Multivariate Direct Filter Approach (MDFA)

Marc Wildi

October 26, 2014
Contents

1 Introduction 1

1.1 Classic Model-Based Paradigm .............................................. 1

1.2 A Shift of Perspective: Tackling the (Relevant) Problem-Structure and Addressing User-Priorities ........................................... 1

1.3 Univariate DFA ...................................................................... 1

1.4 This Book's Contribution: Multivariate (M-) DFA ......................... 2

1.5 R-Code ............................................................................. 2

1.5.1 Getting Started: Setting the Paths ...................................... 2

1.5.2 DFA ............................................................................. 2

1.5.3 MDFA ............................................................................ 5

1.5.4 Feeding and Controlling MDFA ......................................... 8

2 Classic Mean-Square Error (MSE) Perspective 11

2.1 Introduction ........................................................................ 11

2.2 DFA Booster ...................................................................... 11

2.2.1 Discrete Fourier Transform (DFT) and Periodogram .......... 11

2.2.2 Filter Effects: Transfer- Amplitude- and Time-Shift Functions 13

2.2.3 Discrete Finite Sample Convolution .................................... 14

2.2.4 Assembling the Puzzle: the Optimization Criterion .......... 15

2.2.5 Exercises: ‘Uniform Superconsistency’ in a Finite Sample Number-Perspective ......................................................... 16

2.2.6 Exercises: Explaining the Optimization Criterion ............ 19

2.3 MDFA: Problem-Structure and Target (MSE-Perspective) .......... 24

2.3.1 Emphasizing the Filter Error ........................................... 24

2.3.2 One- and Multi-Step Ahead Criteria .................................. 25

2.4 Matrix Notation and Generalized Least-Squares Solution .......... 25

2.4.1 Matrix Notation ............................................................. 26

2.4.2 Generalized Least Squares Solution .................................. 27

2.4.3 R-Code ...................................................................... 28

2.5 Replication of DFA by MDFA ................................................ 29

2.5.1 Exercises ..................................................................... 29

2.6 Qualitative Easing by Leading Indicators: an Empirical Study .... 31

2.6.1 Bivariate MDFA vs. Univariate DFA ................................. 31
## 6.4 Replicate and Customize Classical ARIMA-Based Approaches
- 6.4.1 Forecasting
- 6.4.2 Signal Extraction
- 6.5 Scope

## 7 Exotic Optimization Criteria
- 7.1 Introduction
- 7.2 Hybrid Criterion
  - 7.2.1 SE-Performance
  - 7.2.2 Emphasizing Trading Performance
- 7.3 Double-Decker
  - 7.3.1 Instilling Performance
  - 7.3.2 Extracting Performance

## 8 Inference

## 9 Regularization
- 9.1 Introduction
  - 9.1.1 What is ‘Overfitting’?
  - 9.1.2 Avoiding Overfitting: Brute Force
- 9.2 Regularization Troika: a Triplet of Universal Requirements
  - 9.2.1 Decay
  - 9.2.2 Smoothness
  - 9.2.3 Cross-Sectional Similarity
- 9.3 The Troikaner
  - 9.3.1 Projection Matrix
  - 9.3.2 Generalized Information Criterion
- 9.4 Reconciling Regularization and Filter Constraints
  - 9.4.1 I(1)- (Level) Constraint
  - 9.4.2 I(2)- (Time-Shift) Constraint
  - 9.4.3 I(1)- and I(2)-Constraints
- 9.5 Implementation
  - 9.5.1 The Case $i_1<\neg T, i_2<\neg F$
  - 9.5.2 The Case $i_1<\neg F, i_2<\neg T$
  - 9.5.3 The Case $i_1<\neg T, i_2<\neg T$
- 9.6 Optimization Criterion (Zero-Shrinkage)
- 9.7 General H0-Shrinkage
  - 9.7.1 Zero-Shrinkage vs. Regularization: Potentially Conflicting Requirements
  - 9.7.2 Inclusion of A Priori Knowledge
  - 9.7.3 Replicating (and Enhancing) Clients’ Performances
- 9.8 Optimization Criterion (General H0-Shrinkage)
- 9.9 MDFA-Stages
10 Vintage Data: Working with Data-Revisions

10.1 Introduction
10.2 Data Organization
  10.2.1 Vintage and Release Triangles
  10.2.2 Vintage Triangle in the Frequency-Domain
10.3 Reconcile Real-Time Signalextraction and Data-Revisions
  10.3.1 Vintage-Filtering/Smoothing
  10.3.2 Setting-Up MDFA-Designs: the (Pseudo-) Stationary Case
  10.3.3 Extension to Non-Stationary Time Series

11 Mixed-Frequency Data: Combining and Working with Data Sampled at Different Time Scales

11.1 Introduction
11.2 Target is Low-Frequency Data
  11.2.1 Folding the Frequency Interval
  11.2.2 Generalized Optimization Criterion
11.3 Explaining Series is/are Low-Frequency Data
  11.3.1 Folding and Expanding the Frequency Interval
  11.3.2 Generalized Optimization Criterion
11.4 General Case: Arbitrary Mix
11.5 Unequally Distributed Release Dates
11.6 Missing Data

12 Non-Stationarity: Integrated Processes

12.1 Introduction
12.2 Integration
  12.2.1 Unit-Roots vs. Filter Constraints
  12.2.2 I(1)- (Level) Constraint
  12.2.3 I(2)- (Time-Shift) Constraint
12.3 Optimization Criterion

13 Non-Stationarity: Cointegrated Processes

13.1 Cointegration Relations vs. Filter Constraints
13.2 The Rank-One Case
13.3 Arbitrary Rank
13.4 Universal Time-Domain Decomposition of the Filter Error
13.5 Frequency-Domain Decomposition of the Filter Error
13.6 I(1)-MSE Criterion
13.7 Unveiling the Unit-Root Singularity
13.8 Matrix Notation (Frequency Domain)
13.9 Customization
13.10 Regularization
CONTENTS

13.11 An Application of Cointegration to Data Revisions ........................................... 94

14 Non-Stationarity: Adaptive Filtering ........................................................................ 95
  14.1 Introduction ........................................................................................................... 95
  14.2 Filter Up-Dating .................................................................................................. 95
     14.2.1 Univariate Up-Dating .................................................................................. 95
     14.2.2 Multivariate Up-Dating ............................................................................... 95
  14.3 MDFA-Stages ...................................................................................................... 95

15 Summary and Links .................................................................................................. 97
  15.1 Survey of MDFA Optimization Criteria ............................................................... 97
  15.2 Consistency and Efficiency: a Tale of Two Philosophies ..................................... 97
     15.2.1 Knowing the Truth: Omniscience ............................................................... 97
     15.2.2 Believing in Truth: Faith and Fatalism ......................................................... 97
     15.2.3 From Truth to Effectiveness: Emphasizing Performances ....................... 97
  AppcMDFA: R-Code ................................................................................................... 99
Chapter 1

Introduction

1.1 Classic Model-Based Paradigm

Marc’s perspective:

- Maximum Likelihood, main purpose: determine DGP. If DGP is known then optimality can be invoked, in principle.
- Emphasizes short-term performances, only (contrast with real-time trend extraction: long-term component).
- Rigid criterion: can account neither for relevant problem-structure (signal extraction=one and multi-step ahead forecasts) nor for various user-priorities.

Tucker/Chris’ perspectives:...

1.2 A Shift of Perspective: Tackling the (Relevant) Problem-Structure and Addressing User-Priorities

Refer to DFA and Trilemma papers with Tucker. Refer to chapters 2 (problem structure) and 5 (user-priorities).

1.3 Univariate DFA

Refer to DFA-paper with Tucker and DFA (Sweave environment: replication).

1
CHAPTER 1. INTRODUCTION

1.4 This Book’s Contribution: Multivariate (M-) DFA

Problem structure (chapter 2); forecasting/nowcasting/backcasting and filter revisions (chapter 3); filter constraints (chapter 4); ATS-trilemma, customization and user-priorities (chapter 5); replicating and enhancing classical model-based approaches and HP/CF-filters (chapter 6); addressing more sophisticated gain/loss structures (chapter 7); developing inferential aspects (chapter 8); Regularization Troika and tackling overfitting (chapter 9); data-revisions (chapter 10); solving the mixed-frequency problem (chapter 11); extensions to non-stationary integrated (chapter 12) and cointegrated processes (chapter 13); adaptive filtering (chapter 14).

1.5 R-Code

We here briefly review the main R-code files and provide support for installation.

1.5.1 Getting Started: Setting the Paths

In order to be executable, the R-code requires access to program, output and data folders:

```r
> # Load packages
> library(tseries)
> library(xts)
> # Specify hard-disk
> disk_id<="C"
> # Set paths
> path.main<-paste(disk_id,\":\"\via\desktop\Projects\2014\MDFA-Legacy\Sweave\",sep="")
> # Path to DFA-code: R-files are cut-and-pasted from DFA-book (they are identical)
> path_DFA.pgm<-paste(path.main,"R\I-DFA\",sep="")
> # Path to MDFA-code
> path_MDFA.pgm<-paste(path.main,"R\I-MDFA\",sep="")
> # Path to Latex-folder: all pdfs generated by the R-code are filed there
> path.out<-paste(path.main,"Latex\",sep="")
> # Path to data
> path.dat<-paste(path.main,"Data\",sep="")
```

The (univariate) DFA-code is the same as in DFA; all empirical examples are and will be fully compatible.

1.5.2 DFA

We here briefly review the relevant pieces (anchoring).

1Left-click to activate the hyperlink.
1.5. R-CODE

DFT and Periodogram

The Discrete Fourier Transform (DFT) and the periodogram are defined in DFA, sections 2.2 and 2.3. The following function is cut-and-pasted from section 2.3, p.35:

```r
> per<-function(x,plot_T)
+ {
+   len<-length(x)
+   per<0:(len/2)
+   DFT<-per
+   for (k in 0:(len/2))
+     {
+       cexp <- complex(arg=-(1:len)*2*pi*k/len)
+       DFT[k+1]<-sum(cexp*x*sqrt(1/(2*pi*len)))
+     }
+   per<-abs(DFT)^2
+   if (plot_T)
+     {
+       par(mfrow=c(2,1))
+       plot(per,type="l",xlab="Frequency",ylab="Periodogram",
+            main="Periodogram")
+       axis(1,at=1+0:6*len/12,labels=c("0","pi/6","2pi/6","3pi/6",
+                                    "4pi/6","5pi/6","pi"))
+       axis(2)
+       box()
+       log(per),type="l",xlab="Frequency",ylab="Log-periodogram",
+            main="Log-periodogram")
+       axis(1,at=1+0:6*len/12,labels=c("0","pi/6","2pi/6","3pi/6",
+                                    "4pi/6","5pi/6","pi"))
+       axis(2)
+       box()
+     }
+   return(list(DFT=DFT,per=per))
+ }
```

This function will be generalized in the new multivariate setting.

DFA: Mean-Square Perspective

A simple MSE (Mean-Square Error) version of the DFA is proposed in DFA, section 4.1:

```r
> # This function computes mean-square DFA-solutions
> # L is the length of the MA filter,
```
CHAPTER 1. INTRODUCTION

> # periodogram is the frequency weighting function in the DFA
> # Gamma is the transfer function of the symmetric filter (target) and
> # Lag is the lag-parameter: Lag=0 implies real-time filtering, Lag=L/2
> # The function returns optimal coefficients as well as the transfer function of the
> # optimized real-time filter
> dfa_ms<-function(L,periodogram,Lag,Gamma)
  + {
    + K<-length(periodogram)-1
    + X<-exp(-1.i*Lag*pi*(0:(K))/(K))*rep(1,K+1)*sqrt(periodogram)
    + X_y<-exp(-1.i*Lag*pi*(0:(K))/(K))*rep(1,K+1)
    + for (l in 2:L) #l<-L<-21
    + { + X<-cbind(X,(cos((l-1-Lag)*pi*(0:(K))/(K)) + 1.i*sin((l-1-Lag)*pi*(0:(K))/(K)))*sqrt(periodogram))
    + X_y<-cbind(X_y,(cos((1-Lag)*pi*(0:(K))/(K)))*sqrt(periodogram))
    + } + xtx<-t(Re(X))%*%Re(X)+t(Im(X))%*%Im(X)
    + # MA-Filtercoefficients
    + b<-as.vector(solve(xtx)%*%(t(Re(X_y))%*%(Gamma*periodogram)))
    + # Transfer function
    + trffkt<-1:(K+1)
    + trffkt[1]<-sum(b)
    + for (k in 1:(K))#k<-1
    + { + trffkt[k+1]<-(b%*%exp(1.i*k*(0:(length(b)-1))*pi/(K)))
    + }
    + return(list(b=b,trffkt=trffkt))
  + }

This function is nested in MDFA and can be replicated perfectly, see section 2.5.

DFA: ATS-Trilemma and Customization

A general DFA-function, called $dfa_{analytic}$, is proposed in [DFA] section 4.3.5. Customization
and the generic ATS-trilemma are presented in [DFA] sections 4.3 and 5. Since the generalized
function is lengthier than the previous MSE-implementation we here abstain from editing the
code. Instead, all DFA-functions (including DFT, periodogram and DFA-MSE) can be sourced by
calling the R-file $DFA_code$
1.5. R-CODE

> source(file=paste(path_DFA.pgm,"DFA_code.r",sep=""))

In particular, the head of \texttt{dfa\_analytic} is

> head(dfa\_analytic)

1 function (L, lambda, weight\_func, Lag, Gamma, eta, cutoff, i1, i2)
2  
3 { 
4 K <- length(weight\_func) - 1 
5 omega\_Gamma <- as.integer(cutoff * K/pi) 
6 if ((K - omega\_Gamma + 1) > 0) {

The additional control parameters \texttt{lambda,eta} allow navigation in the generic framework of the ATS-trilemma, see chapter \[5\] and \texttt{i1,i2} account for filter constraints, see chapter \[4\].

1.5.3 MDFA

The R-code for MDFA is more sophisticated and correspondingly more complex and lengthy. As for the DFA-package, the MDFA-code can be sourced. We here briefly review the corresponding pieces.

Data-Matrix

All time series are collected in a data-matrix, say \textit{X}, which is organized as follows:

- the first column \textit{X[,1]} of \textit{X} always corresponds to the target series: the target series \textit{X[,1]} is the time series to be forecasted, nowcasted or backcasted.

- Columns 2,3,... of \textit{X} are allocated to the explaining variables (more than one in a multivariate setting). If the target series is part of the set of explaining variables (it does not have to), then it must be assigned a specific column - by convention always the second one - in \textit{X} i.e. in this case the target series is contained twice: in the first column (target) and in the second column (explaining data).

Example:

> set.seed(1)
> len<-100
> target<-arima.sim(list(ar=0.9),n=len)
> explaining_2<-target+rnorm(len)
> explaining<-cbind(target,explaining_2)
> x<-cbind(target,explaining)
> dimnames(x)[[2]]<-c("target","explaining 1","explaining 2")
> head(x)
CHAPTER 1. INTRODUCTION

In this case we assume a two-dimensional Signal Extraction (SE-) problem whereby the target series (first column) is part of the set of explaining variables. For a one-step ahead forecast problem we might consider lagging of the explaining variables

```R
> x<-cbind(x[,1],lag(x[,2:3],-1))
> dimnames(x)[[2]]<-c("target","lagged explaining 1","lagged explaining 2")
> head(x)
```

However, our frequency-domain approach in such a case will be more general and it will avoid introduction of undesirable NA’s.

DFT

In contrast to the univariate case, which can rely on the periodogram only, the multivariate case requires the (complex) DFT in order to account for cross-sectional dependencies\(^2\). Also, we here extend the scope of the method in order to cover the mixed-frequency case, see chapter 11. Finally, we allow for the possibility of integrated processes, see chapter 12. The additional flexibility requires specific code-extensions which are implemented in the new DFT.r file:

```R
> source(file=paste(path_MDFA.pgm,"DFT.r",sep=""))
```

In order to illustrate some of the new features we briefly look at the main DFT-function called spec_comp:

```R
> spec_comp

function(insamp, x, d) {
  # non-stationary
```

\(^2\)The relative phase-shift(s) of the explaining time series import. In the univariate case the relative phase-shift vanishes since the target and the explaining series are identical.
1.5. R-CODE

```r
if(d == 1) {
    weight_func <- periodogram_bp(diff(x[1 : insamp, 1]), 1,
        insamp - 1)$fourtrans

    # explaining variables
    if(length(weight_func) > 1) {
        for(j in 2 : ncol(x)) {
            # since the data is integrated one uses the pseudo-periodogram:
            # diff(data) and d <- 1
            per <- periodogram_bp(diff(x[1 : insamp, j]), 1, insamp - 1)$fourtrans
            weight_func <- cbind(weight_func, per)
        }
    }
} else {
    #target
    weight_func <- general_spec(x[1 : insamp, 1], 0, insamp)$fourtrans
    # explaining variables
    if(length(weight_func) > 1) {
        for(j in 2 : ncol(x)) #j<-3
        { #
            per <- general_spec(x[1 : insamp, j], 0, insamp)$fourtrans
            weight_func <- cbind(weight_func, per)
        }
    }
}

colnames(weight_func) <- colnames(x)
# return the spectral estimate
return(list(weight_func = weight_func))
```

The inner loop runs across the columns of the data-matrix \(X\) (see above for definition and conventions) and the DFTs are stored in a matrix called \(weight_func\) which is returned by the function. The matrix \(weight_func\) collects all DFTs: the target series is always in the first column whereas the DFTs of the explaining series are in columns 2,3,... The function \(periodogram_bp\), called in the above loop, is slightly more general than the DFA-function \(periodogram\), proposed in the previous section. In particular it can handle various integration orders as well as seasonal peculiarities.

**MDFA**

The complete MDFA-estimation routines can be sourced as follows:

```r
> source(file=paste(path_MDFA.pgm,"I-MDFA_new.r",sep=""))
```
Let us briefly describe the four functions in this file (details are left for later sections):

- **spec_mat_comp**: this function organizes the DFT according to the matrix-notation to be introduced in section [2.4](#).
- **mat_func**: this function provides all regularization features to be discussed in chapter [9](#).
- **mdfa_analytic_new**: this is the main estimation routine. It computes filter coefficients and diagnostic statistics, see chapters [2](#) (classical MSE-criterion), [4](#) (filter constraints), [5](#) (customization), [6](#) (replication of classical forecast paradigm(s)), [8](#) (inference), [9](#) (regularization), [10](#) (vintage data), [11](#) (mixed-frequency approach) and [12, 13, 14](#) (non-stationarity).
- **MS_decomp_total**: this function decomposes the MSE-norm into Accuracy, Timeliness and Smoothness error components (ATS-trilemma), see McElroy/Wildi (2012) and chapter [5](#).

### 1.5.4 Feeding and Controlling MDFA

**Generic Approach: Rich User-Interface**

MDFA is a generic forecast and signal extraction paradigm. Besides replicating classical time series approaches it allows for some unique features like customization and general filter-regularization (Regularization Troika and general H0-shrinkage). Also, it allows for tackling data revisions, mixed-frequency problems and non-stationarity. Accordingly, the user-interface is more sophisticated than the precedent DFA-package. In order to illustrate the topic we here briefly look at the head of the main estimation routine:

```r
> head(mdfa_analytic_new)
```

7 function (K, L, lambda, weight_func, Lag, Gamma, expweight, cutoff,
8   i1, i2, weight_constraint, lambda_cross, lambda_decay, lambda_smooth,
9   lin_expweight, shift_constraint, grand_mean, b0_H0, chris_expweight,
10   weights_only = F, weight_structure, white_noise, synchronicity,
11   lag_mat)
12 {

Besides straightforward entries, like the DFT (weight_func, see previous section), the filter-length (L), or the target specification Gamma there are numerous additional control parameters: the relevance and the modus operandi of most of them will be discussed in this book.

**Default-Settings**

For convenience, we store a so-called default-setting of the parameters in a file called control_default. In order to do so we first need to define the data (initialize the DFT-matrix) and specify the filter-length:

```r
> weight_func<-matrix(1:6)
> L<-2
```
Given these two entries (data and filter-length), the default-settings are as follows:

```r
> K<-nrow(weight_func)-1
> lambda<-0
> Lag<-0
> expweight<-0
> i1<-F
> i2<-F
> weight_constraint<-rep(1/(nrow(weight_func)-1),nrow(weight_func)-1)
> lambda_cross<-lambda_decay<-lambda_smooth<-0
> lin_expweight<-F
> shift_constraint<-rep(0,nrow(weight_func)-1)
> grand_mean<-F
> b0_H0<-NULL
> chris_expweight<-F
> weights_only<-F
> weight_structure<-c(0,0)
> white_noise<-F
> synchronicity<-F
> lag_mat<-matrix(rep(0:(L-1),nrow(weight_func)-1),nrow(L))
```

This particular configuration will be used extensively in chapter 2: mean-square criterion (no customization), no regularization, unconstrained design, no a priori knowledge, nowcasting, common identical sampling frequency, no data-revisions. The default-settings can be obtained by sourcing the corresponding R-file

```r
> source(file=paste(path_MDFA.pgm,"control_default.r",sep=""))
```
Chapter 2

Classic Mean-Square Error (MSE) Perspective

2.1 Introduction

MDFA is a generic forecast and signal-extraction paradigm with a richly parametrized user-interface allowing for sophisticated data analysis. In this chapter we emphasize mean-square performances: the corresponding default-parameters were introduced in section 1.5.4. Specifically, the parameters can be conveniently up-loaded by sourcing a corresponding R-file.

In section 2.2 we provide a brief ‘fresh-up’ of the antecedent DFA paradigm; section 2.3 generalizes the univariate (MSE-) case to a multivariate (MSE-) framework; section 2.4 presents a general matrix-notation which will allow for formal and convenient extensions of the univariate DFA to the MDFA; the DFA is replicated by MDFA in section 2.5; finally, section 2.6 benchmarks a bivariate MDFA against the former DFA and evaluates performance gains by leading-indicator designs.

2.2 DFA Booster

We propose a brief survey or ‘re-fresher’ of the main DFA-concepts. The interested reader is referred to DFA and to McElroy and Wildi (2014) (DFA and Trilemma) for technical details, (R-)code and exercises on the topic.

2.2.1 Discrete Fourier Transform (DFT) and Periodogram

A time series $x_t$, $t = 1, ..., T$, of length $T$, can be mapped to the frequency-domain by the so-called DFT:

$$\Xi_{TX}(\omega) := \frac{1}{\sqrt{2\pi T}} \sum_{t=1}^{T} x_t \exp(-it\omega)$$  \hspace{1cm} (2.1)
CHAPTER 2. CLASSIC MEAN-SQUARE ERROR (MSE) PERSPECTIVE

The DFT $\Xi_{TX}(\omega)$ is generally restricted to the discrete frequency-grid $\omega_k = \frac{k2\pi}{T}$, where $k = -T/2, \ldots, 0, \ldots, T/2$, for even $T$. This (discrete grid) restriction can be justified by the fact that the data could be recovered from the DFT by applying the so-called inverse (DFT-) transformation

$$x_t = \sqrt{\frac{2\pi}{T}} \sum_{k=-[T/2]}^{[T/2]} w_k \Xi_{TX}(\omega_k) \exp(it\omega_k)$$  \hspace{1cm} (2.2)

and where

$$[T/2] = \begin{cases} T/2 & T \text{ even} \\ (T-1)/2 & T \text{ odd} \end{cases}$$

The identity 2.2 is a tautological number identity: it applies to any sequence of numbers $x_t$, $t = 1, \ldots, T$ irrespective of (model-) assumptions. The identity suggests that the data $x_t$ can be decomposed into a linear combination of sines and cosines as weighted by the DFT. The equation can be verified empirically, see DFA, section 2.2.1, exercise 2 (a proof is provided in the appendix).

The periodogram $I_{TX}(\omega_k)$ is defined by

$$I_{TX}(\omega_k) = |\Xi_{TX}(\omega_k)|^2$$  \hspace{1cm} (2.4)

The periodogram is the DFT of the sample autocovariance function $\hat{R}(k)$ of the data:

$$I_{TX}(\omega_k) = \begin{cases} \frac{1}{2\pi} \sum_{j=-(T-1)}^{T-1} \hat{R}(j) \exp(-ij\omega_k) & |k| = 1, \ldots, T/2 \\ \frac{T}{2\pi T^2} & k = 0 \end{cases}$$  \hspace{1cm} (2.5)

where

$$\hat{R}(j) := \frac{1}{T} \sum_{t=1}^{T-|j|} x_t x_{t+|j|}$$  \hspace{1cm} (2.6)

is the sample autocovariance of a zero-mean stationary process. The periodogram can be interpreted as a decomposition of the sample variance:

$$\hat{R}(0) = \frac{2\pi}{T} \sum_{k=-[T/2]}^{[T/2]} I_{TX}(\omega_k) = \frac{2\pi}{T} I_{TX}(0) + 2 \frac{2\pi}{T} \sum_{k=1}^{[T/2]} I_{TX}(\omega_k)$$  \hspace{1cm} (2.7)

\footnote{For odd $T$ one uses $T' = T - 1$ in these expressions instead of $T$.}
The value $2 \frac{\gamma^k}{T]\, I_{TX}(\omega_k), k > 0$ measures dynamic contributions of components with frequency $\omega_k$ to the sample variance of the data.

### 2.2.2 Filter Effects: Transfer- Amplitude- and Time-Shift Functions

Let $y_t$ be the output of a general filter

$$ y_t = \sum_{k=-\infty}^{\infty} \gamma_k x_{t-k} $$

In order to derive the important filter effect(s) we assume a particular (complex-valued) input series $x_t := \exp(it\omega)$. The output signal $y_t$ then becomes

$$ y_t = \sum_{k=-\infty}^{\infty} \gamma_k \exp(i\omega(t-k)) $$  \hspace{1cm} (2.8)

$$ = \exp(i\omega t) \sum_{k=-\infty}^{\infty} \gamma_k \exp(-ik\omega) $$ \hspace{1cm} (2.9)

$$ = \exp(i\omega t) \Gamma(\omega) $$ \hspace{1cm} (2.10)

where the (generally complex-valued) function

$$\Gamma(\omega) := \sum_{k=-\infty}^{\infty} \gamma_k \exp(-ik\omega)$$ \hspace{1cm} (2.11)

is called the transfer function of the filter. We can represent the complex number $\Gamma(\omega)$ in terms of polar coordinates:

$$\Gamma(\omega) = A(\omega) \exp(-i\Phi(\omega))$$ \hspace{1cm} (2.12)

where $A(\omega) = |\Gamma(\omega)|$ is called the amplitude of the filter and $\Phi(\omega)$ is its phase.

If the filter coefficients are real, then the real part of $x_t$ is mapped to the real part of $y_t$. Therefore the cosine (real-part of the input) is mapped to

$$\cos(t\omega) \rightarrow \Re(\exp(i\omega t)\Gamma(\omega)) $$ \hspace{1cm} (2.13)

$$ = A(\omega) [\cos(t\omega) \cos(-\Phi(\omega)) - \sin(t\omega) \sin(-\Phi(\omega))]$$

$$ = A(\omega) \cos(t\omega - \Phi(\omega))$$

$$ = A(\omega) \cos(\omega(t - \Phi(\omega)/\omega))$$ \hspace{1cm} (2.14)

The amplitude function $A(\omega)$ can be interpreted as the weight (damping if $A(\omega) < 1$, amplification if $A(\omega) > 1$) attributed by the filter to a sinusoidal input signal with frequency $\omega$. The function

$$\phi(\omega) := -\Phi(\omega)/\omega$$ \hspace{1cm} (2.15)

\footnote{More precisely: components with frequencies in the interval $[\omega_k - \pi/T, \omega_k + \pi/T].$}
can be interpreted as the time shift of the filter in $\omega$.

The transferfunction of a causal and stable ARMA-filter

$$y_t = \sum_{k=1}^{L'} a_k y_{t-k} + \sum_{j=0}^{L} b_j x_{t-j}$$

is obtained as

$$\Gamma(\omega) = \frac{\sum_{j=0}^{L} b_j \exp(-ij\omega)}{1 - \sum_{k=1}^{L'} a_k \exp(-ik\omega)}$$

Amplitude and time-shift functions of the ARMA-filter can be derived from this expression, see [DFA] section 3.3.4 for comprehensive results and exercises on the topic.

### 2.2.3 Discrete Finite Sample Convolution

The transferfunction or, alternatively, the amplitude and the phase (or time-shift) functions, summarize and describe the effects of a filter as applied to an elementary (periodic and deterministic) trigonometric signal $x_t = \exp(it\omega)$:

$$y_t = \sum_{j=-\infty}^{\infty} \gamma_j x_{t-j} = \Gamma(\omega) x_t$$

An arbitrary sequence $x_1, ..., x_T$, neither periodic nor deterministic, can be decomposed into a weighted sum of trigonometric sinusoids

$$x_t = \frac{\sqrt{2\pi}}{\sqrt{T}} \sum_{k=-[T/2]}^{[T/2]} \Xi_{TX}(\omega_k) \exp(\imath t \omega_k)$$ (2.16)

and similarly for $y_t$

$$y_t = \frac{\sqrt{2\pi}}{\sqrt{T}} \sum_{k=-[T/2]}^{[T/2]} \Xi_{TY}(\omega_k) \exp(\imath t \omega_k)$$ (2.17)

recall the number-identity 2.2 (we omit the weights $w_k$). Therefore, when applying the filter to a general sequence $x_1, ..., x_T$ we might proceed as follows

$$y_t = \sum_{j=-\infty}^{\infty} \gamma_j x_{t-j}$$

$$\approx \sum_{j=-\infty}^{\infty} \gamma_j \left( \frac{\sqrt{2\pi}}{\sqrt{T}} \sum_{k=-[T/2]}^{[T/2]} w_k \Xi_{TX}(\omega_k) \exp(\imath (t-j) \omega_k) \right)$$ (2.18)

$$= \frac{\sqrt{2\pi}}{\sqrt{T}} \sum_{k=-[T/2]}^{[T/2]} w_k \Xi_{TX}(\omega_k) \left( \sum_{j=-\infty}^{\infty} \gamma_j \exp(\imath (t-j) \omega_k) \right)$$

$$= \frac{\sqrt{2\pi}}{\sqrt{T}} \sum_{k=-[T/2]}^{[T/2]} w_k \Xi_{TX}(\omega_k) \Gamma(\omega_k) \exp(\imath t \omega_k)$$ (2.19)

Comparing 2.17 and 2.19 suggests that the DFT $\Xi_{TY}(\omega_k)$ of the output signal is linked to the DFT $\Xi_{TX}(\omega_k)$ of the input signal via

$$\Xi_{TY}(\omega) \approx \Gamma(\omega) \Xi_{TX}(\omega)$$ (2.20)
This result is not a strict equality but from a practical point of view we can ignore the error (one can invoke a ‘uniform super-consistency’ argument, see DFA, Wildi (2005) and (2008)). By definition, see 2.4, we then obtain

\[ I_{TY}(\omega) \approx |\Gamma(\omega)|^2 I_{TX}(\omega) \]  

(2.21)

2.2.4 Assembling the Puzzle: the Optimization Criterion

We assume a general target specification

\[ y_t = \sum_{k=-\infty}^{\infty} \gamma_k x_{t-k} \]  

(2.22)

Note, for example, that (classical one-step ahead) forecasting could be addressed by specifying \( \gamma_{-1} = 1, \gamma_k = 0, k \neq -1 \). We aim at finding filter coefficients \( b_k, k = 0, \ldots, L-1 \) such that the finite sample estimate

\[ \hat{y}_t := \sum_{k=0}^{L-1} b_k x_{t-k} \]  

(2.23)

is ‘closest possible’ to \( y_t \) in mean-square

\[ E \left[ (y_t - \hat{y}_t)^2 \right] \to \min_b \]  

(2.24)

where \( b = (b_0, \ldots, b_{L-1}) \).

As usual, in applications, the expectation is unknown and therefore we could try to replace 2.24 by its sample estimate

\[ \frac{1}{T} \sum_{t=1}^{T} (y_t - \hat{y}_t)^2 = \frac{2\pi}{T} \sum_{k=-[T/2]}^{[T/2]} I_{T\Delta Y}(\omega_k) \]  

(2.25)

where \( I_{T\Delta Y}(\omega_k) \) is the periodogram of \( \Delta y_t := y_t - \hat{y}_t \) and where the identity follows from 2.7. Unfortunately, the output \( y_t \) of the generally bi-infinite filter isn’t observed and therefore \( I_{T\Delta Y}(\omega_k) \) is unknown too. But we could try to approximate \( I_{T\Delta Y}(\omega_k) \) by relying on the finite-sample discrete convolution 2.21

\[ I_{T\Delta Y}(\omega_k) \approx |\Delta \Gamma(\omega_k)|^2 I_{TX}(\omega_k) \]  

where \( \Delta \Gamma(\omega_k) = \Gamma(\omega_k) - \hat{\Gamma}(\omega_k) = \sum_{j=-\infty}^{\infty} \Delta \gamma_j \exp(-ij\omega_k) \) is the difference of target and real-time transfer functions. Then

\[ \frac{1}{T} \sum_{t=1}^{T} (y_t - \hat{y}_t)^2 = \frac{2\pi}{T} \sum_{k=-[T/2]}^{[T/2]} I_{T\Delta Y}(\omega_k) \approx \frac{2\pi}{T} \sum_{k=-[T/2]}^{[T/2]} |\Delta \Gamma(\omega_k)|^2 I_{TX}(\omega_k) \to \min_b \]  

(2.26)

(2.27)

We refer the reader to section 4.1 in DFA for background on this derivation and in particular:
for an extension of these concepts to integrated processes, see also chapter 12 and

for the magnitude of the approximation error which is negligible in practice ('uniform super-
consistency').

The minimization 2.27 is the DFA-MSE criterion.

2.2.5 Exercises: ‘Uniform Superconsistency’ in a Finite Sample Number-Perspective

The empirical design of the following exercises is inspired from DFA, section 4.1.1, exercise 2. It is assumed that the reader is familiar with the concept of an ideal trend (definition, computation of symmetric coefficients) as well as with the DFA-booster proposed above. The series of exercises aims at illustrating the quality of the approximation 2.27 by putting the abstract ‘uniform superconsistency’ argument into a finite-sample number-perspective.

1. Generate realizations of three different stationary processes

\[
\begin{align*}
    x_t &= 0.9x_{t-1} + \epsilon_t \\
    x_t &= \epsilon_t \\
    x_t &= -0.9x_{t-1} + \epsilon_t
\end{align*}
\]

and apply an ideal trend with cutoff \( \pi/6 \) to each of them. The target series \( y_t \) should have length 120 (10 years of monthly data). Hint: since the ideal trend is a bi-infinite filter, we use a truncated finite sample approximation.

- Generate realizations for all three processes. Hint: generate long time series of length 2000 in order to apply the symmetric target filter.

```R
> # Generate series of length 2000
> lenh<-2000
> len<-120
> # Specify the AR-coefficients
> a_vec<-c(0.9,0,-0.9)
> xh<-matrix(nrow=lenh,ncol=length(a_vec))
> x<-matrix(nrow=len,ncol=length(a_vec))
> yhat<-x
> y<-x
> # Generate series for each AR(1)-process
> for (i in 1:length(a_vec))
> { 
>     + # We want the same random-seed for each process
>     + set.seed(10)
>     + # Generate realizations of three different stationary processes
>     + xh[,i]<-arima.sim(list(order=c(1,0,0),ar=a_vec[i]),n=lenh)
>     + x[,i]<-arima.sim(list(order=c(1,0,0),ar=a_vec[i]),n=len)
> }
```

Superconsistency means: the approximation error is asymptotically of smaller order than \( 1/\sqrt{T} \). Uniformity means: this claim remains valid after optimization. The magnitude of the approximation error is computed in DFA, section 3.3.2.
2.2. DFA BOOSTER

```r
+ xh[,i]<-arima.sim(list(ar=a_vec[i]),n=lenh)
+ }
```

- Extract the data (series of length 120) and compute (truncated) ideal trends.

```r
> # Extract 120 observations in the middle of the longer series
> x<-xh[lenh/2+(-len/2):((len/2)-1),]
> # Compute the coefficients of the symmetric target filter
> cutoff<-pi/6
> # Order of approximation
> ord<-1000
> # Filter weights ideal trend (See DFA)
> gamma<-c(cutoff/pi,(1/pi)*sin(cutoff*1:ord)/(1:ord))
> # Compute the outputs yt of the (truncated) symmetric target filter
> for (i in 1:length(a_vec))
+ {
+ for (j in 1:120)
+ {
+ y[j,i]<-gamma[1:900]%*%xh[lenh/2+(-len/2)-1+(j:(j-899)),i]+
+ gamma[2:900]%*%xh[lenh/2+(-len/2)+(j:(j+898)),i]
+ }
+ }
```

Remark: the proposed simulation framework allows computation of the target series \( y_t \) which is generally unobservable. Therefore, we can obtain sample mean-square filter errors, the left-hand side of 2.27. In applications, the latter are generally unobservable.

2. For each of the above realizations: approximate the ideal trend \( y_t \) by a real-time estimate \( \hat{y}_t \) based on a filter of length \( L = 12 \) and compute the criterion value (the right-hand side of 2.27).

- Use filters of length \( L = 12 \) and apply the MSE-DFA criterion 2.27 or, equivalently, the MSE-DFA function proposed in section 1.5.2 to the data samples of length 120 corresponding to \( y_1,...,y_{120} \).

```r
> plot_T<-F
> periodogram<-matrix(ncol=3,nrow=len/2+1)
> trffkt<-periodogram
> perf_mat<-matrix(nrow=3,ncol=2)
> dimnames(perf_mat)[[2]]<-c("Criterion Value","Mean-Square Sample Filter Error")
> dimnames(perf_mat)[[1]]<-c("a1=0.9","a1=0","a1=-0.9")
> # Filter length
> L<-12
> # Real-time design

4So-called nowcasts, see section 3.2.1 for details. A nowcast is obtained by setting \( Lag = 0 \) in the head of the MDFA-function.
> Lag<-0
> # Target ideal trend
> Gamma<-c(1,(1:(len/2))<len/12)
> b<-matrix(nrow=L,ncol=3)
> # Compute real-time filters
> for (i in 1:3)#i<-1
+ {
+    # Compute the periodogram based on the data (length 120)
+    periodogram[,i]<-per(x[,i],plot_T)$per
+    # Optimize filters
+    filt<-dfa_ms(L,periodogram[,i],Lag,Gamma)
+    trffkt[,i]<-filt$trffkt
+    b[,i]<-filt$b
+    # Compute real-time outputs (we can use the longer series in order
+    # to obtain estimates for time points t=1,...,11)
+    for (j in 1:len)
+        yhat[j,i]<-filt$b%*%xh[lenh/2+(-len/2)-1+j:(j-L+1),i]
+    # Compute criterion values
+    perf_mat[i,1]<-(2*pi/length(Gamma))*abs(Gamma-trffkt[,i])^2%*%periodogram[,i]
+ }

- Compute criterion values (the right-hand side of criterion 2.27) for each time series.

> for (i in 1:3)
+ {
+    # Compute criterion values
+    perf_mat[i,1]<-(2*pi/length(Gamma))*abs(Gamma-trffkt[,i])^2%*%periodogram[,i]
+ }

> perf_mat[,1]

\[
\begin{array}{ccc}
a_1=0.9 & a_1=0 & a_1=-0.9 \\
0.31536546 & 0.04688183 & 0.02791224
\end{array}
\]

3. Compute sample mean-square filter errors: the left-hand side of criterion 2.27

> # Compute time-domain MSE
> mse<-apply(na.exclude((yhat-y)^2),2,mean)
> perf_mat[,2]<-mse
> perf_mat[,2]

\[
\begin{array}{ccc}
a_1=0.9 & a_1=0 & a_1=-0.9 \\
0.32154820 & 0.05314833 & 0.02791222
\end{array}
\]

4. Compare left- and right-hand sides of 2.27 see table 2.1 The alleged tightness of the approximation in 2.27 seems confirmed.
2.2. DFA BOOSTER

<table>
<thead>
<tr>
<th>Criterion Value</th>
<th>Mean-Square Sample Filter Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1=0.9</td>
<td>0.315</td>
</tr>
<tr>
<td>a1=0</td>
<td>0.047</td>
</tr>
<tr>
<td>a1=-0.9</td>
<td>0.028</td>
</tr>
</tbody>
</table>

Table 2.1: Criterion values vs. sample (mean-square) filter errors

5. Verify optimality of the estimated filter coefficients (left as an exercise to the reader).

Remark
Consistency of a statistic means that the estimate converges to its target when adding (arbitrarily many) data. Often, the approximation error is of order $1/\sqrt{T}$ in absolute value, where $T$ is the sample size. Super-consistency, of the right-hand side of 2.27, means that the approximation error is of smaller order than $1/\sqrt{T}$. Uniform super-consistency means that this asymptotic tightness remains valid after optimization. Although desirable, at least from a theoretical perspective, the practical relevance of asymptotic concepts is unclear because data-samples are finite and, sometimes, very short (particularly in the case of evolutive environments). It is therefore useful to gain experience by stressing finite sample results\textsuperscript{5} such as summarized in table 2.1.

2.2.6 Exercises: Explaining the Optimization Criterion

The DFA-criterion 2.27 corresponds to a weighted optimization: the real-time filter $\hat{\Gamma}(\cdot)$ should be close to the target $\Gamma(\cdot)$ in ‘loaded’ frequencies whereby the amount of loading is measured by the periodogram (alternative spectral estimates are analyzed in chapter 6). We here briefly illustrate this particular optimization concept by analyzing real-time filter outputs and filter characteristics (amplitude and time-shift functions).

1. Compare graphically target $y_t$ and real-time estimate $\hat{y}_t$ for each realization of the previous exercise, see fig 2.1.

```r
> file = paste("z_dfa_ar1_sym_output.pdf", sep = "")
> pdf(file = paste(path.out,file,sep=""), paper = "special", width = 6, height = 6)
> par(mfrow=c(3,1))
> for (i in 1:3) #i<-1
+ { 
+ ymin<-min(min(y[,i]),min(na.exclude(yhat[,i])))
+ ymax<-max(max(y[,i]),max(na.exclude(yhat[,i])))
+ ts.plot(yhat[,i],main=paste("Time-domain MSE = ",
+ round(mse[i,3],3)," , Frequency-domain MSE = ",
+ round(perf_mat[i,1,3],3)," , a1 = ",a_vec[i],sep=""),col="blue",yylim=c(ymin,ymax),
+ gpars=list(xlab="year", ylab=""))
+ lines(y[,i],col="red")
+ }
```

\textsuperscript{5}The reader is free to generate other realizations (and possibly specify other processes or distributions) and to compute mean-square approximation-errors instead of ‘single realization’ numbers.
Visual inspection seems to conflict with MSE-performances: the real-time filter with the largest MSE (upper panel) appears to fit its target best. This conflict can be alleviated,
to some extent, by adjusting for differences in scale of the time series. But it is obvious that the task of the filter in the upper panel seems easier, in some way, than that of the bottom filter, whose output is much noisier than its target. A more refined analysis reveals, also, that the real-time estimates appear to be systematically shifted to the right: they are delayed. Once again, the upper filter seems least affected. In summary: the difficulty of the estimation task seems to depend on the data-generating process (DGP).

2. Compute and compare graphically amplitude and time-shift functions for all three realizations (processes), see fig.2.8.

```r
omega_k <- pi*0:(len/2)/(len/2)
file = paste("z_dfa_ar1_amp_shift.pdf", sep = ")
pdf(file = paste(path.out,file,sep=""), paper = "special", width = 6, height = 6)
par(mfrow=c(2,2))
amp<-abs(trffkt)
shift<-Arg(trffkt)/omega_k
plot(amp[,1],type="l",main="Amplitude functions", + axes=F,xlab="Frequency",ylab="Amplitude",col="black",ylim=c(0,1))
lines(amp[,2],col="orange")
lines(amp[,3],col="green")
lines(Gamma,col="violet")
mtext("Amplitude a1=0.9", side = 3, line = -1,at=len/4,col="black")
mtext("Amplitude a1=0", side = 3, line = -2,at=len/4,col="orange")
mtext("Amplitude a1=-0.9", side = 3, line = -3,at=len/4,col="green")
mtext("Target", side = 3, line = -4,at=len/4,col="violet")
axis(1,at=c(0,1:6*len/12+1),labels=c("0","pi/6","2pi/6","3pi/6", + "4pi/6","5pi/6","pi"))
axis(2)
box()
plot(shift[,1],type="l",main="Time-shifts", + axes=F,xlab="Frequency",ylab="Shift",col="black", + ylim=c(0,max(na.exclude(shift[,3]))))
lines(shift[,2],col="orange")
lines(shift[,3],col="green")
lines(rep(0,len/2+1),col="violet")
mtext("Shift a1=0.9", side = 3, line = -1,at=len/4,col="black")
mtext("Shift a1=0", side = 3, line = -2,at=len/4,col="orange")
mtext("Shift a1=-0.9", side = 3, line = -3,at=len/4,col="green")
```

\(^6\) A relative (signal-to-noise) measure would seem more appropriate than the raw MSE, see later chapters. \(^7\) Smaller \(a_1\) correspond to noisier realizations \(x_t\) which, in turn, lead to noisier real-time estimates \(\hat{y}_t\) (increasingly difficult estimation problems). \(^8\) Our plots are similar but not identical to the plots in DFA section 4.1.1, exercise 1: here we use data in the middle of the long sample whereas in DFA the first 120 observations are used.
> mtext("Target", side = 3, line = -4, at = len/4, col = "violet")
> axis(1, at = c(0, 1:6*len/12+1), labels = c("0", "pi/6", "2pi/6", "3pi/6",
+ "4pi/6", "5pi/6", "pi"))
> axis(2)
> box()
> plot(periodogram[, 1], type = "l", main = "Periodograms",
+ axes = F, xlab = "Frequency", ylab = "Periodogram", col = "black",
+ ylim = c(0, max(periodogram[, 3])/6))
> lines(periodogram[, 2], col = "orange")
> lines(periodogram[, 3], col = "green")
> mtext("Periodogram a1=0.9", side = 3, line = -1, at = len/4, col = "black")
> mtext("Periodogram a1=0", side = 3, line = -2, at = len/4, col = "orange")
> mtext("Periodogram a1=-0.9", side = 3, line = -3, at = len/4, col = "green")
> axis(1, at = c(0, 1:6*len/12+1), labels = c("0", "pi/6", "2pi/6", "3pi/6",
+ "4pi/6", "5pi/6", "pi"))
> axis(2)
> box()
> dev.off()
Noise and delay of the real-time estimates $\hat{y}_t$ in fig. 2.1 are due to leaking amplitude functions (incomplete stop-band rejection) and to non-vanishing time-shift functions of the real-time filters. Chapter 5 proposes a more general optimization paradigm which addresses both aspects simultaneously.

The time-shift (top right panel) of the black filter ($a_1 = 0.9$) remains comparatively small. Its amplitude function (top left panel) is the farthest away from the target in the stop-band $\omega > \pi/6$ but it is closest to the target in the passband $\omega \leq \pi/6$: the optimization criterion seems to trade (poorer) high-frequency damping against (improved) passband properties. In summary: $\hat{\Gamma}(\cdot)$ tracks $\Gamma(\cdot)$ towards the loaded frequencies, as
CHAPTER 2. CLASSIC MEAN-SQUARE ERROR (MSE) PERSPECTIVE

measured by the periodogram (bottom panel) in 2.27

2.3 MDFA: Problem-Structure and Target (MSE-Perspective)

2.3.1 Emphasizing the Filter Error

The previous (univariate) DFA has been generalized to a multivariate framework in Wildi (2008.2), theorem 7.1, and in McElroy-Wildi (2015) (MDFA-paper). We here briefly summarize the main results in the case of stationary processes (see chapters 12, 13 and 14 for generalizations to non-stationary processes).

Let the target \( y_t \) be defined by 2.22 and let \( x_t, w_{tj}, t = 1, ..., T \) and \( j = 1, ..., m \) be an \( m + 1 \)-dimensional set of explaining variables. Consider

\[
\hat{\Gamma}_X(\omega_k) \Xi_{TX}(\omega_k) + \sum_{n=1}^{m} \hat{\Gamma}_{W_n}(\omega_k) \Xi_{TW_n}(\omega_k)
\]

(2.29)

where

\[
\hat{\Gamma}_X(\omega_k) = \sum_{j=0}^{L-1} b_{Xj} \exp(-ij\omega_k)
\]

(2.30)

\[
\hat{\Gamma}_{W_n}(\omega_k) = \sum_{j=0}^{L-1} b_{w_{n},j} \exp(-ij\omega_k)
\]

(2.31)

are the (one-sided) transfer functions of the (real-time) filters, whose coefficients must be determined, and where \( \Xi_{TX}(\omega_k), \Xi_{TW_n}(\omega_k) \) are the corresponding DFTs. The filter coefficients can be collected in a matrix \( \mathbf{B} = (b_{X}, b_{w_1}, ..., b_{w_m}) \) where \( b_{X} = (b_{X0}, ..., b_{X,L-1})' \) and \( b_{w_n} = (b_{w_{n0}}, ..., b_{w_{n,L-1}})' \) are the vectors of filter coefficients. Then the following multivariate MSE-criterion

\[
\frac{2\pi}{T} \sum_{k=-T/2}^{T/2} \left| \left( \Gamma(\omega_k) - \hat{\Gamma}_X(\omega_k) \right) \Xi_{TX}(\omega_k) - \sum_{n=1}^{m} \hat{\Gamma}_{W_n}(\omega_k) \Xi_{TW_n}(\omega_k) \right|^2 \to \min_{\mathbf{B}}
\]

(2.32)

generalizes the univariate DFA-MSE criterion 2.27 see theorem 7.1 in Wildi (2008.2) and McElroy-Wildi (2015)\(^{10}\).

Remarks:

- Due to the explicit link between \( x_t \) and the target \( y_t \), the former is generally informative about \( y_t \). There might be exceptions\(^{11}\) though, and in such a case we assume \( b_{X} = 0 \) or, equivalently, \( \hat{\Gamma}_X \equiv 0 \) in the above expressions.

- In the absence of additional explaining series \( (w_{tn}) \) the multivariate criterion 2.32 reduces to 2.27. The proposed (MSE-) MDFA-criterion thus generalizes the previous DFA.

\(^{9}\)The explicit link between \( y_t \) and \( x_t, \) as defined by 2.22 justifies to distinguish \( x_t \) from the other explaining series.

\(^{10}\)MDFA-paper.

\(^{11}\)GDP-series: official releases are revised and subject to publication lags which might alter real-time performances when including GDP in the set of explaining series, see chapter 10.
2.3.2 One- and Multi-Step Ahead Criteria

The univariate and multivariate criteria \(2.27\) and \(2.32\) rely on a general target specification. The classic one-step ahead mean-square criterion could be replicated by specifying \(\gamma_{-1} = 1, \gamma_k = 0, k \neq 0\) in \(2.22\):

\[
y_t = \sum_{k=-\infty}^{\infty} \gamma_k x_{t-k} = x_{t+1}
\]

In this case, criterion \(2.32\) becomes

\[
\frac{2\pi}{T} \sum_{k=-T/2}^{T/2} \left( \exp(i\omega_k) - \hat{\Gamma}_X(\omega_k) \right) \Xi_{TX}(\omega_k) - \sum_{n=1}^{m} \hat{\Gamma}_W_n(\omega_k) \Xi_{TW_n}(\omega_k) \right)^2 \rightarrow \min_B
\]

where the anticipative allpass target filter \(\Gamma(\omega_k) := \exp(i\omega_k)\) rotates the DFT \(\Xi_{TX}(\omega_k)\) in the frequency-domain (shifts the data in the time-domain). A direct link to classical (pseudo-) maximum likelihood approaches is provided in chapter \(6\) for details (replication of model-based performances by MDFA). An interesting consequence of the (anticipative) allpass target will be analyzed in chapter \(5\) where it is shown that the general ATS-Trilemma, inherent to the MDFA, collapses to a less rich (more rigid) AT-dilemma in the case of forecasting. To conclude, we note that \(h\)-step ahead forecasting could be obtained by specifying the allpass target \(\Gamma(\omega_k) := \exp(ih\omega_k)\).

Remarks

- Typically, in the time-domain, \(h\) observations are lost when estimating the coefficients of a direct \(h\)-step ahead forecast equation. In contrast, the whole sample remains at disposal in the frequency-domain because time-shifts, of the data, are handled by rotations, of the (full-sample) DFTs. This ‘magic trick’ is obtained by an implicit assumption about the data whose discovery is left as an exercise to the reader (hint: it is discussed in \(\text{DFA}\) and further below).

- We here proposed forecasts of the original data. In section \(3.2.1\) we generalize this concept to forecasts (and nowcasts/backcasts) of a general signal specification i.e. transformed data.

2.4 Matrix Notation and Generalized Least-Squares Solution

We here introduce a convenient matrix notation which will reveal useful when tackling filter constraints (see chapter \(4\)) as well as more sophisticated optimization criteria (customization, regularization, mixed-frequency). We then derive the solution of the MSE-criterion \(2.32\).
2.4.1 Matrix Notation

By symmetry of its summands around \( \omega_0 = 0 \), criterion 2.32 can be rewritten in a numerically more efficient form

\[
\frac{2\pi}{T} \left| \left( \Gamma(0) - \hat{\Gamma}_X(0) \right) \Xi_{TX}(0) - \sum_{n=1}^{m} \hat{\Gamma}_{W_n}(0) \Xi_{TW_n}(0) \right|^2 + \frac{2\pi}{T} \sum_{k>0}^{T/2} \left| \left( \Gamma(\omega_k) - \hat{\Gamma}_X(\omega_k) \right) \Xi_{TX}(\omega_k) - \sum_{n=1}^{m} \hat{\Gamma}_{W_n}(\omega_k) \Xi_{TW_n}(\omega_k) \right|^2 \rightarrow \min_B \tag{2.34}
\]

Note that frequency zero is counted once, only, whereas all strictly positive frequencies are duplicated. We now derive a more convenient vector notation for the above criterion.

Let \( X \) be a matrix whose \( k \)-th row \( X_k \) is defined as

\[
X_k = (1 + I_{k>0}) \cdot \begin{pmatrix}
\Xi_{TX}(\omega_k) & \exp(-i\omega_k)\Xi_{TX}(\omega_k) & \ldots & \exp(-i(L-1)\omega_k)\Xi_{TX}(\omega_k) \\
\Xi_{TW_1}(\omega_k) & \exp(-i\omega_k)\Xi_{TW_1}(\omega_k) & \ldots & \exp(-i(L-1)\omega_k)\Xi_{TW_1}(\omega_k) \\
\Xi_{TW_2}(\omega_k) & \exp(-i\omega_k)\Xi_{TW_2}(\omega_k) & \ldots & \exp(-i(L-1)\omega_k)\Xi_{TW_2}(\omega_k) \\
\vdots & \vdots & \ddots & \vdots \\
\Xi_{TW_m}(\omega_k) & \exp(-i\omega_k)\Xi_{TW_m}(\omega_k) & \ldots & \exp(-i(L-1)\omega_k)\Xi_{TW_m}(\omega_k)
\end{pmatrix}
\]

where the Vec\(_{\text{row}}\)-operator appends rows (we use this notation in order to avoid margin-overflow) and where the indicator function \((1 + I_{k>0}) = \begin{cases} 1 & k = 0 \\ 2 & k = 1, \ldots, T/2 \end{cases}\) accounts for the fact that frequency zero occurs once only in the criterion. The length of the \( k \)-th row is \((L-1)(m+1)\) and the dimension of the design-matrix \( X \) is \((T/2 + 1) \times (L-1)(m+1)\). Next, define a coefficient vector \( b \) and a target vector \( Y \)

\[
b = \text{Vec}_{\text{col}}(B) = \begin{pmatrix}
b_{X0} & b_{W_01} & b_{W_02} & \ldots & b_{W_0m} \\
b_{X1} & b_{W_11} & b_{W_12} & \ldots & b_{W_1m} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
b_{XL-1} & b_{W_{L-1}1} & b_{W_{L-1}2} & \ldots & b_{W_{L-1}m}
\end{pmatrix}
\]

\[
Y = \begin{pmatrix}
\Gamma(\omega_0)\Xi_{TX}(\omega_0) \\
2\Gamma(\omega_1)\Xi_{TX}(\omega_1) \\
2\Gamma(\omega_2)\Xi_{TX}(\omega_2) \\
\vdots \\
2\Gamma(\omega_{T/2})\Xi_{TX}(\omega_{T/2})
\end{pmatrix}
\]

where Vec\(_{\text{col}}\) stacks the columns of the coefficient matrix \( B \). Note, once again, that all frequencies larger than zero are duplicated in \( Y \). Criterion 2.32 or, equivalently 2.34, can now be expressed more conveniently in vector notation:

\[
(Y - Xb)'(Y - Xb) \rightarrow \min_b \tag{2.36}
\]
2.4. MATRIX NOTATION AND GENERALIZED LEAST-SQUARES SOLUTION

where \((Y - Xb)'\) is the Hermitian conjugate of \(Y - Xb\) (transpose and complex conjugate). If all vectors and matrices were real (real numbers) then the solution to this minimization problem would be the well-known least-squares estimate

\[
\hat{b} = (X'X)^{-1} X'Y
\]

Unfortunately, the above vectors and matrices are complex-valued. Before presenting a correct least-squares estimate we propose to rotate all DFT’s \(^\text{14}\): this transformation is not strictly necessary but it simplifies later expressions.

\[
\frac{2\pi}{T} \sum_{k=-T/2}^{T/2} \left| \left( \Gamma(\omega_k) - \hat{\Gamma}_X(\omega_k) \right) \Xi_{TX}(\omega_k) - \sum_{n=1}^{m} \hat{\Gamma}_W(\omega_k) \Xi_{TW_n}(\omega_k) \right|^2
\]

\[
= \frac{2\pi}{T} \sum_{k=-T/2}^{T/2} \left| \Gamma(\omega_k) \Xi_{TX}(\omega_k) \right| \left| \exp (i * \arg (\Gamma(\omega_k) \Xi_{TX}(\omega_k))) - \hat{\Gamma}_X(\omega_k) \Xi_{TX}(\omega_k) \right|^2
\]

\[
- \sum_{n=1}^{m} \hat{\Gamma}_W(\omega_k) \Xi_{TW_n}(\omega_k)
\]

\[
= \frac{2\pi}{T} \sum_{k=-T/2}^{T/2} \left| \Gamma(\omega_k) \Xi_{TX}(\omega_k) \right| \left| \exp (-i * \arg (\Gamma(\omega_k) \Xi_{TX}(\omega_k))) - \hat{\Gamma}_X(\omega_k) \Xi_{TX}(\omega_k) \right|^2 \tag{2.37}
\]

where the arg-function corresponds to the phase (angle) of a complex number and where the function is applied component-by-component to the complex-valued vector. Let us define a rotated design-matrix \(X_{\text{rot}}\) and a rotated target vector \(Y_{\text{rot}}\)

\[
X_{k,\text{rot}} = X_k \exp (-i * \arg (\Gamma(\omega_k) \Xi_{TX}(\omega_k))) \tag{2.38}
\]

where \(X_{k,\text{rot}}\) designates the \(k\)-th row of \(X_{\text{rot}}\) and where

\[
Y_{\text{rot}} = |Y|
\]

is a (real) positive target vector.

2.4.2 Generalized Least Squares Solution

The optimization criterion becomes

\[
(Y_{\text{rot}} - X_{\text{rot}}b)'(Y_{\text{rot}} - X_{\text{rot}}b) \to \min_b \tag{2.39}
\]

The general (matrix derivative) formula for tackling this complex-valued minimization problem is

\[
d/db \text{ Criterion} = d/db (Y_{\text{rot}} - X_{\text{rot}}b)'(Y_{\text{rot}} - X_{\text{rot}}b)
\]

\[
= -(Y_{\text{rot}} - X_{\text{rot}}b)'X_{\text{rot}} - (Y_{\text{rot}} - X_{\text{rot}}b)'X_{\text{rot}}
\]

\[
= -2Y_{\text{rot}}'R(X_{\text{rot}}) + 2b'\Xi_{TX}(X_{\text{rot}})
\]

\[^{13}\text{For simplicity we omitted the normalization } \frac{2\pi}{T} \text{ which is irrelevant for optimization.}\]

\[^{14}\text{The criterion } 2.32 \text{ is invariant to such a transformation.}\]
where $X_{\text{rot}}'$ is the Hermitian conjugate (transposed and complex conjugate); $X_{\text{rot}}^T$ is the transposed (but not complex conjugate) matrix; $\overline{X_{\text{rot}}}$ is the complex conjugate (but not transposed) matrix; $\Re(\cdot)$ means the real part of a complex number. Note that $b' = b^T$ and $Y' = Y^T$ because both vectors are real. The generalized least-squares estimate is obtained by equating the previous expression to zero

$$\hat{b} = (\Re(X_{\text{rot}}'X_{\text{rot}}))^{-1} \Re(X_{\text{rot}})' Y_{\text{rot}} \quad (2.43)$$

The resulting $\hat{b}$ is the solution to the MDFA-MSE signal extraction problem 2.32 (or 2.27).

### 2.4.3 R-Code

**DFA**

The proposed matrix notation and the MSE (generalized least-squares) estimate $\hat{b}$ can be traced-back in the R-code of the (MSE-) DFA proposed in section 1.5.2:

```r
> dfa_ms

function(L,periodogram,Lag,Gamma)
{
  K<-length(periodogram)-1
  X<-exp(-1.i*Lag*pi*(0:(K))/(K))*rep(1,K+1)*sqrt(periodogram)
  X_y<-exp(-1.i*Lag*pi*(0:(K))/(K))*rep(1,K+1)
  for (l in 2:L) #l<-L<-21
  {
    X<-cbind(X,(cos((l-1-Lag)*pi*(0:(K))/(K))+
              1.i*sin((l-1-Lag)*pi*(0:(K))/(K))))*sqrt(periodogram))
    X_y<-cbind(X_y,(cos((l-1-Lag)*pi*(0:(K))/(K))+
                   1.i*sin((l-1-Lag)*pi*(0:(K))/(K))))
  }
  xtx<-t(Re(X))%*%Re(X)+t(Im(X))%*%Im(X)
  # MA-Filter coefficients
  b<-as.vector(solve(xtx)%*%(t(Re(X_y))%*%(Gamma*periodogram)))
  # Transferfunction
  trffkt<-1:(K+1)
  trffkt[1]<-sum(b)
  for (k in 1:(K)) #k<-1
  {
    trffkt[k+1]<-(b[k]*exp(1.i*k*(0:(length(b)-1)))*pi/(K)))
  }
  return(list(b=b,trffkt=trffkt))
}
```
2.5. REPLICATION OF DFA BY MDFA

The code replicates the above formulas up to the particular weighting of frequency zero (which is not duplicated in 2.34), see exercise 1, section 2.5.1.

MDFA

The MDFA estimation routine introduced in section 1.5.3 is more general than the MSE-problem discussed in this chapter. The least-squares solution 2.43 is nested as a special case. The DFA is nested too, see section 2.5 (replication). Other nested solutions replicate classic model-based approaches, see chapter 6. Conceptually, ‘nesting’ is obtained by specifying hyperparameters in the head of the function call.

2.5 Replication of DFA by MDFA

We rely on the data in the previous exercises, see section 2.2.6, and replicate the obtained DFA-results (criterion 2.27) by MDFA (criterion 2.32).

2.5.1 Exercises

1. Define the data-matrix (target and explaining variable) and compute the DFTs. For ease of exposition we consider the first AR(1)-process only ($a_1 = 0.9$).

   > # Select the first process
   > i_process<-1
   > # Define the data-matrix:
   > # The first column must be the target series.
   > # Columns 2,3,... are the explaining series. In a univariate setting
   > # target and explaining variable are identical
   > data_matrix<-cbind(x[,i_process],x[,i_process])
   > # Determine the in-sample period (fully in sample)
   > insample<-nrow(data_matrix)
   > # Compute the DFT by relying on the multivariate DFT-function: d=0 for stationary data (default settings)
   > weight_func<-spec_comp(insample, data_matrix, d)$weight_func
   > # For this replication exercise we have to define a second DFT which is altered in frequency zero only
   > weight_func_mod<-weight_func
   > # Frequency zero is scaled by $\sqrt{2}$ in order to be able to replicate DFA
   > weight_func_mod[1,]<-sqrt(2)*weight_func[1,]

   We here use two different DFTs: weight_func, the original DFT, and weight_func_mod which is altered by a constant scaling term in frequency zero. As we shall see, the latter (modified) DFT allows to replicate DFA perfectly. To be clear: although the effect of this modification is marginal (see below) we do not recommend it, except for replication purposes.

2. Estimate optimal filter coefficients and compare DFA- and MDFA-estimates, for original as well as modified DFTs.
> # Source the default (MSE-) parameter settings
> source(file=paste(path_MDFA.pgm,"control_default.r",sep=""))
> # Estimate filter coefficients: original (unmodified) DFTs
> mdfa_obj_original<-mdfa_analytic_new(K,L,lambda,weight_func,Lag,Gamma,expweight,cutoff,ii,ii)
> # Estimate filter coefficients: modified DFTs (for replication purpose only)
> mdfa_obj_mod<-mdfa_analytic_new(K,L,lambda,weight_func_mod,Lag,Gamma,expweight,cutoff,ii,ii)
> # Filter coefficients: compare MDFA and previous DFA
> b_mat<-cbind(mdfa_obj_original$b,mdfa_obj_mod$b,b[,i_process])
> dimnames(b_mat)[[2]]<-c("MDFA original","MDFA modified","DFA")
> dimnames(b_mat)[[1]]<-paste("lag ",0:(L-1),sep="")
> b_mat

<table>
<thead>
<tr>
<th></th>
<th>MDFA original</th>
<th>MDFA modified</th>
<th>DFA</th>
</tr>
</thead>
<tbody>
<tr>
<td>lag 0</td>
<td>0.53727437</td>
<td>0.53923140</td>
<td>0.53923140</td>
</tr>
<tr>
<td>lag 1</td>
<td>0.10239982</td>
<td>0.10211162</td>
<td>0.10211162</td>
</tr>
<tr>
<td>lag 2</td>
<td>0.17192360</td>
<td>0.17216056</td>
<td>0.17216056</td>
</tr>
<tr>
<td>lag 3</td>
<td>0.11457416</td>
<td>0.11459648</td>
<td>0.11459648</td>
</tr>
<tr>
<td>lag 4</td>
<td>0.07839113</td>
<td>0.07839963</td>
<td>0.07839963</td>
</tr>
<tr>
<td>lag 5</td>
<td>0.02200310</td>
<td>0.02217871</td>
<td>0.02217871</td>
</tr>
<tr>
<td>lag 6</td>
<td>0.05489930</td>
<td>0.05507491</td>
<td>0.05507491</td>
</tr>
<tr>
<td>lag 7</td>
<td>-0.03093838</td>
<td>-0.03092987</td>
<td>-0.03092987</td>
</tr>
<tr>
<td>lag 8</td>
<td>-0.05126126</td>
<td>-0.05123894</td>
<td>-0.05123894</td>
</tr>
<tr>
<td>lag 9</td>
<td>-0.03594181</td>
<td>-0.03570485</td>
<td>-0.03570485</td>
</tr>
<tr>
<td>lag 10</td>
<td>-0.08953608</td>
<td>-0.08982428</td>
<td>-0.08982428</td>
</tr>
<tr>
<td>lag 11</td>
<td>0.04272186</td>
<td>0.04467888</td>
<td>0.04467888</td>
</tr>
</tbody>
</table>

The modified MDFA replicates DFA exactly, as desired. Also, the differences between original and modified MDFAs are marginal, as expected. In the following we shall always rely on the original MDFA (unmodified DFTs), because frequency zero is tackled the right way.

3. The value of the multivariate criterion 2.32 is computed explicitly by the MDFA-function:

> criterion_mdfa<-mdfa_obj_original$rever
> # DFA-numbers are stored in perf_mat
> crit_mdfa<-matrix(c(criterion_mdfa,perf_mat[,i_process],1),ncol=1)
> dimnames(crit_mdfa)[[1]]<-c("MDFA criterion (original)","DFA criterion","sample MSE")
> dimnames(crit_mdfa)[[2]]<-"MSE estimates"
> t(crit_mdfa)

<table>
<thead>
<tr>
<th>MDFA criterion (original)</th>
<th>DFA criterion</th>
<th>sample MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE estimates</td>
<td>0.316615</td>
<td>0.3153655</td>
</tr>
</tbody>
</table>

Note that the original MDFA is marginally closer to the sample MSE (correct handling of frequency zero).
2.6 Qualitative Easing by Leading Indicators: an Empirical Study

We here try to quantify performance gains obtained by including an additional idealized leading indicator into the univariate design of the previous section. Specifically, the new explaining series \( w_{1t} \) is defined as

\[
w_{1t} = x_{t+\delta} + s \cdot \epsilon_{t+\delta} \tag{2.44}
\]

where \( x_t \) is the data of the previous univariate design, \( \epsilon_t \) is an idiosyncratic noise component (iid zero-mean Gaussian standardized), \( s \) is a scaling and \( \delta \) is a time-shift. In the first exercise, section 2.6.1, we select \( s = 0.1 \) (a weak idiosyncratic component) and \( \delta = 1 \) (lead by one time unit). We do not argue about the practical relevance or pertinence of this particular setting; instead, our main purpose is to illustrate potential gains obtainable by including a leading indicator: the reader is free to experiment with alternative designs (larger \( s \) for example). In section 2.6.2 we disentangle and quantify time-shift effects (\( \delta \)) and signal-to-noise effects (\( s \)).

2.6.1 Bivariate MDFA vs. Univariate DFA

1. Use the data in the previous section 2.5, construct the leading indicator 2.44 and specify the (3-dim) data-matrix.

```r
> set.seed(12)
> # Select the AR(1)-process with coefficient 0.9
> i_process<-1
> # Scaling of the idiosyncratic noise
> scale_idiosyncratic<-0.1
> eps<-rnorm(nrow(xh))
> indicator<-xh[,i_process]+scale_idiosyncratic*eps
> # Data: first column=target, second column=x, third column=shifted (leading) indicator
> data_matrix<-cbind(xh[,i_process],xh[,i_process],c(indicator[2:nrow(xh)],NA))
> dimnames(data_matrix)[[2]]<-c("target", "x", "leading indicator")
> # Extract 120 observations from the long sample
> data_matrix_120<-data_matrix[lenh/2+(-len/2):((len/2)-1),]
> head(data_matrix_120)
```

<table>
<thead>
<tr>
<th>target</th>
<th>x</th>
<th>leading indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0686889</td>
<td>1.0686889</td>
<td>0.8136893</td>
</tr>
<tr>
<td>0.9496807</td>
<td>0.9496807</td>
<td>0.6149977</td>
</tr>
<tr>
<td>0.6627989</td>
<td>0.6627989</td>
<td>1.7381388</td>
</tr>
<tr>
<td>1.6465327</td>
<td>1.6465327</td>
<td>1.6747908</td>
</tr>
</tbody>
</table>

\(^{15}\)This is a degenerate common factor model in which the factor is identical with \( x_t \): it is directly observable or, stated otherwise, the idiosyncratic component of \( x_t \) vanishes.

\(^{16}\)A larger \( s \) implies that the indicator is less informative about the target \( y_t \).
The first two series are identical; the new third one leads by one-time unit and it is contaminated by noise.

2. Compute the DFTs of the data.

   > # Fully in sample
   > insample<-nrow(data_matrix_120)
   > # d=0 for stationary series: see default settings
   > weight_func<-spec_comp(insample, data_matrix_120, d)$weight_func

3. Estimate optimal (MSE-) filter coefficients.

   > # Source the default (MSE-) parameter settings
   > source(file=paste(path_MDFA.pgm,"control_default.r",sep=""))
   > # Estimate filter coefficients
   > mdfa_obj<-mdfa_analytic_new(K,L,lambda,weight_func,Lag,Gamma,expweight,cutoff,i1,i2,weight_structure,white_noise,synchronicity,lag_mat)
   > # Filter coefficients
   > b_mat<-mdfa_obj$b
   > dimnames(b_mat)[[2]]<-c("x","leading indicator")
   > dimnames(b_mat)[[1]]<-paste("Lag ",0:(L-1),sep=""))
   > head(b_mat)

   x     leading indicator
   Lag 0  0.20483766  0.39974339
   Lag 1  0.36053853 -0.07963108
   Lag 2  0.21569004 -0.18757559
   Lag 3  0.14461613 -0.06042970
   Lag 4  0.13622064 -0.02550296
   Lag 5  0.06506111 -0.09642673

   The solution is ‘clever’: the filter of the leading indicator (second column) assigns most weight to the last anticipative observation (top coefficient in the second column). Since the other observations are contaminated by noise, the bivariate design prefers to assign weight to the original data, instead (coefficients in first column).

4. Compute the minimal criterion value.

   > # Criterion value
   > mdfa_obj$rever

   [1]  0.1461433

   The new criterion value is substantially smaller than the DFA (0.315, see previous exercise).
5. Verify that the mean-square sample error is indeed smaller (recall that the sample MSE is generally not observable because it involves knowledge of the target $y_t$).

- Apply the (bivariate) filter to the data

```r
> yhat_multivariate<-rep(NA,len)
> for (j in 1:len)
  + yhat_multivariate[j]<-sum(apply(b_mat*data_matrix[lenh/2+(-len/2)-1+j:(j-L+1),2:3],1,sum))
```

- Derive the (time-domain) sample mean-square error and compare performances with the DFA, above.

```r
> perf_mse<-matrix(c(mean(na.exclude((yhat_multivariate-y[,i_process]))^2),mean(na.exclude((yhat[,]-y[,i_process]))^2)),nrow=1)
> dimnames(perf_mse)[[2]]<-c("bivariate MDFA","DFA")
> dimnames(perf_mse)[[1]]<"Sample MSE"
> perf_mse

<table>
<thead>
<tr>
<th></th>
<th>bivariate MDFA</th>
<th>DFA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample MSE</td>
<td>0.1389454</td>
<td>0.3215482</td>
</tr>
</tbody>
</table>
```

Sample MSE-performances of the bivariate design are substantially improved, as expected: the leading indicator is informative (this result depends on the magnitude of the idiosyncratic noise in the construction of the indicator, of course). Also, the criterion value 0.146 is fairly close to the sample MSE 0.139, as desired.

6. Plot and compare target $y_t$ as well as DFA and MDFA real-time estimates $\hat{y}_t$.

```r
> file = paste("z_mdfadfa_ar1_sym_output.pdf", sep = "")
> pdf(file = paste(path.out,file,sep=""), paper = "special", width = 6, height = 6)
> i<-1
> ymin<-min(min(y[,i]),min(na.exclude(yhat[,i])))
> ymax<-max(max(y[,i]),max(na.exclude(yhat[,i])))
> ts.plot(yhat[,i],main=paste("Sample MSE MDFA: ",ylab="",
+ round(perf_mse[1,3]),", DFA: ",round(perf_mse[2,3],sep=""),col="blue",ylim=c(ymin,ymax))
> lines(y[,i],col="red")
> lines(yhat_multivariate,col="green")
> mtext("DFA", side = 3, line = -2,at=len/2,col="blue")
> mtext("target", side = 3, line = -1,at=len/2,col="red")
> mtext("MDFA", side = 3, line = -3,at=len/2,col="green")
> dev.off()
```
The MDFA-output (green) is both smoother and faster than the DFA (blue): noisy ripples are smaller in magnitude and turning points can be detected earlier (lead).

7. Verify that filter coefficients of the bivariate design are optimal: this is left as an exercise to the reader (any other set of bivariate filter coefficients, of length 12, should increase the criterion value 0.146 and/or the sample MSE 0.139).

2.6.2 Measuring Lead and Signal-to-Noise Effects of a Leading Indicator

Experience suggests that the lead and the signal-to-noise ratio are conflicting requirements: faster indicators are generally noisier.\(^\text{17}\) We here attempt to disentangle and to quantify both effects by relying on our previous toy-model

\[
W_{1t} = x_{t+\delta} + s \cdot \epsilon_{t+\delta}
\]

For fixed \(x_t\) we can alter the shift \(\delta\) and the (inverse) signal-to-noise ratio \(s\). To be more specific, we want to analyze non-integer\(^\text{18}\) \(\delta\) in order to be able to quantify (daily) up-dating effects of a monthly indicator in the framework of a mixed-frequency approach, see chapter 11: can we expect gains by up-dating information on a weekly basis, as information flows in, or can we collect the data and defer an up-date to the end of the month without performance losses?

1. Specify candidate time-shifts and (inverse) signal-to-noise ratios

> # Inverse SNR: the variance of the standardized noise is one: we thus normalize by
> # the standard deviation of the data \(x\) (second column of the data matrix)

---

\(^{17}\)In part this is due to – incorrect – usage of anticipating difference filters.

\(^{18}\)Fractional \(\delta\) can be implemented very easily in the frequency-domain since shifts become rotations.
> scale_idiosyncratic_vec<-c(0,0.1,0.5,1,2)/sqrt(var(data_matrix_120[,2]))
> # We select fractional leads: multiples of 0.25
> # A fractional lead of 0.25 corresponds roughly to a week on a monthly time scale
> delta_vec<-0.25*0:4

2. Generate the leading indicators for all combinations of \((\delta, s)\) and compute corresponding mean-square filter errors (criterion values).

> # Initialize the performance matrix
> lead_snr_mat<-matrix(ncol=length(scale_idiosyncratic_vec),nrow=length(delta_vec))
> dimnames(lead_snr_mat)[[2]]<-paste("1/SNR=",sqrt(var(data_matrix_120[,1]))*scale_idiosyncratic_vec,paste"
> dimnames(lead_snr_mat)[[2]][1]<-paste("Univ. design: ",dimnames(lead_snr_mat)[[2]][1],sep=""
> dimnames(lead_snr_mat)[[1]]<-paste("Shift ",delta_vec,paste"
> # Generate the idiosyncratic noise
> set.seed(20)
> eps<-rnorm(nrow(data_matrix_120))
> # Loop over all combinations of leads and SNR-ratios
> for (i in 1:length(scale_idiosyncratic_vec))#i<-1
> + {#j<-1
> + # Add the (suitably scaled) noise: no lead yet.
> + indicator<-data_matrix_120[,2]+scale_idiosyncratic_vec[i]*eps
> + # Overwrite the indicator column with the new time series
> + data_matrix_120[,3]<-indicator
> + # Compute the DFTs (full in-sample, for stationary series d=0)
> + insample<-nrow(data_matrix_120)
> + weight_func<-spec_comp(insample, data_matrix_120, d)$weight_func
> + # Compute the discrete frequency-grid \(\omega_k\): from zero to \(\pi\)
> + omega_k<-(0:(nrow(weight_func)-1))*pi/(nrow(weight_func)-1)
> + # Introduce the fractional time-shift by rotation of the DFT of the indicator (last column)
> + weight_func[,ncol(weight_func)]<-exp(-1.1*delta_vec[j]*omega_k)*weight_func[,ncol(weight_func)]
> + # If the idiosyncratic noise is zero, then we use a univariate design
> + if (i==1)
> + weight_func<-weight_func,-2]
> + # Compute optimal filters and derive the (frequency-domain) MSE
> + mdfa_obj<-mdfa_analytic_new(K,L,lambda,weight_func,Lag,Gamma,expweight,cutoff,ii,2,white_noise,synchronicity,lag_mat)
> + # Store the MSE
> + lead_snr_mat[j,i]<-mdfa_obj$rever
> + }
> }

Comments
CHAPTER 2. CLASSIC MEAN-SQUARE ERROR (MSE) PERSPECTIVE

- In the above R-code, the signal-to-noise feature is implemented in the time-domain; the fractional lead is implemented in the frequency-domain\(^{19}\) by rotation of the DFT by \(\exp(-i\delta \omega_k), k = 0, ..., T/2\).
- The previous exercises confirmed tightness of the approximation\(^{2,32}\) therefore we can rely on the criterion value (computed by our MDFA-routine), as a measure for the MSE. We don’t need to filter the time series explicitly and to compute sample MSEs in the time-domain!

3. Collect all MSEs (criterion values) in a table and analyse the obtained results:

<table>
<thead>
<tr>
<th>Shift</th>
<th>1/SNR= 0</th>
<th>1/SNR= 0.1</th>
<th>1/SNR= 0.5</th>
<th>1/SNR= 1</th>
<th>1/SNR= 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.317</td>
<td>0.266</td>
<td>0.266</td>
<td>0.266</td>
<td>0.266</td>
</tr>
<tr>
<td>0.25</td>
<td>0.263</td>
<td>0.175</td>
<td>0.232</td>
<td>0.256</td>
<td>0.267</td>
</tr>
<tr>
<td>0.5</td>
<td>0.217</td>
<td>0.141</td>
<td>0.190</td>
<td>0.218</td>
<td>0.251</td>
</tr>
<tr>
<td>0.75</td>
<td>0.179</td>
<td>0.125</td>
<td>0.159</td>
<td>0.181</td>
<td>0.222</td>
</tr>
<tr>
<td>1</td>
<td>0.147</td>
<td>0.122</td>
<td>0.130</td>
<td>0.148</td>
<td>0.191</td>
</tr>
</tbody>
</table>

Table 2.2: Effect of lead and of (inverse) signal-to-noise ratio on filter MSE

**Analysis: Design**

- If the noise and the lead vanish \((s = \delta = 0)\), then the design is singular because

\[
\omega_{1t} = x_t + se_t = x_t
\]

Therefore we have to skip one of the redundant explaining variables in the DFT (otherwise the code would issue an error).

- If the idiosyncratic noise component vanishes but the lead does not \((s = 0, \delta > 0)\) then, in principle, the data is not perfectly colinear, at least in the time-domain. In the frequency-domain, though, the two DFT-columns are linearly dependent (rotation) and therefore the proposed design is still singular\(^{20}\). Therefore, all results in the first column correspond to univariate designs, where the single explanatory variable is the noise-free leading indicator.

- Performances reported in columns 2-5 of the first row \((0.266: \delta = 0)\) are identical because all designs are strictly equivalent in informational terms\(^{21}\) one can subtract \(x_t\) (second data-column) from \(\omega_{1t} = x_t + se_t\) to obtain \(se_t\) i.e. the data-matrix \((x_t, \omega_{1t})\) could be substituted by \((x_t, \epsilon_t)\) for all \(s \neq 0\).

- Since \(\epsilon_t\) is independent of \(x_t\) (different random seed) one expects that performances of bivariate designs (columns 2-5) and of the univariate design (column 1) in the first

\(^{19}\) A fractional lead in the time-domain would assume a higher sampling frequency.

\(^{20}\) The DFT assumes that the data is periodic: therefore original and shifted data are perfectly colinear. The singularity could be avoided by computing DFTs of original and shifted series explicitly (instead of rotating the DFT).

\(^{21}\) Recall footnote\(^{15}\) p.31: the proposed design is a degenerate common factor model.
row \((\delta = 0)\) should be identical, at least in theory. In practice, the spurious decrease \(0.266 - 0.317 = 0.051\) of the bivariate designs is entirely due to overfitting. see chapter 9 for a comprehensive development of the topic.

**Analysis: Results**

- Column 1 in the above table corresponds to a univariate design; columns 2-5 are bivariate designs. The first column measures time-shift effects of a single noise-free leading indicator\(^{22}\). Columns 2-4 consider a classic bivariate design, where the original data is augmented by a noisy leading indicator.

- The top-left number \((0.317)\) (univariate design without lead) replicates performances as reported in section 2.5.

- The MSE generally decreases with increasing lead (increasing \(\delta\)) and/or with decreasing (inverse) signal-to-noise ratio (smaller \(s\), except the degenerate case \(s = 0\)). A stronger noise could be compensated, to some extent, by a larger lead, at least in mean-square terms. As an example, a relative lead by half a month could be compensated by an increase of the (inverse) SNR from 0.1 to 1: see cells (3,2) and (5,4) in the above table.

- Delays larger than one time unit (one month) do not seem to add significant value if the noise is weak (second column). If the noise is strong (last column) larger leads may be required in order to compensate for the augmented noise\(^{23}\).

- Postponing the up-date of the filter-output \(\hat{y}_t\) to the end of the month (week 4) might be costly in terms of performance-loss: if the noise component is weak (second column) then a lead of 0.25 (up-date in week 3 instead of week 4) reduces the MSE from 0.266 to 0.175. If noise and signal are equally strong (4-th column) then an up-date of \(\hat{y}_t\) in the middle of the month (shift 0.5) would reduce the MSE from 0.266 to 0.218.

These results suggests pertinence of a mixed-frequency approach for which the filter output is aligned on the inflowing data-stream and continuously up-dated in real-time, see chapter 11.

2.7 Summary

- We emphasized MSE-performances of unconstrained, unregularized and stationary designs.

- We provided a DFA-reminder and confirmed pertinence of the abstract uniform superconsistency argument. We illustrated and interpreted the univariate DFA-criterion.

- We generalized the DFA-criterion to a multivariate framework and derived a formal vector-solution.

\(^{22}\)The DFA cannot be used to replicate these results if the explaining variable is shifted because target and explaining variables wouldn’t be identical.

\(^{23}\)Stronger noise suppression by the filter induces larger delays of the output signal which must be compensated by an increasing lead of the series.
• We replicated the DFA by the more general MDFA and we quantified efficiency gains obtained by a bivariate leading-indicator design (over the univariate approach).

• Our results suggested pertinence and practical relevance of a mixed-frequency approach, for which filter-outputs would be up-dated continuously with inflowing information.
Chapter 3

Filter Revisions

3.1 Introduction

GDP-growth rates as released by the BEA\textsuperscript{1} are summarized in table 3.1:

\begin{verbatim}
> US_GDP<-read.csv(paste(path.dat,"US_GDP.csv",sep=""),header=T)
\end{verbatim}

Columns correspond to publication dates: the first column was published in the first quarter 2009 and the last column in the first quarter 2013. Rows correspond to historical time. The fifth row (2008:Q4) addresses GDP in the last quarter 2008: We see how the initial estimate or first release $-0.96\%$ is revised as new information in subsequent years flows in, ending in a substantial negative growth of $-2.3\%$ according to data up to the first quarter 2013. The magnitude of the revisions depends, among others, on the information content of a time series (degree of aggregation), on filter-adjustments (GDP-data is typically seasonally adjusted) and on forecast-quality of unobserved components (early GDP-releases heavily rely on estimates of components which are not available at first publication). Since the GDP-series is noisy, it might be interesting to apply a filter which suppresses the short-term random fluctuations of the series: one could use the same filter $b_0, \ldots, b_{L-1}$ for all time points $t = L, \ldots, T$ or one could use a sequence of specialized filters $b_{tt}, b_{L-1+t, t}$ which are optimized to extract the relevant information in each time point $t$ in the sample. The resulting time-dependent filter-sequence would induce a new type of revisions, the so-called filter-revisions. The latter are analyzed in this chapter; data-revisions are discussed in chapter 10.

### Table 3.1: US-GDP: yearly vintages starting in Q1 2009 and ending in Q1 2013

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>US.GDP</th>
<th>X09Q1</th>
<th>X10Q1</th>
<th>X11Q1</th>
<th>X12Q1</th>
<th>X13Q1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007</td>
<td>Q4</td>
<td>-0.04%</td>
<td>0.53%</td>
<td>0.72%</td>
<td>0.42%</td>
<td>0.42%</td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td>Q1</td>
<td>0.22%</td>
<td>-0.18%</td>
<td>-0.18%</td>
<td>-0.44%</td>
<td>-0.44%</td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td>Q2</td>
<td>0.70%</td>
<td>0.36%</td>
<td>0.15%</td>
<td>0.33%</td>
<td>0.33%</td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td>Q3</td>
<td>-0.13%</td>
<td>-0.68%</td>
<td>-1.01%</td>
<td>-0.93%</td>
<td>-0.93%</td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td>Q4</td>
<td>-0.96%</td>
<td>-1.37%</td>
<td>-1.74%</td>
<td>-2.30%</td>
<td>-2.30%</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Q1</td>
<td>-1.65%</td>
<td>-1.24%</td>
<td>-1.71%</td>
<td>-1.34%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Q2</td>
<td>-0.18%</td>
<td>-0.18%</td>
<td>-0.17%</td>
<td>-0.08%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Q3</td>
<td>0.55%</td>
<td>0.40%</td>
<td>0.42%</td>
<td>0.36%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Q4</td>
<td>1.40%</td>
<td>1.23%</td>
<td>0.94%</td>
<td>0.99%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Q1</td>
<td>0.92%</td>
<td>0.97%</td>
<td>0.58%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Q2</td>
<td>0.43%</td>
<td>0.93%</td>
<td>0.56%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Q3</td>
<td>0.63%</td>
<td>0.62%</td>
<td>0.64%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Q4</td>
<td>0.78%</td>
<td>0.58%</td>
<td>0.59%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Q1</td>
<td>0.09%</td>
<td>0.02%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Q2</td>
<td>0.33%</td>
<td>0.61%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Q3</td>
<td>0.45%</td>
<td>0.32%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Q4</td>
<td>0.68%</td>
<td>1.01%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>Q1</td>
<td>0.49%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>Q2</td>
<td>0.31%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>Q3</td>
<td>0.77%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>Q4</td>
<td>-0.04%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.2 Filter Revisions

#### 3.2.1 Forecasting, Nowcasting and Smoothing

For simplicity of exposition and ease of notation we first assume a univariate design. Let the target $y_t$ be specified by 2.22

$$y_t = \sum_{k=-\infty}^{\infty} \gamma_k x_{t-k}$$

Until now we sought filter coefficients $b_0, ..., b_{L-1}$ such that $\hat{y}_t = \sum_{k=0}^{L-1} b_k x_{t-k}$ is close to $y_t$ in mean-square (a so-called nowcast). But we could have targeted $y_{t+1}$ (forecast) or $y_{t-1}$ (backcast), instead; more generally, we could be interested in estimating $y_{t+h}$ by relying on data $x_t, ..., x_{t-L-1}$ where $h \in \mathbb{Z}$. In this more general perspective, we aim at finding filter coefficients $b_{kh}, k = h, ..., L-1+h$ such that the finite sample estimate

$$\hat{y}_{t+h}^h := \sum_{k=h}^{L-1+h} b_{kh} x_{t-k}$$ (3.1)
3.2. FILTER REVISIONS

is ‘closest possible’ to \( y_t, h \in \mathbb{Z} \), in mean-square

\[
E \left[ (y_t - \hat{y}_t^h)^2 \right] \rightarrow \min_{b_h}
\]

where \( b_h = (b_{hh}, \ldots, b_{L-1+h,h}) \).

- If \( h = 0 \) we use data \( x_t, \ldots, x_{t-(L-1)} \) for estimating \( y_t \): \( \hat{y}_t^0 \) is a nowcast and \( b_{k0}, k = 0, \ldots, L-1 \) is a real-time filter.
- If \( h = 1 \) we use data \( x_{t-1}, \ldots, x_{t-L} \) for estimating \( y_t \): \( \hat{y}_t^1 \) is a forecast and \( b_{k,1}, k = 1, \ldots, L \) is a forecast filter.
- If \( h = -1 \) we use data \( x_{t+1}, \ldots, x_{t-(L-2)} \) for estimating \( y_t \): \( \hat{y}_t^{-1} \) is a backcast and \( b_{k,-1}, k = -1, \ldots, L - 2 \) is a smoother.

In contrast to classical one-step ahead forecasting which emphasize the original data, see section 2.3.2, we here extend the concept to general signal specifications: we forecast, nowcast or backcast the output \( y_t \) of a possibly bi-infinite filter \( \Gamma(\cdot) \). In this more general perspective the proposed nowcast- and backcast-problems are non-trivial estimation tasks.

Intuitively, a backcast should improve (the filter-MSE should decrease) with decreasing horizon \( h < 0 \) because future information \( x_{t+1}, \ldots, x_{t-h} \) becomes available for tracking the historical target \( y_t \). We now analyze these effects: section 3.2.2 emphasizes filter characteristics (amplitude and time-shifts) as a function of \( h \); filter vintages and the revision error are discussed in section 3.2.3; section 3.2.4 proposes a convenient graphical summary, the so called tentacle plot.

### 3.2.2 Backcasting: Analysis of Filter Characteristics

We here analyze amplitude and time-shift functions of univariate DFA-filters as a function of \( h \leq 0 \). For this purpose we rely on the empirical design introduced in section 2.2.6. Specifically, we estimate optimal filters for \( h = 0, -1, \ldots, -6 \) for the three stationary processes

\[
\begin{align*}
x_t &= 0.9x_{t-1} + \epsilon_t \\
x_t &= \epsilon_t \\
x_t &= -0.9x_{t-1} + \epsilon_t
\end{align*}
\]

The target is the ideal (bi-infinite) lowpass filter with cutoff \( \pi/6 \). Note that we assume the data \( x_t \) to be fixed: data revisions are discussed in chapter 10. The horizon parameter \( h \leq 0 \) corresponds to the \texttt{Lag}-variable in the head of the DFA-function call

```r
> head(dfa_ms)
```

1 function (L, periodogram, Lag, Gamma)

```r
2 {
3     K <- length(periodogram) - 1
4     X <- exp(-(0+1i) * Lag * pi * (0:(K))/(K)) * rep(1, K + 1) *
5       sqrt(periodogram)
6     X_y <- exp(-(0+1i) * Lag * pi * (0:(K))/(K)) * rep(1, K +
```

...
CHAPTER 3. FILTER REVISIONS

> #head(mdfa_analytic_new)

Note however that $Lag = -h$ by convention i.e. a positive Lag means a backcast.

1. Compute optimal finite sample filters for $Lag = 0, ..., (L - 1)/2$ for the above three processes.
   
   **Hint:** we set $L = 13$ in order to obtain a symmetric filter in $Lag = 6$

   ```
   > L<-13
   > yhat_Lag<-array(dim=c(len,3,L/2+2))
   > trffkt<-array(dim=c(len/2+1,3,L/2+2))
   > b<-array(dim=c(L,3,L/2+2))
   > # Compute real-time filters for Lag=,...,L/2 and for the above three AR-processes
   > for (i in 1:3)
   + {
   +    periodogram[,]<-per(x[,i],plot_T)$per
   +    for (Lag in 0:((L/2)+1))
   +    {
   +      filt<-dfa_ms(L,periodogram[,i],Lag,Gamma)
   +      trffkt[,]<-filt$trffkt
   +      b[,]<-filt$b
   +    }
   + # Compute outputs
   +    for (j in L:len)
   +    {
   +      yhat_Lag[j,i,Lag+1]<-filt$b%*%x[j:(j-L+1),i]
   +    }
   + }
   ```

2. Focus on the second process (white noise) and analyze the outcome as a function of Lag, see fig.3.1.

   ```
   > # Discrete frequency grid
   > omega_k<-pi*0:(len/2)/(len/2)
   > colo<-rainbow(L/2+2)
   > file = paste("z_dfa_ar1_amp_shift_Lag_0.pdf", sep = "")
   > pdf(file = paste(path.out,file,sep=""), paper = "special", width = 6, height = 6)
   > par(mfrow=c(2,2))
   > amp<-abs(trffkt)
   > shift<-Arg(trffkt)/omega_k
   > for (i in 2:2)
   + {
   +    ymin<-min(amp[,i,],na.rm=T)
   +    ymax<-max(amp[,i,],na.rm=T)
   +    plot(amp[,i,],type="l",main=paste("Amplitude functions, a1 = ",a_vec[i],sep=""),
   +    axes=F,xlab="Frequency",ylab="Amplitude",col=colo[i],ylim=c(ymin,ymax))
   ```
3.2. FILTER REVISIONS

```r
+ mtext("Lag=0", side = 3, line = -1,at=len/4,col=colo[1])
+ for (j in 2:(L/2+2))
+ {
+   lines(amp[,i,j],col=colo[j])
+   mtext(paste("Lag="j-1,sep=""), side = 3, line = -j,at=len/4,col=colo[j])
+ }
+ axis(1,at=c(0,1:6*len/12+1),labels=c("0","pi/6","2pi/6","3pi/6",
+ "4pi/6","5pi/6","pi"))
+ axis(2)
+ box()
+ ymin<-min(shift[,i,],na.rm=T)
+ ymax<-max(shift[,i,],na.rm=T)
+ plot(shift[,i,1],type="l",main=paste("Time-Shifts, a1 = ",a_vec[i],sep=""),
+ axes=F,xlab="Frequency",ylab="Shift",col=colo[1],ylim=c(ymin,ymax))
+ mtext("Lag=0", side = 3, line = -1,at=len/4,col=colo[1])
+ for (j in 2:(L/2+2))
+ {
+   lines(shift[,i,j],col=colo[j])
+   mtext(paste("Lag="j-1,sep=""), side = 3, line = -j,at=len/4,col=colo[j])
+ }
+ axis(1,at=c(0,1:6*len/12+1),labels=c("0","pi/6","2pi/6","3pi/6",
+ "4pi/6","5pi/6","pi"))
+ axis(2)
+ box()
+ ymin<-min(b[,i,],na.rm=T)
+ ymax<-max(b[,i,],na.rm=T)
+ plot(b[,i,1],col=colo[1],ylim=c(ymin,ymax),main=paste("Filter coefficients"),
+ ylab="Output",xlab="lag",axes=F,type="l")
+ mtext("Lag=0", side = 3, line = -1,at=L/2,col=colo[1])
+ for (j in 2:(L/2+2))
+ {
+   lines(b[,i,j],col=colo[j],type="l")
+   mtext(paste("Lag="j-1,sep=""), side = 3, line = -j,at=L/2,col=colo[j])
+ }
+ axis(1,at=1:L,labels=-1+1:L)
+ axis(2)
+ box()
+ ymin<-min(yhat_Lag[,i,],na.rm=T)
+ ymax<-max(yhat_Lag[,i,],na.rm=T)
+ ts.plot(yhat_Lag[,i,1],col=colo[1],ylim=c(ymin,ymax),
```
+ main=paste("Output series"),ylab="Output")
+ mtext("Lag=0", side = 3, line = -1, at=len/2, col=colo[1])
+ for (j in 2:(L/2+2))
+ {
+   lines(yhat_Lag[,i,j], col=colo[j])
+   mtext(paste("Lag=",j-1,sep=""), side = 3, line = -j, at=len/2, col=colo[j])
+ }
+ }
+ }
> dev.off()
As expected, the time-shift (top-right) increases with increasing Lag (decreasing $h$). For $\text{Lag} = (L - 1)/2 = 6$ the filter is symmetric (see bottom left graph) and therefore the corresponding time-shift (violet line) is constant.

In contrast to the symmetric target filter, which is centered about $x_t$, the time-shift of the symmetric $\text{Lag} = 6$-filter does not vanish because the filter is causal: its coefficients are centered about $x_{t-6}$. We can see that the output series (bottom-right panel) are shifted accordingly: larger shifts are associated to stronger noise rejection (amplitude closer to zero in the stop-band) and to smoother series.

\footnote{The shift is constant (flat line) in the passband. The variable shift in the stopband is an artifact of the Arg-function: the physical shift must be constant since the filter weights (violet line bottom left panel) are symmetric.}

Figure 3.1: Amplitude (left) and time-shift (right) functions as a function of Lag (rainbow colors) for the white noise process ($a_1=0$)
CHAPTER 3. FILTER REVISIONS

3.2.3 Filter Vintages

In the GDP table 3.1 historical releases are up-dated (revised) when new information becomes available. We could proceed analogously for the above filter-designs:

- in time point \( t \) a new observation \( x_t \) becomes available. Therefore we can compute a nowcast \( \hat{y}_t^0 \) of \( y_t \) based on the real-time filter \( \text{Lag} = 0 \).

- But we can also improve our previous estimate \( \hat{y}_{t-1} \) of \( y_{t-1} \) since the new observation \( x_t \) is informative about \( y_{t-1} \). For this purpose we can rely on the \( \text{Lag} = 1 \) filter and obtain a better estimate \( \hat{y}_{t-1}^{-1} \).

- We do similarly for \( \hat{y}_{t-L}^{-L} \), \( \text{Lag} > 1 \): all historical estimates can be up-dated.

Assume now that we have a sample of length \( T \) and define a filter-vintage according to

\[
\hat{y}_{T-t}^{t}, t = 0, \ldots, T-1
\]

In each time point \( T-t \), the data point \( \hat{y}_{T-t}^{t} \) is the last observation of the output of the \( \text{Lag} = t \) filter. The series \( \hat{y}_{T-t}^{t}, t = 0, \ldots, T-1 \) is called a filter-vintage: we obtain a filter-vintage for each \( T \). Filter vintages can be arranged in a so-called filter-vintage triangle.

Exercises

1. Compute a filter-vintage triangle for each of the above AR(1)-processes. Specifically, use the filters for \( \text{Lag} = 0, \ldots, 6 \). Note that our maximal Lag-value is six: therefore, \( \hat{y}_{T-t}^{-t}, t > 6 \) is identified with \( \hat{y}_{T-t}^{-6} \), i.e. the filter-vintage becomes

\[
\hat{y}_{T-t}^{-\text{min}(6,t)}, t = 0, \ldots, T-1
\]

2. Compute the last 7 vintages (last 7 columns and rows) for the third process \((a_1 = -0.9)\), see table 3.2.

---

\[^3\text{The new observation } x_1 \text{ is informative because the target filter is bi-infinite.}\]
3.2. FILTER REVISIONS

> # We select the third DGP with a1=-0.9
> i<-3
> vintage_triangle<-vintage[,i,]
> dimnames(vintage_triangle)[[2]]<-paste("Publ. ",1:len,sep=" ")
> dimnames(vintage_triangle)[[1]]<-paste("Target ",1:len,sep=" 

Table 3.2: Last few vintages for the AR(1)-process with a1=-0.9: columns correspond to vintages and are indexed by corresponding publication dates; rows correspond to revisions of estimates for a fixed historical target date; diagonals correspond to releases

<table>
<thead>
<tr>
<th>Target 114</th>
<th>Publ. 114</th>
<th>Publ. 115</th>
<th>Publ. 116</th>
<th>Publ. 117</th>
<th>Publ. 118</th>
<th>Publ. 119</th>
<th>Publ. 120</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target 115</td>
<td>-0.208</td>
<td>-0.193</td>
<td>-0.119</td>
<td>-0.159</td>
<td>-0.110</td>
<td>-0.117</td>
<td>-0.122</td>
</tr>
<tr>
<td>Target 116</td>
<td>-0.127</td>
<td>-0.046</td>
<td>-0.094</td>
<td>-0.026</td>
<td>-0.028</td>
<td>-0.045</td>
<td></td>
</tr>
<tr>
<td>Target 117</td>
<td>0.018</td>
<td>-0.035</td>
<td>0.048</td>
<td>0.058</td>
<td>0.029</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Target 118</td>
<td>0.011</td>
<td>0.103</td>
<td>0.127</td>
<td>0.088</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Target 119</td>
<td></td>
<td>0.132</td>
<td>0.170</td>
<td>0.123</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Target 120</td>
<td></td>
<td></td>
<td>0.182</td>
<td>0.130</td>
<td>0.113</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The last column collects the last vintage $\hat{y}_{120-t}$ for $t = 0, 1, ..., 6$: $\hat{y}_{120}^0 = 0.113$ is the real-time estimate (first release) of the target $y_{120}$ based on data $x_1, ..., x_{120}$; $y_{119}^{-1} = 0.13$ is the second release of the target $y_{119}$ based on data $x_1, ..., x_{120}$ (the output of the $Lag = 1$ filter) and so on. In analogy to table 3.1, the column-date in table 3.2 refers to the publication date and the row-date refers to the target time i.e. the index $t$ of $y_t$. The initial release in the first column (publication date 114) is $-0.208$; the second release of the same target value $y_{114}$ in the second column is $-0.193$; the third release in the third column is $-0.119$, and so on. The differences between the various releases are due to filter-revisions: previous estimates of $y_{114}$ are up-dated when new information becomes available. The initial release $-0.208$ is up-dated until it reaches its final value $-0.122$ in the last column.

Remarks

- Let us repeat that the data $x_t$ is assumed to be fixed. Formal treatment of data revisions is postponed to chapter 10.
- The diagonals of the filter-vintage triangle are the releases: the initial release (first diagonal), the second release (second diagonal), and so on. A diagonal is generated by a fixed filter. Therefore diagonals are stationary time series (assuming stationarity of the data $x_t$).
- A filter-vintage (a column) is a non-stationary series: the DGP changes because different observations are generated by different filters (at least until the final values are obtained). As an example, the last observation of a vintage is generally ‘noisier’ than earlier observations (because it is based on a causal real-time filter $Lag = 0$).

\[^4\text{Recall that } \hat{y}_{T-t}^{T-1} = \hat{y}_{T-t}^0 \text{ if } t > 6 \text{ i.e. the estimate of } y_{114} \text{ won’t be revised anymore in later vintages (} T > 120).\]
• Therefore it would be a mistake to fit a model to a vintage series.

• It would be an even bigger mistake to infer a real-time estimate from such a model (because the fitted historical data is smoothed). See chapter 10 for a formal treatment of the problem.

3.2.4 Tentacle Plot

A simultaneous plot of all vintages is called a tentacle plot.

1. Plot the vintages obtained for the three AR(1)-processes \( 3.2 \) in the previous section, see fig. 3.2.

\[
\begin{align*}
&\text{> colo<-rainbow(len)} \\
&\text{> file = paste("z_vintages.pdf", sep = "")} \\
&\text{> pdf(file = paste(path.out,file,sep=""), paper = "special", width = 6, height = 6)} \\
&\text{> par(mfrow=c(3,1))} \\
&\text{> for (i in 1:3)} \\
&\text{\hspace{1em}+ {}} \\
&\text{\hspace{2em}+ ymin<-min(vintage[,i,],na.rm=T)} \\
&\text{\hspace{2em}+ ymax<-max(vintage[,i,],na.rm=T)} \\
&\text{\hspace{2em}+ ts.plot(vintage[,i,L],col=colo[1],ylim=c(ymin,ymax),} \\
&\text{\hspace{2em}\hspace{1em}+ main=paste("Tentacle plot: vintages and full revision sequence,} \\
&\text{\hspace{2em}\hspace{2em}+ a1 = ",a_vec[i],sep=""),ylab="Vintages")} \\
&\text{\hspace{2em}+ for (j in (L+1):len)} \\
&\text{\hspace{3em}+ {}} \\
&\text{\hspace{4em}+ lines(vintage[,i,j],col=colo[j])} \\
&\text{\hspace{3em}+ }} \\
&\text{\hspace{2em}+ lines(vintage[,i,len],col="red",lwd=2)} \\
&\text{\hspace{1em}+ } \\
&\text{\textgreater dev.off()} \\
\end{align*}
\]
The rate of convergence of early releases to their final values and the dynamic pattern of the revisions can be summarized and analyzed conveniently by this graphical tool. The behavior in the vicinity of the turning points deserves particular attention. These practically relevant features were not addressed by our previous diagnostic tools (MSEs, amplitude and time-shift functions). A comparison of the three graphs confirms earlier results: the difficulty of the estimation problem – the magnitude of the revision error – depends on the DGP: the signal-extraction task is less demanding when the data is positively autocorrelated.

2. Focus attention on the second process (white noise) and emphasize final and initial releases
CHAPTER 3. FILTER REVISIONS

graphically, see fig.3.3. Provide a short analysis of the salient features.

```r
> file = paste("z_vintages_2.pdf", sep = "")
> pdf(file = paste(path.out, file, sep=""), paper = "special", width = 6, height = 6)
> par(mfrow=c(2,1))
> i<-2
> ymin<-min(vintage[,i,],na.rm=T)
> ymax<-max(vintage[,i,],na.rm=T)
> ts.plot(vintage[,i,L],col=colo[1],ylim=c(ymin,ymax),
> + main="Vintages: full revision sequence and final release (black)",ylab="Vintages")
> for (j in (L+1):len)
+ { 
+ lines(vintage[,i,j],col=colo[j])
+ }
> lines(vintage[,i,len],col="black",lwd=2)
> i<-2
> ymin<-min(vintage[,i,],na.rm=T)
> ymax<-max(vintage[,i,],na.rm=T)
> ts.plot(vintage[,i,L],col=colo[1],ylim=c(ymin,ymax),
> + main="Vintages: full revision sequence and real-time initial release (black)",
> + ylab="Vintages")
> for (j in (L+1):len)
+ { 
+ lines(vintage[,i,j],col=colo[j])
+ }
> lines(yhat_Lag[,i,1],col="black",lty=1)
> dev.off()
```

RStudioGD

2
Convergence to the final values is achieved after $(L - 1)/2 = 6$ time steps\footnote{The full revision span can be substantially longer for macroeconomic aggregates: US-GDP is still substantially revised after after two years, see table 3.1.}. The top graph emphasizes the final values; the bottom graph emphasizes the first release. The latter (black line) is obtained by linking the end-points of the tentacles: this way the noisy real-time dynamics as well as the inherent delay are unmasked. Indeed, the amplitude function is leaking in the stopband and the time-shift is positive in the passband, see fig.\ref{fig:3.1}. Such a design is not well-suited for detecting turning-points in real time. Better designs are proposed in the following section\ref{sec:3.3} (bivariate leading indicator) or in chapter \ref{chap:5} (customization).
3.3 Bivariate Leading Indicator Design

We here apply the leading indicator design introduced in section 2.6. For simplicity we restrict the analysis to the second process (white noise).

1. Generate a leading indicator and define the (3-dim) data matrix.

```r
> set.seed(12)
> # Select the AR(1)-process with coefficient 0.9
> i_process<-2
> # Scaling of the idiosyncratic noise
> scale_idiosyncratic<-0.1
> eps<-rnorm(nrow(x))
> indicator<-x[,i_process]+scale_idiosyncratic*eps
> # Data: first column=target, second column=x, third column=shifted (leading) indicator
> data_matrix_120<-cbind(x[,i_process],x[,i_process],c(indicator[2:nrow(x)],indicator[nrow(x)]))
> dimnames(data_matrix_120)[[2]]<-'target','x','leading indicator'
> head(data_matrix_120)

    target         x leading indicator
   [1,] -0.1842525 -0.1842525 -1.2136136
   [2,] -1.3713305 -1.3713305 -0.6948422
   [3,] -0.5991677 -0.5991677  0.2025446
   [4,]  0.2945451  0.2945451  0.1900301
   [5,]  0.3897943  0.3897943 -1.2353058
   [6,] -1.2080762 -1.2080762 -0.3952109
```

Note that we implemented the lead in the time-domain, by shifting the data corresponding to the leading indicator (in order to avoid a NA at the end we just replicated the last observation). As an alternative, we could have rotated the DFT in the frequency-domain (but we didn’t...).

2. Compute the DFTs.

```r
> # Fully in sample
> insample<-nrow(data_matrix_120)
> # d=0 for stationary series: see default settings
> weight_func<-spec_comp(insample, data_matrix_120, d)$weight_func
```

3. Estimate optimal (MSE-) filter coefficients as a function of Lag=0,...,6

```r
> yhat_Lag_mdfa<-matrix(nrow=len,ncol=L/2+2)
> # Source the default (MSE-) parameter settings
> source(file=paste(path_MDFA.pgm,"control_default.r",sep=""))
> # Estimate filter coefficients
> for (Lag in 0:(L/2)))mdfa_obj$rever
```
3.3. BIVARIATE LEADING INDICATOR DESIGN

```r
+ {  
  + mdfa_obj<-mdfa_analytic_new(K,L,lambda,weight_func,Lag,Gamma,expweight,cutoff,i1,i2,weight_structure)
  + print(paste("Lag"," Criterion="
  + round(mdfa_obj$rever,4),sep=""))
  + # Filter coefficients
  + b_mat<-mdfa_obj$b
  + # Compute outputs
  + for (j in L:len)
  +   yhat_Lag_mdfa[j,Lag+1]<-sum(apply(b_mat*data_matrix_120[j:(j-L+1),2:3],1,sum))
+ }

[1] "Lag=0 Criterion=0.0424"
[1] "Lag=1 Criterion=0.0261"
[1] "Lag=2 Criterion=0.0172"
[1] "Lag=3 Criterion=0.0142"
[1] "Lag=4 Criterion=0.0145"
[1] "Lag=5 Criterion=0.0156"
[1] "Lag=6 Criterion=0.0155"
```

The criterion values do no more decay after Lag=3. Therefore, we expect the filter vintages to converge quite fast to the final values in the tentacle plot.

4. Define the vintage triangle.

```r
> vintage_mdfa<-matrix(nrow=len,ncol=len)
> # For each of the three AR(1)-processes We compute the vintage series
> for (j in L:len)#j<-len
+ {  
+   vintage_mdfa[(j-as.integer(L/2)):j,j]<-yhat_Lag_mdfa[j,(as.integer(L/2)+1):1]
+   vintage_mdfa[1:(j-as.integer(L/2)-1),j]<-
+   yhat_Lag_mdfa[(as.integer(L/2)+1):(j-1),as.integer(L/2)+1]
+   }
```

5. Generate a tentacle plot, see fig3.4

```r
> file = paste("z_vintages_mdfa.pdf", sep = "")
> pdf(file = paste(path.out,file,sep=""), paper = "special", width = 6, height = 6)
> par(mfrow=c(2,1))
> ymin<-min(vintage_mdfa,na.rm=T)
> ymax<-max(vintage_mdfa,na.rm=T)
> ts.plot(vintage_mdfa[,L],col=colo[1],ylim=c(ymin,ymax),
+ main="Vintages: full revision sequence and final release (black)",ylab="Vintages")
> for (j in (L+1):len)
+ {  
+   lines(vintage_mdfa[,j],col=colo[j])
```
+ } } 
> lines(vintage_mdfa[,len],col="black",lwd=2)
> ymin<-min(vintage_mdfa,na.rm=T)
> ymax<-max(vintage_mdfa,na.rm=T)
> ts.plot(vintage_mdfa[,L],col=colo[1],ylim=(ymin,ymax),
+ main="Vintages: full revision sequence and final release (black)",ylab="Vintages")
> for (j in (L+1):len)
+ {
+ lines(vintage_mdfa[,j],col=colo[j])
+ }
> lines(yhat_Lag_mdfa[,1],col="black",lwd=1)
> dev.off()
3.3. BIVARIATE LEADING INDICATOR DESIGN

Visual inspections and comparisons of figs. 3.4 and 3.3 (univariate DFA) are more eloquent than a convoluted comparative analysis.

Final Remark

- When applying a real-time filter (Lag = 0) of length L to the data, the output $\hat{y}_t^0, t = 1, ..., L - 1$ corresponding to the first $(L - 1)$-observations is missing. However, these missing initial values could be obtained quite easily by applying the Lag-operator judiciously. Specifically, estimates of $y_t, t = 1, ..., L - 1$ are obtainable in terms of $\hat{y}_t^{-L+t}, t = 1, ..., L - 1$. Verification of this claim is left as an exercise to the reader.
3.4 Summary

- We distinguished forecast \((h > 0)\), nowcast \((h = 0)\) and backcast \((h < 0)\) applications: the corresponding estimation problems can be handled by the hyperparameter \(Lag = -h\) in the function-calls.

- We analysed an important revision error component which is attributable to chained back-casting as expressed by the filter-vintages.

- We assumed the data to be fixed: no data-revisions. The latter are analysed in chapter 10.

- We proposed a useful new diagnostic tool, the so-called tentacle plot, and benchmarked the bivariate leading-indicator design against the univariate DFA.

- The pertinence – practical relevance – of filter-vintages and of backcasts is closely linked to the MSE-perspective because the multivariate criterion \(2.32\) minimizes the mean-square filter error: revisions are minimized. More sophisticated criteria proposed in chapter 5 will emphasize nowcast and forecast applications instead.
Chapter 4

Filter Constraints

We propose and analyze filter constraints which address particular user-priorities. Formally, the constraints may be invoked in order to ensure finiteness of the mean-square filter error in the case of integrated processes: they address filter characteristics in frequency zero\(^1\). The proposed constraints do not address cross-sectional links: the relevant multivariate extensions are proposed in chapter 13.

4.1 Level and Time-Shift Constraints

4.1.1 Level: i1==T

A first-order (level-) restriction of the coefficients \(b_0, ..., b_{L-1}\) of a filter with transfer function \(\hat{\Gamma}(\cdot)\) is specified as

\[
\hat{\Gamma}(0) = w
\]

or, equivalently

\[
b_0 + b_1 + ... + b_{L-1} = w \tag{4.1}
\]

where \(w\) is a constant. Typically, \(w := \Gamma(0)\) such that the fit of the target filter \(\Gamma(\cdot)\) by \(\hat{\Gamma}(\cdot)\) is perfect in frequency zero. If \(x_t\) is integrated I(1) then the perfect fit in the unit-root frequency ensures a finite mean-square filter error (assuming some mild regularity conditions, see McElroy and Wildi (2014) DFA and Wildi (2005)). Figuratively, the level of \(\hat{y}_t\) tracks the level of the target \(y_t\).

R Code

The Boolean \(i1\) in the function call of MDFA determines imposition (or not) of the level constraint. The constant \(w\) can be set in a vector called \textit{weight\_constraint}. In a multivariate design one must specify multiple constraints \(w^u, u = 1, ..., m + 1\). The restrictions are independent: each time

---

\(^1\)Arbitrary frequencies could be tackled but we did not find any relevant practical application so far.
series receives its own weight (cross-sectional dependencies among the constraints are proposed in chapter [13]).

4.1.2 Time-shift: $i2==T$

The time-shift $\hat{\phi}(\omega) = \Phi(\omega)/\omega$ of a filter is subject to a singularity in frequency zero. However, the limiting value, as $\omega \to 0$, could be obtained, see section 3.2.3 in DFA:

$$\hat{\phi}(0) = \lim_{\omega \to 0} \frac{\Phi(\omega)}{\omega} = \frac{d}{d\omega} \hat{\Phi}(\omega)\bigg|_{\omega=0} = \frac{d}{d\omega} \hat{\Gamma}(\omega)\bigg|_{\omega=0} = -i\hat{A}(0) = \frac{\sum_{j=0}^{L-1} jb_j}{\sum_{j=0}^{L-1} b_j}$$ (4.2)

Second and third equalities are obtained from

$$-i \sum_{j=0}^{L-1} jb_j = \frac{d}{d\omega} \hat{\Gamma}(\omega)\bigg|_{\omega=0} = \frac{d}{d\omega} \hat{\Phi}(\omega)\bigg|_{\omega=0} -i\hat{A}(0) \exp(-i\Phi(0)) = -i\hat{A}(0) \frac{d}{d\omega} \hat{\Phi}(\omega)\bigg|_{\omega=0}$$

The derivative of the amplitude vanishes in zero because the amplitude is a continuous even function i.e. $\hat{A}(-\omega) = \hat{A}(\omega)$. We are now in a position to formulate a time-shift restriction in frequency zero:

$$\frac{\sum_{j=0}^{L-1} jb_j}{\sum_{j=0}^{L-1} b_j} = s$$

This expression can be rewritten as

$$\sum_{j=0}^{L-1} (j - s)b_j = 0$$ (4.3)

In practice, a vanishing time-shift in frequency zero, $s = 0$

$$\sum_{j=1}^{L-1} jb_j = 0$$ (4.4)

is often a desirable ‘feature’ because turning-points (of the trend) should not be delayed too much. Also, a vanishing time-shift is necessary in order to ensure finiteness of the mean-square filter error if the data is I(2), although this argument cannot be invoked in practice\(^2\).

\(^2\)After years of intense search we lost hope to find an economic time series of integration order two.
4.1. LEVEL AND TIME-SHIFT CONSTRAINTS

R Code

The Boolean \( i2 \) in the function call of MDFA determines imposition (or not) of the time-shift constraint. The constant \( s \) can be set in a vector called \( \text{shift\_constraint} \). In a multivariate design one must specify multiple constraints \( s^u, u = 1, ..., m+1 \). The restrictions are independent: each time series receives its own weight (cross-sectional dependencies among the constraints are proposed in chapter 13).

4.1.3 Level and Time-Shift: \( i1==T, i2==T \)

Both constraints can be imposed simultaneously by solving the above expressions for \( b_{L-1} \) and \( b_{L-2} \):

\[
\begin{align*}
\hat{b}_{L-2} &= (L - 1 - s)w - (L - 1)b_0 - (L - 2)b_1 - ... - 2b_{L-3} \\
\hat{b}_{L-1} &= (2 + s - L)w + (L - 2)b_0 + (L - 3)b_1 + (L - 4)b_2 + ... + b_{L-3}
\end{align*}
\]

4.1.4 Exercises: Checking the Formulas

1. We verify the filter constraints by a simple number experiment.

   - Define arbitrary filter coefficients \( b_0, ..., b_3 \) and constants \( w, s \) and derive \( b_4 \) and \( b_5 \) according to 4.5 and 4.6:

     ```
     > # Filter length
     > L<-5
     > set.seed(1)
     > # The first three coefficients are random numbers
     > b<-rnorm(1:(L-2))
     > # Define the constants: the following are classical restrictions (amplitude is 1 and time shift is zero)
     > w<-1
     > s<-0
     > b<-c(b,(L-1-s)*w-(L-1:(L-2))%*%b,(2+s-L)*w+(L-2:(L-1))%*%b)
     ```

   - Verify pertinence of the filter constraints.

     ```
     > # Level constraint
     > sum(b)-w
     [1] 8.881784e-16
     > # time-shift constraint
     > sum(b*0:(L-1))/sum(b)-s
     [1] 5.051515e-15
     ```

   Both expressions are (virtually) zero, as desired.

2. To be filled: we verify the effects of the filter constraints on amplitude and time-shift functions

   - univariate: (problem: DFA is cheating (explain why) i.e. one must use MDFA univariate)
4.2 Filter Constraints and Pseudo-Periodogram, -DFT

To be filled: Wildi 2005 and 2008, I(1) and I(2) constraints. Refer to chapter 12.

4.3 General Parametrization

4.3.1 Caveats

The particular parametrization of the filter constraints in terms of $b_{L-1}$ and $b_{L-2}$ in 4.5 and 4.6 is to some extent arbitrary. Indeed, we could have selected $b_0$ and $b_1$, instead. Of course, our particular choice does not affect the estimation result, at least as long as the freely determined coefficients are indeed ‘freely determined’. Unfortunately, the regularization features to be introduced in chapter 9 conflict with this assumption. Therefore we need a more general approach.

4.3.2 Nowcast, Forecasting and Smoothing

We want to estimate $y_{T+h}$ for $h > 0$ (forecasting), $h = 0$ (nowcasting) or $h < 0$ (backcasting), given data $x_T, x_{T-1}, ..., x_{T-(L-1)}$, see section 3.2.1. If $h = 0$ (nowcasting) then imposing a vanishing time shift ($s = 0$) means that $\hat{y}_T$ and $y_T$ are ‘synchronized’ (at least in frequency zero). If $h \neq 0$, then the synchronization should apply between $\hat{y}_{T-h}$ and $y_T$ i.e. the vanishing time-shift should apply to the shifted output signal. Relying on 4.4 we obtain:

$\hat{\phi}(0) = \frac{\sum_{j=h}^{L-1+h} j b_j}{\sum_{j=h}^{L-1+h} b_j}$

and the time-shift constraint 4.3 becomes

$\sum_{j=h}^{L-1+h} (j - s) b_j = 0$ (4.7)

The level constraint is

$b_h + b_{h+1} + ... + b_{h+(L-1)} = \omega$ (4.8)

In order to avoid later conflicts (see chapter 9) we select $b_0$ and/or $b_1$ as natural candidate(s) for implementing the constraints.

4.3.3 Matrix Notation

The proposed filter constraints can be re-written in the form

$b = Rb_f + c$ (4.9)

where $b_f$ is the vector of freely determined coefficients. We now specify the right-hand side of this equation for each of the three relevant cases: i1=T, i2=F (simple level-constraint), i1=F, i2=T (simple time-shift constraint) and i1=i2=T (both constraints imposed).
4.3. GENERAL PARAMETRIZATION

The case \( i_1 < -T, \ i_2 < -F \)

We consider a general multivariate framework with \( m + 1 \) explaining variables \((x_t, w_{1t}, \ldots, w_{mt})\). In this case we obtain

\[
b^u_{\text{max}(0,h)} = w^u - \sum_{k=h,k\neq0} b^u_k
\]

where the index \( u = 0, \ldots, m \) runs across series \((u = 0\) corresponds to \(x_t\)). Note that \( b^u_{\text{max}(0,h)} = b^u_0 \) if \( h \leq 0 \) (nowcast/backcast). For \( h > 0 \) (\( h \)-step ahead forecast) we target \( y_{T+h} \), based on \( x_T, x_{T-1}, \ldots; \)
then \( b^u_{\text{max}(0,h)} = b^u_h \); the weight attributed to \( x_T \), by the forecast filter, is the lag-\(h\) coefficient.

The entries in \( \mathbf{R} \) become

\[
\mathbf{R} = \begin{pmatrix}
\mathbf{C} & 0 & \ldots & 0 \\
0 & \mathbf{C} & \ldots & 0 \\
\vdots & & \ddots & \vdots \\
0 & 0 & \ldots & \mathbf{C}
\end{pmatrix}
\]  \quad (4.10)

\[
\mathbf{C} = \begin{pmatrix}
1 & 0 & 0 & \ldots & 0 & 0 \\
0 & 1 & 0 & \ldots & 0 & 0 \\
\vdots & & \ddots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \ldots & 1 & 0 \\
0 & 0 & 0 & \ldots & 0 & 1
\end{pmatrix}
\]  \quad (4.11)

\[
\mathbf{c}' = (0, \ldots, 0, w^0_0, 0, \ldots, 0 \| 0, \ldots, 0, w^1_0, \ldots, 0 \| \ldots \| 0, \ldots, 0, w^m_0, 0, \ldots, 0)
\]

\[
\mathbf{b}\mathbf{r}' = (b^0_0, \ldots, b^1_0, \ldots, b^0_{h+L-1} \| b^1_0, \ldots, b^1_{h-1}, b^1_{h+L-1} \| \ldots \| b^m_0, \ldots, b^m_{h-1}, b^m_{h+L-1})
\]

The vector of -1’s in \( \mathbf{C} \) is in row-position \( \text{max}(0, -h) + 1 \) whereas the constant \( w^u \) \((u = 0, \ldots, m)\) in \( \mathbf{c} \) is in position \( u * (L-1) + \text{max}(0, -h) + 1 \). The vector \( \mathbf{b}\mathbf{r}' \) collects all freely determined parameters (thus \( b^u_0 \) is missing) whereas \( \mathbf{b} \) collects all coefficients: the former vector is used for optimization and the latter is required for filtering.

The case \( i_1 < -F, \ i_2 < -T \)

We consider a simple time-shift constraint without level requirement\[^3\] and we first assume \( s \neq 0 \) and \( h < 0 \) (backcast). From \( 4.7 \) we obtain

\[
-sb^u_0 = -(h-s)b^u_h - (h+1-s)b^u_{h+1} + \ldots - (-1-s)b^u_{h-1} - (1-s)b^u_1 - (2-s)b^u_2 + \ldots - (h+L-1-s)b^u_{h+L-1}
\]

or equivalently

\[
b^u_0 = \frac{(h-s)b^u_h + (h+1-s)b^u_{h+1} + \ldots + (-1-s)b^u_{h-1} + (1-s)b^u_1 + (2-s)b^u_2 + \ldots + (h+L-1-s)b^u_{h+L-1}}{s}
\]

\[^3\] This case cannot be replicated by model-based approaches because of the hierarchical structure of unit-roots: the I(2)-case presupposes the I(1)-requirement.
Such that

\[ C_{h<0} = \begin{pmatrix} 1 & 0 & 0 & \ldots & \ldots & \ldots & 0 & 0 \\ 0 & 1 & 0 & \ldots & \ldots & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \frac{h-s}{s} & \frac{h+1-s}{s} & \frac{h+2-s}{s} & \ldots & -\frac{1-s}{s} & \frac{1-s}{s} & \frac{2-s}{s} & \ldots & \frac{h+L-1-s}{s} \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & \ldots & \ldots & 0 & 1 \\ 0 & 0 & 0 & \ldots & \ldots & \ldots & 0 & 1 \end{pmatrix} \] (4.12)

\[ c'_{h<0} = 0 \] (4.13)

\[ b'_f = (b^0_{h}, \ldots, b^0_{h-1}, b^0_{h+L-1} \parallel b^1_{h}, \ldots, b^1_{h-1}, b^1_{h+L-1} \parallel \ldots \parallel b^m_{h}, \ldots, b^m_{h-1}, b^m_{h+L-1}) \]

The vector \( \left( \frac{h-s}{s}, \ldots, \frac{h+L-1-s}{s} \right) \) in \( C_{h<0} \) is in row-position \(-h+1\). The vector \( b_f \) collects the freely determined coefficients: all coefficients except \( b^0_{u} \).

The case \( h = 0 \) (nowcast) is handled by

\[ C_{h=0} = \begin{pmatrix} 1-s & 2-s & \ldots & L-1-s \\ \frac{s}{s} & \frac{s}{s} & \ldots & \frac{s}{s} \\ 1 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \end{pmatrix} \]

\[ c'_{h=0} = 0 \]

\[ b'_f = (b^0_{h+1}, \ldots, b^0_{h+L-1} \parallel b^1_{h+1}, \ldots, b^1_{h+L-1} \parallel \ldots \parallel b^m_{h+1}, \ldots, b^m_{h+L-1}) \]

and the case \( h > 0 \) (forecast) corresponds to

\[(h-s)b^u_{h} = -(h+1-s)b^u_{h+1} - (h+2-s)b^u_{h+2} - \ldots - (h+L-1-s)b^u_{h+L-1} \]

or, equivalently

\[ b^u_{h} = \frac{-(h+1-s)b^u_{h+1} - (h+2-s)b^u_{h+2} - \ldots - (h+L-1-s)b^u_{h+L-1}}{h-s} \]

We obtain

\[ C_{h>0} = \begin{pmatrix} \frac{-h+1-s}{h-s} & \frac{h+2-s}{h-s} & \ldots & \frac{h+L-1-s}{h-s} \\ \frac{h-s}{h-s} & \frac{h-s}{h-s} & \ldots & \frac{h-s}{h-s} \\ 1 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \end{pmatrix} \]

\[ c'_{h>0} = 0 \]

\[ b'_f = (b^0_{h+1}, \ldots, b^0_{h+L-1} \parallel b^1_{h+1}, \ldots, b^1_{h+L-1} \parallel \ldots \parallel b^m_{h+1}, \ldots, b^m_{h+L-1}) \]
Note that we implicitly assumed \( h - s \neq 0 \) in the above derivation. Otherwise we would have to isolate \( b_{h+1} \) instead of \( b_h \) (left as an exercise to the reader). Recall, also, that we assumed \( s \neq 0 \) in the case \( h \leq 0 \): if \( s = 0 \) then we isolate \( b_1 \), instead of \( b_0 \), in the above expressions\(^4\) (left as an exercise to the reader).

The case \( i1 < -T, i2 < -T \)

As in the previous section, we first tackle the backcast-problem: \( h < 0 \). Solving for \( b^u_0 \) and \( b^u_1 \) in \([4.7]\) and \([4.8]\) leads to

\[
\begin{align*}
    b^u_1 &= s^u w^u - h b^u_h - (h + 1) b^u_{h+1} - \ldots - (-1) b^u_{h-1} = 0 - 2 b^u_{h} - 3 b^u_{h-1} - \ldots - (h + L - 1) b^u_{h+L-1} \\
    b^u_0 &= w^u (1 - s^u) + (h - 1) b^u_h + h b^u_{h+1} + \ldots + (-2) b^u_{h-1} + b^u_h + 2 b^u_{h+1} + \ldots + (L - 2 + h) b^u_{L-1+h}
\end{align*}
\]

We obtain

\[
C_{h<0} = \begin{pmatrix}
1 & 0 & 0 & \ldots & 0 & 0 \\
0 & 1 & 0 & \ldots & 0 & 0 \\
\vdots & & & & & \\
-1 & h & h+1 & \ldots & -2 & 1 & 2 & \ldots & (L - 2 + h) \\
-1 & -(h+1) & -(h+2) & \ldots & -2 & -3 & \ldots & -(h + L - 1) \\
\vdots & & & & & \\
0 & 0 & 0 & \ldots & 1 & 0 \\
0 & 0 & 0 & \ldots & 0 & 1 
\end{pmatrix}
\]

\[
c'_{h<0} = \begin{pmatrix}
0, \ldots, 0, w^0 (1 - s^0), s^0 w^0, 0, \ldots, 0 \parallel 0, \ldots, 0, w^4 (1 - s^4), s^4 w^4, 0, \ldots, 0 \parallel \ldots \parallel 0, \ldots, 0, w^m (1 - s^m), s^m w^m, 0, \ldots, 0
\end{pmatrix}
\]

\[
\mathbf{b}_r' = (b^0_{-h}, \ldots, b^0_{-1}, b^0_0, b^0_{L-1-h} \parallel b^1_{-h}, \ldots, b^1_{-1}, b^1_0, b^1_{L-1-h} \parallel \ldots \parallel b^m_{-h}, \ldots, b^m_{-1}, b^m_0, b^m_{L-1-h})
\]

The non-trivial weighting-vectors in \( C_{h<0} \) are located in row-positions \( -h + 1 \) and \( -h + 2 \) whereas the constants \( w^u (1 - s^u) \) and \( s^u w^u \) in \( C_{h<0} \) are to be found in positions \( u \ast (L - 1) - h + 1 \) and \( u \ast (L - 1) - h + 2 \), respectively. Note that both \( b^0_0 \) and \( b^1_0 \) are now missing in the vector of freely determined coefficients \( \mathbf{b}_r \).

For \( h = 0 \) we obtain

\[
C_{h=0} = \begin{pmatrix}
1 & 2 & \ldots & (L - 2) \\
-2 & -3 & \ldots & -(L - 1) \\
1 & 0 & 0 \\
\vdots & & & \\
0 & \ldots & 0 & 1 
\end{pmatrix}
\]

\[
c'_{h=0} = \begin{pmatrix}
w^0 (1 - s^0), s^0 w^0, 0, \ldots, 0 \parallel w^4 (1 - s^4), s^4 w^4, 0, \ldots, 0 \parallel \ldots \parallel w^m (1 - s^m), s^m w^m, 0, \ldots, 0
\end{pmatrix}
\]

\[
\mathbf{b}_r' = (b^0_0, \ldots, b^0_{L-1} \parallel b^1_0, \ldots, b^1_{L-1} \parallel \ldots \parallel b^m_0, \ldots, b^m_{L-1})
\]

Finally, for \( h > 0 \) the constraints become

\[
\begin{align*}
    (h+1) b^u_{h+1} &= s^u w^u - (h + 2) b^u_{h+2} - (h + 3) b^u_{h+3} - \ldots - (h + L - 1) b^u_{h+L-1} \\
    h b^u_h &= w^u (1 - s^u) + (h + 1) b^u_{h+2} + (h + 2) b^u_{h+3} + \ldots + (h + L - 2) b^u_{h+L-1}
\end{align*}
\]

\(^4\text{If } s = 0 \text{ then, obviously, } 1 - s \neq 0 \text{ and therefore the quotients will be defined.}\)
or, equivalently
\[
\begin{align*}
\hat{b}_{h+1}^u & = \frac{s^u w^u - (h + 2)b_{h+2}^u - (h + 3)b_{h+3} - \ldots - (h + L - 1)b_{h+L-1}^u}{h + 1} \\
\hat{b}_h^u & = \frac{w^u (1 - s^u) + (h + 1)b_{h+2}^u + (h + 2)b_{h+3}^u + \ldots + (h + L - 2)b_{h+L-1}^u}{h}
\end{align*}
\]
so that
\[
\begin{pmatrix}
\frac{h + 1}{h + 2} & \frac{h + 2}{h + 3} & \frac{3}{h + 4} & \ldots & \frac{h + L - 2}{h + L - 1} \\
\frac{h + 2}{h + 1} & \frac{h + 3}{h + 1} & \frac{h + 4}{h + 1} & \ldots & \frac{h + L - 2}{h + L - 1} \\
1 & 0 & 0 & \ldots & 0 \\
0 & 1 & 0 & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \ldots & 1
\end{pmatrix}
\]
\[
C_{h>0} = \begin{pmatrix}
\hat{b}_0^0 (1 - s^0) & \hat{b}_0^1 & \ldots & \hat{b}_0^{m-1}
\end{pmatrix} = \begin{pmatrix}
\hat{b}_0^0 & \hat{b}_0^1 & \ldots & \hat{b}_0^{m-1}
\end{pmatrix}
\]
\[
b^f = \begin{pmatrix}
\hat{b}_0^0 & \hat{b}_0^1 & \ldots & \hat{b}_0^{m-1}
\end{pmatrix}
\]
Obviously, all expressions on the right-hand side of 4.9 depend on \(i_f\) as well as on \(h\) but for notational simplicity we refrain from attaching a cumbersome triple index to them.

4.4 Constrained Optimization

4.4.1 Generalized Criterion

Optimal constrained filter coefficients can be obtained by plugging 4.9 into 2.39 and by taking derivatives
\[
\frac{d}{d b_f} \text{Criterion} = \frac{d}{d b_f} (Y_{\text{rot}} - X_{\text{rot}} (R b_f + c))' (Y_{\text{rot}} - X_{\text{rot}} (R b_f + c))
\]
\[
= -2 (Y_{\text{rot}})' \mathbb{R} (X_{\text{rot}}) R - 2 \Re \left\{ (X_{\text{rot}})' (X_{\text{rot}} R) \right\} + 2 b_f' \Re \left\{ (X_{\text{rot}} R)' X_{\text{rot}} R \right\}
\]
The constrained solution is obtained by equating this expression to zero
\[
\hat{b}_f^{\text{Const}}(i_1, i_2) = \left\{ \Re \left[ (X_{\text{rot}} R)' X_{\text{rot}} R \right] \right\}^{-1} \left( \Re (X_{\text{rot}} R)' Y_{\text{rot}} + \Re \left\{ (X_{\text{rot}} R)' X_{\text{rot}} c \right\} \right)
\]
\[
= \left\{ \Re \left[ (X_{\text{rot}} R)' X_{\text{rot}} R \right] \right\}^{-1} \left( \Re (X_{\text{rot}} R)' Y_{\text{rot}} + \text{Level} \right)
\] (4.16)
where the vector
\[
\text{Level} := \Re \left\{ (X_{\text{rot}} R)' X_{\text{rot}} c \right\}
\]
can be interpreted as a generalized level term.\footnote{If \(w^u = 0\) for all \(u = 0, \ldots, m\) then \(\text{Level} = 0\).} A comparison with 2.43 illustrates that constrained and unconstrained solutions are similar up to the presence of the transformation \(R\) and the occurrence of the new level-term \text{Level}. In particular \(R = \mathbf{I}\) and \(c = 0\) replicates 2.43 as expected (no constraints imposed). The ‘full-coefficient’ vector \(b\), indispensable for filtering, is obtained by plugging the obtained constrained solution \(\hat{b}_f^{\text{Const}}(i_1, i_2)\) into 4.9
4.5 Exercises: Implementing Constraints in MDFA

We rely on the bivariate leading indicator design proposed in chapter 2 and compute amplitude and time-shift functions for all possible combinations of the Boolean \((i_1, i_2)\).

1. Rely on the data of section 2.6.1 (leading indicator) and use the default settings \((Lag = 0, i_1 = i_2 = F)\): unconstrained design) for estimating filter coefficients for filters of length \(L = 13\). Use the third series \((a_1 = -0.9)\) and compute and plot amplitude and time-shift functions, see fig. 4.1.

```
> # Filter length
> L<-13
> # Fully in sample
> insample<-nrow(data_matrix_120)
> # d=0 for stationary series: see default settings
> weight_func<-spec_comp(insample, data_matrix_120, d)$weight_func
> # Source the default (MSE-) parameter settings
> source(file=paste(path_MDFA.pgm,"control_default.r",sep=""))
> # Source a convenient plot function
> source(file=paste(path_MDFA.pgm,"mplot_func.r",sep=""))
> # Estimate filter coefficients
> mdfa_obj<-mdfa_analytic_new(k,L,lambda,weight_func,Lag,Gamma,expweight,cutoff,i1,i2,weight_func)
> file = paste("z_mdfa_ar1_amp_shift_Lag_0_iF_i2F.pdf", sep = "")
> pdf(file = paste(path.out,file,sep=""), paper = "special", width = 6, height = 6)
> par(mfrow = c(2, 1))
> # amplitude functions
> mplot <- abs(mdfa_obj$trffkt)
> # x-axis
> freq_axe <- rep(NA, K + 1)
> freq_axe[1] <- 0
> freq_axe[1 + (1 : 6) * K / 6] <- c(paste0(c("", 2 : 5), "pi/6"), "pi")
> ax <- freq_axe
> # colors, title and additional titles
> insamp <- 1.e+90
> colo <- NULL
```
> plot_title <- "Amplitude Functions"
> title_more <- colnames(x[, -1])
> mplot_func(mplot, ax, plot_title, title_more, insamp, colo)
> # time-shift
> mplot <- Arg(t(sign(apply(mdfa_obj$b, 2, sum)) * t(mdfa_obj$trffkt))) / 
+ ((0 : (nrow(mdfa_obj$trffkt) - 1)) * pi / (nrow(mdfa_obj$trffkt) - 1))
> # We use the exact formula for the time-shift in frequency zero
> mplot[1, ] <- apply(mdfa_obj$b * ((0 : (L - 1))), 2, sum) / 
+ apply(mdfa_obj$b, 2, sum)
> plot_title <- "Time-Shift"
> mplot_func(mplot, ax, plot_title, title_more, insamp, colo)
> # filter coefficients
> #mplot <- mdfa_obj$b
> #ax <- Lag + 0 : (L-1)
> #plot_title <- "Filter Coefficients"
> #mplot_func(mplot, ax, plot_title, title_more, insamp, colo)
> dev.off()
4.5. EXERCISES: IMPLEMENTING CONSTRAINTS IN MDFA

Figure 4.1: Amplitude (top) and time-shift (bottom) functions: unconstrained $i_1=i_2=F$

No constraints are imposed in frequency zero.

2. Same as above but impose a simple level constraint: $i_1 = T, i_2 = F$ and $w_0 = w_1 = 0.5$. Plot and compare the resulting amplitude functions, see fig. 4.2.

> # Source the default (MSE-) parameter settings
> source(file=paste(path_MDFA.pgm,"control_default.r",sep=""))
> # Impose level constraint
> i1<-T
> # Constraints: for series 1 and 2
> weight_constraint<-c(0.5,0.5)
> mdf_obj <- M DFA_analytic_new(K, L, lambda, weight_func, Lag, Gamma, expweight, cutoff, i1, i2, weight = weight, shift = 0, "ar1", "shift", "grand_mean", b0_H0, chris_expweight, weights_only = F, weight_structure, white_noise, synchronicity, lag_mat)
> file = paste("z_mdfa_ar1_amp_shift_Lag_0_iT_i2F.pdf", sep = "")
> pdf(file = paste(path.out, file, sep=""), paper = "special", width = 6, height = 6)
> par(mfrow = c(2, 1))
> # amplitude functions
> mplot <- abs(mdfa_obj$trffkt)
> # x-axis
> freq_axe <- rep(NA, K + 1)
> freq_axe[1] <- 0
> freq_axe[1 + (1 : 6) * K / 6] <- c(paste0(c("", 2 : 5), "pi/6"), "pi")
> ax <- freq_axe
> # colors, title and additional titles
> insamp <- 1.e+90
> colo <- NULL
> plot_title <- "Amplitude Functions"
> title_more <- colnames(x[, -1])
> mplot_func(mplot, ax, plot_title, title_more, insamp, colo)
> # time-shift
> mplot <- Arg(t(sign(apply(mdfa_obj$b, 2, sum)) * t(mdfa_obj$trffkt))) /
> + ((0 : (nrow(mdfa_obj$trffkt) - 1)) * pi / (nrow(mdfa_obj$trffkt) - 1))
> # We use the exact formula for the time-shift in frequency zero
> mplot[1, ] <- apply(mdfa_obj$b * ((0 : (L - 1))), 2, sum) /
> + apply(mdfa_obj$b, 2, sum)
> plot_title <- "Time-Shift"
> mplot_func(mplot, ax, plot_title, title_more, insamp, colo)
> # filter coefficients
> # mplot <- mdfa_obj$b
> # ax <- Lag + 0 : (L-1)
> # plot_title <- "Filter Coefficients"
> # mplot_func(mplot, ax, plot_title, title_more, insamp, colo)
> dev.off()
As expected, the amplitude functions satisfy the first order restriction: both start in 0.5 in frequency zero.

3. Same as above but impose a simple time-shift restriction, $i_1 = F, i_2 < -T$, and select a vanishing time-shift $s^0 = s^1 = 0$. Compute and compare the time-shift functions, see fig. 4.3

```r
> # Source the default (MSE-) parameter settings
> source(file=paste(path_MDFA.pgm,"control_default.r",sep=""))
> # Estimate filter coefficients: note that i1<-F by sourcing the default parameters
> i2<-T
> shift_constraint<-rep(0,2)
```
> mdfa_obj <- mdfa_analytic_new(K, L, lambda, weight_func, Lag, Gamma, expweight, cutoff, i1, i2, weight, white_noise, shift_constraint, grand_mean, b0_H0, chris_expweight, weights_only = F, weight_structure)
> file = paste("z_mdfa_ar1_amp_shift_Lag_0_iF_i2T.pdf", sep = "")
> pdf(file = paste(path.out, file, sep=""), paper = "special", width = 6, height = 6)
> par(mfrow = c(2, 1))
> # amplitude functions
> mplot <- abs(mdfa_obj$trffkt)
> # x-axis
> freq_axe <- rep(NA, K + 1)
> freq_axe[1] <- 0
> freq_axe[1 + (1 : 6) * K / 6] <- c(paste0(c("", 2 : 5), "pi/6"), "pi")
> ax <- freq_axe
> # colors, title and additional titles
> insamp <- 1.e+90
> colo <- NULL
> plot_title <- "Amplitude Functions"
> title_more <- colnames(x[, -1])
> mplot_func(mplot, ax, plot_title, title_more, insamp, colo)
> # time-shift
> mplot <- Arg(t(sign(apply(mdfa_obj$b, 2, sum)) * t(mdfa_obj$trffkt))) /
> + ((0 : (nrow(mdfa_obj$trffkt) - 1)) / (nrow(mdfa_obj$trffkt) - 1))
> # We use the exact formula for the time-shift in frequency zero
> mplot[1, ] <- apply(mdfa_obj$b * ((0 : (L - 1))), 2, sum) /
> + apply(mdfa_obj$b, 2, sum)
> plot_title <- "Time-Shift"
> mplot_func(mplot, ax, plot_title, title_more, insamp, colo)
> # filter coefficients
> #mplot <- mdfa_obj$b
> #ax <- Lag + 0 : (L-1)
> #plot_title <- "Filter Coefficients"
> #mplot_func(mplot, ax, plot_title, title_more, insamp, colo)
> dev.off()
The time-shift are now vanishing in frequency zero, as desired. Note that we used the exact expression [4.4] for computing the time-shifts in frequency zero.

4. Impose both constraints simultaneously, see fig. [4.4]

```r
> # Source the default (MSE-) parameter settings
> source(file=paste(path_MDFA.pgm,"control_default.r",sep=""))
> # Impose both constraints
> i1<-T
> i2<-T
> # Specify the constraints
```
> weight_constraint<-c(0.5,0.5)
> shift_constraint<-rep(0,2)
> mdfa_obj<-mdfa_analytic_new(K,L,lambda,weight_func,Lag,Gamma,expweight,cutoff,i1,i2,weight_structure,weight_constraint,shift_constraint,grand_mean,b0_H0,chris_expweight,weights_only=F,weight_structure,white_noise,synchronicity,lag_mat)
> file = paste("z_mdfa_ar1_amp_shift_Lag_0_iT_i2T.pdf", sep = "")
> pdf(file = paste(path.out,file,sep=""), paper = "special", width = 6, height = 6)
> par(mfrow = c(2, 1))
> # amplitude functions
> mplot <- abs(mdfa_obj$trffkt)
> # x-axis
> freq_axe <- rep(NA, K + 1)
> freq_axe[1] <- 0
> freq_axe[1 + (1 : 6) * K / 6] <- c(paste0(c(", 2 : 5), "pi/6"), "pi")
> ax <- freq_axe
> # colors, title and additional titles
> insamp <- 1.e+90
> colo <- NULL
> plot_title <- "Amplitude Functions"
> title_more <- colnames(x[, -1])
> mplot_func(mplot, ax, plot_title, title_more, insamp, colo)
> # time-shift
> mplot <- Arg(t(sign(apply(mdfa_obj$b, 2, sum)) * t(mdfa_obj$trffkt))) /
+ ((0 : (nrow(mdfa_obj$trffkt) - 1)) * pi / (nrow(mdfa_obj$trffkt) - 1))
> # We use the exact formula for the time-shift in frequency zero
> mplot[1, ] <- apply(mdfa_obj$b * ((0 : (L - 1))), 2, sum) /
+ apply(mdfa_obj$b, 2, sum)
> plot_title <- "Time-Shift"
> mplot_func(mplot, ax, plot_title, title_more, insamp, colo)
> # filter coefficients
> #mplot <- mdfa_obj$b
> #ax <- Lag + 0 : (L-1)
> #plot_title <- "Filter Coefficients"
> #mplot_func(mplot, ax, plot_title, title_more, insamp, colo)
> dev.off()
Both functions satisfy the restrictions, as desired.

4.6 Summary

- We proposed a set of practically relevant filter restrictions in frequency zero: the constraints affect the level and the time-shift of the constrained filter. Similar constraints could be applied to arbitrary frequencies (left as an exercise).

- Formally, a constrained filter would be able to track a non-stationary (integrated) signal of integration order one or two. However, in practice one is often interested in a vanishing
time-shift, whether the series is integrated or not.

- We proposed a formal matrix implementation and derived a generalized optimization criterion. The former unconstrained case \((i_1 = i_2 = F)\) is nested in the general solution.
Chapter 5

ATS-Trilemma

5.1 Introduction

- DFA-MSE tackles the relevant problem structure. But user-priorities are not addressed yet.

- Replicate Smoothness and Timeliness performances (use function in MDFA-code) of leading indicator design in section 2.6.2 by a univariate customized design. Costs: Accuracy worsens.

5.2 DFA: ATS-Trilemma and Customization

Refer to trilemma paper with Tucker on DFA (univariate). Eventually discuss numerically closed-form approximation.

5.3 From DFA to MDFA

Refer to elements-paper (extension of DFA to multivariate case)

5.4 Multivariate Customization

5.4.1 Emphasizing Smoothness

5.4.2 Emphasizing Timeliness

5.4.3 Emphasizing Smoothness AND Timeliness
Chapter 6

MDFA: Replicating and Enhancing (Customizing) Classical Filter-Designs and Model-Based Approaches

6.1 Introduction: a Generic For-, Now- and Backcast Approach

6.2 Replicate (Univariate) DFA

The following general DFA-code is pasted from section 4.3.5 in [DFA]

```r
> # This function computes analytic DFA-solutions
> # L is the length of the MA-filter,
> # weight_func is the periodogram,
> # lambda emphasizes phase artifacts in the customized criterion,
> # eta emphasizes noise-suppression/smoothness
> # Gamma is the transferfunction of the symmetric filter (target) and
> # Lag is the lag-parameter: Lag=0 implies real-time filtering, Lag=L/2
> # i1 and i2 allow for filter restrictions in frequency zero
> # The function returns the weights of the MA-Filter as well as its transferfunction
> 
> dfa_analytic<-function(L,lambda,weight_func,Lag,Gamma,eta,cutoff,i1,i2)
+ {
```
+ K<-length(weight_func)-1
+ # Define the amplitude weighting function weight_h (W(omega_k))
+ omega_Gamma<-as.integer(cutoff*K/pi)
+ if ((K-omega_Gamma+1)>0)
+ {
+     weight_h<-weight_func*(c(rep(1,omega_Gamma),(1:(K-omega_Gamma+1)))^eta))
+ } else
+ {
+     weight_h<-weight_func*rep(1,K+1)
+ }
+ # First order filter restriction: assigning a large weight to frequency zero
+ if (i1)
+     weight_h[1]<-max(1.e+10,weight_h[1])
+ + X<-exp(-1.i*Lag*pi*(0:(K))/(K))*rep(1,K+1)*sqrt(weight_h)
+ X_y<-exp(-1.i*Lag*pi*(0:(K))/(K))*rep(1,K+1)
+ if (i2)
+ {
+     # Second order restriction: time shift in frequency zero vanishes
+     for (l in 2:(L-1))
+     {
+         X<-cbind(X,(cos((1-Lag)*pi*(0:(K))/(K))-(1-Lag)/(L-1))*
+             cos((1-Lag)*pi*(0:(K))/(K))+
+             sqrt(1+Gamma*lambda)*1.i*(sin((1-Lag)*pi*(0:(K))/(K))-(1-Lag)/(L-1))*
+             sin((1-Lag)*pi*(0:(K))/(K)))*sqrt(weight_h))
+         X_y<-cbind(X_y,(cos((1-Lag)*pi*(0:(K))/(K))-(1-Lag)/(L-1))*
+             cos((1-Lag)*pi*(0:(K))/(K))+
+             1.i*(sin((1-Lag)*pi*(0:(K))/(K))-(1-Lag)/(L-1))*sin((1-Lag)*pi*(0:(K))/(K)))*sqrt(weight_h))
+     }
+     xtx<-t(Re(X))%^%Re(X)+t(Im(X))%^%Im(X)
+     # MA-Filterweights
+     b<-as.vector(solve(xtx)%*%(t(Re(X_y))%^%(Gamma*weight_h)))
+     # the last weight is a function of the previous ones through the second order restriction
+     b<-c(b,-sum(b*(0:(length(b)-1)))/(length(b)))
+ } else
+ {
+     for (l in 2:L)
+     {
+         X<-cbind(X,(cos((1-Lag)*pi*(0:(K))/(K)))+
+             sqrt(1+Gamma*lambda)*1.i*sin((1-Lag)*pi*(0:(K))/(K)))*sqrt(weight_h))
+         X_y<-cbind(X_y,(cos((1-Lag)*pi*(0:(K))/(K)))+
+             1.i*sin((1-Lag)*pi*(0:(K))/(K)))*sqrt(weight_h))
6.3 Replicate and Customize Classical Filter Designs (HP, CF)

Discuss implicit time-series models; derive corresponding spectra; replicate classical designs.

6.4 Replicate and Customize Classical ARIMA-Based Approaches

Derive (pseudo-) spectrum from model; replicate exact finite sample model-based performances up to arbitrary precision.

6.4.1 Forecasting

Replication only: the allpass forecast target does not allow for an application of the Trilemma because the Smoothness-term is missing (allpass target).

6.4.2 Signal Extraction

Replication and enhancing: refer to DFA-book, section 8.

6.5 Scope

Contrast scope and (statistical) relevance of classical model-based (one-step ahead mean-square) maximum likelihood approach vs. MDFA.
CHAPTER 6. MDFA: REPLICATING AND ENHANCING (CUSTOMIZING) CLASSICAL FILTER-DESIGNS AND MODEL-BASED APPROACHES
Chapter 7

Exotic Optimization Criteria

7.1 Introduction
Classic MSE and Customization: From AT-dilemma to ATS-trilemma. Alternative performances (not directly signal extraction): exotic criteria.

7.2 Hybrid Criterion
7.2.1 SE-Performance
7.2.2 Emphasizing Trading Performance
Maximizing Profits: Time vs. Frequency-Domain Perspectives
A Scaling Problem
A Projection Approach

7.3 Double-Decker
7.3.1 Instilling Performance
7.3.2 Extracting Performance
Chapter 8

Inference

Refer to DFA-article with Tucker. Refer to 2005 and 2008 books + elements paper. I won’t assign too much time to this topic because it is more or less irrelevant for applications; it is less relevant than for model-based approaches because we emphasize the filter (not the DGP). Generalization(s) of well-known model-based statistics. Maybe additional contributions by Tucker about the multivariate case. My own results include asymptotic distribution of filter coefficients (2005-book) and a generalized unit-root test (2008-book).
Chapter 9

Regularization

9.1 Introduction

Tackle overfitting: refer to elements paper.

9.1.1 What is ‘Overfitting’?

Exercises:

9.1.2 Avoiding Overfitting: Brute Force

Filter Constraints
Filter Length

9.2 Regularization Troika: a Triplet of Universal Requirements

9.2.1 Decay
9.2.2 Smoothness
9.2.3 Cross-Sectional Similarity

9.3 The Troikaner

9.3.1 Projection Matrix
9.3.2 Generalized Information Criterion

9.4 Reconciling Regularization and Filter Constraints

Refer to section [12] for a formal background to the constraints.
9.4.1 I(1)- (Level) Constraint
9.4.2 I(2)- (Time-Shift) Constraint
9.4.3 I(1)- and I(2)-Constraints

9.5 Implementation
9.5.1 The Case $i_1 < -T$, $i_2 < -F$
9.5.2 The Case $i_1 < -F$, $i_2 < -T$
9.5.3 The Case $i_1 < -T$, $i_2 < -T$

9.6 Optimization Criterion (Zero-Shrinkage)

9.7 General H0-Shrinkage
9.7.1 Zero-Shrinkage vs. Regularization: Potentially Conflicting Requirements
9.7.2 Inclusion of A Priori Knowledge
9.7.3 Replicating (and Enhancing) Clients’ Performances

9.8 Optimization Criterion (General H0-Shrinkage)

9.9 MDFA-Stages
- Numerically (very) fast
- Statistically efficient
- Immunized against Overfitting
- Highly Adaptive
Chapter 10

Vintage Data: Working with Data-Revisions

refer to 2011-paper

10.1 Introduction

10.2 Data Organization

10.2.1 Vintage and Release Triangles

10.2.2 Vintage Triangle in the Frequency-Domain

10.3 Reconcile Real-Time Signalextraction and Data-Revisions

10.3.1 Vintage-Filtering/Smoothing

10.3.2 Setting-Up MDFA-Designs: the (Pseudo-) Stationary Case

10.3.3 Extension to Non-Stationary Time Series

Refer and link to section 13
Chapter 11

Mixed-Frequency Data:  
Combining and Working with  
Data Sampled at Different Time Scales

11.1 Introduction

- Refer to section 2.6.2 for a quantitative analysis of up-dating effects.

- Refer to recent work with Chris
11.2 Target is Low-Frequency Data

11.2.1 Folding the Frequency Interval

11.2.2 Generalized Optimization Criterion

11.3 Explaining Series is/are Low-Frequency Data

11.3.1 Folding and Expanding the Frequency Interval

11.3.2 Generalized Optimization Criterion

11.4 General Case: Arbitrary Mix

11.5 Unequally Distributed Release Dates

11.6 Missing Data
Chapter 12

Non-Stationarity: Integrated Processes

12.1 Introduction

Briefly review classical model-based concepts

12.2 Integration

12.2.1 Unit-Roots vs. Filter Constraints

Refer to DFA-paper with Tucker and to 2005 and 2008-books (generalized unit-root test)

12.2.2 I(1)- (Level) Constraint

12.2.3 I(2)- (Time-Shift) Constraint

12.3 Optimization Criterion
Chapter 13

Non-Stationarity: Cointegrated Processes

Contribution by Tucker? Refer to elements paper with regards to empirical criteria.
13.1 Cointegration Relations vs. Filter Constraints
13.2 The Rank-One Case
13.3 Arbitrary Rank
13.4 Universal Time-Domain Decomposition of the Filter Error
13.5 Frequency-Domain Decomposition of the Filter Error
13.6 I(1)-MSE Criterion
13.7 Unveiling the Unit-Root Singularity
13.8 Matrix Notation (Frequency Domain)
13.9 Customization
13.10 Regularization
13.11 An Application of Cointegration to Data Revisions

Link to section [10.3.3]
Chapter 14

Non-Stationarity: Adaptive Filtering

14.1 Introduction

Contrast with difference-stationary processes

14.2 Filter Up-Dating

14.2.1 Univariate Up-Dating
14.2.2 Multivariate Up-Dating

14.3 MDFA-Stages
CHAPTER 14. NON-STATIONARITY: ADAPTIVE FILTERING
Chapter 15

Summary and Links

15.1 Survey of MDFA Optimization Criteria

15.2 Consistency and Efficiency: a Tale of Two Philosophies

15.2.1 Knowing the Truth: Omniscience

15.2.2 Believing in Truth: Faith and Fatalism

15.2.3 From Truth to Effectiveness: Emphasizing Performances
Appendix

Appendix MDFA: R-Code