}^{\frac{1}{2}}_{\boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{L}}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}^{\frac{1}{2}}_{\boldsymbol{\mathcal{X}}\boldsymbol{\mathcal{X}}} \end{bmatrix}, \tag{5.70}$$

such that the observation equations (5.61) can be homogenized.

$$\begin{bmatrix} \Delta\boldsymbol{l} \\ \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{D}^{\frac{1}{2}}_{\boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{L}}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}^{\frac{1}{2}}_{\boldsymbol{\mathcal{X}}\boldsymbol{\mathcal{X}}} \end{bmatrix} \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{I} \end{bmatrix} \Delta\boldsymbol{x} = \begin{bmatrix} \boldsymbol{D}^{\frac{1}{2}}_{\boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{L}}}\boldsymbol{A} \\ \boldsymbol{D}^{\frac{1}{2}}_{\boldsymbol{\mathcal{X}}\boldsymbol{\mathcal{X}}} \end{bmatrix} \Delta\boldsymbol{x} \tag{5.71}$$

In order to compute the least squares estimates the QR decomposition

$$\begin{bmatrix} \boldsymbol{D}^{\frac{1}{2}}_{\boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{L}}}\boldsymbol{A} \\ \boldsymbol{D}^{\frac{1}{2}}_{\boldsymbol{\mathcal{X}}\boldsymbol{\mathcal{X}}} \end{bmatrix} = \boldsymbol{Q}\boldsymbol{R} \tag{5.72}$$

is performed and

$$\boldsymbol{R}\widetilde{\Delta\boldsymbol{x}} =, \boldsymbol{Q}^\top \Delta\boldsymbol{l} \tag{5.73}$$

is solved for $\widetilde{\Delta\boldsymbol{x}}$. The inverse weights d_1, d_2, …, d_n can only be computed after a least squares solution $\widetilde{\boldsymbol{x}} = \boldsymbol{x}^{(0)} + \widetilde{\Delta\boldsymbol{x}}$ is known. This means, that we need to iterate the whole procedure until the inverse weights converge. Starting values for the inverse weights can be computed with the approximate values $\boldsymbol{x}^{(0)}$ of the parameters of the notch filters. In practice, the approximate values $\boldsymbol{x}^{(0)}$ are often manually chosen.

6. Application GOCE

The gravity field and steady-state ocean circulation (GOCE) mission is a project of the European Space Agency (ESA). It is dedicated to the determination of the Earth's gravity field. The GOCE mission aims at the precise determination of the steady-state part of the Earth's gravity field with a very high resolution. The accuracy of the resulting quasi geoid shall be 1 cm for a spatial resolution of 100 km. In order to achieve this ambitious goal, the GOCE mission is based on a sensor fusion concept. The low frequency signals of the gravity field will be observed mainly by satellite-to-satellite tracking (SST) while the high frequency signals will be observed mainly by satellite gravity gradiometry (SGG). A detailed description of the GOCE mission and its goals can be found in ESA (2002).

The gradiometer is the main instrument on board of the GOCE satellite. A description of this measurement device is can be found in Rummel (1985). It will measure at a rate of one Hertz and provide six observations per epoch. Thus, during twelve months of measurement the gradiometer will collect approximately 100 million SGG observations. This large number of observations allows only for tailored numerical algorithms to be applied in order to take the high correlations of the SGG observations into account. All conditions listed in section 2.2 (time series, normal distribution, stationarity, ergodicity, correlation length) including the additional conditions (number of observations and parameters, high correlations) are met for the SGG observations. Therefore, the high-level processing of the SGG observations is the ideal application scenario for this thesis.

Within this chapter we will first introduce the deterministic model for the observations. This is important for the computation of the observation equations. After that, we investigate the numerical method for the computation of the least squares solution. The method of choice is in this case a special variant of the conjugate gradient method (cf. Hestenes and Stiefel 1952, Schwarz 1970) called preconditioned conjugate gradient multiple adjustment (PCGMA) method. The next step is to describe in which way the filtering is best integrated into the PCGMA method (cf. Schuh 1996). Then, the test data will be described, followed by the description of the used filter models. Finally, the results of numerical simulations will be presented in this chapter.

6.1 The Observation Equations

It is common use to express the gravitational potential V of the Earth in a series of normalized spherical harmonics

$$V(\lambda,\vartheta,r) = \frac{GM}{R}\sum_{n=0}^{n_{\max}}\sum_{m=0}^{n}\left(\frac{R}{r}\right)^{n+1}\big(c_{nm}\cos(m\lambda)+s_{nm}\sin(m\lambda)\big)P_{nm}(\cos\vartheta). \tag{6.1}$$

Herein, c_{nm} and s_{nm} are the fully normalized spherical harmonic coefficients, whereas the normalization is performed such that

$$\frac{1}{4\pi}\iint_{\Omega}\big(\cos(m\lambda)P_{nm}(\cos\vartheta)\big)^2 d\Omega(\lambda,\vartheta) = 1 \quad\text{and}\quad \frac{1}{4\pi}\iint_{\Omega}\big(\sin(m\lambda)P_{nm}(\cos\vartheta)\big)^2 d\Omega(\lambda,\vartheta) = 1 \tag{6.2}$$

holds for the basis functions $\cos(m\lambda)P_{nm}(\cos\vartheta)$ and $\sin(m\lambda)P_{nm}(\cos\vartheta)$. The meaning of each letter in equation (6.1) is listed the following table.

V	gravitational potential of the Earth	r	radius
G	geocentric gravitational constant	λ	longitude
M	Earth's mass	ϑ	polar distance
R	reference radius	P_{nm}	fully normalized associated Legendre function of the first kind
n	degree		
m	order	c_{nm}, s_{nm}	fully normalized harmonic coefficients

Within the gravity field recovery from GOCE data the zero order term $C_{00} = 1$ and the first order terms C_{10}, C_{11}, $S_{11} = 0$ are usually excluded. There is also a maximum degree $n_{\max} \approx 250$, which will not be exceeded, as the gradiometer on board of GOCE is only sensitive for a certain frequency bandwidth (cf. ESA 1999).

The gradiometer does not measure the gravitational potential V itself, but its second derivatives $V_{ij} = \frac{\partial^2 V}{\partial i \partial j}$, $i, j \in \{x,\, y,\, z\}$. The actual measurement are the elements of the gravitational tensor

$$\boldsymbol{V} = \begin{bmatrix} V_{xx} & V_{xy} & V_{xz} \\ V_{yx} & V_{yy} & V_{yz} \\ V_{zx} & V_{zy} & V_{zz} \end{bmatrix}. \tag{6.3}$$

The gravitational tensor is given in the gradiometer reference frame, which is oriented along the axis of the gradiometer. The attitude of the satellite is controlled such that the x-axis of the gradiometer reference frame points approximately along the satellites orbital track while the z-axis approximately points perpendicular to the Earth's surface. In order to maintain this attitude, the satellite is equipped by magnetic torquers. The gravitational tensor needs to be transformed into an Earth-fixed reference frame by a rotation. For this purpose, the attitude of the satellite is determined by a combination of star-tracker and gradiometer information: The star-tracker provides angular velocities while the gradiometer provides angular accelerations. In the processing of the measurements, only the diagonal elements of the gravitational tensor are used. The reason for this lies in the accuracy of the individual elements and their effect on the recovery of the gravity field. The V_{zz} element is the most important one, as it points approximately perpendicular to the Earth's surface.

In the least squares adjustment, the measurements V_{xx}, V_{yy} and V_{zz} are stored in the observation vector $\boldsymbol{l}$. The design matrix $\boldsymbol{A}$ connects these measurements with the normalized harmonic coefficients c_{nm} and s_{nm}, which are the parameters of the gravity field. These parameters are contained in the parameter vector $\boldsymbol{x}$. Now, we know the contents of the terms of the observation equation (2.4)

$$\boldsymbol{l} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e}. \tag{6.4}$$

6.2 Solution Strategy

Our goal will be to determine the gravity field parameters contained in the parameter vector $\boldsymbol{x}$ as accurately as possible. After assuming that the measurement noise follows the normal distribution, the best estimate for $\boldsymbol{x}$ is the least squares estimate

$$\widetilde{\boldsymbol{x}} = (\boldsymbol{A}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{A})^{-1} \boldsymbol{A}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{l}, \tag{6.5}$$

according to equation (2.8). However, the computation of this least squares estimate is a computationally very demanding task. The reason for this is given by the sheer number of observations and the high resolution of the gravity field. The sampling rate of the measurements will be one second, resulting in about 100 million observations for a measurement period of one year. The gravity field will be resolved approximately up to degree and order 250, resulting in about 60 000 parameters. If we use double precision, the design matrix for the SGG observations then needs 45 000 gigabytes (GB) of memory and the full covariance matrix of the observations needs 73 000 terabytes (TB). Table 6.1 shows the size of the design matrix $\boldsymbol{A}$ and covariance matrix $\boldsymbol{\Sigma}$ for various numbers of observations and parameters.

From table 6.1 we can easily see that it is currently not possible to store the design matrix or the covariance matrix in the working memory of a computer. Therefore, we need tailored methods to numerically compute the least squares estimate as well as to represent and incorporate the covariance matrix of the observations. The latter aspect is subject to this thesis. Therefore, it will be discussed in the following in more detail. But first we make the choice for the numerical method for computing the least squares estimates.

(a) Size of the design matrix $\boldsymbol{A}$.

	maximum degree	
measurement period	100	250
1 month	$5.9 \cdot 10^2$ GB	$3.6 \cdot 10^3$ GB
6 months	$3.5 \cdot 10^3$ GB	$2.2 \cdot 10^4$ GB
12 months	$7.1 \cdot 10^3$ GB	$4.3 \cdot 10^4$ GB

(b) Size of the covariance matrix $\boldsymbol{\Sigma}$.

measurement period	
1 month	$4.5 \cdot 10^5$ GB
6 months	$1.6 \cdot 10^7$ GB
12 months	$6.5 \cdot 10^7$ GB

Table 6.1: Size of the design matrix and the covariance matrix of the observations in gigabytes with respect to the measurement period and the resolution of the gravity field. If the maximum degree is 100, the size of the normal equation matrix $\boldsymbol{N} = \boldsymbol{A}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{A}$ is 0.7 GB. For the maximum degree 250 the normal equation matrix requires 29.6 GB.

According to Schuh (1996), p. 111, the method of choice for the numerical computation of the least squares estimates is a variant of the conjugate gradient algorithm, the PCGMA algorithm. Schuh (1996) highlights that the algorithm has very low computational costs. Moreover it is capable of consistently combining different kinds of data, which is a necessity in the case of the GOCE mission because of the two kinds of observations: SST and SGG observations. It is also very well suited to be parallelized, that is using a cluster of computers instead of a single computer to determine the least squares estimate. For these reasons we use the PCGMA algorithm to compute the least squares estimate $\widetilde{\boldsymbol{x}}$ in equation (6.5).

In the case of the GOCE mission, we incorporate the SGG and SST observations into the least squares adjustment in different ways. The SGG observations are incorporated based on observation equations while the SST observations are incorporated based on normal equations. The least squares estimate then reads

$$\widetilde{\boldsymbol{x}} = (\boldsymbol{A}_{\text{SGG}}^\top \boldsymbol{\Sigma}_{\text{SGG}}^{-1} \boldsymbol{A}_{\text{SGG}} + p\,\boldsymbol{N}_{\text{SST}})^{-1} (\boldsymbol{A}_{\text{SGG}}^\top \boldsymbol{\Sigma}_{\text{SGG}}^{-1} \boldsymbol{l}_{\text{SGG}} + p\,\boldsymbol{n}_{\text{SST}}), \tag{6.6}$$

wherein p is a weighting factor for the relative weighting of the SGG and SST observations. The incorporation of SST observations based on normal equations is possible due to the fact, that the SST observations are insensitive to the high frequency parts of the gravity field, such that their normal equations are comparably small. Because of the large number of SGG observations, it is not possible to directly use the covariance matrix $\boldsymbol{\Sigma}_{\text{SGG}}$ of the SGG observations. Fortunately, all conditions listed in section 2.2 (time series, normal distribution, stationarity, ergodicity, correlation length) are met for the SGG observations, such that we can use ARMA processes to represent the covariance matrix $\boldsymbol{\Sigma}_{\text{SGG}}$. We can split the inverse covariance matrix into the product

$$\boldsymbol{\Sigma}_{\text{SGG}}^{-1} = \boldsymbol{F}_{\text{SGG}}^\top \boldsymbol{F}_{\text{SGG}} \tag{6.7}$$

of an upper and a lower triangular matrix. The lower triangular matrix

$$\boldsymbol{F}_{\text{SGG}} = \boldsymbol{S}\boldsymbol{H} + \begin{bmatrix} \boldsymbol{W} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \tag{6.8}$$

is the filter matrix in equation (4.36). The multiplication with this matrix corresponds to filtering including the new method for the filter warm-up developed in section 4.2. Inserting (6.7) into (6.6) yields

$$\widetilde{\boldsymbol{x}} = (\boldsymbol{A}_{\text{SGG}}^\top \boldsymbol{F}_{\text{SGG}}^\top \boldsymbol{F}_{\text{SGG}} \boldsymbol{A}_{\text{SGG}} + p\,\boldsymbol{N}_{\text{SST}})^{-1} (\boldsymbol{A}_{\text{SGG}}^\top \boldsymbol{F}_{\text{SGG}}^\top \boldsymbol{F}_{\text{SGG}} \boldsymbol{l}_{\text{SGG}} + p\,\boldsymbol{n}_{\text{SST}}). \tag{6.9}$$

The basic form of the PCGMA algorithm is given in algorithm 6.1. In this basic form, there is no indication how the filtering or the parallelization is performed. Furthermore, extensions of the PCGMA algorithm, like variance component estimation (cf. Alkhatib 2007) or regularization (cf. Alkhatib 2007) are not depicted in the basic form for the sake of simplicity.

input	$\boldsymbol{x}^{(0)}$	initial solution
	$\boldsymbol{l}_{\text{SGG}}$	SGG observations
	$\boldsymbol{N}_{\text{SST}}$, $\boldsymbol{n}_{\text{SST}}$	SST normal equations
	$\boldsymbol{F}_{\text{SGG}}$	decorrelating filter matrix for SGG observations
	I	number of iterations
	p	weighting parameter for relative weighting of SGG and SST observations
output	$\boldsymbol{x}^{(I)}$	final solution
	$\boldsymbol{v}_{\text{SGG}}^{(I)}$	decorrelated SGG residuals
	$\boldsymbol{K}$	preconditioner

```
initialization
$\boldsymbol{K} = \text{select}(\boldsymbol{A}_{\text{SGG}}^\top \boldsymbol{F}_{\text{SGG}}^\top \boldsymbol{F}_{\text{SGG}} \boldsymbol{A}_{\text{SGG}}) + p\,\boldsymbol{N}_{\text{SST}}$
$\boldsymbol{v}_{\text{SGG}}^{(0)} = \boldsymbol{F}_{\text{SGG}}(\boldsymbol{A}_{\text{SGG}}\boldsymbol{x}^{(0)} - \boldsymbol{l}_{\text{SGG}})$
$\boldsymbol{r}^{(0)} = \boldsymbol{A}_{\text{SGG}}^\top \boldsymbol{F}_{\text{SGG}}^\top \boldsymbol{v}^{(0)} + p\left(\boldsymbol{N}_{\text{SST}}\boldsymbol{x}^{(0)} - \boldsymbol{n}_{\text{SST}}\right)$
$\boldsymbol{f}^{(0)} = \text{solve}(\boldsymbol{K},\, \boldsymbol{r}^{(0)})$
$\boldsymbol{d}^{(0)} = -\boldsymbol{f}^{(0)}$

loop
for $i = 0, 1, \ldots, I-1$
    if $i > 0$
        $e = \frac{\boldsymbol{r}^{(i)\top}\boldsymbol{d}^{(i)}}{\boldsymbol{r}^{(i-1)\top}\boldsymbol{d}^{(i-1)}}$
        $\boldsymbol{d}^{(i)} = \boldsymbol{d}^{(i)} + e\boldsymbol{d}^{(i-1)}$
    end
    $\boldsymbol{g} = \boldsymbol{F}_{\text{SGG}}\boldsymbol{A}_{\text{SGG}}\boldsymbol{d}^{(i)}$
    $\boldsymbol{h} = \boldsymbol{A}_{\text{SGG}}^\top \boldsymbol{F}_{\text{SGG}}^\top \boldsymbol{g} + p\left(\boldsymbol{N}_{\text{SST}}\boldsymbol{d}^{(i)} - \boldsymbol{n}_{\text{SST}}\right)$
    $q = \frac{\boldsymbol{r}^{(i)\top}\boldsymbol{f}^{(i)}}{\boldsymbol{d}^{(i)\top}\boldsymbol{h}}$
    $\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} + q\boldsymbol{d}^{(i)}$
    $\boldsymbol{r}^{(i+1)} = \boldsymbol{r}^{(i)} + q\boldsymbol{h}$
    $\boldsymbol{f}^{(i+1)} = \text{solve}(\boldsymbol{K},\, \boldsymbol{r}^{(i+1)})$
    $\boldsymbol{v}_{\text{SGG}}^{(i+1)} = \boldsymbol{v}_{\text{SGG}}^{(i)} + q\boldsymbol{g}$
end
```

Algorithm 6.1: The preconditioned conjugate gradient multiple adjustment (PCGMA) algorithm in its basic form. In this basic form, there is no indication how the filtering or the parallelization is performed. Furthermore, extensions of the PCGMA algorithm, like variance component estimation or regularization, are not depicted in the basic form for the sake of simplicity.

Now we discuss how to perform the filtering. In order to do so, we need to recall that the design matrix does not fit into the working memory of nowadays computers. Therefore, we need to subdivide it into smaller parts. The smaller parts have to be computed each time they are involved in the computation. For the PCGMA algorithm, this is one time for the computation of the preconditioner and one or two times within the initialization and each iteration (cf. algorithm 6.1). Therefore, the subdivision should accommodate a fast computation of the parts of the design matrix. In our case, the computation of each line of the design matrix requires the computation of Legendre functions P_{nm}, which is typically performed recursively starting at degree $n = 0$ and order $m = 0$. Thus, it makes sense to subdivide the design matrix horizontally into blocks. Note, that each block would contain at most 4 000 rows for 2 GB of working memory and 60 000 parameters. In practice, the number of rows per block is much smaller, because the design matrix is not the only matrix within the working memory.

The fact that we can only have a very limited number of rows of the design matrix in the working memory requires a very space-saving implementation of the filter equation (2.53) for the filtering of the design matrix. If we want a good modeling of the SGG error characteristic by AR processes according to equation (2.27), we must choose the process order P high in order to obtain a good fit of the power spectral density of the filter to the power spectral density of the measurement noise, especially in the area below the measurement bandwidth (cf. figure 5.1). This is in contrast to the memory requirements of the filter implementation and therefore rules the following possible implementations out:

- all MA filter methods: block method, overlap-add method, FFT method (cf. section 3.1)
- ARMA overlap-add method (cf. section 3.2)
- ARMA FFT method (cf. section 3.2)

The possibility to use the Toeplitz approach is ruled out by the high computational costs due to the high correlations of the SGG observations. Therefore, the only applicable filter implementation for the filtering of the design matrix is the ARMA block method. For this reason we favor the usage of ARMA filters.

Having identified the ARMA process approach for the modeling of the covariance matrix of the SGG observations, we can start to investigate the best way of integrating the filtering into the adjustment procedure. First of all we need to find the locations, where the filtering has to be integrated. The locations are the computation of the SGG part

$$\text{select}(\boldsymbol{A}_{\text{SGG}}^{\top}\boldsymbol{F}_{\text{SGG}}^{\top}\boldsymbol{F}_{\text{SGG}}\boldsymbol{A}_{\text{SGG}}) \tag{6.10}$$

of the preconditioner $\boldsymbol{K}$ and the products of the type

$$\boldsymbol{A}_{\text{SGG}}^{\top}\boldsymbol{F}_{\text{SGG}}^{\top}\boldsymbol{F}_{\text{SGG}}\boldsymbol{y}, \tag{6.11}$$

where

$$\boldsymbol{y} = \begin{cases} \boldsymbol{A}_{\text{SGG}}\boldsymbol{x}^{(0)} - \boldsymbol{l}_{\text{SGG}}, & \text{within the initialization} \\ \boldsymbol{A}_{\text{SGG}}\boldsymbol{d}^{(i)}, & \text{within the iterations} \end{cases} \tag{6.12}$$

(cf. algorithm 6.1). Herein, the select instruction for the preconditioner means that only those elements of the normal equation matrix are computed and stored, which differ significantly from zero. For further details on the preconditioner confer section C.3. Furthermore, it should be mentioned that the intermediate product $\boldsymbol{F}_{\text{SGG}}\boldsymbol{y}$ is used within the computations of the PCGMA algorithm 6.1.

It should be also mentioned that for other application scenarios it may be feasible to obtain a suitable preconditioner by means of analytical formulas instead of selecting elements of the normal equation matrix. In this case the design matrix is not needed for the computation of the preconditioner and thus the filtering of the design matrix is also not needed for the computation of the preconditioner.

First, we look at the computation of the SGG part of the preconditioner. The best way is to filter the design matrix and then multiply the result with each other, i. e.

$$\text{select}((\boldsymbol{F}_{\text{SGG}}\boldsymbol{A}_{\text{SGG}})^\top(\boldsymbol{F}_{\text{SGG}}\boldsymbol{A}_{\text{SGG}})). \tag{6.13}$$

The sequence of operations is indicated by the brackets. The advantage of this sequence of operations is, that the computation of the design matrix and the filtering are both performed only once. It should be mentioned, that for our application scenario the filtering of the warm-up can only be approximated according to equation (4.40) due to the long filter warm-up resulting from the SGG error characteristic. However, this approximation can affect only the convergence rate of the PCGMA algorithm 6.1. For the simulations described in the sections 6.5 and 6.6 the approximation had no effect.

Next, we look at the computation of the products of the type $\boldsymbol{A}_{\text{SGG}}^\top\boldsymbol{F}_{\text{SGG}}^\top\boldsymbol{F}_{\text{SGG}}\boldsymbol{y}$. In principle there exist two efficient sequences of operations. The first sequence aims at filtering only vectors instead of the design matrix. Within the initialization of the PCGMA algorithm 6.1 the sequence reads

$$\boldsymbol{A}_{\text{SGG}}^\top(\boldsymbol{F}_{\text{SGG}}^\top(\boldsymbol{F}_{\text{SGG}}(\boldsymbol{A}_{\text{SGG}}\boldsymbol{x}^{(0)} - \boldsymbol{l}_{\text{SGG}}))). \tag{6.14}$$

However, since for our application scenario the design matrix is too large to fit into the working memory, it has to be computed twice. The second sequence avoids computing the design matrix twice by filtering it. The sequence of operations then reads

$$(\boldsymbol{F}_{\text{SGG}}\boldsymbol{A}_{\text{SGG}})^\top(\boldsymbol{F}_{\text{SGG}}\boldsymbol{A}_{\text{SGG}})\boldsymbol{x}^{(0)} - \boldsymbol{F}_{\text{SGG}}\boldsymbol{l}_{\text{SGG}}. \tag{6.15}$$

However, the filtering is not applied to a vector, but to the whole design matrix. The numerical effort increases dramatically, because the number of columns of the design matrix is large for our application scenario.

For the decision that we have to make now, we need to consider a feature of the used ARMA filters. The relevant feature is that the used ARMA filters consist of several filter sections. For this reason the filtering of the design matrix is more time consuming than the computation of the design matrix. Thus, the best way of integrating the filtering into the adjustment procedure is the sequence according to equation (6.14), which avoids to filter the design matrix.

Now that the integration of the filtering is chosen, we discuss the parallelization of the PCGMA algorithm 6.1. For the parallelization we follow a classic master-worker concept. One computing node, the master, controls the whole computation while all other computing nodes, the workers, do the parts of the computation, which can be parallelized. Within the PCGMA algorithm 6.1 the computation of the design matrix is parallelized, because this is the computationally most intensive part after the filtering in the case of our application scenario. Each worker computes one block of the design matrix, whereas the design matrix is subdivided horizontally.

The parallelization and the subdivision of the design matrix are strongly interwoven. For the computation of the SGG part of the preconditioner, the workers have to compute the design matrix and filter it according to equation (6.13). For the filtering of the design matrix each column can be understood as a time series. If we horizontally subdivide the design matrix, the time series are interrupted. At the beginning of each block of the design matrix the filtering produces a warm-up. In order to circumvent this problem, the blocks of the design matrix overlap by the length of the warm-up. This technique is called the overlap-save method (cf. OPPENHEIM and SCHAFER 1999, p. 674). For the sake of clarity, the partitioning of the design matrix is depicted in figure 6.1 and the whole procedure is described in more detail in the following.

In order to compute the SGG part of the preconditioner, we use in principle the equations (4.37) and (4.38). The first step is to filter the design matrix, which is accomplished by the multiplication with the filter matrix according to equation (6.8).

$$\boldsymbol{D}_{\text{SGG}} = \boldsymbol{F}_{\text{SGG}}\boldsymbol{A}_{\text{SGG}} \tag{6.16}$$

The normal equation matrix of the SGG observations is then obtained by

$$(\boldsymbol{F}_{\text{SGG}}\boldsymbol{A}_{\text{SGG}})^\top(\boldsymbol{F}_{\text{SGG}}\boldsymbol{A}_{\text{SGG}}) = \boldsymbol{D}_{\text{SGG}}^\top\boldsymbol{D}_{\text{SGG}}. \tag{6.17}$$

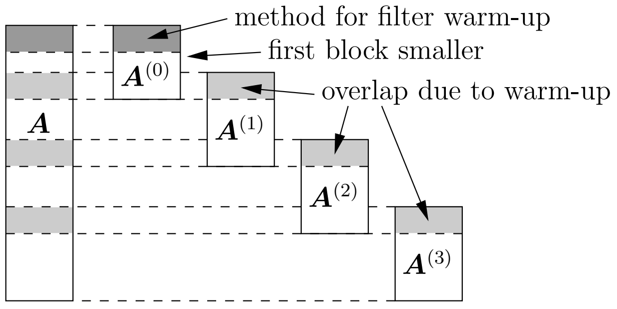


Figure 6.1: Subdivision of the design matrix for the parallel computation of the SGG part of the preconditioner. The matrix is subdivided into overlapping horizontal blocks, whereas the first block is smaller than the rest, because the method for the filtering of the warm-up causes additional computational effort. The size of the overlap corresponds to the length of the warm-up of the decorrelating filter.

In order to parallelize the computation, the filtered design matrix is subdivided into non-overlapping blocks $\boldsymbol{D}^{(k)}_{\text{SGG}}$. These blocks are computed by the product of a horizontal slice of the filter matrix and the whole design matrix.

$$\boldsymbol{D}^{(k)}_{\text{SGG}} = (\boldsymbol{D}_{\text{SGG}})_{kB_1+1:(k+1)B_1,:} = (\boldsymbol{F}_{\text{SGG}})_{kB_1+1:(k+1)B_1,:}\boldsymbol{A}_{\text{SGG}} \tag{6.18}$$

Herein, B_1 is the block size for the subdivision of the design matrix for the computation of the SGG part of the preconditioner. If each worker had to compute the whole design matrix, there would be no gain in the subdivision of $\boldsymbol{D}_{\text{SGG}}$. However, the filter matrix $\boldsymbol{F}_{\text{SGG}}$ can be approximated by a band matrix according to equation (4.59). If we use this approximation, we can leave the zero parts of the filter matrix $\boldsymbol{F}_{\text{SGG}}$ aside.

$$(\boldsymbol{F}_{\text{SGG}})_{kB_1+1:(k+1)B_1,:}\boldsymbol{A}_{\text{SGG}} = (\boldsymbol{F}_{\text{SGG}})_{kB_1+1:(k+1)B_1,kB_1+1-R:(k+1)B_1}(\boldsymbol{A}_{\text{SGG}})_{kB_1+1-R:(k+1)B_1,:}. \tag{6.19}$$

Herein, R is the length of the warm-up while $R+1$ is the number of non-zero elements in each row of the approximated filter matrix. Note, that the approximation can only affect the convergence rate of the PCGMA algorithm, if it is performed improperly. After defining

$$\boldsymbol{F}^{(k)}_{\text{SGG}} = \begin{cases} (\boldsymbol{F}_{\text{SGG}})_{1:(k+1)B_1,1:(k+1)B_1}, & \text{for } k=0 \\ (\boldsymbol{F}_{\text{SGG}})_{kB_1+1:(k+1)B_1,kB_1-R+1:(k+1)B_1}, & \text{for } k>0 \end{cases} \tag{6.20}$$

and

$$\boldsymbol{A}^{(k)}_{\text{SGG}} = \begin{cases} (\boldsymbol{A}_{\text{SGG}})_{1:(k+1)B_1,:}, & \text{for } k=0 \\ (\boldsymbol{A}_{\text{SGG}})_{kB_1-R+1:(k+1)B_1,:}, & \text{for } k>0 \end{cases}, \tag{6.21}$$

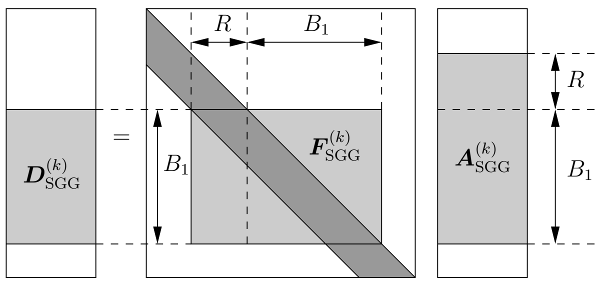
equation (6.18) becomes

$$\boldsymbol{D}^{(k)}_{\text{SGG}} = \boldsymbol{F}^{(k)}_{\text{SGG}}\boldsymbol{A}^{(k)}_{\text{SGG}}. \tag{6.22}$$

Figure 6.2 illustrates this subdivision into blocks.

According to equation (6.13) and (6.16) the SGG part of the preconditioner can be computed by

$$\begin{aligned} \text{select}((\boldsymbol{F}_{\text{SGG}}\boldsymbol{A}_{\text{SGG}})^\top(\boldsymbol{F}_{\text{SGG}}\boldsymbol{A}_{\text{SGG}})) &= \text{select}(\boldsymbol{D}^\top_{\text{SGG}}\boldsymbol{D}_{\text{SGG}}) \\ &= \sum_k \text{select}((\boldsymbol{D}^{(k)}_{\text{SGG}})^\top\boldsymbol{D}^{(k)}_{\text{SGG}}) \\ &= \sum_k \text{select}((\boldsymbol{F}^{(k)}_{\text{SGG}}\boldsymbol{A}^{(k)}_{\text{SGG}})^\top(\boldsymbol{F}^{(k)}_{\text{SGG}}\boldsymbol{A}^{(k)}_{\text{SGG}})). \end{aligned} \tag{6.23}$$



$=$
$\boldsymbol{D}^{(k)}_{\mathrm{SGG}}$
R
B_1
B_1
$\boldsymbol{F}^{(k)}_{\mathrm{SGG}}$
$\boldsymbol{A}^{(k)}_{\mathrm{SGG}}$
R
B_1

Figure 6.2: The relation between the blocks for the filtering of the design matrix. The filter matrix $\boldsymbol{F}_{\mathrm{SGG}}$ is approximated by a band matrix, where the band is indicated by dark gray. The brighter gray indicates the used blocks of the matrices.

Now, we analyze the operation $\boldsymbol{F}^{(k)}_{\mathrm{SGG}}\boldsymbol{A}^{(k)}_{\mathrm{SGG}}$ in more detail. First, we assume that the block size for the subdivision of the design matrix exceeds the length of the warm-up, i.e. $B_1 \geq R$. Then, the method for the filtering of the warm-up needs not to be parallelized. If the block size B_1 cannot be chosen greater than the length of the warm-up, i.e. $B_1 < R$, we simply increase the size of the first block such that its number of rows is equal to the length of the warm-up. According to equation (6.8) and (6.20) the blocks of the filter matrix are then

$$\boldsymbol{F}^{(k)}_{\mathrm{SGG}} = \begin{cases} (\boldsymbol{SH})_{1:(k+1)B_1,0:(k+1)B_1-1} + \begin{bmatrix} \underset{R\times R}{\boldsymbol{W}} & \underset{R\times B_1-R}{\boldsymbol{0}} \\ \underset{B_1-R\times R}{\boldsymbol{0}} & \underset{B_1-R\times B_1-R}{\boldsymbol{0}} \end{bmatrix}, & \text{for } k=0 \\ (\boldsymbol{SH})_{kB_1+1:(k+1)B_1,kB_1-R+1:(k+1)B_1}, & \text{for } k>0 \end{cases}. \tag{6.24}$$

Note, that the dimension of the first block $\boldsymbol{F}^{(0)}_{\mathrm{SGG}}$ is $B_1 \times B_1$ while the dimension of all other blocks $\boldsymbol{F}^{(k)}_{\mathrm{SGG}}$ for $k > 0$ is $B_1 \times B_1 + R$.

The filtering by the first block $\boldsymbol{F}^{(0)}_{\mathrm{SGG}}$ simply corresponds to filtering of the first B_1 rows of the design matrix instead of N rows. For the first filter matrix we find according to equation (6.24)

$$\boldsymbol{F}^{(0)}_{\mathrm{SGG}} = \boldsymbol{S}_{1:B_1,1:B_1}\boldsymbol{H}_{1:B_1,1:B_1} + \begin{bmatrix} \underset{R\times R}{\boldsymbol{W}} & \underset{R\times B_1-R}{\boldsymbol{0}} \\ \underset{B_1-R\times R}{\boldsymbol{0}} & \underset{B_1-R\times B_1-R}{\boldsymbol{0}} \end{bmatrix}. \tag{6.25}$$

Multiplying the first block of the design matrix by the first block of the filter matrix yields

$$\boldsymbol{F}^{(0)}_{\mathrm{SGG}}\boldsymbol{A}^{(0)}_{\mathrm{SGG}} = \boldsymbol{S}_{1:B_1,1:B_1}\boldsymbol{H}_{1:B_1,1:B_1}\boldsymbol{A}^{(0)}_{\mathrm{SGG}} + \begin{bmatrix} \boldsymbol{W}(\boldsymbol{A}^{(0)}_{\mathrm{SGG}})_{1:R,:} \\ \boldsymbol{0} \end{bmatrix}. \tag{6.26}$$

This means that the first block $\boldsymbol{A}^{(0)}_{\mathrm{SGG}}$ of the design matrix is filtered from the first row to the last row and after that, the first R rows of the filtered block, which correspond to the warm-up of the filter, are replaced by the rows obtained by the method for the filtering of the warm-up.

Now we look at the blocks $\boldsymbol{F}^{(k)}_{\mathrm{SGG}}$ for $k > 0$. For these blocks we find

$$\begin{aligned} \boldsymbol{F}^{(k)}_{\mathrm{SGG}} &= (\boldsymbol{SH})_{kB_1+1:(k+1)B_1,kB_1-R+1:(k+1)B_1} \\ &= \boldsymbol{S}_{kB_1+1:(k+1)B_1,kB_1+1:(k+1)B_1}\boldsymbol{H}_{kB_1+1:(k+1)B_1,kB_1-R+1:(k+1)B_1}, \quad \text{for } k>0. \end{aligned} \tag{6.27}$$

Note, that $\boldsymbol{S}_{kB_1+1:(k+1)B_1,kB_1+1:(k+1)B_1} = \boldsymbol{I}$ is an identity matrix. With this knowledge, we can write

$$\boldsymbol{F}^{(k)}_{\mathrm{SGG}} = \begin{bmatrix} \underset{B_1\times R}{\boldsymbol{0}} & \underset{B_1\times B_1}{\boldsymbol{I}} \end{bmatrix} \boldsymbol{H}_{kB_1-R+1:(k+1)B_1,kB_1-R+1:(k+1)B_1}, \quad \text{for } k>0. \tag{6.28}$$

Furthermore, $\boldsymbol{H}$ is a Toeplitz matrix. Thus, it holds that

$$\boldsymbol{H}_{kB_1-R+1:(k+1)B_1,kB_1-R+1:(k+1)B_1} = \boldsymbol{H}_{1:B_1+R,1:B_1+R}. \tag{6.29}$$

Therefore, we find

$$\boldsymbol{F}_{\text{SGG}}^{(k)} = \begin{bmatrix} \underset{B_1\times R}{\mathbf{0}} & \underset{B_1\times B_1}{\boldsymbol{I}} \end{bmatrix} \boldsymbol{H}_{1:B_1+R,1:B_1+R}, \quad \text{for } k>0 \tag{6.30}$$

for the blocks of the filter matrix. For the filtering of the blocks $\boldsymbol{A}_{\text{SGG}}^{(k)}$ for $k>0$ we obtain then

$$\begin{aligned}\boldsymbol{F}_{\text{SGG}}^{(k)}\boldsymbol{A}_{\text{SGG}}^{(k)} &= \begin{bmatrix} \underset{B_1\times R}{\mathbf{0}} & \underset{B_1\times B_1}{\boldsymbol{I}} \end{bmatrix} \boldsymbol{H}_{1:B_1+R,1:B_1+R}\boldsymbol{A}_{\text{SGG}}^{(k)} \\ &= (\boldsymbol{H}_{1:B_1+R,1:B_1+R}\boldsymbol{A}_{\text{SGG}}^{(k)})_{R+1:R+B_1,:}, \quad \text{for } k>0.\end{aligned} \tag{6.31}$$

This means that each block $\boldsymbol{A}_{\text{SGG}}^{(k)}$ of the design matrix is filtered from its first row to its last row and after that, the first R rows, which correspond to the warm-up, are set to zero.

The parallelization of the computation of the SGG part of the preconditioner is accomplished by the fact, that each worker computed one summand select$((\boldsymbol{F}_{\text{SGG}}^{(k)}\boldsymbol{A}_{\text{SGG}}^{(k)})^\top(\boldsymbol{F}_{\text{SGG}}^{(k)}\boldsymbol{A}_{\text{SGG}}^{(k)}))$ of the sum in equation (6.23). Note, that each block $\boldsymbol{A}^{(k)}$ will still be too big to fit in the working memory. Therefore, each block $\boldsymbol{A}^{(k)}$ is again subdivided into blocks. The block size for this subdivision is chosen equal to the block size L of the block ARMA filter method described in section 3.2. For performance issues we may choose to compute several filtered blocks, before the inner product of the selected columns according to equation (6.13) are computed, if we the working nodes possess enough memory. It should be mentioned that the first block may be chosen smaller, because the method for the filter warm-up causes an additional computational effort.

After the computation of the SGG part of the preconditioner the subdivision should be chosen differently. The reason for this is that the design matrix is not filtered by the workers anymore. The workers only compute the products

$$\boldsymbol{y} = \begin{cases} \boldsymbol{A}_{\text{SGG}}\boldsymbol{x}^{(0)} - \boldsymbol{l}_{\text{SGG}}, & \text{within the initialization} \\ \boldsymbol{A}_{\text{SGG}}\boldsymbol{d}^{(i)}, & \text{within the iterations} \end{cases} \tag{6.32}$$

and

$$\boldsymbol{A}_{\text{SGG}}^\top\boldsymbol{z}, \tag{6.33}$$

whereas

$$\boldsymbol{z} = \boldsymbol{F}_{\text{SGG}}^\top\boldsymbol{F}_{\text{SGG}}\boldsymbol{y}. \tag{6.34}$$

For these computations, there is no overlap needed for the subdivision of the design matrix. Therefore, the blocks of the design matrix are obtained by

$$\boldsymbol{A}_{\text{SGG}}^{(k)} = (\boldsymbol{A}_{\text{SGG}})_{kB_2:(k+1)B_2-1,:}, \tag{6.35}$$

whereas B_2 is the block size, which may differ from the block size B_1 for the computation of the SGG part of the preconditioner. Figure 6.3 illustrates this subdivision of the design matrix.

The vectors $\boldsymbol{y}$ and $\boldsymbol{z}$ are subdivided according to the subdivision of the design matrix, i.e.

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}^{(0)} \\ \vdots \\ \boldsymbol{y}^{(K_2)} \end{bmatrix}, \quad \text{whereas } \boldsymbol{y}^{(k)} = \boldsymbol{y}_{kL_2+1:(k+1)L_2} \tag{6.36}$$

and

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{z}^{(0)} \\ \vdots \\ \boldsymbol{z}^{(K_2)} \end{bmatrix}, \quad \text{whereas } \boldsymbol{z}^{(k)} = \boldsymbol{z}_{kL_2+1:(k+1)L_2}. \tag{6.37}$$

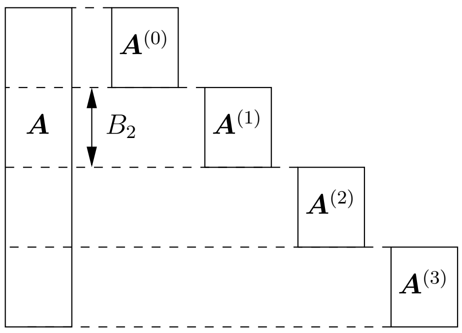


Figure 6.3: The subdivision of the design matrix for the parallel computations after computation of the preconditioner. The block size is chosen such that all blocks are of equal size. Furthermore, the blocks do not overlap.

For clarity, it should be mentioned that the filtering is performed according to equation (4.53). Together with equation (3.142) and (4.55) we find

$$\boldsymbol{z} = \boldsymbol{F}_{\mathrm{SGG}}^{\top} \boldsymbol{F}_{\mathrm{SGG}} \boldsymbol{y} = \boldsymbol{PHPSHy} + \boldsymbol{W}^{\top} \boldsymbol{W} \boldsymbol{y}_{1:R}. \tag{6.38}$$

These operations are performed by the master. Now, we can describe the parallel version in algorithm 6.2 of the basic PCGMA algorithm 6.1.

input	$\boldsymbol{x}^{(0)}$	initial solution
	$\boldsymbol{l}_{\text{SGG}}$	SGG observations
	$\boldsymbol{N}_{\text{SST}}$, $\boldsymbol{n}_{\text{SST}}$	SST normal equations
	$\boldsymbol{F}_{\text{SGG}}$	decorrelating filter matrix for SGG observations
	I	number of iterations
	p	weighting parameter for relative weighting of SGG and SST observations
output	$\boldsymbol{x}^{(I)}$	final solution
	$\boldsymbol{v}_{\text{SGG}}^{(I)}$	decorrelated SGG residuals
	$\boldsymbol{K}$	preconditioner

initialization

```
$\boldsymbol{K} = \text{select}((\boldsymbol{F}_{\text{SGG}}\boldsymbol{A}_{\text{SGG}})^\top(\boldsymbol{F}_{\text{SGG}}\boldsymbol{A}_{\text{SGG}})) + p\,\boldsymbol{N}_{\text{SST}}$    computed parallel
$\boldsymbol{y}^{(k)} = \boldsymbol{A}_{\text{SGG}}^{(k)}\boldsymbol{x}^{(0)} - \boldsymbol{l}_{\text{SGG}}^{(k)}$   for $k = 0, 1, 2, \ldots$    computed parallel, collected from master
$\boldsymbol{v}_{\text{SGG}}^{(0)} = \boldsymbol{F}_{\text{SGG}}\boldsymbol{y} = \boldsymbol{SHy} + \begin{bmatrix}(\boldsymbol{W}\boldsymbol{y}_{1:R})^\top & \boldsymbol{0}\end{bmatrix}^\top$
$\boldsymbol{z} = \boldsymbol{F}_{\text{SGG}}^\top\boldsymbol{v}_{\text{SGG}}^{(0)} = \boldsymbol{PHP}\boldsymbol{v}_{\text{SGG}}^{(0)} + \boldsymbol{W}^\top(\boldsymbol{v}_{\text{SGG}}^{(0)})_{1:R}$
$\boldsymbol{r}^{(0)} = \sum_k(\boldsymbol{A}_{\text{SGG}}^{(k)})^\top\boldsymbol{z}^{(k)} + p\left(\boldsymbol{N}_{\text{SST}}\boldsymbol{x}^{(0)} - \boldsymbol{n}_{\text{SST}}\right)$    computed parallel, accumulated on master
$\boldsymbol{f}^{(0)} = \text{solve}(\boldsymbol{K}, \boldsymbol{r}^{(0)})$
$\boldsymbol{d}^{(0)} = -\boldsymbol{f}^{(0)}$
```

loop

```
for $i = 0, 1, \ldots, I-1$
    if $i > 0$
        $e = \frac{\boldsymbol{r}^{(i)\top}\boldsymbol{d}^{(i)}}{\boldsymbol{r}^{(i-1)\top}\boldsymbol{d}^{(i-1)}}$
        $\boldsymbol{d}^{(i)} = \boldsymbol{d}^{(i)} + e\boldsymbol{d}^{(i-1)}$
    end
    $\boldsymbol{y}^{(k)} = \boldsymbol{A}_{\text{SGG}}^{(k)}\boldsymbol{d}^{(i)}$   for $k = 0, 1, 2, \ldots$    computed parallel, collected from master
    $\boldsymbol{g} = \boldsymbol{F}_{\text{SGG}}\boldsymbol{y} = \boldsymbol{SHy} + \begin{bmatrix}(\boldsymbol{W}\boldsymbol{y}_{1:R})^\top & \boldsymbol{0}\end{bmatrix}^\top$
    $\boldsymbol{z} = \boldsymbol{F}_{\text{SGG}}^\top\boldsymbol{g} = \boldsymbol{PHPg} + \boldsymbol{W}^\top\boldsymbol{g}_{1:R}$
    $\boldsymbol{h} = \sum_k(\boldsymbol{A}_{\text{SGG}}^{(k)})^\top\boldsymbol{z}^{(k)} + p\left(\boldsymbol{N}_{\text{SST}}\boldsymbol{d}^{(i)} - \boldsymbol{n}_{\text{SST}}\right)$    computed parallel, accumulated on master
    $q = \frac{\boldsymbol{r}^{(i)\top}\boldsymbol{f}^{(i)}}{\boldsymbol{d}^{(i)\top}\boldsymbol{h}}$
    $\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} + q\boldsymbol{d}^{(i)}$
    $\boldsymbol{r}^{(i+1)} = \boldsymbol{r}^{(i)} + q\boldsymbol{h}$
    $\boldsymbol{f}^{(i+1)} = \text{solve}(\boldsymbol{K}, \boldsymbol{r}^{(i+1)})$
    $\boldsymbol{v}_{\text{SGG}}^{(i+1)} = \boldsymbol{v}_{\text{SGG}}^{(i)} + q\boldsymbol{g}$
end
```

Algorithm 6.2: The preconditioned conjugate gradient multiple adjustment (PCGMA) algorithm in its parallel form. As the computation of the design matrix is the computationally most intensive part of the algorithm, it is performed in parallel.

6.3 Test Data Set

In the different numerical simulations the data of an ESA GOCE end-to-end simulation (cf. De Sanctis et al. 2002) was used. This test configuration was also used during the official ESA Acceptance Review 2 for the testing of the final operational software at the end of the development phase in the framework of the GOCE High-level Processing Facility (cf. Rummel et al. 2004), which was performed in spring/summer 2006. The test data set consists of:

Gravity gradients The gravity gradients cover a period of 60 days and are defined in the gradiometer reference frame (GRF). The measurement rate is 1 Hz. The gravity gradients are based on the gravity model EGM96 (cf. Lemoine et al. 1998) complete to degree and order 360 and they are superimposed by colored noise (cf. figure 5.1).

Orbit The gradients are defined along an orbit with GOCE characteristics (inclination $i = 96.5°$, eccentricity $e < 2 \cdot 10^{-3}$, mean altitude ~ 240 km). The orbit positions and velocities were generated by orbit integration, which was based on the gravity model EGM96, complete to degree and order 200. Within the orbit simulation a full external force model was included, too, as well as the drag free and attitude control system (cf. ESA 1999).

Attitude The orientation of the satellite body axes (and hence the GRF) with respect to the inertial frame is given in terms of quaternions, which are computed from a combination of star tracker and gradiometer information. Correspondingly, they include attitude biases and noise, related to the star tracker and gradiometer inaccuracies modeled in the end-to-end simulation (cf. Pail 2005).

The test data set consists of two groups of data: SST normal equations and SGG observations. The SST data are very sensitive to the low frequency part of the gravity field while the SGG observations are very sensitive to the high frequency part of the gravity field. The combination of SST and SGG observations of the two months measurement period allows for the determination of the gravity field approximately up to degree and order 255. The SST normal equations result from kinematic orbit solutions, which provide position and velocity information, in combination with the principle of energy conservation in a closed system. A short summary of the SST observations is given in table 6.2.

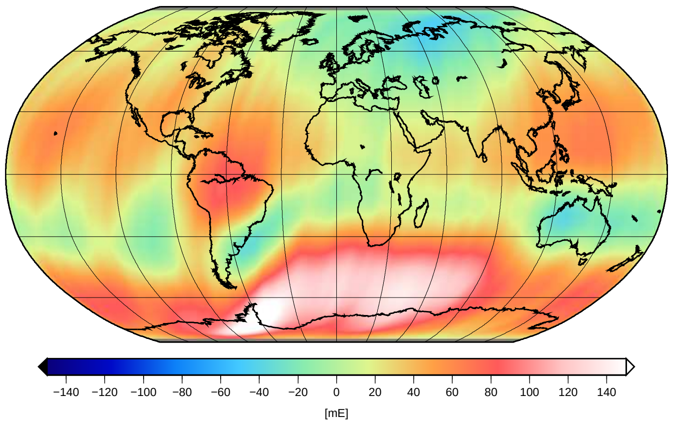
observation type	SST	integration type	normal equations
positions	5 011 200	possible resolution	degree and order 90
observations	5 011 200	unknown parameters	8 277
observation period	60 days	gravity field model	EGM96
earth revolutions	928	error characteristics	white noise

Table 6.2: Summary of the features of the SST data.

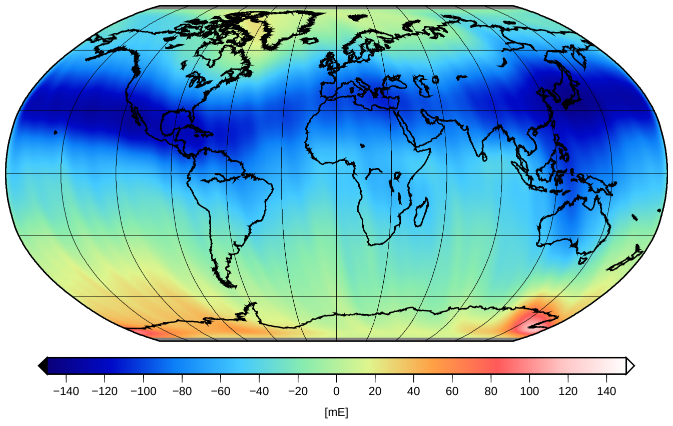
The characteristics of the colored noise of the SGG observations were already discussed in section 5.1. One feature of special interest for the decorrelation is that the measurement noise of the SGG observations contains a geographically dependent error, which is shown in figure 6.4. Due to the satellite orbit this error has a periodic characteristic in the noise time series and thus results in peaks in the error spectrum. As the geographically dependent error is rather systematic than random, it may affect the gravity field solution. The question is, whether it is possible to prevent this by a combination of down-weighting this error with tailored decorrelating filters and an additional group of observations, which are in our case the SST observations, or not. A short summary of the remaining features of the SGG observations is given in table 6.3.

6.4 Used Filter Models

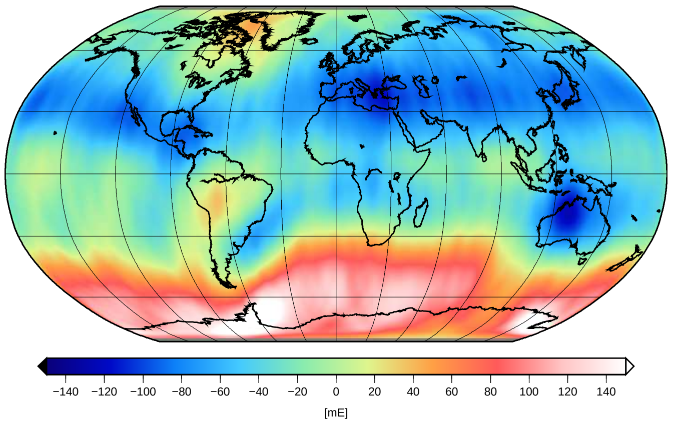
As the decorrelating filters are the item under test for the performed simulations, there were three different filter models generated for the test data set described in section 6.3. The original measurement noise time



(a) descending orbit tracks



(b) ascending orbit tracks



(c) all orbit tracks

Figure 6.4: Geographically dependent error of the SGG observations. The SGG observations contain a systematic error, which is geographically dependent. The error is different for the descending and ascending orbit tracks of the orbit of the satellite. After interpolating the errors of the descending and ascending orbit tracks of the orbit of the satellite onto a grid, they are added. The result is shown in the lower most panel. This sum of the error of the descending and ascending orbit tracks of the satellite may affect the gravity field solution.

observation type	SGG diagonal components	integration type	observation equations
positions	5 011 200	possible resolution	degree and order 255
observations	15 033 600	unknown parameters	65 021
observation period	60 days	gravity field model	EGM96
earth revolutions	928	error characteristics	colored noise

Table 6.3: Summary of the features of the SGG data.

series e_n was used as the basis for the filter design. This time series is only known due to the fact, that the error free observations are known within the simulation. Thus, the filter models are designed under ideal conditions. In practice, we would have to use the residual time series as an approximation of the measurement noise time series. Since the power spectral density of the residual time series is very similar to the power spectral density of the measurement noise series, the accuracy of the gravity field would not be decreased.

It should be mentioned that the peaks within the power spectral density occur approximately at the frequencies $f_k = \frac{k}{\text{day}} = \frac{k}{86400}$ Hz, $k = 1, 2, \ldots, 400$. The high number of peaks and the very small bandwidth of the notch filters, which could model the peaks, would result in a filter model, which we can not apply for the following two reasons: The warm-up would be unacceptably long and the computational effort of filtering would be incredibly large. However, the highest peaks occur at the frequencies $f_k = \frac{k}{\text{revolution}} \approx \frac{k}{5400}$ Hz, $k = 1, 2, \ldots, 24$. The rest of the peaks are smaller, the farther they are apart from the highest peaks. Therefore, the strategy is to model only the highest peaks by notch filters and to treat the rest of the peaks as side lobes (cf. figure 6.6(d)) of the highest peaks. This is a reasonable compromise between accurate modeling and computational demands. Table 6.4 gives a short overview over the designed filters. Details on the filter design are given in the following sections. Note, that illustrative figures are presented only for the V_{zz} component, because figures of the V_{xx} and V_{yy} components would look very similar.

filter model	component	sections	total filter order P	total filter order Q	warm-up
A	V_{xx}	3	53	53	15000 epochs
	V_{yy}	2	52	52	
	V_{zz}	3	73	73	
B	V_{xx}	20	137	137	45000 epochs
	V_{yy}	25	118	118	
	V_{zz}	27	126	126	
C	V_{xx}	18	133	133	65000 epochs
	V_{yy}	25	98	98	
	V_{zz}	27	121	121	

Table 6.4: Summary of statistics on the used filter models. The total filter order is the sum of the orders of the filter sections. The warm-up is equal for all three components.

It should be mentioned that there is an alternative way of taking the peaks in the power spectral density into account. Besides the stochastic treatment of the peaks, which is performed within this thesis by using notch filters, we may as well treat the in a deterministic way. In this case, we extend the observation equations (6.4) by additional absorbing parameters. This is feasible due to the fact that the frequencies of the peaks are known and that the source of the peaks is the geographically dependent error, which has a systematic characteristic. However, as we aim at investigating decorrelating filters, we do not use this approach.

Simple Averaging Filter (Model A)

Filter model A is a simple averaging filter, which consists of a Butterworth highpass filter and a general ARMA filter. It does not contain notch filters (cf. section 5.5). Therefore, this filter model consists only of few sections, which is an advantage concerning the computational effort of filtering. However, not to model the peaks in the power spectral density (cf. section 5.1) may turn out to perform worse than more complicated filter models. The design can be split into the following steps.

Step 1: Power Spectral Density First, we compute the power spectral density of the correlated time series. Whether we use Welch's method to obtain a smoothed estimate or not, does not matter.

Step 2: Butterworth Highpass Filter Looking at the power spectral density, we design a Butterworth highpass filter by cut and try.

Step 3: Highpass-Filtered Time Series Then, we filter the correlated time series by the Butterworth highpass filter in order to remove the low frequency part of the time series.

Step 4: Remove Warm-Up After having filtered the time series, we need to remove the warm-up of the Butterworth highpass filter.

Step 5: General ARMA Filter Based on the highpass-filtered time series, from which we have remove the warm-up, we estimate a general ARMA filter. Any insufficiencies in the design of the Butterworth highpass filter will be caught up here.

Step 6: Simple Averaging Filter The simple averaging filter is the combination of the Butterworth highpass filter and the general ARMA filter.

Aggressive Filter (Model B)

The second filter model aims at radically down-weighting the peaks within the power spectral density. The down-weighting is achieved by designing notch filters, which are adjusted only to the peaks and by using the weights in equation (5.68). Besides the notch filters, the filter model consists of a Butterworth highpass filter and a general ARMA filter. The design can be split into the following steps, which are also illustrated in figure 6.6.

Step 1: Power Spectral Density First, we compute the power spectral density of the correlated time series (cf. figure 6.6(b)). We do not use Welch's method.

Step 2: Median Power Spectral Density Then, we apply a moving-median filter to the power spectral density. The result is a smoothed version of the power spectral density, which does not contain the peaks (cf. figure 6.6(b)).

Step 3: Butterworth Highpass Filter Based on the median power spectral density, we design a Butterworth highpass filter by cut and try (cf. figure 6.6(b)). We have to take care, that the filter does not underestimate the power spectral density.

Step 4: Generated Time Series Next, we generate a time series from the median power spectral density. This can be performed in the following way: We compute the autocorrelation function from the median power spectral density using the inverse discrete Fourier transform. Based on the autocorrelation function we estimate an MA filter using the Levinson-Durbin algorithm 5.1. Using this MA filter as a process, we create a correlated time series by filtering a white noise time series.

Step 5: General ARMA Filter Based on the generated time series, we estimate a general ARMA filter.

Step 6: Quotient of Power Spectral Densities The peaks of the power spectral density are isolated by dividing the power spectral density by the median power spectral density (cf. figure 6.6(c) and 6.6(d)).

Step 7: Detect Peaks Now, we detect the peaks in the quotient of the power spectral densities. We distinguish between the highest peaks and the side lobes peaks (cf. figure 6.6(c) and 6.6(d)).

Step 8: Approximated Notch Filters Based on the frequency and the value of the power spectral density of the detected peaks, we chose the notch frequencies and scaling factor. In addition we chose a suitable notch bandwidth by cut and try (cf. figure 6.6(e) and 6.6(f)).

Step 9: Estimated Notch Filters The approximated notch filters are the basis for the estimation of the actual notch filters. Within the estimation we use the weights according to equation (5.68) in order to obtain aggressive notch filters (cf. figure 6.6(e) and 6.6(f)).

Step 10: Aggressive Filter The aggressive filter is the combination of the Butterworth highpass filter, the general ARMA filter and the estimated notch filters (cf. figure 6.6(a)).

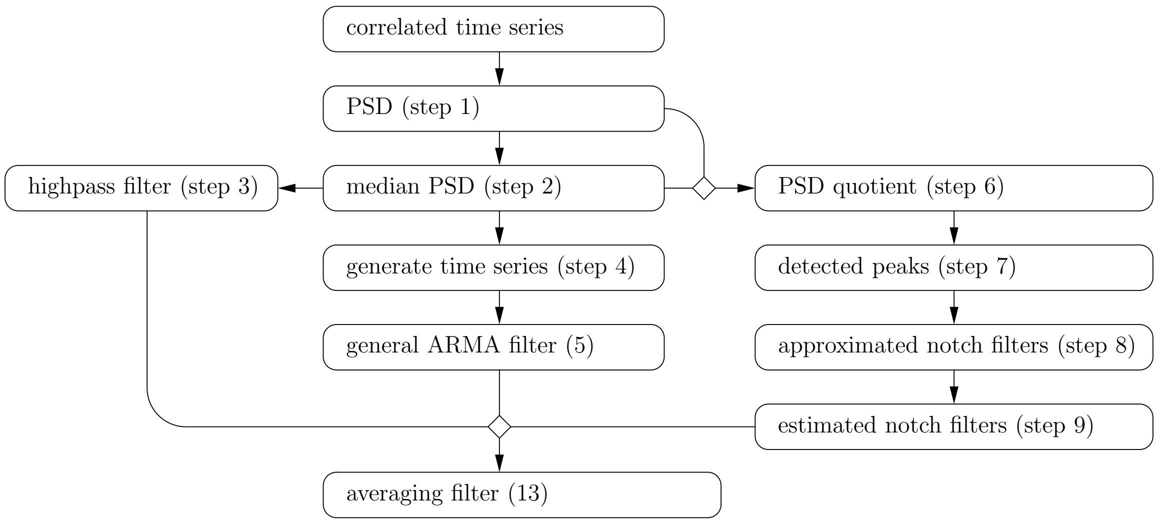


Figure 6.5: Sequence of steps within the estimation of the aggressive filter (model B).

Averaging Filter (Model C)

The third filter model is a compromise between the simple averaging filter model and the aggressive filter model. It also contains notch filters, but they are not directly adjusted to the peaks in the power spectral density. First, a smoothed estimate of the power spectral density was obtained by applying Welch's method (cf. WELCH 1967). Then, the notch filters are adjusted to the peaks in the smoothed power spectral density. As a result of this procedure the notch filters are by far not as aggressive as those of the aggressive filter model. Besides the notch filters, the filter model consists of a Butterworth highpass filter and a general ARMA filter. The design can be split into the following steps, which are also illustrated in figure 6.8.

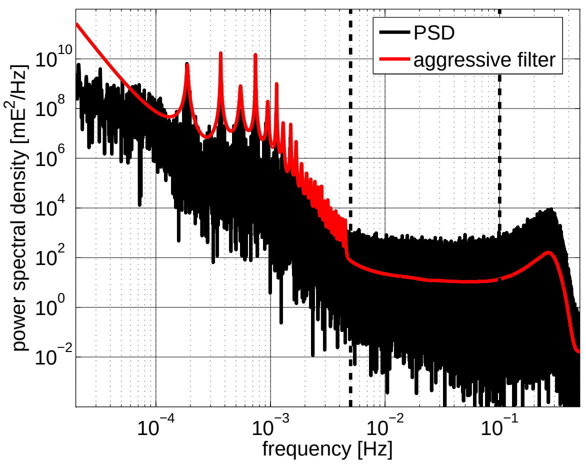
Step 1: Power Spectral Density First, we compute the power spectral density of the correlated time series (cf. figure 6.8(b)). We do not use Welch's method.

Step 2: Median Power Spectral Density Then, we apply a moving-median filter to the power spectral density (cf. figure 6.8(b)). The result is a smoothed version of the power spectral density, which does not contain the peaks.

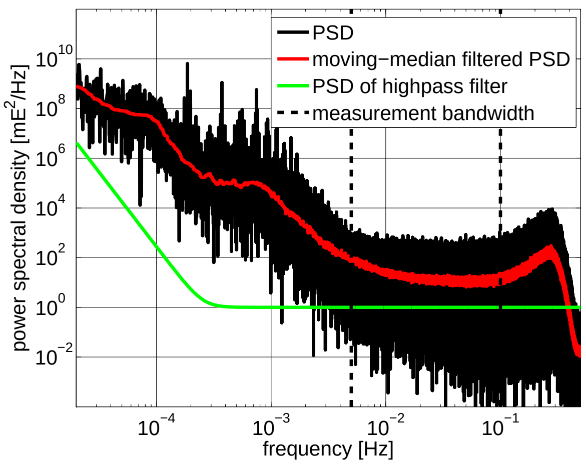
Step 3: Butterworth Highpass Filter Based on the median power spectral density, we design a Butterworth highpass filter by cut and try (cf. figure 6.8(b)). We have to take care, that the filter does not underestimate the power spectral density.

Step 4: Smoothed Power Spectral Density We compute a smoothed estimate of the power spectral density of the correlated time series by means of Welch's method (cf. figure 6.8(c)).

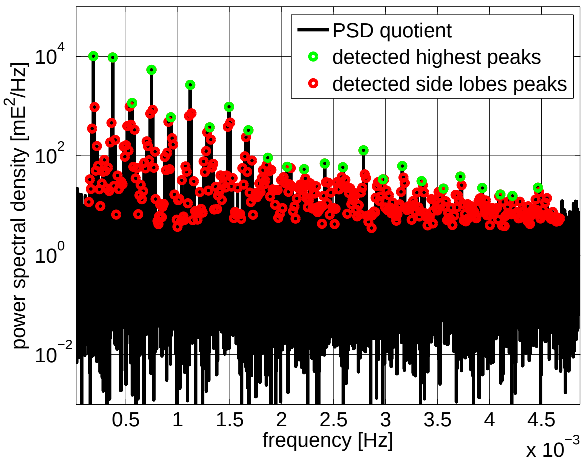
Step 5: Interpolated Median Power Spectral Density Now we interpolate the median power spectral density such that it is given for the same frequencies as the smoothed power spectral density. A linear interpolation is sufficient.



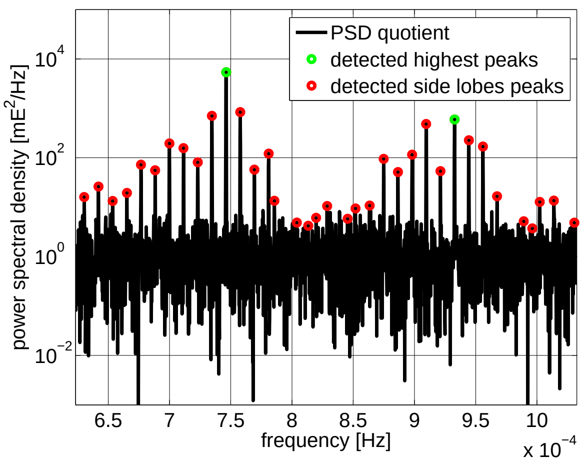
(a) The filter is designed to be very aggressive in those regions, where the peaks are.



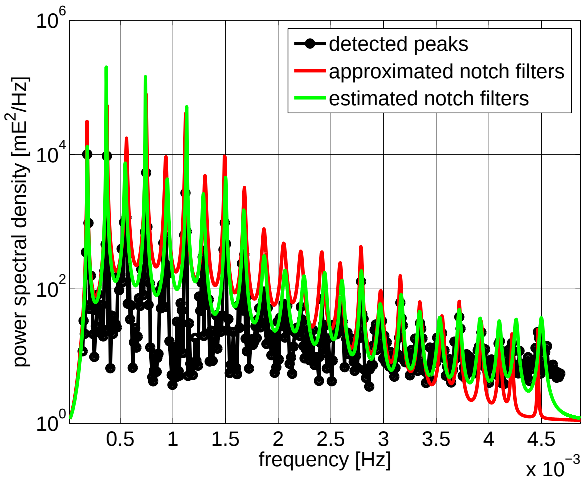
(b) The median PSD does not contain the peaks. The highpass filter is designed by cut and try.



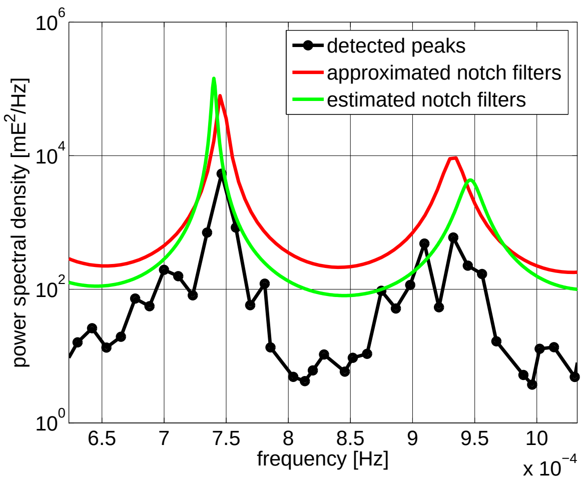
(c) Within the Quotient of the PSD and the median PSD the peaks can be easily detected.



(d) zoomed view of 6.6(c)



(e) The notch filters are estimated such that they are very aggressive.



(f) zoomed view of 6.6(e)

Figure 6.6: Intermediate products within the estimation of the aggressive filter (model B). Note, that always the inverse of the power spectral density of the filter is depicted for simplicity.

Step 6: Quotient of Power Spectral Densities The peaks of the power spectral density are isolated by dividing the smoothed power spectral density by the interpolated median power spectral density (cf. figure 6.8(d)).

Step 7: Detect Peaks Now, we detect the main peaks in the quotient of the power spectral densities (cf. figure 6.8(d)). Due to the smoothing, there are no side lobes peaks.

Step 8: Approximated Notch Filters Based on the frequency and the value of the power spectral density of the detected peaks, we chose the notch frequencies and scaling factor. In addition we chose a suitable bandwidth by cut and try (cf. figure 6.8(e)).

Step 9: Estimated Notch Filters The approximated notch filters are the basis for the estimation of the actual notch filters (cf. figure 6.8(e)). Within the estimation we use the weights according to equation (5.68).

Step 10: Filtered Time Series We filter the correlated time series by the Butterworth highpass filter and the estimated notch filters.

Step 11: Remove Warm-Up After having filtered the time series, we need to remove the warm-up of the combined Butterworth highpass and estimated notch filters.

Step 12: General ARMA Filter Based on the filtered time series, we estimate a general ARMA filter (cf. figure 6.8(f)).

Step 13: Averaging Filter The averaging filter is the combination of the Butterworth highpass filter, the general ARMA filter and the estimated notch filters (cf. figure 6.8(a)).

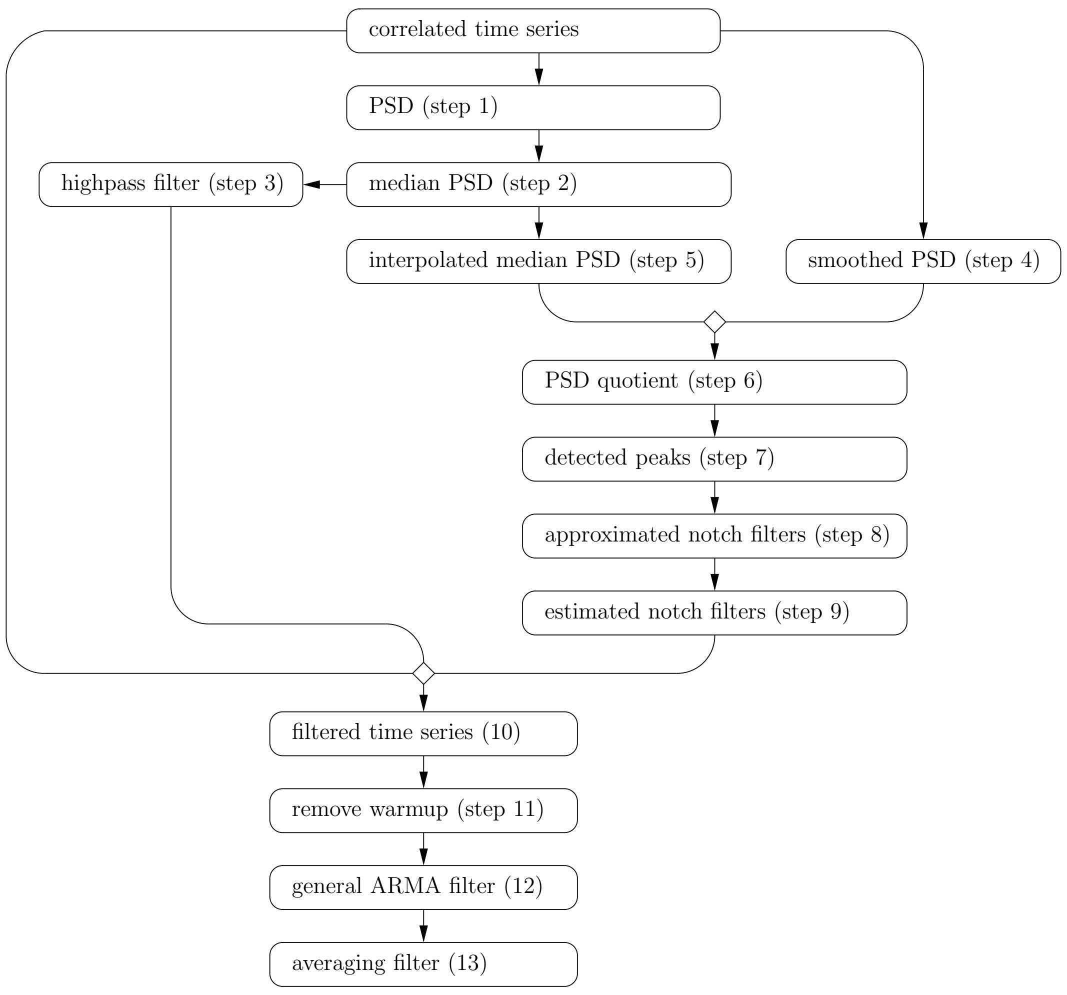
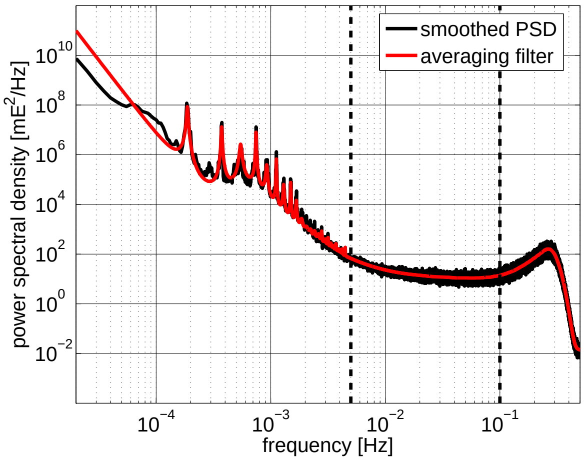
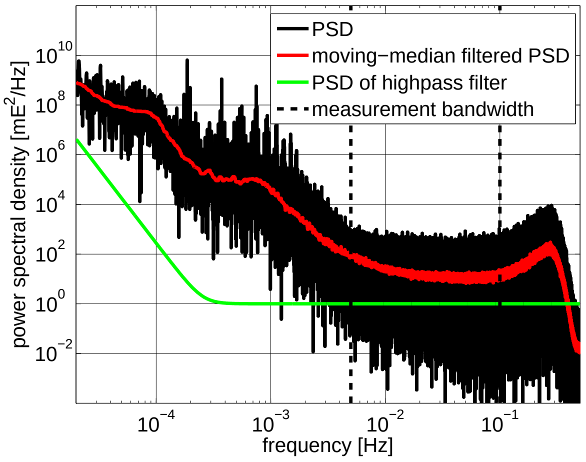


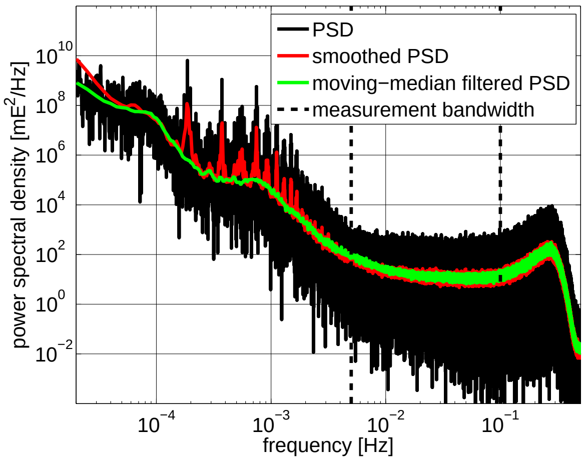
Figure 6.7: Sequence of steps within the estimation of the averaging filter model C.



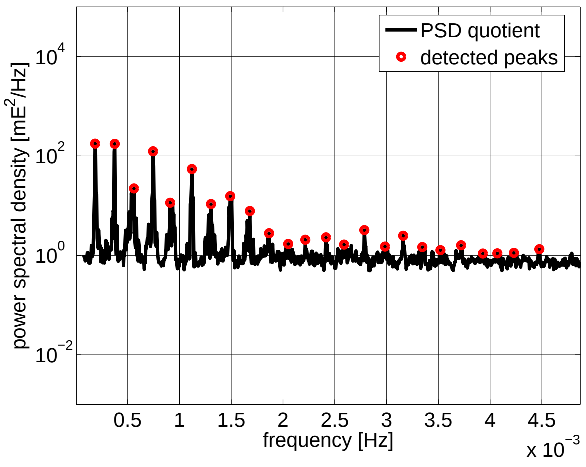
(a) The PSD estimated by Welch's method is quite smooth. The averaging filter is designed to adjust to this smooth estimate of the PSD.



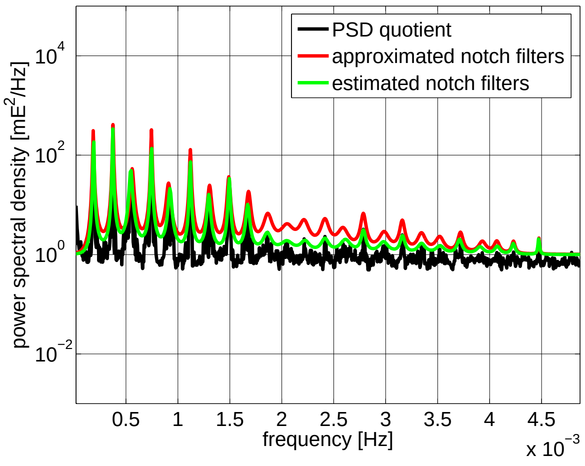
(b) The median PSD does not contain the peaks. The highpass filter is designed by cut and try.



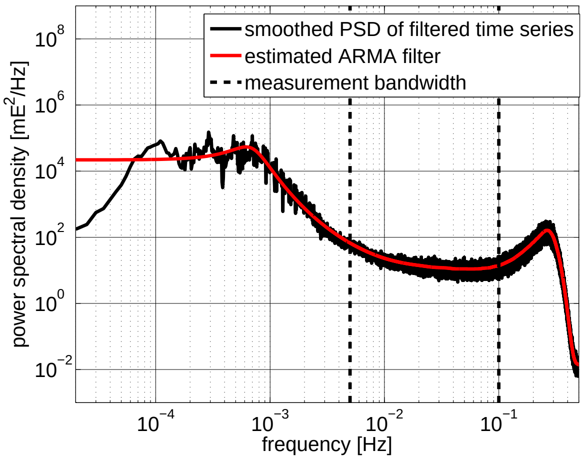
(c) The median PSD and the smoothed estimate of the PSD distinguish from each other only by the peaks.



(d) Within the Quotient of the smoothed PSD and the median PSD the peaks can be easily detected.



(e) The notch filters are estimated such that they are very aggressive.



(f) The remaining correlations after the application of the highpass and notch filters are modeled by a general ARMA filter.

Figure 6.8: Intermediate products within the estimation of the averaging filter (model C). Note, that always the inverse of the power spectral density of the filter is depicted for simplicity.

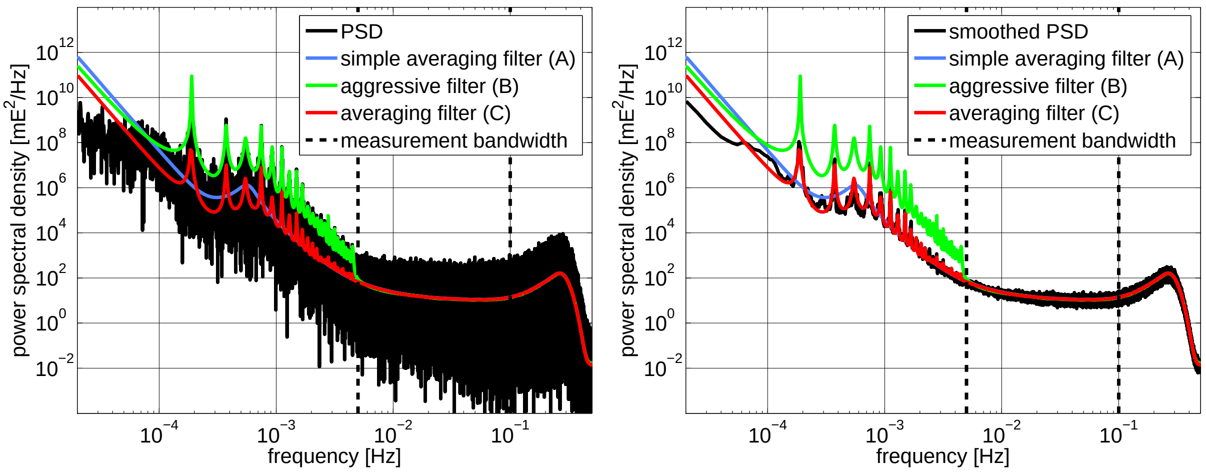


Figure 6.9: Comparison of the different filter models for the measurement noise of the V_{zz} component. The left panel shows the power spectral density of the measurement noise and the filters. The right panel shows a smoothed estimate of power spectral density of the measurement noise and the power spectral density of the filters. The simple averaging filter is optimized concerning the computational effort for filtering. The aggressive filter is designed to down-weight the peaks below the measurement bandwidth as much as reasonably possible. The averaging filter is a compromise between the other two filters concerning its decorrelation capabilities. Its computational effort for filtering is approximately the same as for the aggressive filter. Note, that always the inverse of the power spectral density of the filter is displayed for simplicity.

6.5 Simulation Results for Comparison of Filter Designs

The goal of this simulation is to identify the best filter design for the test data set. Therefore, the test data set described in section 6.3 has been processed using each of the three different filter models described in section 6.4. It should be mentioned that regularization (cf. section C.1) and variance component estimation (cf. section C.2) of the PCGMA algorithm were used within all three computations. Within this section the results of these three computations are compared to each other. The comparison is performed with respect to:

- the gravity field solution in terms of quasi geoid heights,
- the spherical harmonic coefficients, i.e. the parameters of the least squares solution, and
- the residuals of the least squares adjustment.

Quasi geoid heights are obtained in the following manner. First, the potential U of a rotational ellipsoid is subtracted from the potential V in order to obtain the so-called disturbing potential T.

$$T = V - U \tag{6.39}$$

Then, the disturbing potential T is divided by the normal gravity γ of the reference ellipsoid (cf. Heiskanen and Moritz 1967), which yields the quasi geoid heights

$$N = \frac{T}{\gamma} = \frac{V - U}{\gamma}. \tag{6.40}$$

As the test data set is simulated, we know the true quasi geoid heights $N^{(\text{true})}$. These quasi geoid heights are then compared to the quasi geoid heights $N^{(\text{estimated})}$ resulting from the three computations using the different filter models by looking at their difference

$$\Delta N = N^{(\text{estimated})} - N^{(\text{true})}, \tag{6.41}$$

which is shown in figure 6.10. Furthermore, the estimated quasi geoid heights $N^{(\text{estimated})}$ are compared to each other also in figure 6.10. It should be mentioned that the true quasi geoid heights $N^{(\text{true})}$ are computed using only the spherical harmonic coefficients up to degree and order 255, because this is the same maximum degree and order as used within the adjustment procedure.

The table in figure 6.10 shows that the overall performance of the models A, B and C (cf. section 6.4) is more or less the same. We can find significant differences between the models only by investigating the differences between the estimated quasi geoid heights. From figure 6.10(f) we can see that the simple averaging filter (model A) and the averaging filter (model C) produce only slightly different quasi geoid heights, which is confirmed by the statistics in the table. But the quasi geoid heights obtained with the aggressive filter (model B) show a distinctive pattern of the magnitude $\pm 2\,\text{cm}$. Comparing this pattern to the pattern in figure 6.4(c), it becomes obvious, the aggressive down-weighting of the geographically dependent error within the SGG observations has an impact on the gravity field solution. Looking closely at the figures 6.10(a), 6.10(c) and 6.10(e), we find that the geographically dependent error is contained in the solutions A and C, but not in solution B. In order to verify this, the quasi geoid differences in figure 6.10 were lowpass filtered by a Gauss filter with a radius of 200 km (cf. Wahr et al. 1998). Figure 6.11 shows the results of this lowpass filtering. Model A and C clearly contain the characteristic pattern of the geographically dependent error within the SGG observations. In model B the error is also present, but on a much lower scale. From this we can already conclude, that the design of the aggressive filter (model B) was successful.

Next, we investigate the spherical harmonic coefficients s_{nm} and c_{nm} introduced in equation (6.1). We do this comparison in two ways. First, we compute the absolute difference of the true coefficients $s_{nm}^{(\text{true})}$ and $c_{nm}^{(\text{true})}$

and the estimated coefficients $s_{nm}^{(\text{estimated})}$ and $c_{nm}^{(\text{estimated})}$ and normalize it by dividing it by the corresponding standard deviation $\sigma_{s_{nm}}$ and $\sigma^2_{c_{nm}}$.

$$\Delta s_{nm}^{(\text{normalized})} = \frac{|s_{nm}^{(\text{true})} - s_{nm}^{(\text{estimated})}|}{\sigma_{s_{nm}}}, \quad \Delta c_{nm}^{(\text{normalized})} = \frac{|c_{nm}^{(\text{true})} - c_{nm}^{(\text{estimated})}|}{\sigma_{c_{nm}}} \tag{6.42}$$

The variances $\sigma_{s_{nm}}$ and $\sigma_{c_{nm}}$ are computed by means of the inverse preconditioning matrix $\boldsymbol{K}$ (cf. section C.3). This means that they are only approximated as the preconditioning matrix is an approximation of the normal equation matrix. However, if this approximation is of good quality, the derived variances will be of good quality, too. The normalized errors $\Delta s_{nm}^{(\text{normalized})}$ and $\Delta c_{nm}^{(\text{normalized})}$ are shown in the left panels of figure 6.12. We can see that the zonal coefficients are poorly determined due to fact, that the ground tracks of the GOCE satellite do not cover the polar caps. Furthermore, the normalized errors of model A and model C look very much the same. Only model B shows visible differences. Between degree 30 and 100 there are areas at the edges of the coefficient triangle, which are plotted in a darker blue than the rest. This means that the sectoral coefficients and those tesseral coefficients, which are close the sectoral coefficients, of model B turn out to have a smaller error, than their variance indicates between degrees 30 to 100. Looking very closely at the left panels of figure 6.12, we see that the zonal coefficients of model B show higher errors than those of model A and C between degrees 2 to 20. This becomes more clearly visible in the right panels of figure 6.12. These panels show the median degree error

$$\Delta_n^{(\text{median})} = \operatorname*{median}_m(|s_{nm}^{(\text{true})} - s_{nm}^{(\text{estimated})}|, |c_{nm}^{(\text{true})} - c_{nm}^{(\text{estimated})}|), \quad \text{where } m \in \{0, \ldots, n\}, \tag{6.43}$$

mean degree error

$$\Delta_n^{(\text{mean})} = \frac{1}{2n+1}\sum_{m=0}^{n}(|s_{nm}^{(\text{true})} - s_{nm}^{(\text{estimated})}| + |c_{nm}^{(\text{true})} - c_{nm}^{(\text{estimated})}|), \tag{6.44}$$

and maximum error

$$\Delta_n^{(\max)} = \max_m(|s_{nm}^{(\text{true})} - s_{nm}^{(\text{estimated})}|, |c_{nm}^{(\text{true})} - c_{nm}^{(\text{estimated})}|), \quad \text{where } m \in \{0, \ldots, n\}, \tag{6.45}$$

in comparison to the median degree standard deviation

$$\sigma_n^{(\text{median})} = \operatorname*{median}_m(\sigma_{s_{nm}}, \sigma_{c_{nm}}), \quad \text{whereas } m \in \{0, \ldots, n\}, \tag{6.46}$$

and coefficient magnitude, given in terms of degree variances σ_n^2. The degree variances

$$\sigma_n^2 = \sum_{m=0}^{n}(s_{nm}^2 + c_{nm}^2) \tag{6.47}$$

can be approximated by Kaula's rule of thumb (cf. Kaula 1966), which states that

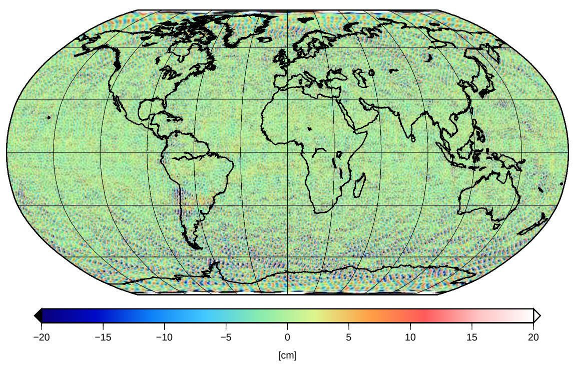
$$\frac{\sigma_n}{\sqrt{2n+1}} \approx \frac{10^{-5}}{n^2} = \sigma_n^{(\text{Kaula})}. \tag{6.48}$$

The term $\sigma_n^{(\text{Kaula})}$, which is depicted by the black line in figure 6.12, is an approximation of the magnitude of the coefficients. Because the median degree errors $\Delta_n^{(\text{median})}$ are a robust estimate of the accuracy, computed by the median degree standard deviation $\sigma_n^{(\text{median})}$, they have to be multiplied by the factor 1.483, before both are compared to each other (cf. Hettmansperger and McKean 1998, p. 199). This is performed in the right panels of figure 6.12. Furthermore, it should be mentioned, that though regularization was used within the adjustment, the median degree variance was computed without the regularization information, i.e. only with SGG and SST information. Again, the models A and C do not differ significantly. Model B shows a higher median degree error and median degree variance than model A and C between degrees 30 to 100. Between degrees 10 to 30 the median degree error and median degree variance of model B is lower than those of the models A and C. This can be seen more clearly in figure 6.13, where the median degree errors and the median degree variances of all three models are directly compared.

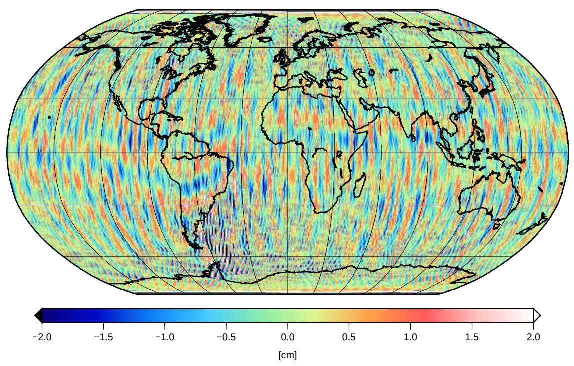
The last item for comparison are the residuals of the models. In order to perform a fair comparison, the estimated correlated residuals are compared to true correlated residuals. The comparison is based on the power spectral densities of the residuals time series. This enables us to investigate how well the correlations of the true residuals are recovered by the model residuals. From figure 6.14 and 6.15 we can see that the correlations are very well recovered by all three models. The blue curve shows the relative error of the power spectral densities, which is defined to be

$$P^{(\text{error})}(f) = \max\Big(\frac{P^{(\text{true})}(f)}{P^{(\text{estimated})}(f)}, \frac{P^{(\text{estimated})}(f)}{P^{(\text{true})}(f)}\Big). \tag{6.49}$$

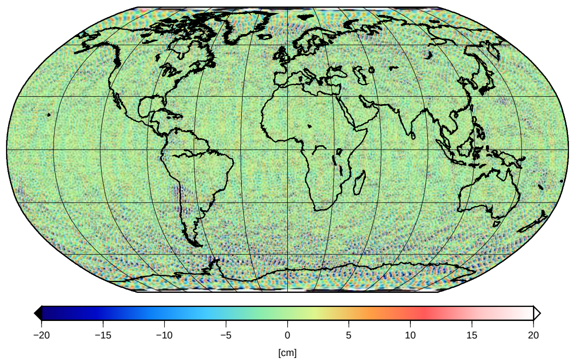
The relative error of the power spectral densities thus is a factor of misjudgment. Figure 6.14 and 6.15 shows, that this error is only high in the frequency range of the peaks. Apparently, the first main peak at approximately $f = \frac{1}{\text{revolution}} \approx \frac{1}{5400}$ Hz is not recovered at all by all three models. This means that this error has entered the gravity field solution completely. Even the aggressive filter model could not prevent this.



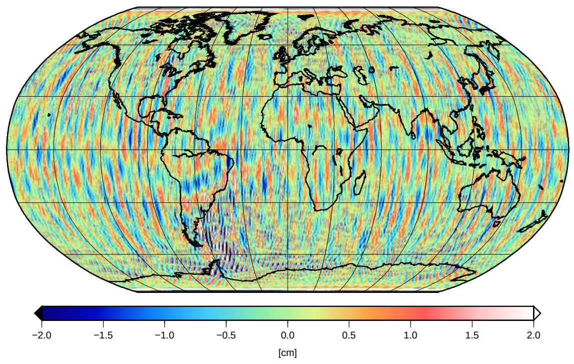
(a) model A minus EGM96



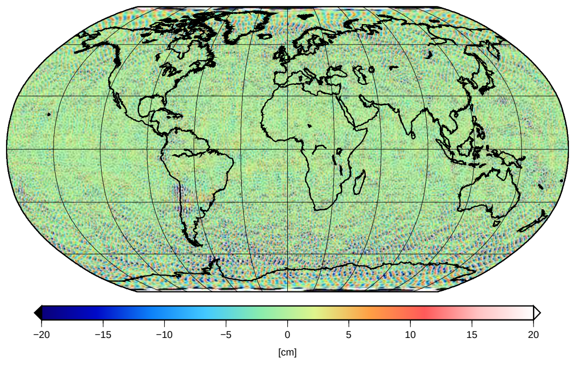
(b) model A minus model B



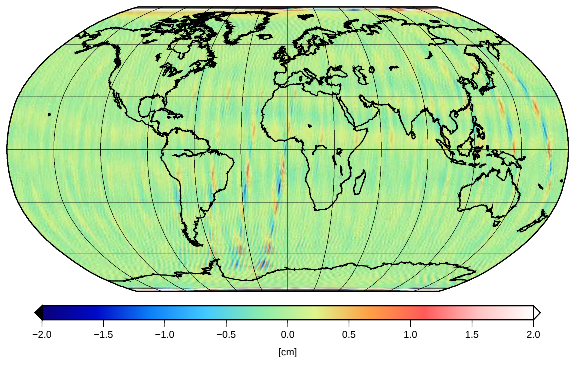
(c) model B minus EGM96



(d) model C minus model B



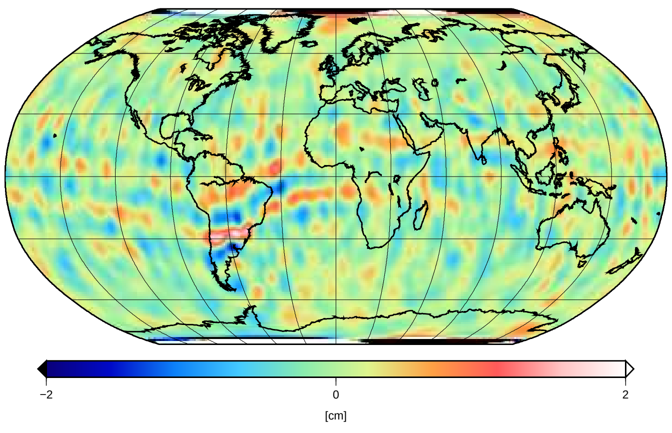
(e) model C minus EGM96



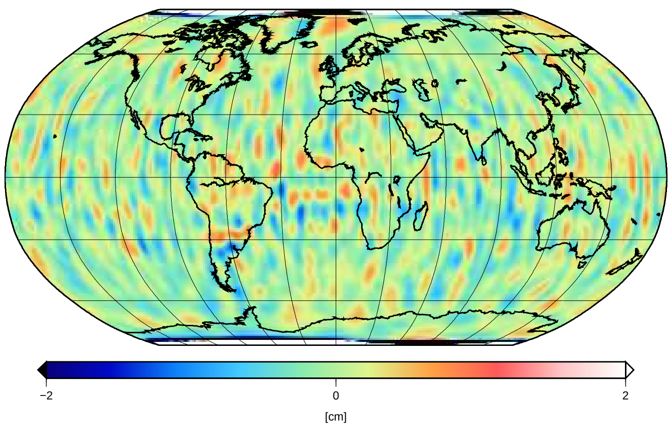
(f) model C minus model A

sector	model	min	max	mean	rms	model	min	max	mean	rms
$\pm 90°$	A	-3.602 m	3.307 m	-2.1 cm	34.3 cm	A - B	-0.977 m	0.317 m	-1.7 cm	11.7 cm
$\pm 80°$		-0.639 m	0.599 m	0.0 cm	9.3 cm		-0.041 m	0.046 m	0.0 cm	0.9 cm
$\pm 90°$	B	-4.534 m	32.82 m	-3.8 cm	41.1 cm	C - B	-1.124 m	0.219 m	-2.4 cm	15.6 cm
$\pm 80°$		-0.647 m	0.601 m	0.0 cm	9.4 cm		-0.042 m	0.045 m	0.0 cm	0.8 cm
$\pm 90°$	C	-3.532 m	3.341 m	-1.4 cm	32.7 cm	C - A	-0.567 m	0.249 m	-0.7 cm	5.7 cm
$\pm 80°$		-0.637 m	0.597 m	0.0 cm	9.3 cm		-0.029 m	0.026 m	0.0 cm	0.2 cm

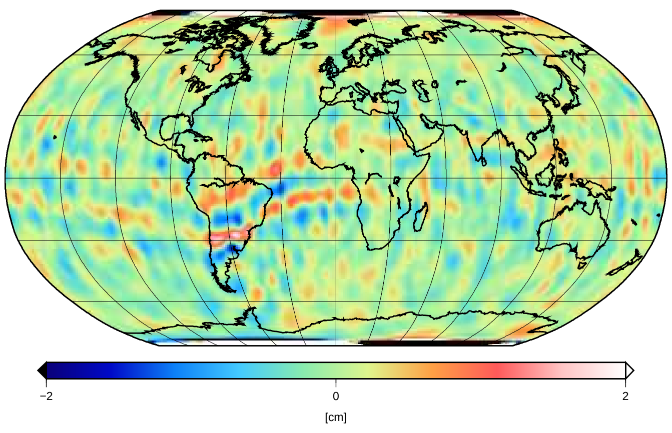
Figure 6.10: Comparison of quasi geoid heights according to equation (6.41). The quasi geoid heights are compared up to degree and order 255. Model A and model C do not differ much. The difference between model A and model B as well as the difference between model C and model B show a distinctive pattern, which results from the geographically dependent error of the SGG observations shown in figure 6.4. As model B uses the aggressive filter, this error is down-weighted and consequently has not much influence on the gravity field solution of model B.



(a) model A minus EGM96

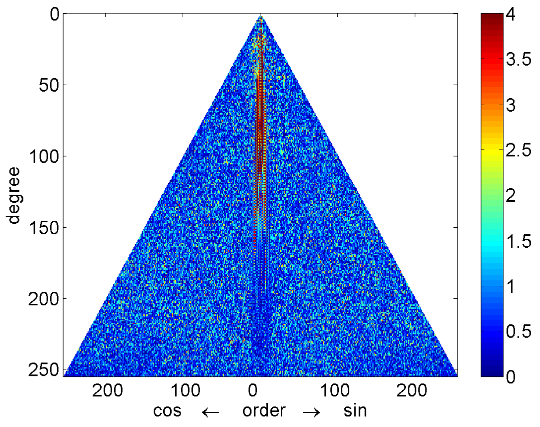


(b) model B minus EGM96

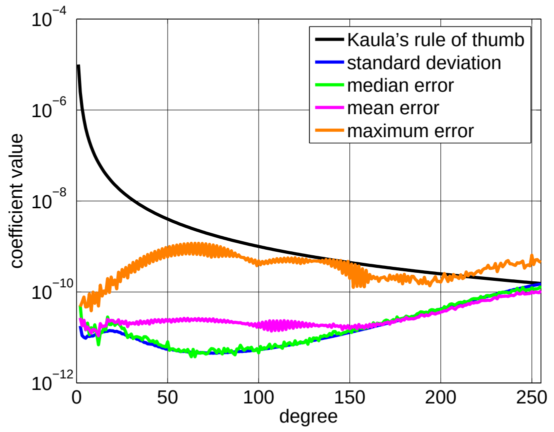


(c) model C minus EGM96

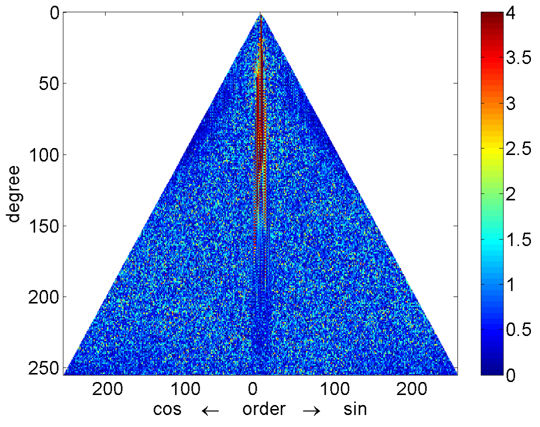
Figure 6.11: Comparison of lowpass filtered quasi geoid heights according to equation (6.41). Model A and C contain a characteristic pattern along the equator. This pattern is very similar to the geographically dependent error within the SGG observation shown in figure 6.4(c). For this reason we can conclude that this systematic error of the SGG observations has affected the gravity field solution of model A and C. In model B the effect is still present, but on a much lower scale. We would not recognize the effect, if we did not know that it is present. Therefore, we can conclude that the down-weighting of geographically dependent errors within the SGG observation by the decorrelating filters was successful and, consequently, that the model B provides better results.



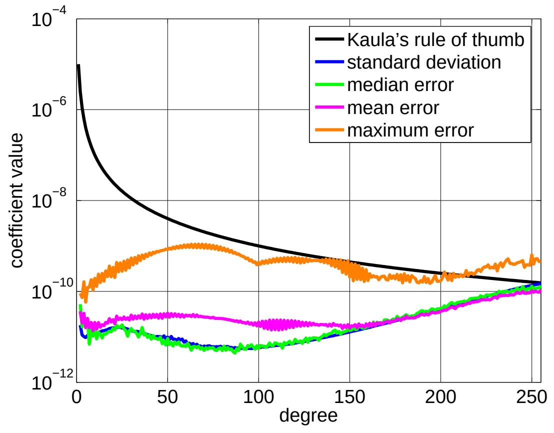
(a) model A



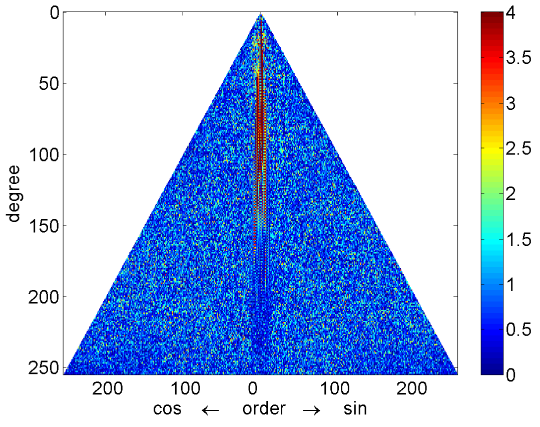
(b) model A



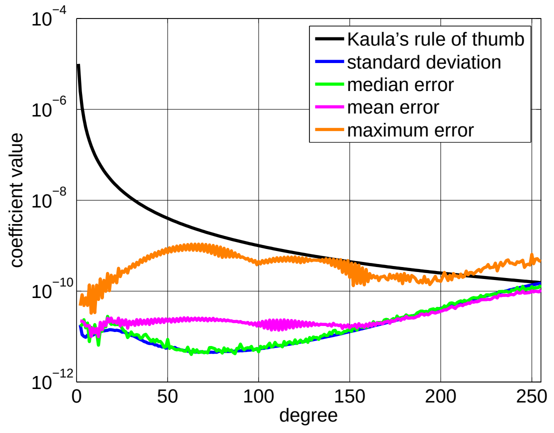
(c) model B



(d) model B

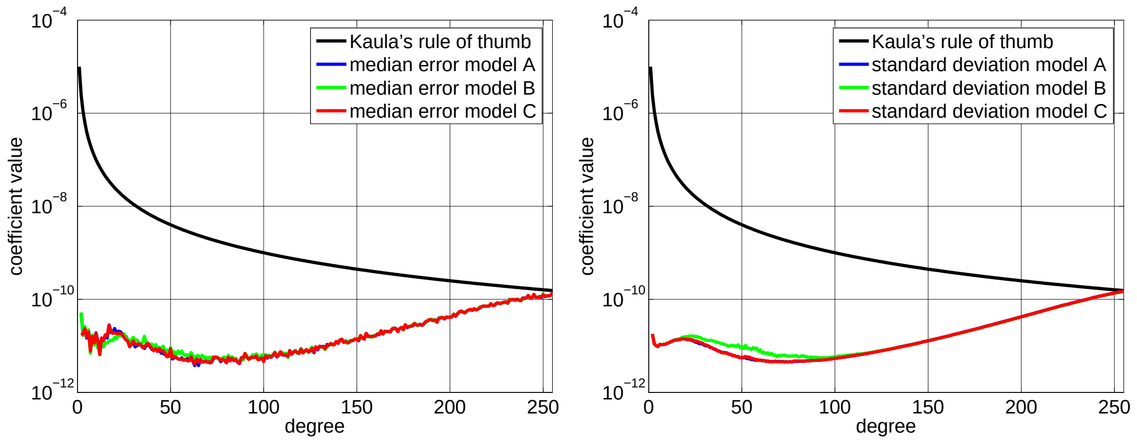


(e) model C



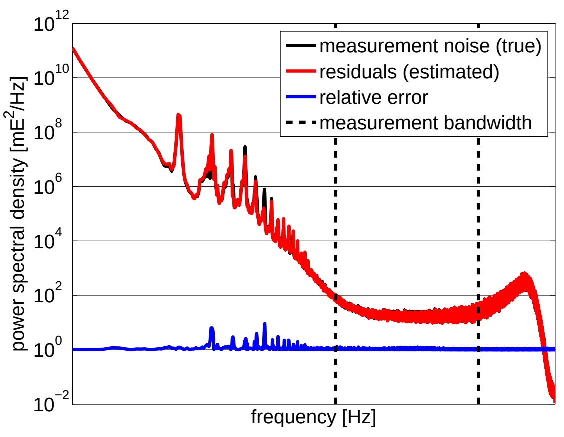
(f) model C

Figure 6.12: Normalized coefficient errors and degree evaluation of coefficients. The left panels show the normalized coefficient errors according to equation (6.42). The right panels show the median degree error according to (6.43), mean degree error according to (6.44), maximum error according to (6.43), median degree standard deviation according to (6.46) and the coefficient magnitude according to (6.47) and (6.48). Only between the degrees 10 to 100 model B differs from model A and C significantly. Model A and C appear to perform equally well.

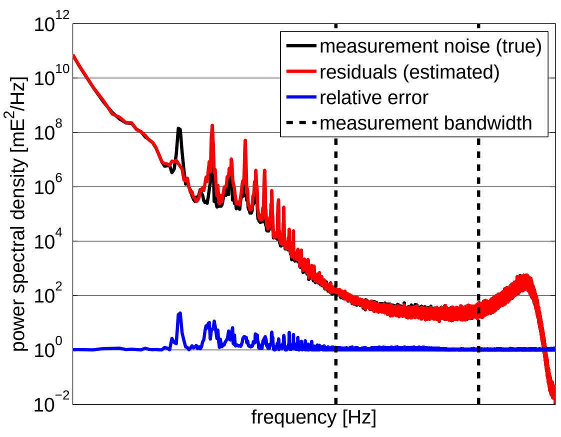


(a) median degree errors (b) median degree standard deviations

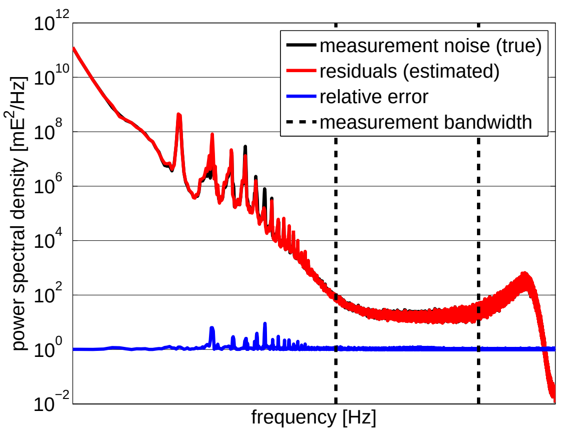
Figure 6.13: Direct comparison of median degree errors (6.43) and median degree standard deviations (6.46). Only between the degrees 10 to 100 model B differs from model A and C significantly. Model A and C appear to perform equally well. The use of the aggressive filter in model B results in higher errors and standard deviations between degrees 30 to 100. However, the errors of model B between degrees 10 to 30 are lower than those of model A and C.



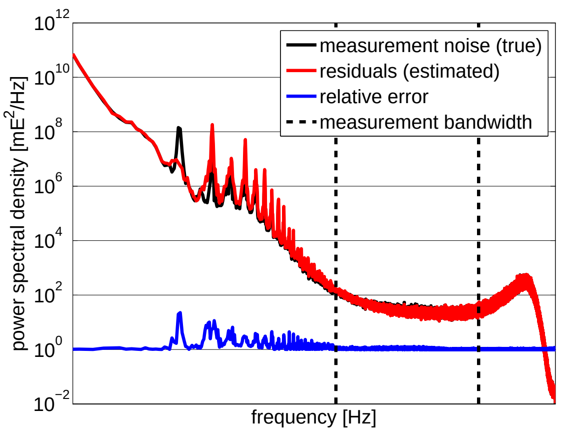
(a) XX Component, model A



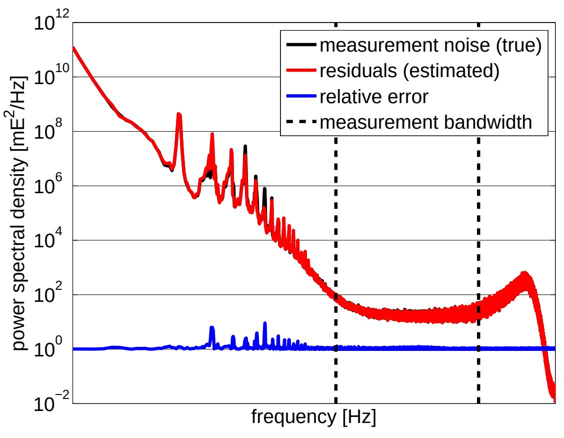
(b) ZZ Component, model A



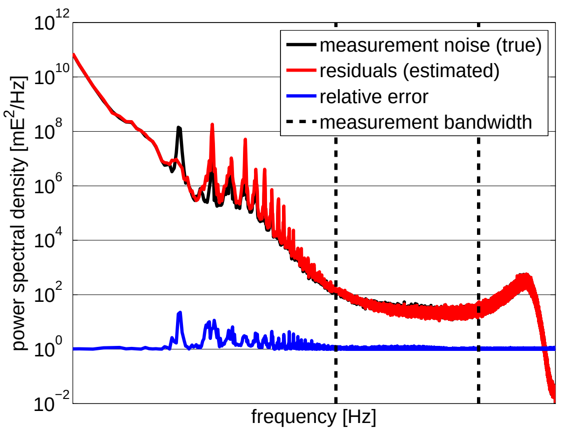
(c) XX Component, model B



(d) ZZ Component, model B

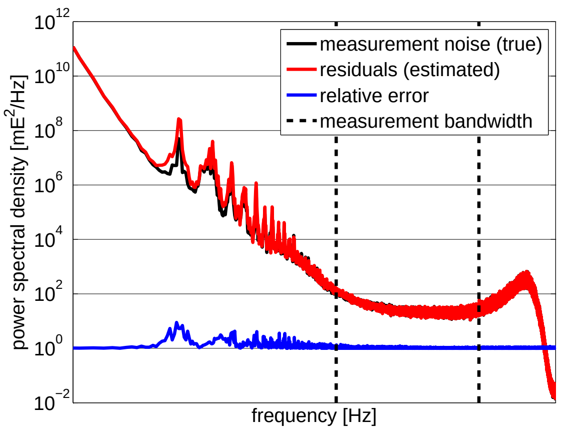


(e) XX Component, model C

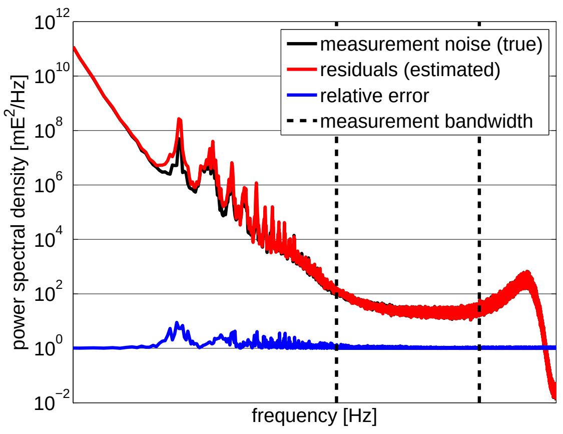


(f) ZZ Component, model C

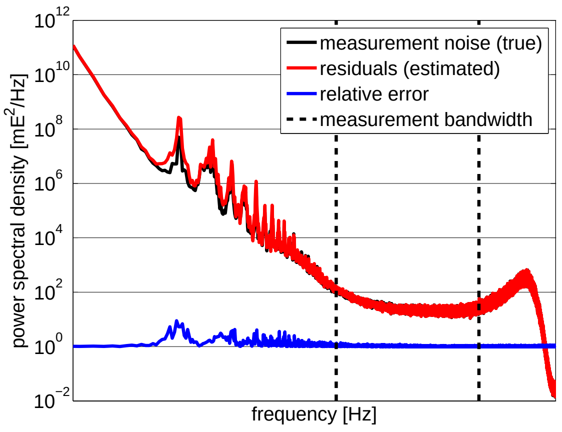
Figure 6.14: Recovery of the correlations investigated by means of the power spectral density of the residuals. As the power spectral density is the Fourier transform of the autocovariance function, this figure illustrates how well the correlations of the true residuals are recovered by the estimated residuals. There seem to be no significant differences between all three models. One interesting fact is that the first main peaks of the ZZ component of the SGG observations cannot be recovered by the estimated residuals. Thus, this error enters the gravity field solution. Confer also figure 6.15.



(a) YY Component, model A



(b) YY Component, model B



(c) YY Component, model C

Figure 6.15: Recovery of the correlations investigated by means of the power spectral density of the residuals. As the power spectral density is the Fourier transform of the autocovariance function, this figure illustrates how well the correlations of the true residuals are recovered by the estimated residuals. There seem to be no significant differences between all three models. Confer also figure 6.14.

6.6 Simulation Results for Short and Long Data Gaps

Within this section we investigate whether the developed procedures for short and long data gaps work, after integrating them into the whole processing. For this reason some of the SGG observations have been declared as data gaps. All other data, i.e. the SST observations as well as gradiometer positions and orientations, are assumed to be without any gaps for simplicity. For the decorrelation the simple averaging filter (model A, cf. section 6.4) was chosen, because it needs by far the least calculation time. Furthermore, the deterministic part of the fill-in values for short data gaps was set equal to the gravity field solution without gaps of model A. This way the stochastic part of the fill-in values is computed under ideal conditions, such that it is the only item under test besides the method for long data gaps.

Altogether, there are 105 data gaps in the SGG observations, whereas 5 are long data gaps and 100 are short data gaps. The size of long data gaps varies from 59 999 records to 189 999 records while the size of short data gaps varies from 1 to 1779 records. One record contains three SGG observations: V_{xx}, V_{yy} and V_{zz}. Table 6.5 summarizes the settings for the simulation.

filter model	simple averaging filter (model A)
deterministic part of fill-in values	solution of model A without gaps
stochastic part of fill-in values	computation based on the simple averaging filter and the residuals of model A without gaps
number of SGG data gaps	105
number of missing SGG observations	680434 records = 2041302 SGG observations
number of short SGG data gaps	100
number of missing SGG observations	50439 records = 151317 SGG observations
number of long SGG data gaps	5
number of missing SGG observations	629995 records = 1889985 SGG observations
number of other missing data	none

Table 6.5: Settings for the simulation concerning short and long data gaps.

For the analysis of the gravity field solution it is not only interesting how many SGG observations are missing, but also where these observations are located geographically. We expect the gravity field solution to be degraded compared to the solution without gaps exactly at those geographic locations, where many observations are missing. Figure 6.16 shows the geographic distribution of the missing SGG observations. The stripes within the figure result from the ground tracks of the satellite orbit. Especially at latitude $\pm 30^\circ$ the ground tracks for the missing data cross each other. Therefore, we can expect the gravity field solution to be degraded at these locations. Furthermore, near the polar regions the location of missing data becomes more dense than figure 6.16 indicates. Note, that the chosen geographical mapping draws the pole as a line. Thus, near the polar region we can expect the gravity field solution to be degraded as well.

Figure 6.17 shows the simulation results in terms of quasi geoid heights of the gravity field solutions. The gravity field solution with gaps is compared to the corresponding solution without gaps. The table in figure 6.17 shows that the solution with gaps is less accurate than the solution without gaps, which can be explained by the fact, that a smaller number of observations was used in the adjustment. Furthermore, the degradation of the gravity field solution is exactly at the geographic locations, where we expect it.

Finally, we look at the decorrelated residuals directly after a short and a long data gap. Within both the residuals directly after a short and a long data gap, there is no indication of warm-up effects. Therefore, we can conclude that the computation of the fill-in values for the stochastic part was successful and that the filter method for the warm-up works well.

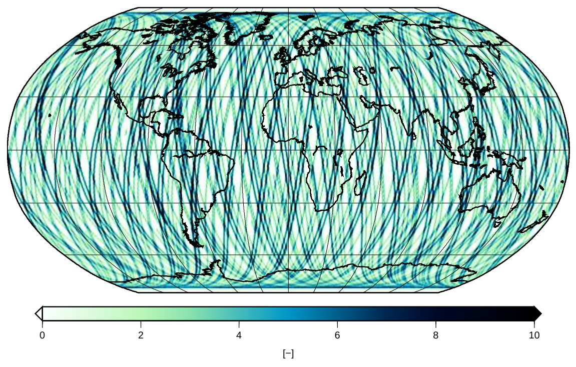
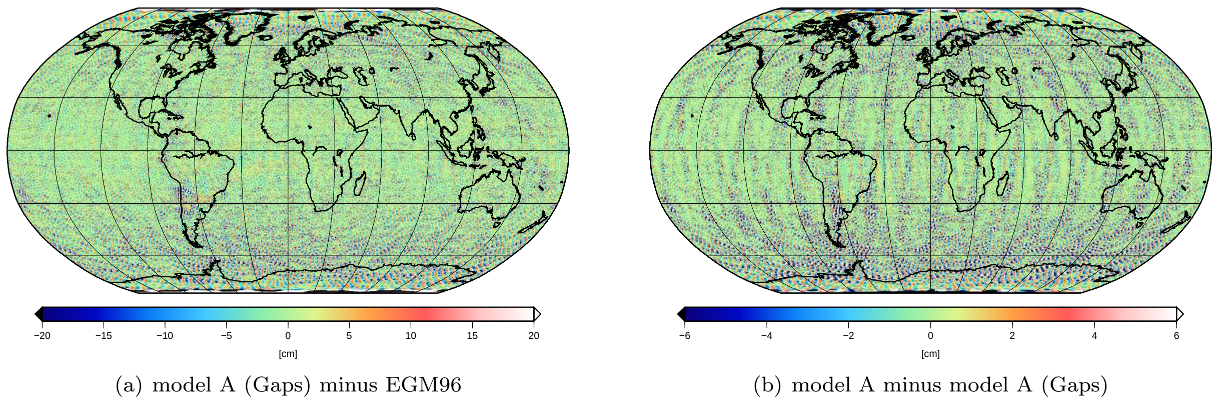


Figure 6.16: Location and number of missing SGG observations due to data gaps. The minimum number is 0 missing SGG observations and the maximum number is 29 missing SGG observations in one square of 0.5°× 0.5°.

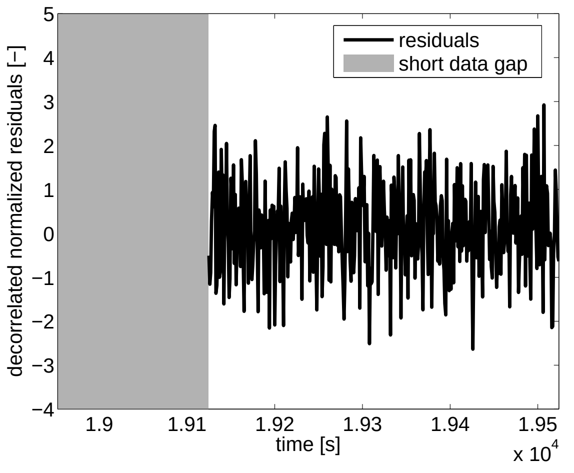


(a) model A (Gaps) minus EGM96

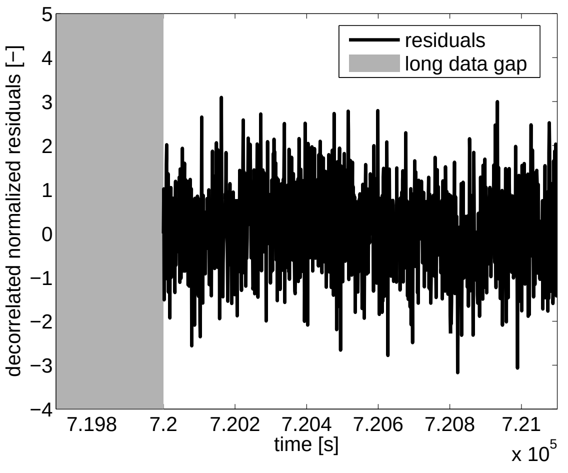
(b) model A minus model A (Gaps)

sector	model	min	max	mean	rms
±90°	A (Gaps)	-3.585 m	3.348 m	-1.7 cm	33.3 cm
±80°		-0.815 m	0.703 m	0.0 cm	9.9 cm
±90°	A	-3.602 m	3.307 m	-2.1 cm	34.3 cm
±80°		-0.639 m	0.599 m	0.0 cm	9.3 cm
±90°	A (Gaps) - A	-0.268 m	0.253 m	-0.4 cm	4.6 cm
±80°		-0.268 m	0.253 m	0.0 cm	3.5 cm

Figure 6.17: Comparison of quasi geoid heights according to equation (6.41). The left panel shows the differences of the solution in the case of data gaps to the true quasi geoid heights, computed using spherical harmonic coefficients up to degree and order 255. The right panel shows the differences of the solution in the case of data gaps to the solution in the case of no data gaps. The geographic pattern of the degradation can be explained by the geographic pattern of the missing values (cf. figure 6.16).



(a) decorrelated residuals after a short gap



(b) decorrelated residuals after a long gap

Figure 6.18: Decorrelated residuals directly after data gaps. As there are no signs of any warm-up effects for both the residuals directly after a short and a long data gap, we can conclude that the algorithm for the computation of fill-in values for the stochastic part of the SGG observations as well as the filter method for the warm-up work well.

7. Conclusions

In this chapter we start by making a short summary of the major results for each chapter. After this, we highlight the contributions of this thesis. Finally, we finish this thesis by an outlook.

7.1 Summary of results

Chapter 1 We started by describing the application scenario. There are two problems concerning a least squares adjustment due to the large number of observations and parameters: The design matrix is very large and the covariance matrix of the observations is even larger. The first problem is solved by parallel computing while the second problem is solved by tailored filter algorithms. The latter is based on finding a sparse representation of the covariance matrix. However, this is only possible, if the observations meet certain conditions.

Chapter 2 This chapter served as the basis for the algorithms and methods developed in this thesis. Here, the necessary conditions for the observations were listed: time series, normal distribution, stationarity, ergodicity and correlation length. If any of these conditions are not met, from a theoretical point of view the methods and algorithms developed in this thesis should not be applied. Based on the conditions we found three approaches of sparsely representing the covariance matrix: the Toeplitz approach, the AR process approach and the ARMA process approach. The first action, was performing variance propagation in order to obtain the covariance matrix, which results from using the approaches. After that, we derived the direct link between the covariance matrix of the observations and the filter coefficients. With this link, we were able to formulate the design goal for the filters. An interesting fact is, that only the amplitude response is important for the decorrelation capabilities of the filters while the phase response can be left aside. Therefore, the power spectral density is of special concern.

Chapter 3 - Section 1 and 2 Here we investigated candidate methods for implementing the decorrelating filters. In principle there are four methods: the straightforward method, the block method, the overlap-add method and the FFT method. We identified the block method and the overlap-add method to be superior to the others concerning the numerical effort of filtering. Which one of these two methods should be preferred, depends on the length of the filter. If the filter is short, the block method should be used. If the filter is longer, the overlap-add method should be used. Of special concern was the ARMA block method, which allows for sophisticated filter designs to be implemented very fast and memory-saving at the same time. Here, we found clear rules for building filter sections, fast algorithms for the computation of the filter matrices and the filtering itself. We analyzed the block size for the ARMA block method, which resulted in the advice to determine the optimum block size empirically.

Chapter 3 - Section 3 Based on the different methods for implementing the decorrelating filters, we explored ways of integrating the filtering into least squares adjustment procedures. This included the Toeplitz approach, though it does not correspond to filtering. However, the Toeplitz approach turned out to perform worse with respect to the numerical effort and thus calculation time. For the process approaches there are two ways of integrating them into iterative adjustment procedures like the conjugate gradient algorithm. The first way is depicted by Schuh (1996) and involves the filtering of the design matrix in each iteration. The second way is depicted by Klees et al. (2003) and avoids to filter the design matrix in each iteration at the cost of computing the design matrix twice. Which one is faster depends on whether the filtering of the design matrix or the computation of the design matrix is faster. Furthermore, we investigated how to avoid to compute the design matrix twice and to avoid to filter the design matrix at the same time by synchronous filter methods. However, the synchronous filter methods can only be applied, if sufficient memory is available.

Chapter 4 This chapter was concerned with two special problems, which arise by the use of the filtering methods. The first problem is that the first section of filtered observations cannot be used within the least square adjustment due to a phenomenon, which we call the filter warm-up. In order to avoid to loose observations due to the filter warm-up, we developed a method to determine the length of the filter warm-up and, more important, a method to avoid the filter warm-up. The second problem arises in the case of short data gaps within the observation time series. Here, we developed a suitable algorithm to compute the stochastic part of the fill-in values.

Chapter 5 In the first part of this chapter we analyzed the correlations of the measurement noise of the observations to be decorrelated of the application scenario. We found out, that we needed three classes of filters: highpass filters, general MA or ARMA filters and notch filters. Consequently, we put a suitable set of tools together to design such filters.

Chapter 6 In this chapter we introduced the application scenario and the test data sets in detail. We investigated, which of the filter methods should be used. It turned out that, due to memory demands and the high correlations of the observations, the only applicable method was the ARMA block method. Furthermore, due to the sophisticated filter designs, the filtering of the design matrix within the iterations should be avoided. In addition, we introduced the PCGMA algorithm for solving the least squares problem and highlighted that a different load balancing should be used for the computation of the preconditioning matrix and the rest of the algorithm. Finally, we performed simulations in order to find out which filter design provides the best results and to investigate, if the developed filter methods work well in the case of data gaps. The simulation results indicated, that the aggressive filter design was the best, because it could sufficiently down-weight a geographically dependent error of the observations, and that the filter methods work well in the case of data gaps.

7.2 Contributions of this Thesis

The first contribution of this thesis is the **ARMA block filter method**. The basic ideas of Burrus (1972) were picked up to develop a very fast and memory-saving filter algorithm. Especially the fact, that the algorithm is optimized by means of the BLAS/ATLAS library for filtering large matrices, makes it very valuable for the application scenario described in chapter 6. Another important contribution of this thesis is the solution to the problem of the filter warm-up. In this context a **method for the determination of warm-up length** was developed as well as a **method to prevent the filter warm-up**. Thanks to these methods no observations are lost anymore due to the filter warm-up. One special problem was to compute fill-in values for short data gaps. Previous investigations by Klees and Ditmar (2004) showed, that neglecting the stochastic part of the observations leads to bad fill-in values. For this reason, a suitable **algorithm for the computation of the stochastic part of the fill-in values for short data gaps** was developed in this thesis. Finally, different filter designs were compared to each other with respect to the gravity field recovery. The contribution of this part is the **identification of the best filter design** in this context.

7.3 Outlook

Though many problems were solved in this thesis, there is still room for further research. For example the design of suitable filters requires a great deal of interaction with the operator, who designs the filter. Here, one may try to advance techniques to automate the design process. This is not as easy as it seems, because we have to teach the computer, which feature of the power spectral density should be modeled and which should not. Another point is the possibility to take advantage of having access to the decorrelated residuals within the adjustment procedure. For example, we may investigate how the PCGMA algorithm can be transformed from a least squares solver into a robust solver by weight-iteration for the residuals. Especially the fast filter algorithm will be of advantage here. Of course, it will be interesting to transfer the developed methods to other application scenarios and to see, how they perform. Finally, the launch of the GOCE satellite is planed for May 2008. To process the real GOCE mission data instead of simulated test data will reveal the true potential of the gradiometer on board the GOCE satellite. In this spirit, the author likes to finish this thesis by quoting Ernst Ferstl, an Austrian teacher, poet and aphorist:

The difference between theory and practice is in practice much larger than in theory.

Translated from the original quote by Ernst Ferstl: *Der Unterschied zwischen Theorie und Praxis ist in der Praxis weit höher als in der Theorie. (cf.* FERSTL 1996*)*

List of Figures

List of Tables

List of Algorithms

A. Tables of Symbols

general

$\Re(x)$	real part of complex variable x
$\Im(x)$	imaginary part of complex variable x
$\sphericalangle(x)$	angle of complex variable x, i.e. $\arctan \frac{\Im(x)}{\Re(x)}$
$\lvert x \rvert$	absolute value of x or radius of complex variable x, i.e. $\sqrt{\Im^2(x) + \Re^2(x)}$
x_n	time series (always defined for $-\infty < n < \infty$)
$\circledast$	cyclic convolution
$*$	convolution
α, β, ...	angles (also true values)
$\widetilde{a}$, $\widetilde{b}$, ...	best linear unbiased estimates
$\widehat{a}$, $\widehat{b}$, ...	arbitrary estimates
varepsilon	numerical zero

transforms

$\mathcal{F}\{x_n\}$	discrete Fourier-transform (DFT) of time series x_n (result is continuous with respect to frequency)
$\mathcal{F}\{\boldsymbol{x}\}$	discrete Fourier-transform (DFT) of vector $\boldsymbol{x}$ (result is discrete with respect to frequency)
$\mathcal{F}^{-1}\{y_n\}$	discrete inverse Fourier-transform (IDFT) of time series y_n (result is discrete with respect to frequency)
$\mathcal{F}^{-1}\{\boldsymbol{y}\}$	discrete inverse Fourier-transform (IDFT) of vector $\boldsymbol{y}$ (result is discrete with respect to frequency)
$\mathcal{F}^*\{\boldsymbol{x}\}$	complex conjugate of discrete Fourier-transform (IDFT) of vector $\boldsymbol{x}$
$\mathcal{Z}\{x_n\}$	Z-transform of time series x_n

samples, random values, true values

α, β, ...	true values (also angles)
$\mathcal{A}$, $\mathcal{B}$, ...	random variables
a, b, ...	sample values or analytic values
$E\{\mathcal{A}\}$, $E\{\mathcal{B}\}$, ...	expectation of random variables
$\boldsymbol{E}\{\boldsymbol{\mathcal{A}}\}$, $\boldsymbol{E}\{\boldsymbol{\mathcal{B}}\}$, ...	expectation of random vectors
$\Sigma\{\mathcal{A}\}$, $\Sigma\{\mathcal{B}\}$, ...	variance of random variables
$\boldsymbol{\Sigma}\{\boldsymbol{\mathcal{A}}\} = \boldsymbol{\Sigma}_{\boldsymbol{\mathcal{A}\mathcal{A}}}$, $\boldsymbol{\Sigma}\{\boldsymbol{\mathcal{B}}\} = \boldsymbol{\Sigma}_{\boldsymbol{\mathcal{B}\mathcal{B}}}$, ...	covariance matrix of random vectors

instructions

$\mathrm{diag}(\boldsymbol{X})$	diagonal of matrix $\boldsymbol{X}$, result is a standing vector
$\mathrm{diag}(\boldsymbol{x})$	diagonal matrix $\boldsymbol{X}$, constructed by vector $\boldsymbol{x}$
$\mathrm{select}(\boldsymbol{X})$	matrix, which contains only a subset of the elements of matrix $\boldsymbol{X}$ (not selected elements are set to zero)
$\mathrm{ceil}(x)$	value x rounded up to the next integer
$\mathrm{floor}(x)$	value x rounded down to the next integer
$\boldsymbol{x} = \mathrm{solve}(\boldsymbol{A}, \boldsymbol{b})$	computation of the solution of the equation system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ using a suitable algorithm

vectors and matrices

The indexes of vectors and matrices start by one, i.e.

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} X_{1,1} & \cdots & X_{1,n} \\ \vdots & & \vdots \\ X_{m,1} & \cdots & X_{m,n} \end{bmatrix}$$

x	scalar
$\boldsymbol{x}$	vector
$\boldsymbol{X}$	matrix
$\boldsymbol{x}_a$	element with index a of vector $\boldsymbol{x}$
$\boldsymbol{x}_{a:b}$	part of vector $\boldsymbol{x}$
$\boldsymbol{X}_{a:b,c:d}$	part of vector $\boldsymbol{x}$
$\boldsymbol{X}_{:,c}$	column c of matrix $\boldsymbol{x}$
$\boldsymbol{X}_{a,:}$	row a of matrix $\boldsymbol{x}$
$\boldsymbol{X}_{:,c:d}$	vertical slice of matrix $\boldsymbol{x}$, consisting of the columns c to d
$\boldsymbol{X}_{a:b,:}$	horizontal slice of matrix $\boldsymbol{x}$, consisting of the rows a to b
$\boldsymbol{X}^k_{a:b,c:d}$	part of matrix $\boldsymbol{X}^k$

vector and matrix operations

$\boldsymbol{z} = \boldsymbol{x} \circ \boldsymbol{y}$	element-wise multiplication: $\boldsymbol{z}_n = \boldsymbol{x}_n \cdot \boldsymbol{y}_n$ for each n
$\boldsymbol{z} = \boldsymbol{x} \diamond \boldsymbol{y}$	element-wise division: $\boldsymbol{z}_n = \frac{\boldsymbol{x}_n}{\boldsymbol{y}_n}$ for each n
$\boldsymbol{X}^{-1}$	inverse matrix of $\boldsymbol{X}$
$\boldsymbol{X}^\top$	transpose matrix of $\boldsymbol{X}$
$\boldsymbol{X}^k$	k-th power of matrix $\boldsymbol{X}$ ($\boldsymbol{X}^k = \boldsymbol{X} \cdot \boldsymbol{X} \cdot \ldots \cdot \boldsymbol{X}$)

B. Abbreviations

AR	AutoRegressive
ARMA	AutoRegressive Moving-Average
ATLAS	Automatically Tuned Linear Algebra Software
BLAS	Basic Linear Algebra Subprograms
BLUE	Best Linear Unbiased Estimator
CHAMP	CHAllenging Mini-satellite Payload
DFT	Discrete Fourier Transform
EGM96	Earth Geopotential Model 1996
ESA	European Space Agency
FFT	Fast Fourier Transform
GB	GigaBytes
GLONASS	GLObal Navigation Satellite System
GOCE	Gravity field and steady-state Ocean Circulation Explorer
GPS	Global Positioning System
GRACE	Gravity Recovery And Climate Experiment
GRF	Gradiometer Reference Frame
IDFT	Inverse Discrete Fourier Transform
ITG	former Institute for Theoretical Geodesy, since 2007 department for Theoretical Geodesy of the Institute for Geodesy and Geoinformation
JUMP	JUelich Multi Processor
MA	Moving-Average
PCCG	PreConditioned Conjugate Gradient
PCGMA	Preconditioned Conjugate Gradient Multiple Adjustement
PSD	Power Spectral Density
REG	REGularization
SGG	Satellite Gravity Gradiometry
SST	Satellite-to-Satellite Tracking
TB	TeraBytes

C. Features and Extensions of the PCGMA algorithm

C.1 Regularization

Regularization can be interpreted as a technique to integrate external information about the parameters into a least squares adjustment. In the case of the application scenario described in section 6, this is needed because of the following two problems:

- downward continuation problem
- polar gap problem

The downward continuation problem results from the fact that the GOCE satellite will measure in about 270 km above the Earth's surface. If we try to compute the quasi geoid, we will do so on the Earth's surface, i.e. at a height of approximately 0 km. The problem is that the high frequency information is much stronger on the Earth's surface than at satellite height. The polar gap problem results from the sun-synchronous orbit of the satellite. Here, the problem is that we have no measurements over the polar regions, which results in a degraded gravity field solution in these regions. In the case of our application scenario the polar gap problem is the more important one compared to the downward continuation problem (cf. Metzler and Pail 2005 or Metzler 2007).

The problems can be solved by integrating external information about the spherical harmonic coefficients (cf. Koch and Kusche 2002). This can be performed by introducing additional observation equations, which leads to minimizing the residual sum of squares

$$\boldsymbol{v}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{v} + (\boldsymbol{x} - \boldsymbol{x}_{\mathrm{REG}})^\top \boldsymbol{\Sigma}_{\mathrm{REG}}^{-1} (\boldsymbol{x} - \boldsymbol{x}_{\mathrm{REG}}) \longrightarrow \mathrm{Min.} \tag{C.1}$$

(cf. Alkhatib 2007). Herein, the vectors $\boldsymbol{x}$ and $\boldsymbol{v}$ are the parameter and residual vectors of the Gauss-Markov model (cf. section 2.1). Matrix $\boldsymbol{\Sigma}$ is the covariance matrix of the observations of the Gauss-Markov model. The first term in equation (C.1) is responsible for the adjustment to the observations while the second term stabilizes the least squares solution. The second term can also be thought of as a penalty term. $\boldsymbol{x}_{\mathrm{REG}}$ is a parameter vector, which represents the external information. The accuracy of the external information is reflected by the covariance matrix $\boldsymbol{\Sigma}_{\mathrm{REG}}$. The higher the accuracy, the strongly is the penalty of deviating from the external information. The least squares solution (2.8) then becomes

$$\widetilde{\boldsymbol{x}} = (\boldsymbol{A}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{A} + \boldsymbol{\Sigma}_{\mathrm{REG}}^{-1})^{-1} (\boldsymbol{A}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{l} + \boldsymbol{\Sigma}_{\mathrm{REG}}^{-1} \boldsymbol{x}_{\mathrm{REG}}). \tag{C.2}$$

In the case of our application scenario we use degree-dependent Kaula regularization. The entries of the covariance matrix $\boldsymbol{\Sigma}_{\mathrm{REG}}$ are chosen according to Kaula's rule of thumb (cf. Kaula 1966), i.e.

$$\sigma_{ij} = \begin{cases} 10^{-10} \cdot n^{-4}, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}, \tag{C.3}$$

whereas n is the degree. The parameter vector, which represents the external information, is

$$\boldsymbol{x}_{\mathrm{REG}} = \boldsymbol{0}. \tag{C.4}$$

The degree-dependent Kaula regularization can be though of as a smoothness condition for the gravity field. The reason for this is, that Kaula's rule of thumb states that the higher the degree of spherical harmonic coefficient is, the smaller the magnitude of this coefficient must be.

The integration into the PCGMA algorithm (cf. 6.2) has to be performed at two points. The first is the preconditioner. Here, we have to add a diagonal matrix to the preconditioning matrix.

$$\boldsymbol{K} = \boldsymbol{K} + \boldsymbol{\Sigma}_{\text{REG}}^{-1} \tag{C.5}$$

As the covariance matrix $\boldsymbol{\Sigma}_{\text{REG}}$ is a diagonal matrix in our case, the computation can be integrated by adding the equation

$$\operatorname{diag}(\boldsymbol{K}) = \operatorname{diag}(\boldsymbol{K}) + \operatorname{diag}(\boldsymbol{\Sigma}_{\text{REG}}^{-1}) \tag{C.6}$$

after the computation of the preconditioner $\boldsymbol{K}$. Furthermore, the regularization has to be integrated into the initialization and iterations of the PCGMA algorithm. If the covariance matrix $\boldsymbol{\Sigma}_{\text{REG}}$ is a diagonal matrix, the integration into the initialization of the PCGMA algorithm affects the computation of $\boldsymbol{r}^{(0)}$, which then becomes

$$\boldsymbol{r}^{(0)} = \boldsymbol{r}^{(0)} + \boldsymbol{\Sigma}_{\text{REG}}^{-1}(\boldsymbol{x}^{(0)} - \boldsymbol{x}_{\text{REG}}) = \boldsymbol{r}^{(0)} + \operatorname{diag}(\boldsymbol{\Sigma}_{\text{REG}}^{-1}) \circ (\boldsymbol{x}^{(0)} - \boldsymbol{x}_{\text{REG}}), \tag{C.7}$$

whereas the circle stands for element-wise multiplication. Within the iterations of the PCGMA algorithm the computation of vector $\boldsymbol{h}$ has to be altered to

$$\boldsymbol{h} = \boldsymbol{h} + \boldsymbol{\Sigma}_{\text{REG}}^{-1}\boldsymbol{d}^{(i)} = \boldsymbol{h} + \operatorname{diag}(\boldsymbol{\Sigma}_{\text{REG}}^{-1}) \circ \boldsymbol{d}^{(i)}. \tag{C.8}$$

Again, the simplification is possible, if $\boldsymbol{\Sigma}_{\text{REG}}$ is a diagonal matrix.

C.2 Variance Component Estimation

Gravity field models typically result from numerous types of observations. The GOCE gravity field for example will result from SST observations, SGG observations and as the case may be regularization information. These observation types are typically uncorrelated, but need a proper relative weighting within the least squares adjustment. Variance component estimation is a tool to determine optimal relative weights (cf. Koch and Kusche 2002 and Alkhatib and Schuh 2006). The least squares estimates (C.2) with weight factors p_i for each observation type can be obtained by the normal equations

$$\begin{aligned}(p_{\text{SGG}}\boldsymbol{A}_{\text{SGG}}^{\top}\boldsymbol{\Sigma}_{\text{SGG}}^{-1}\boldsymbol{A}_{\text{SGG}} + p_{\text{SST}}\boldsymbol{N}_{\text{SST}} + p_{\text{REG}}\boldsymbol{\Sigma}_{\text{REG}}^{-1})\widetilde{\boldsymbol{x}}& \\ = p_{\text{SGG}}\boldsymbol{A}_{\text{SGG}}^{\top}\boldsymbol{\Sigma}_{\text{SGG}}^{-1}\boldsymbol{l}_{\text{SGG}} + p_{\text{SST}}\boldsymbol{n}_{\text{SST}} + p_{\text{REG}}\boldsymbol{\Sigma}_{\text{REG}}^{-1}\boldsymbol{x}_{\text{REG}}.&\end{aligned} \tag{C.9}$$

Herein, the weight factors $p_i = \frac{1}{s_i^2}$, whereas $i \in \{\text{SGG, SST, REG}\}$, are reciprocal to the variance components

$$s_i^2 = \frac{\boldsymbol{v}_i^{\top}\boldsymbol{\Sigma}_i^{-1}\boldsymbol{v}_i}{r_i}. \tag{C.10}$$

In order to determine the weight factors, we need to compute the square sums of residuals $\boldsymbol{v}_i^{\top}\boldsymbol{\Sigma}_i^{-1}\boldsymbol{v}_i$ and the redundancy r_i for each observation type. The redundancy r_i can be computed by

$$r_{\text{SST}} = n_{\text{SST}} - \operatorname{trace}\left(p_{\text{SST}}\boldsymbol{N}_{\text{SST}}\boldsymbol{N}^{-1}\right) \tag{C.11}$$

and

$$r_{\text{SGG}} = n_{\text{SGG}} - \operatorname{trace}\left(p_{\text{SGG}}\boldsymbol{A}_{\text{SGG}}^{\top}\boldsymbol{\Sigma}_{\text{SGG}}^{-1}\boldsymbol{A}_{\text{SGG}}\boldsymbol{N}^{-1}\right), \tag{C.12}$$

respectively. Herein, n_{SGG} is the number of SGG observations and n_{SST} is the number of SST observations, respectively. Furthermore,

$$\boldsymbol{N} = p_{\text{SGG}}\boldsymbol{A}_{\text{SGG}}^{\top}\boldsymbol{\Sigma}_{\text{SGG}}^{-1}\boldsymbol{A}_{\text{SGG}} + p_{\text{SST}}\boldsymbol{N}_{\text{SST}} + p_{\text{REG}}\boldsymbol{\Sigma}_{\text{REG}}^{-1} \tag{C.13}$$

is the whole normal equation matrix. Using equation (3.135), we can also write for the SGG observations

$$r_{\text{SGG}} = n_{\text{SGG}} - \text{trace}\left(p_{\text{SGG}}\boldsymbol{F}_{\text{SGG}}\boldsymbol{A}_{\text{SGG}}\boldsymbol{N}^{-1}(\boldsymbol{F}_{\text{SGG}}\boldsymbol{A}_{\text{SGG}})^{\top}\right), \tag{C.14}$$

because it holds that trace $(\boldsymbol{BC})$ = trace $(\boldsymbol{CB})$ (cf. Koch 1997, p. 44). The redundancy of the regularization can be obtained by either using the same technique as for the SST component, which is recommended, or by the following formula.

$$r_{\text{REG}} = n_{\text{SGG}} + n_{\text{SST}} - m + r_{\text{SGG}} + r_{\text{SST}} \tag{C.15}$$

Herein, m is the number of parameters within the least squares adjustment. For the variance component of the regularization information we find

$$s^2_{\text{REG}} = \frac{(\widetilde{\boldsymbol{x}} - \boldsymbol{x}_{\text{REG}})^{\top}\boldsymbol{\Sigma}^{-1}_{\text{REG}}(\widetilde{\boldsymbol{x}} - \boldsymbol{x}_{\text{REG}})}{r_{\text{REG}}}. \tag{C.16}$$

For our application scenario the determination of the trace terms is a computationally very demanding task, because it involves the inverse of the normal equation matrix $\boldsymbol{N}$. In addition the computations have to be performed several times, until the weight factors p_i converge. In the following it is shown, how the problem of the high computational complexity can be solved.

According to Koch 1997, pp. 144-145, if $\boldsymbol{E}\{\boldsymbol{\mathcal{Z}}\} = 0$ and $\boldsymbol{\Sigma}\{\boldsymbol{\mathcal{Z}}\} = \boldsymbol{I}$, it holds that

$$\boldsymbol{E}\{\boldsymbol{\mathcal{Z}}^{\top}\boldsymbol{B}\boldsymbol{\mathcal{Z}}\} = \text{trace}\,(\boldsymbol{B}) = t, \tag{C.17}$$

whereas $\boldsymbol{B}$ is a symmetric matrix. The trick is to approximate the trace term by a Monte Carlo simulation

$$\widehat{t} = \frac{1}{Q}\sum_{q=1}^{Q}\boldsymbol{z}_q^{\top}\boldsymbol{B}\boldsymbol{z}_q \tag{C.18}$$

(cf. Alkhatib 2007, p. 44). The best distribution for the random vector $\boldsymbol{\mathcal{Z}}$ is

$$\boldsymbol{\mathcal{Z}}_i = \begin{cases} -1, & \text{with probability } \frac{1}{2} \\ +1, & \text{with probability } \frac{1}{2} \end{cases}, \tag{C.19}$$

whereas $\boldsymbol{\mathcal{Z}}_i$ and $\boldsymbol{\mathcal{Z}}_j$ are uncorrelated for $i \neq j$. Note, that for this distribution, is holds that $\boldsymbol{E}\{\boldsymbol{\mathcal{Z}}\} = 0$ and $\boldsymbol{\Sigma}\{\boldsymbol{\mathcal{Z}}\} = \boldsymbol{I}$. This distribution is the best, because if we use it, the estimate $\widehat{t}$ has the smallest possible variance. In practice, for the SST and SGG observations a single realization $\boldsymbol{z}$ is sufficient in order to determine $\widehat{t}$ accurately (cf. Alkhatib 2007, p. 44). Thus, for SST and SGG observations we can set $Q = 1$. For the regularization it is advised to use more than one realization $\boldsymbol{z}$. Therefore, we find for the SST and SGG observation

$$\widehat{t} = \boldsymbol{z}^{\top}\boldsymbol{B}\boldsymbol{z} \tag{C.20}$$

as an estimation of the trace term.

The practical computation of (C.20) can be integrated into the least squares adjustment. First, we find by means of the Cholesky decomposition $\boldsymbol{N}_{\text{SST}} = \boldsymbol{R}^{\top}_{\text{SST}}\boldsymbol{R}_{\text{SST}}$ and then by means of the approximation by Monte Carlo simulations

$$\begin{aligned} t_{\text{SST}} &= \text{trace}\left(p_{\text{SST}}\boldsymbol{N}_{\text{SST}}\boldsymbol{N}^{-1}\right) \\ &= \text{trace}\left(p_{\text{SST}}\boldsymbol{R}^{\top}_{\text{SST}}\boldsymbol{R}_{\text{SST}}\boldsymbol{N}^{-1}\right) \\ &= \text{trace}\left(p_{\text{SST}}\boldsymbol{R}_{\text{SST}}\boldsymbol{N}^{-1}\boldsymbol{R}^{\top}_{\text{SST}}\right) \\ &\approx p_{\text{SST}}\boldsymbol{z}^{\top}_{\text{SST}}\boldsymbol{R}_{\text{SST}}\boldsymbol{N}^{-1}\boldsymbol{R}^{\top}_{\text{SST}}\boldsymbol{z}_{\text{SST}} \\ &= \widehat{t}_{\text{SST}}, \end{aligned} \tag{C.21}$$

whereas $\boldsymbol{\mathcal{Z}}_{\text{SST}}$ follows the same distribution as $\boldsymbol{\mathcal{Z}}$, and respectively

$$\begin{aligned} t_{\text{SGG}} &= \operatorname{trace}\left(p_{\text{SGG}} \boldsymbol{F}_{\text{SGG}} \boldsymbol{A}_{\text{SGG}} \boldsymbol{N}^{-1} (\boldsymbol{F}_{\text{SGG}} \boldsymbol{A}_{\text{SGG}})^\top\right) \\ &\approx p_{\text{SGG}} \boldsymbol{z}_{\text{SGG}}^\top \boldsymbol{F}_{\text{SGG}} \boldsymbol{A}_{\text{SGG}} \boldsymbol{N}^{-1} (\boldsymbol{F}_{\text{SGG}} \boldsymbol{A}_{\text{SGG}})^\top \boldsymbol{z}_{\text{SGG}} = \widehat{t}_{\text{SGG}}, \end{aligned} \tag{C.22}$$

whereas $\boldsymbol{\mathcal{Z}}_{\text{SST}}$ also follows the same distribution as $\boldsymbol{\mathcal{Z}}$. Now, $\boldsymbol{y}_{\text{SST}} = p_{\text{SST}} \boldsymbol{N}^{-1} \boldsymbol{R}_{\text{SST}}^\top \boldsymbol{z}_{\text{SST}}$ is nothing but an additional solution of the normal equations $\boldsymbol{N} \boldsymbol{y}_{\text{SST}} = p_{\text{SST}} \boldsymbol{R}_{\text{SST}}^\top \boldsymbol{z}_{\text{SST}}$, whereas $p_{\text{SST}} \boldsymbol{R}_{\text{SST}}^\top \boldsymbol{z}_{\text{SST}}$ is the right hand side of the normal equations. Thus, we can compute the estimation of the trace term by introducing an additional parameter vector $\boldsymbol{y}_{\text{SST}}$ into the least squares adjustment. After the least squares adjustment we only have to compute $\widehat{t}_{\text{SST}} = \boldsymbol{z}_{\text{SST}}^\top \boldsymbol{R}_{\text{SST}} \boldsymbol{y}_{\text{SST}}$. Therefore, we find the estimation of the redundancy

$$\widehat{r}_{\text{SST}} = n_{\text{SST}} - \boldsymbol{z}_{\text{SST}}^\top \boldsymbol{R}_{\text{SST}} \boldsymbol{y}_{\text{SST}}. \tag{C.23}$$

Analogously, $\boldsymbol{y}_{\text{SGG}} = p_{\text{SGG}} \boldsymbol{N}^{-1} (\boldsymbol{F}_{\text{SGG}} \boldsymbol{A}_{\text{SGG}})^\top \boldsymbol{z}_{\text{SGG}}$ is an additional solution of the normal equations $\boldsymbol{N} \boldsymbol{y}_{\text{SGG}} = p_{\text{SGG}} (\boldsymbol{F}_{\text{SGG}} \boldsymbol{A}_{\text{SGG}})^\top \boldsymbol{z}_{\text{SGG}}$, whereas $p_{\text{SGG}} (\boldsymbol{F}_{\text{SGG}} \boldsymbol{A}_{\text{SGG}})^\top \boldsymbol{z}_{\text{SGG}}$ is the right hand side of the normal equations. If we look closer at this term, then we realize that this right hand side arises, if we introduce $\boldsymbol{z}_{\text{SGG}}$ as an additional observation vector in the system of observation equations. Consequently, we introduce $\boldsymbol{y}_{\text{SGG}}$ as an additional parameter vector. In this case, we obtain the redundancy by

$$\widehat{r}_{\text{SGG}} = n_{\text{SGG}} - \boldsymbol{z}_{\text{SGG}}^\top \boldsymbol{F}_{\text{SGG}} \boldsymbol{A}_{\text{SGG}} \boldsymbol{y}_{\text{SGG}} = n_{\text{SGG}} - \boldsymbol{y}_{\text{SGG}}^\top \boldsymbol{N} \boldsymbol{y}_{\text{SGG}}. \tag{C.24}$$

Note, that variance components for other observation types can be integrated in the same manner. Equation (C.21) is used for observation types, which are incorporated into the least squares adjustment on the basis of normal equations. Equation (C.22) is used for observation types, which are incorporated into the least squares adjustment on the basis of observation equations. With the estimation r_{SGG} we then obtain the variance components according to equation (C.10).

C.3 Preconditioning

If we solve the normal equations

$$\boldsymbol{N}\boldsymbol{x} = \boldsymbol{n} \tag{C.25}$$

with an iterative procedure like the conjugate gradient algorithm, we need to consider the rate of convergence. The rate of convergence is rule by the condition number of the normal equation matrix

$$\kappa(\boldsymbol{N}) = \frac{\lambda_{\max}}{\lambda_{\min}}. \tag{C.26}$$

Herein, $\lambda_{\max}$ and $\lambda_{\min}$ are the largest and the smallest eigenvalue of the positive definite normal equation matrix $\boldsymbol{N}$. The higher the condition number is, the more iterations are needed within the iterative procedure in order to achieve convergence. Preconditioning is a technique to improve the condition number of the equation system, which has to be solved iteratively. In principle, the normal equations are multiplied by the inverse preconditioner $\boldsymbol{K}$.

$$\boldsymbol{K}^{-1}\boldsymbol{N}\boldsymbol{x} = \boldsymbol{K}^{-1}\boldsymbol{n} \tag{C.27}$$

Note, that this multiplication does not alter the solution of the normal equations. Only the condition number

$$\kappa(\boldsymbol{K}^{-1}\boldsymbol{N}) \tag{C.28}$$

is different. If we choose the preconditioner $\boldsymbol{K}$ to be a suitable approximation of the normal equation matrix $\boldsymbol{N}$, the condition number $\kappa(\boldsymbol{K}^{-1}\boldsymbol{N})$ will be much smaller than $\kappa(\boldsymbol{N})$. Thus, the rate of convergence of the iterative procedure will improve massively. However, we need to choose the preconditioner $\boldsymbol{K}$ such that the solution $\boldsymbol{y}$ of

$$\boldsymbol{K}\boldsymbol{y} = \boldsymbol{k} \tag{C.29}$$

is easy to compute, because the symbolic multiplication by the inverse $\boldsymbol{K}^{-1}$ in (C.27) requires to solve such a system of equations. Furthermore, we have to be careful to choose the preconditioner $\boldsymbol{K}$ such that it needs not too much memory. This is especially important for our application scenario described in chapter 6, because here the number of parameters is very large. In addition the preconditioner should also be easy to compute, because otherwise the calculation time of the whole procedure may by dominated by the calculation time for the computation of the preconditioner.

Summarizing, we can say that there are the following requirements for the preconditioner:

- $\boldsymbol{K}$ has to be a good approximation of the normal equation matrix $\boldsymbol{N}$.
- $\boldsymbol{K}$ has to be easy to compute.
- $\boldsymbol{K}$ has to be easy to store.
- The solution of $\boldsymbol{K}\boldsymbol{y} = \boldsymbol{k}$ has to be easy to compute.

In the case of our application scenario the observations are distributed almost regularly over the Earth. Therefore, a tailored preconditioner, the so-called kite matrix (cf. Boxhammer 2006), is a very suitable preconditioner. The special structure of the kite matrix has two reasons: The first is that the SGG observations are combined with SST observations. The latter contain only information about the low frequency part of gravity field and thus contribute only to the determination of a part of the parameter vector $\boldsymbol{x}$ of the least squares adjustment. Therefore, the normal equations are set up only partially for the SST observations. This part is fully transferred to the kite matrix. The second reason is the assumption that only certain spherical harmonic coefficients, which have the same order, are correlated. With this assumption we would obtain a block diagonal structure for the preconditioner, if we sorted the spherical harmonic coefficients in the parameter vector $\boldsymbol{x}$ by their order. If now SGG and SST observations are combined and sorted according to the kite numbering scheme (cf. Boxhammer 2006), we will find the typical kite structure shown in figure C.1.

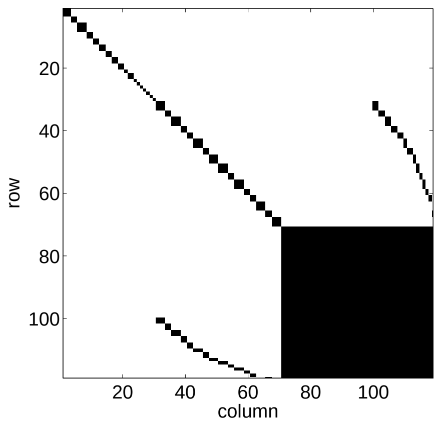


Figure C.1: Typical structure of the kite matrix. Black elements are non-zero while white elements are zero. In the lower right corner of the matrix is the kite with two tails, one on each side. The diagonal blocks in the upper left corner of the matrix are the line.

The kite matrix arises because of the special sorting of the parameters and selecting only specific elements of the normal equation matrix. For these reasons, we may write for the computation of the kite matrix

$$\boldsymbol{K} = p_{\mathrm{SGG}}\mathrm{select}(\boldsymbol{A}_{\mathrm{SGG}}^{\top}\boldsymbol{F}_{\mathrm{SGG}}^{\top}\boldsymbol{F}_{\mathrm{SGG}}\boldsymbol{A}_{\mathrm{SGG}}) + p_{\mathrm{SST}}\boldsymbol{N}_{\mathrm{SST}}. \tag{C.30}$$

The special structure of the kite matrix has the advantage that a Cholesky decomposition can be performed in-place without the occurrence of additional fill-in elements (cf. Boxhammer 2006). The solution $\boldsymbol{y}$ of

$$\boldsymbol{K}\boldsymbol{y} = \boldsymbol{k} \tag{C.31}$$

can therefore be easily computed by means of the Cholesky decomposition of the kite matrix

$$\boldsymbol{K} = \boldsymbol{R}^\top \boldsymbol{R} \tag{C.32}$$

with subsequent solving of the equation systems

$$\boldsymbol{R}^\top \boldsymbol{z} = \boldsymbol{k} \tag{C.33}$$

and

$$\boldsymbol{R}\boldsymbol{y} = \boldsymbol{z} \tag{C.34}$$

by forward and backward substitution. We abbreviate this by

$$\boldsymbol{y} = \text{solve}(\boldsymbol{K}, \boldsymbol{k}). \tag{C.35}$$

C.4 The all inclusive PCGMA algorithm

The PCGMA algorithm including regularization and variance component estimation is given in algorithm C.1. This algorithm is depicted for the sake of clarity, as it may not be perfectly clear, at which points regularization and variance component estimation have to be integrated. The simulations described in chapter 6 were performed using all features of algorithm C.1. For simplicity, only one additional right hand side for the variance component of the regularization is introduced. Within the simulation ten additional right hand sides were used for the variance component of the regularization in order to increase its accuracy.

input	$\boldsymbol{X}^{(0)} = \left[\boldsymbol{x}^{(0)}\ \boldsymbol{y}^{(0)}_{\text{SGG}}\ \boldsymbol{y}^{(0)}_{\text{SST}}\ \boldsymbol{y}^{(0)}_{\text{REG}}\right]$, I	initial solution and number of iterations
	$\boldsymbol{l}_{\text{SGG}}$, $\boldsymbol{F}_{\text{SGG}}$	SGG observations and decorrelating filter matrix
	$\boldsymbol{N}_{\text{SST}} = \boldsymbol{R}_{\text{SST}}^\top \boldsymbol{R}_{\text{SST}}$, $\boldsymbol{n}_{\text{SST}}$, $\boldsymbol{l}_{\text{SST}}^\top \boldsymbol{\Sigma}_{\text{SST}}^{-1} \boldsymbol{l}_{\text{SST}}$	SST normal equations
	$\boldsymbol{x}_{\text{REG}}$, $\boldsymbol{\Sigma}_{\text{REG}} = (\boldsymbol{R}_{\text{REG}}^\top \boldsymbol{R}_{\text{REG}})^{-1}$	regularization information
	p_{SGG}, p_{SST}, p_{REG}	initial variance components
output	$\boldsymbol{X}^{(I)}$, $\boldsymbol{K}$	final solution and preconditioner
	p_{SGG}, p_{SST}, p_{REG}	variance components

```
initialization
$\boldsymbol{K} = p_{\text{SGG}}\,\text{select}((\boldsymbol{F}_{\text{SGG}}\boldsymbol{A}_{\text{SGG}})^\top(\boldsymbol{F}_{\text{SGG}}\boldsymbol{A}_{\text{SGG}})) + p_{\text{SST}}\,\boldsymbol{N}_{\text{SST}} + p_{\text{REG}}\boldsymbol{\Sigma}_{\text{REG}}^{-1}$        computed parallel
$\boldsymbol{Y}^{(k)} = \boldsymbol{A}^{(k)}_{\text{SGG}}\boldsymbol{X}^{(0)} - \left[\boldsymbol{l}^{(k)}_{\text{SGG}}\ \ \boldsymbol{z}^{(k)}_{\text{SGG}}\ \ \boldsymbol{0}\ \ \boldsymbol{0}\right]$   for $k = 0, 1, 2, \dots$        computed parallel
$\boldsymbol{V}^{(0)} = \boldsymbol{F}_{\text{SGG}}\boldsymbol{Y} = \boldsymbol{SHY} + \left[(\boldsymbol{WY}_{1:R,:})^\top\ \ \boldsymbol{0}\right]^\top$
$\boldsymbol{Z} = \boldsymbol{F}_{\text{SGG}}^\top\boldsymbol{V}^{(0)} = \boldsymbol{PHPV}^{(0)} + \boldsymbol{W}^\top\boldsymbol{V}^{(0)}_{1:R,:}$
$\boldsymbol{R}^{(0)} = p_{\text{SGG}}\sum_k(\boldsymbol{A}^{(k)}_{\text{SGG}})^\top\boldsymbol{Z}^{(k)} + p_{\text{SST}}\left(\boldsymbol{N}_{\text{SST}}\boldsymbol{X}^{(0)} - \left[\boldsymbol{n}_{\text{SST}}\ \ \boldsymbol{0}\ \ \boldsymbol{R}_{\text{SST}}^\top\boldsymbol{z}_{\text{SST}}\ \ \boldsymbol{0}\right]\right) + \dots$
    $\dots + p_{\text{REG}}\left(\boldsymbol{\Sigma}_{\text{REG}}^{-1}\boldsymbol{X}^{(0)} - \left[\boldsymbol{x}_{\text{REG}}\ \ \boldsymbol{0}\ \ \boldsymbol{0}\ \ \boldsymbol{R}_{\text{REG}}^\top\boldsymbol{z}_{\text{REG}}\right]\right)$        computed parallel
$\boldsymbol{F}^{(0)} = \text{solve}(\boldsymbol{K}, \boldsymbol{R}^{(0)})$
$\boldsymbol{D}^{(0)} = -\boldsymbol{F}^{(0)}$

loop
for $i = 0, 1, \dots, I-1$
    if $i > 0$
        $\boldsymbol{e} = \text{diag}({\boldsymbol{R}^{(i)}}^\top\boldsymbol{D}^{(i)}) \diamond \text{diag}({\boldsymbol{R}^{(i-1)}}^\top\boldsymbol{D}^{(i-1)})$        ($\diamond$ means element-wise division: $a \diamond b = \frac{a}{b}$)
        $\boldsymbol{D}^{(i)} = \boldsymbol{D}^{(i)} + \boldsymbol{D}^{(i-1)}\text{diag}(\boldsymbol{e})$
    end
    $\boldsymbol{Y}^{(k)} = \boldsymbol{A}^{(k)}_{\text{SGG}}\boldsymbol{D}^{(i)}$   for $k = 0, 1, 2, \dots$        computed parallel
    $\boldsymbol{G} = \boldsymbol{F}_{\text{SGG}}\boldsymbol{Y} = \boldsymbol{SHY} + \left[(\boldsymbol{WY}_{1:R,:})^\top\ \ \boldsymbol{0}\right]^\top$
    $\boldsymbol{Z} = \boldsymbol{F}_{\text{SGG}}^\top\boldsymbol{G} = \boldsymbol{PHPG} + \boldsymbol{W}^\top\boldsymbol{G}_{1:R,:}$
    $\boldsymbol{H} = p_{\text{SGG}}\sum_k(\boldsymbol{A}^{(k)}_{\text{SGG}})^\top\boldsymbol{Z}^{(k)} + p_{\text{SST}}\,\boldsymbol{N}_{\text{SST}}\boldsymbol{D}^{(i)} + p_{\text{REG}}\boldsymbol{\Sigma}_{\text{REG}}^{-1}\boldsymbol{D}^{(i)}$        computed parallel
    $\boldsymbol{q} = \text{diag}({\boldsymbol{R}^{(i)}}^\top\boldsymbol{F}^{(i)}) \diamond \text{diag}({\boldsymbol{D}^{(i)}}^\top\boldsymbol{H})$        ($\diamond$ means element-wise division: $a \diamond b = \frac{a}{b}$ )
    $\boldsymbol{X}^{(i+1)} = \boldsymbol{X}^{(i)} + \boldsymbol{D}^{(i)}\text{diag}(\boldsymbol{q})$
    $\boldsymbol{R}^{(i+1)} = \boldsymbol{R}^{(i)} + \boldsymbol{H}\text{diag}(\boldsymbol{q})$
    $\boldsymbol{F}^{(i+1)} = \text{solve}(\boldsymbol{K}, \boldsymbol{R}^{(i+1)})$
    $\boldsymbol{V}^{(i+1)} = \boldsymbol{V}^{(i)} + \boldsymbol{G}\text{diag}(\boldsymbol{q})$
end

variance components
$p_{\text{SGG}} = \frac{r_{\text{SGG}}}{(\boldsymbol{v}^{(I)}_{\text{SGG}})^\top\boldsymbol{v}^{(I)}_{\text{SGG}}} = \frac{n_{\text{SGG}} - \boldsymbol{z}_{\text{SGG}}^\top(\boldsymbol{V}^{(I)}_{:,2} + \boldsymbol{F}_{\text{SGG}}\boldsymbol{z}_{\text{SGG}})}{(\boldsymbol{V}^{(i+1)}_{:,1})^\top\boldsymbol{V}^{(i+1)}_{:,1}}$
$p_{\text{SST}} = \frac{r_{\text{SST}}}{(\boldsymbol{v}^{(I)}_{\text{SST}})^\top\boldsymbol{\Sigma}_{\text{SST}}^{-1}\boldsymbol{v}^{(I)}_{\text{SST}}} = \frac{n_{\text{SST}} - \boldsymbol{z}_{\text{SST}}^\top\boldsymbol{R}_{\text{SST}}\boldsymbol{X}^{(I)}_{:,3}}{(\boldsymbol{X}^{(I)}_{:,1})^\top\boldsymbol{N}_{\text{SST}}\boldsymbol{X}^{(I)}_{:,1} - 2\boldsymbol{n}_{\text{SST}}^\top\boldsymbol{X}^{(I)}_{:,1} + \boldsymbol{l}_{\text{SST}}^\top\boldsymbol{\Sigma}_{\text{SST}}^{-1}\boldsymbol{l}_{\text{SST}}}$
$p_{\text{REG}} = \frac{r_{\text{REG}}}{(\boldsymbol{v}^{(I)}_{\text{REG}})^\top\boldsymbol{\Sigma}_{\text{REG}}^{-1}\boldsymbol{v}^{(I)}_{\text{REG}}} = \frac{n_{\text{REG}} - \boldsymbol{z}_{\text{REG}}^\top\boldsymbol{R}_{\text{REG}}\boldsymbol{X}^{(I)}_{:,4}}{(\boldsymbol{X}^{(I)}_{:,1})^\top\boldsymbol{\Sigma}_{\text{REG}}^{-1}\boldsymbol{X}^{(I)}_{:,1} - 2\boldsymbol{x}_{\text{REG}}^\top\boldsymbol{\Sigma}_{\text{REG}}^{-1}\boldsymbol{X}^{(I)}_{:,1} + \boldsymbol{x}_{\text{REG}}^\top\boldsymbol{\Sigma}_{\text{REG}}^{-1}\boldsymbol{x}_{\text{REG}}}$
```

Algorithm C.1: The preconditioned conjugate gradient multiple adjustment (PCGMA) algorithm in its parallel form including regularization and variance component estimation. The algorithm may have to be repeatedly executed until the variance components converge.

References

Alkhatib, H. (2007) *On Monte Carlo methods with applications to the current satellite gravity missions.* Dissertation. Institut für Geodäsie und Geoinformation der Universität Bonn.

Alkhatib, H. and W.-D. Schuh (2006) Integration of the Monte Carlo covariance estimation strategy into tailored solution procedures for large-scaled least squares problems. *Journal of Geodesy*, 81(1):53–66.

Anderson, E., Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney and D. Sorensen (1999) *LAPACK Users' Guide.* Third Edition. Siam, Philadelphia.

Box, G. E. P. and G. M. Jenkins (1970) *Time series analysis - forecasting and control.* Holden-Day, San Francisco, Cambridge, London, Amsterdam.

Boxhammer, C. (2006) *Effiziente numerische Verfahren zur sphärischen harmonischen Analyse von Satellitendaten.* Dissertation. Institut für Theoretische Geodäsie der Universität Bonn.

Boxhammer, C. and W.-D. Schuh (2006) GOCE Gravity field modeling: computational aspects - free kite numbering scheme. Rummel, R., C. Reigber, M. Rothacher, G. Boedecker, U. Schreiber and J. Flury (Hrsg.), *Observation of the Earth System from Space*, Springer, Berlin - Heidelberg, 209–224.

Brigham, E. O. (1988) *The fast Fourier transform and its applications.* Prentice-Hall, Englewood Cliffs, New Jersey.

Burrus, C.S. (1972) Block realization of digital filters. *IEEE transactions on audio and electroacoustics*, AU-20(4):230–235.

Chornoboy, E. S. (1992) Initialization for improved IIR filter performance. *IEEE transactions on signal processing*, 40(3):543–550.

Cuddington, K. M. and P. Yodzis (1999) Black noise and population persistence. *Proceedings of the Royal Society of London*, 266:969–973.

De Sanctis, S., A. Guiotto and M. Parisch (2002) *Simulator Design Specification.* Technical Note, GO-SP-AI-0001. Alenia Spazio, Turin, Italy.

Dehner, G.F. (2003) Noise optimized IIR digital digital filter design — tutorial and some new aspects. *Signal processing*, 83:1565–1582.

Demmel, J. W. (1997) *Applied Numerical Linear Algebra.* Siam.

Ditmar, P., R. Klees and F. Kostenko (2003) Fast and accurate computation of spherical harmonic coefficients from satellite gravity gradiometry data. *Journal of geodesy*, 76:690–705.

Durbin, J. (1960) The fitting of time series models. *Revue Inst. Int. De Statist.*, 28:233–244.

ESA (1999) *The four candidate Earth explorer core missions - gravity field and steady-state ocean circulation mission.* ESA Report SP-1233(1), Granada.

ESA (2000) *From Eötvös to mGal - final report.* ESA-Project, ESA/ESTEC Contract No. 13392/98/NL/GD.

ESA (2002) *From Eötvös to mGal+ - final report.* ESA Project, ESA/ESTEC Contract No. 14287/00/NL/DC.

Farhang-Boroujeny, B. (1998) *Adaptive Filters - Theory and Applications.* John Wiley and Sons, Chichester.

Ferstl, E. (1996) *Unter der Oberfläche - Gedanken mit Tiefgang.* Va bene-Verlag, Wien.

Forster, O. (1983) *Analysis 1.* Differential- und Integralrechnung einer Veränderlichen. Vieweg Verlag, Braunschweig Wiesbaden.

Friedlander, B. and B. Porat (1984) The modified Yule-Walker method of ARMA spectral estimation. *IEEE transactions on aerospace and electronic systems*, AES-20(2):158–173.

Hamilton, J. D. (1994) *Time Series Analysis.* Princeton University Press, Princeton, New Jersey.

Heiskanen, W. A. and H. Moritz (1967) *Physical geodesy.* Freeman, San Francisco.

Hestenes, M. and E. Stiefel (1952) Methods of conjugate gradients for solving linear systems. *Jounal of Research of the National Bureau of Standards*, 49(6):409–436.

Hettmansperger, T.H. and J.W. McKean (1998) *Kendall's library of statistics 5 - robust nonparametric statistical methods.* Arnold, a member of the Hodder Headline Group, London.

Ilk, K.H., J. Flury, R. Rummel, P. Schwintzer, W. Bosch, C. Haas, J. Schröter, D. Stammer, W. Zahel, H. Miller, R. Dietrich, P. Huybrechts, H. Schmeling, D. Wolf, H.J. Götze, J. Riegger, A. Bardossy, A. Güntner and T. Gruber (2005) *Mass transport and mass distribution in the Earth system - contribution of the new generation of satellite gravity and altimetry missions to geosciences.* Proposal for a German priority research program. GOCE-Projektbüro Deutschland, Technische Universität München, GeoForschungszentrum Potsdam.

JUMP. (2007) , (JU)elich (M)ulti(P)rozessor (JUMP) Dokumentation. `http://jumpdoc.fz-juelich.de/`, 2007. State: 2008.

Kan, E.P.F. and J.K. Aggarwal (1971) Error analysis of digital filter employing floating-point arithmetic. *IEEE transactions on circuit theory*, CT-18(6):678–686.

Kaufman, I. (1969) Evaluation of an analytical function of a companion matrix with distinct eigenvalues. *Proceedings of the IEEE*, 57:1180–1181.

Kaula, W. M. (1966) *Theory of satellite geodesy.* Blaisdell, Waltham, Massachusetts.

Klees, R. and P. Broersen (2002) *How to handle colored noise in large least-squares problems - building the optimal filter.* Delft University Press.

Klees, R. and P. Ditmar (2004) How to handle colored noise in large least-squares problems in the presence of data gaps? *Proceedings of the V Hotine-Marussi symposium on mathematical geodesy*, 127:39–48.

Klees, R., P. Ditmar and P. Broersen (2003) How to handle colored noise in large least-squares problems. *Journal of geodesy*, 76:629–640.

Klees, R., P. Ditmar and J. Kusche (2004) Numerical techniques for large least-squares problems with appications to GOCE. *Proceedings of the V Hotine-Marussi symposium on mathematical geodesy*, 127:12–21.

Koch, K. R. (1997) *Parameterschätzung und Hypothesentests.* Dümmler, Bonn.

Koch, K.-R. and J. Kusche (2002) Regularization of the geopotential determination from satellite data by variance components. *Journal of Geodesy*, 76:259–268.

Koch, K.R. (2005) Determining the maximum degree of harmonic coefficients in geopotential models by Monte Carlo methods. *Studia Geophysica et Geodaetica*, 49:259–275.

Laakso, T. I. and V. Välimäki (1998) Energy-based effective length of the impulse responce of a recursive filter. *Proceedings of the 1998 IEEE international conference on acoustics, speech an signal processing (ICASSP'98)*, 3:1253–1256. `www.acoustics.hut.fi/~vpv/publications/icassp98-impl.pdf`.

Lemoine, F.G., S.C. Kenyon, J.K. Factor, R.G. Trimmer, N.K. Pavlis, D.S. Chinn, C.M. Cox, S.M. Klosko, S.B. Luthcke, M.H. Torrence, Y.M. Wang, R.G. Williamson, E.C. Pavlis, R.H. Rapp and T.R. Olson (1998) *The Development of the Joint NASA GSFC and the National Imagery and Mapping Agency (NIMA) Geopotential Model EGM96.* National Aeronautics and Space Administration, Goddard Space Flight Center, Greenbelt, Maryland.

Levinson, N. (1947) The Wiener RMS (root mean square) error criterion in filter design and prediction. *Jounal of Mathematics and Physics*, 25:261–278.

Liu, B. and T. Kaneko (1969) Error analysis of digital filters realized with floating point arithmetic. *Proceedings of the IEEE*, 57(10):1735–1747.

Lomb, N.R. (1975) Least-squares frequency analysis of unequally spaced data. *Astrophysics and Space Science*, 39:447–462.

Mayer-Gürr, T. (2006) *Gravitationsfeldbestimmung aus der Analyse kurzer Bahnbögen am Beispiel der Satellitenmissionen CHAMP und GRACE.* Dissertation. Institut für Theoretische Geodäsie der Universität Bonn.

Mayer-Gürr, T., K.H. Ilk, A. Eicker and M. Feuchtinger (2005) ITG-CHAMP01: A CHAMP gravity field model from short kinematical arcs of a one-year observation period. *Journal of Geodesy*, 78:462–480.

Meek, J.W. and A.S. Veletsos (1972) Fast convolution for recursive digital filters. *IEEE transactions on audio and electroacoustics*, Au-20(1):93–94.

Metzler, B. (2007) *Spherical cap regularization - a spatially restricted regularization method tailored to the polar gap problem.* Dissertation. Institute of Navigation and Satellite Geodesy at the Graz University of Technology.

Metzler, B. and R. Pail (2005) GOCE data processing: the spherical cap regularization approach. *Stud. Geophys. Geod.*, 49:441–462.

Meyer, C. D. (2000) *Matrix Analysis and Applied Linear Algebra.* Siam, Philadelphia.

Ng, S. and P. Perron (2003) A Note on Selection of Time Series Models. *Boston College Working Papers in Economics*, 500:1–17. `http://fmwww.bc.edu/ec-p/wp500.pdf`.

NIC. (2007) , John von Neumann-Institut für Computing. `http://www.fz-juelich.de/nic/`, 2007. State:2008.

Oppenheim, A. V. and R. W. Schafer (1999) *Zeitdiskrete Signalverarbeitung.* 3. Auflage. Oldenbourg Verlag, München Wien.

Oppenheim, A. V., A. S. Willsky and I. T. Young (1983) *Signals and systems.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Pail, R. (2005) A parametric study on the impact of satellite attitude errors on GOCE gravity field recovery. *Journal of geodesy*, 79:231–241.

Pail, R. and G. Plank (2002) Assessment of three numerical solution strategies for gravity field recovery from GOCE satellite gravity gradiometry implemented on a parallel platform. *Journal of Geodesy*, 76:462–474.

Pail, R., W.-D. Schuh and T. Wermuth (2005) GOCE gravity field processing. In: Jekeli, C., L. Bastos and J. Fernandes (Hrsg.), *Gravity, geoid and space missions*, Band 129 der Reihe International Association of Geodesy symposia, 36–41.

Pintelon, R., P. Guillaume, Y. Rolain, J. Schoukens and H. Van hamme (1994) Parametric Identification of transfer functions in the frequency domain - a survey. *IEEE transactions on automatic control*, 39(11):2245–2259.

Plank, G. (2004) *Numerical solution strategies for the GOCE mission by using cluster technologies.* Dissertation. Institut für Navigation und Satellitengeodäsie der Technischen Universität Graz.

Press, W.H. and G.B. Rybicke (1989) Fast algorithm for spectral analysis of unevenly sampled data. *The astrophysical journal*, 338:277–280.

ATLAS Project. (2007) . `http://math-atlas.sourceforge.net/`, 2007. State: 2008.

ITG-GRACE03. (2007) . `http://www.geod.uni-bonn.de/itg-grace03.html/`, 2007. State: 2008.

Rao, B. D. (1992) Floating point arithmetic and digital filters. *IEEE transactions on signal processing*, 40(1):85–95.

Reigber, C., P. Schwintzer, K. H. Neumayer, F. Barthelmes, R. König, C. Förste, G. Balmino, R. Biancale, J. M. Lemoine, S. Loyer, S. Bruinsma, F. Perosanz and T. Fayard (2003) The CHAMP-only Earth gravity field model EIGEN-2. *Advances in Space Research*, 31(8):1883–1888.

Reigber, C., H. Lühr, P. Schwintzer and J. Wickert (2004) *Earth observation with CHAMP: results from three years in orbit.* Springer, Berlin-Heidelberg.

Reigber, C., R. Schmidt, F. Flechtner, R. Konig, U. Meyer, K. H. Neumayer, P. Schwintzler and S. Y. Zhu (2005) Earth gravity field model complete to degree and order 150 from GRACE: EIGEN-GRACE02S. *Journal of Geodynamics*, 39(1):1–10.

Rummel, R. (1985) Satellitengradiometrie. *ZfV*, 6:242–257.

Rummel, R., G. Balmino, J. Johannessen, P. Visser and P. Woodworth (2002) Dedicated gravity field missions - principles and aims. *Journal of geodynamics*, 33:3–20.

Rummel, R., R. Koop and T. Gruber (2004) High Level Processing Facility for GOCE: products and processing strategy. In: *Proceedings of Second International GOCE User Workshop "GOCE, The Geoid and Oceanography"*. ESA-ESRIN, Frascati.

Sandberg, I.W. (1967) Floating–point–roundoff accumulation in digital–filter realizations. *Bell system technical journal*, 1775–1791.

Scargle, J.D. (1982) Studies in astronomical time series analysis. II. Statistical aspects of spectral analysis of unevenly spaced data. *The astrophysical journal*, 263:835–853.

Schlittgen, R. and B. H. J. Streitberg (2001) *Zeitreihenanalyse.* 9. Auflage. Oldenbourg Verlag, München Wien.

Schuh, W.-D. (1996) *Tailored numerical solution strategies for the global determination of the earth's gravity field.* Number 81 in Mitteilungen der Universität Graz. Universität Graz.

Schuh, W.-D. (2002) Improved modeling of SGG-data sets by advanced filter strategies. Sünkel, H. (Hrsg.), *From Eötvös to mGal+ - final report*, ESA Project, ESA/ESTEC Contract No. 14287/00/NL/DC, 113–182.

Schuh, W.-D. (2003) The processing of band-limited measurements - filtering techniques in the least squares context and in the presence of data gaps. Beutler, G., M.R. Drinkwater, R. Rummel and R. von Steiger (Hrsg.), *Earth gravity field from space - from sensors to Earth sciences*, Band 108, Space Science Reviews, 67–78. ISSI Workshop, Bern (March 11-15,2002).

Schuh, W.-D. and B. Kargoll (2004) The numerical treatment of the downward continuation problem for the gravity potential. *Proceedings of the V Hotine-Marussi symposium on mathematical geodesy*, 127:22–31.

Schwarz, H. R. (1970) Die Methode der konjugierten Gradienten in der Ausgleichungsrechnung. *ZfV*, 95:130–140.

Shewchuk, J. R. (1994) , An introduction to the conjugate gradient method without the agonising pain. `http://www.cs.cmu.edu/~jrs/jrspapers.htm`, 1994. State: 2008.

Sidman, M. D., F.E. DeAngelis and G. C. Verghese (1991) Parametric system identification on logarithmic frequency responce data. *IEEE transactions on automatic control*, 36(9):1065–1070.

Sneeuw, N. (2000) *A semi-analytical approach to gravity field analysis from satellite observations.* Reihe C, 527. Deutsche Geodätische Kommission, München.

Stier, W. (2001) *Methoden der Zeitreihenanalyse.* Springer-Verlag, Berlin, Heidelberg.

Sung, W. and S.K. Mitra (1986) Efficient multi-processor implementation of recursive digital filters. *Proceedings of the IEEE ICASSP*, 257–260.

Tapley, B., J. Ries, S. Bettadpur, D. Chambers, M. Cheng, F. Condi, B. Gunter, Z. Kang, P. Nagel, R. Pastor, T. Pekker, S. Poole and F. Wang (2005) GGM02 - An improved Earth gravity model from GRACE. *Journal of Geodesy*, 79:467–478.

Trench, W. F. (1964) An algorithm for the inversion of finite Toeplitz Matrices. *Journal of the Society for Industrial and Applied Mathematics*, 12(3):515–522.

Trench, W. F. (1974) Inversion of Toeplitz Band Matrices. *Mathematics of Computation*, 28(128):1089–1095.

Wahr, J., M. Molenaar and F. Bryan (1998) Time variability of the Earth's gravity field: hydrological and oceanic effects and their possible detection using GRACE. *Journal of Geophysical Research*, 103 (B12):30205–30229.

Welch, P.D. (1967) The use of Fast Fourier Transform for the estimation of power spectra: A method based on time averaging over short modified periodograms. *IEEE transactions on audio and electroacoustics*, Au-15(2):70–73.

Widrow, B., J.R. Glover, J.M. McCool, J. Kaunitz, C.S. Williams, H.R. Hearn, J.R. Zeidler, E. Dong and R. C. Goodlin (1975) Adaptive noise cancelling: Principles and applications. *Proceedings of the IEEE*, 63(12):1692–1716.

Acknowledgments

This research was financially supported by the BMBF Geotechnologien program GOCE-GRAND II (Grant 03F0421B) and the GOCE High-level Processing Facility (HPF).

Part of the simulations were performed using the JUMP supercomputer in Jülich. The calculation time was granted by the John von Neumann Computing Institute at Jülich (project 1827).