7 Computational Aspects and Numerical Implementation

This chapter concerns the computational aspects of system identification using discrete-time parametric models. In chapter 4, the ARMAV model of a structural system was derived. The estimation of the ARMAV model and the equivalent innovation state space system was the subject of chapter 5. Some of the applications of system identification using these models were given in chapter 6. However, the actual implementation as well as the practical problems encountered in this context have not yet been discussed. These subjects are put in focus in this chapter. Section 7.1 concerns some of the aspects that must be taken into account in order to improve the accuracy in the numerical manipulation of multivariate discrete-time systems. Section 7.2 describes what a good system identification software must include. It also describes what to require from the programming environment and why the actual implementation in this thesis has been performed in the MATLAB programming language. Finally, in section 7.3 the actual numerical implementation of the theory presented in this thesis is considered.

7.1 Improving Computational Accuracy of Numerical Algorithms

In chapter 5, numerical algorithms for identification of multivariate linear systems were considered. However, it is necessary to account for the finite length numerical arithmetic and possible errors in the data. This section will describe some of the typical problems encountered in practical system identification, and give guidelines as to how they can accounted for.

7.1.1 Prefiltering, Resampling and Detrending of the Data

The discrete-time stochastic models presented in this thesis assume that the measurements are realizations of a zero-mean stochastic process. This means that response measurements having a DC component and components of slow harmonic excitation will cause trouble in the interpretation of identification results. It may therefore be beneficial to prefilter the data prior to the actual system identification. A high-pass filter can effectively remove any constant DC component, and harmonic components such as 50 Hz noise can be removed by either lowpass or bandpass filtering. The same filters should be applied to all measurement channels in order to avoid introduction of different phase distortion of the channels. If the filters only are applied to some of the channels a two-ways filtering approach should used. It is also important to eliminate the possibility of aliasing either by applying an appropriate analog anti-aliasing filter or by rapid sampling. For most identification procedures though, it is important that the sampling interval is optimal with respect to the content of the dynamical amplified response.

If the sampled response has no significant energy content above a frequency that is an integer factor of the Nyquist frequency the sampled response should be decimated. In other words, it should be resampled with a higher sampling interval. If the sampling interval is too small, the estimates of low-frequency modal parameters will be poor. How to select the optimal sampling interval for identification of ARMA models has been investigated in Kirkegaard [53]. Finally, it might also occur that the DC component is time-varying for some reasons. The measured data will then have a drifting trend. This trend might e.g. be linear or harmonic but in any case, it should be removed prior to the identification of time-invariant linear models. The removal of trends from the measured data is called detrending.

7.1.2 Scaling of the Measured Data

The dynamic response of a structure may be measured in several ways. For system identification purposes structural response is often measured as accelerations. However, displacements, velocities, material strain and stress, and pore pressure may also be used. In any case, to get as reliable information as possible from the measurements, the measurement range of the sensors is typically scaled according to the level of the response at the sensor location. For most sensors, a calibration of the sensor signal must be performed to get correct physical units. This calibration is often carried out as a part of the data acquisition procedure. Because of the different physical units that might occur in a set of measurements, and because the response level may deviate considerably from one sensor position to another, it is reasonable to expect that the standard deviation of the measured records may be significantly different from one another. Such measurements will most certainly result in a large range of the system matrices of the identified model. A large range implies that there is a large difference between the values of the elements of these matrices. This would not be a problem if infinite precision arithmetic was used. However, this is not the case in practice and the result is ill-conditioned system matrices. Such system matrices will e.g. result in an ill-conditioned modal decomposition and therefore modal parameter estimates.

These reasons motivate the application of a scaling procedure to the measured records prior to the actual system identification. Let $\boldsymbol{y}(t_k)$ be the measured zero-mean response. The scaled measurements $\boldsymbol{y}_s(t_k)$ are then given by

$$\boldsymbol{S}\,\boldsymbol{y}_s(t_k) = \boldsymbol{y}(t_k) \qquad \textbf{(1)}$$

where $\boldsymbol{S}$ is a scaling matrix. If a system identification of a stochastic state space realization is based on $\boldsymbol{y}_s(t_k)$ a correct scaling of the response and of the mode shapes is obtained by using the following observation equation

$$\boldsymbol{y}(t_k) = \boldsymbol{S}^{-1}\boldsymbol{C}\hat{\boldsymbol{x}}(t_k|t_{k-1}) + \boldsymbol{S}^{-1}\boldsymbol{e}(t_k) \qquad \textbf{(2)}$$

If a system identification of an ARMAV model is based on $\boldsymbol{y}_s(t_k)$ a correct scaling is obtained by the extension

$$\boldsymbol{y}(t_k) \;=\; \boldsymbol{S}^{-1}\boldsymbol{A}^{-1}(q)\boldsymbol{C}(q)\boldsymbol{e}(t_k) \qquad \textbf{(3)}$$

A natural choice of scaling matrix is a diagonal matrix where the diagonal elements are the sampled standard deviations of the measurements. In this way all channels will have unit standard deviation and be non-dimensional.

7.1.3 Reducing the Range of the System Matrices

In the previous chapters, it has been shown how the ARMAV model can be realized in state space by the observability canonical state space form, see e.g. (2.62) and (2.63). This particular realization has been emphasized since it is easy to obtain directly from the ARMAV model. However, for numerical computations this realization should be avoided since it typically has an ill-conditioned modal decomposition. The reason is that the transition matrix is in companion form which implies that its range is large. Due to the finite precision arithmetic the eigenvalues and eigenvectors of the companion matrix will be highly sensitive to perturbations and therefore sensitive to noise.

The condition of the companion matrix should therefore be improved before e.g. a modal decomposition is applied. The condition can be improved by balancing of the companion matrix. When a matrix is ill-conditioned there is a large difference between the singular values of the matrix. Such a matrix is balanced by use of a proper similarity transformation. The effect of this transformation is to reduce the difference between the singular values of the matrix, and in principle make all singular values equal. This similarity transformation has to be applied to all system matrices according to the transformation rules given in definition 2.2. Since the balancing affects all system matrices of the realization, the obtained system is referred to as a balanced state space realization. Various techniques for balancing of state space realizations exist, see e.g. Aoki [11], Hoen [38] and McKelvey [76].

7.1.4 Robustifying the Prediction Filter

This section addresses two practical problems which can be encountered with the use of the prediction filter $\boldsymbol{L}(q,\boldsymbol{\theta})$ of the PEM method reviewed in chapter 5. Suppose that prediction errors with a very small probability of occurrence can assume certain large values. Measured data that create such prediction errors are called outliers and these can seriously affect the variance of the prediction errors. It is obvious that such an influence is not acceptable. The solution is a robustification of the criterion function defined in section 5.2

$$V_N(\boldsymbol{\theta}) \;=\; det\left(\frac{1}{N}\sum_{k=1}^{N} g\big(\boldsymbol{\varepsilon}(t_k,\boldsymbol{\theta})\big)g\big(\boldsymbol{\varepsilon}(t_k,\boldsymbol{\theta})\big)^T\right) \qquad \textbf{(4)}$$

The function $g(\boldsymbol{\varepsilon}(t_k,\theta))$ should behave as $\boldsymbol{\varepsilon}(t_k,\boldsymbol{\theta})$ for small values of $\boldsymbol{\varepsilon}(t_k,\boldsymbol{\theta})$, and then saturate as $\boldsymbol{\varepsilon}(t_k,\theta)$ increases. It can be shown, see Ljung [71], that choosing $g(\boldsymbol{\varepsilon}(t_k,\theta))$ as

$$\begin{aligned} g\left(\boldsymbol{\varepsilon}(t_k,\theta)\right) &= \boldsymbol{\varepsilon}(t_k,\boldsymbol{\theta}) \ , \ \textit{for} \quad \left|\boldsymbol{\varepsilon}(t_k,\theta)\right| < \rho\cdot\boldsymbol{\sigma}_\varepsilon \\ g\left(\boldsymbol{\varepsilon}(t_k,\theta)\right) &= \rho\cdot\boldsymbol{\sigma}_\varepsilon \ , \quad \textit{for} \quad \boldsymbol{\varepsilon}(t_k,\theta) > \rho\cdot\boldsymbol{\sigma}_\varepsilon \\ g\left(\boldsymbol{\varepsilon}(t_k,\theta)\right) &= -\rho\cdot\boldsymbol{\sigma}_\varepsilon \ , \ \textit{for} \quad \boldsymbol{\varepsilon}(t_k,\boldsymbol{\theta}) < -\rho\cdot\boldsymbol{\sigma}_\varepsilon \end{aligned} \qquad \textbf{(5)}$$

reduces the variance of the prediction errors efficiently. ρ is a scalar in the range $1 \le \rho \le 1.8$, and $\boldsymbol{\sigma}_\varepsilon$ is a vector of estimated standard deviations of the prediction errors. The vector $\boldsymbol{\sigma}_\varepsilon$ is calculated robustly, i.e. in a way that is highly insensitive to the presence of outliers as

$$\boldsymbol{\sigma}_\varepsilon = \frac{\textit{median}\left(\boldsymbol{\varepsilon}(t_k,\theta) - \textit{median}\left(\boldsymbol{\varepsilon}(t_k,\theta)\right)\right)}{0.7} \qquad \textbf{(6)}$$

The loss of optimality due to this robustification is insignificant in the case of a Gaussian distributed probability density function of the true system, see Ljung [71].

So far, it has been assumed that the prediction errors obtained as the output of a prediction filter subjected to a stationary data sequence $\{\boldsymbol{y}(t_k)\}$ is also a stationary sequence. However, using standard formulas for predicting the output from an ARMAV model, the prediction errors are in general not white before a transient phase has passed. During this transient phase the effects of the choice of initial conditions will dissipate. This is true even when the correct model and parameters are used. Thus, in cases where the amount of data is limited this transient behaviour might influence the performance of the criterion function significantly. When applying the prediction filter $\boldsymbol{L}(q,\boldsymbol{\theta})$ of the ARMAV(na,nc), $na \ge nc$, for prediction of the response at time step k, measured data are in principle needed for time step k-1 back to infinite past. However, since data is not available back to infinite past an initialization procedure has to be chosen.

Defining the time of the first measurement as 1, the predictor of the ARMAV(na,nc) model obtained in (5.12) can be calculated for $t_k \ge t_s$, $t_s = na+1$, as

$$\begin{aligned} \hat{\boldsymbol{y}}(t_k|t_{k-1}) = &-\boldsymbol{A}_1\boldsymbol{y}(t_{k-1}) - \boldsymbol{A}_2\boldsymbol{y}(t_{k-2}) - \ldots - \boldsymbol{A}_{na}\boldsymbol{y}(t_{k-na}) + \\ &\boldsymbol{C}_1\boldsymbol{\varepsilon}(t_{k-1}) + \boldsymbol{C}_2\boldsymbol{\varepsilon}(t_{k-2}) + \ldots + \boldsymbol{C}_{nc}\boldsymbol{\varepsilon}(t_{k-nc}) \end{aligned} \qquad \textbf{(7)}$$

when it is initialized by the following sequences

$$\begin{aligned} &\left\{\boldsymbol{y}(t_{s-1}),\ \boldsymbol{y}(t_{s-2}),\ \ldots\ ,\ \boldsymbol{y}(t_{s-na})\right\} \\ &\left\{\boldsymbol{\varepsilon}(t_{s-1}),\ \boldsymbol{\varepsilon}(t_{s-2}),\ \ldots\ ,\ \boldsymbol{\varepsilon}(t_{s-nc})\right\} \end{aligned} \qquad \textbf{(8)}$$

The usual procedure, recommended in e.g. Ljung [71], for initialization of the unknown prediction error sequence in (7.8) is to use the unconditional mean of the innovations, i.e. zero

$$\boldsymbol{\varepsilon}(t_{s-1}) = \boldsymbol{\varepsilon}(t_{s-2}) = \ldots = \boldsymbol{\varepsilon}(t_{s-nc}) = \mathbf{0} \qquad \textbf{(9)}$$

This initialization procedure is called the direct start, see Knudsen [64]. Since the transient behaviour is related to the moving average, all types of model structure having a moving average might experience bad transient behaviour due to improper initialization. The transient behaviour of 20 prediction error sequences of a 2-channel ARMAV(2,2) model is shown in the left-hand side plots in figure 7.1. On the right-hand side, the variations standard deviation of the 20 prediction error sequences of the two channels are shown.

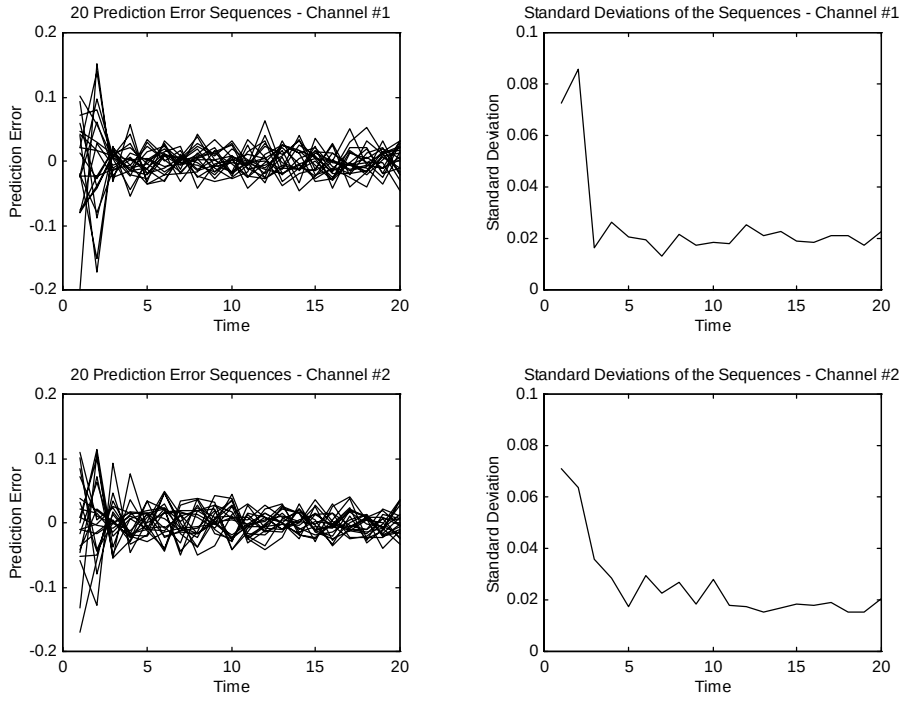

Figure 7.1: Transient behaviour of the prediction errors of an ARMAV(2,2). The left-hand side plots show 20 different prediction error sequences, and the right-hand side plots the variation of the standard deviation of these.

From figure 7.1, it seems obvious that if the amount of data is limited this kind of transient behaviour will seriously affect the criterion function and therefore destroy the statistical properties of the estimator. In Knudsen [65], it is analysed how the variance of the prediction errors of a univariate ARMAX model depends on the amount of data, and how close the eigenvalues of the moving average polynomial $\boldsymbol{C}(q)$ are to the unit circle. It is concluded that the transient behaviour depends both on the order of $\boldsymbol{C}(q)$ and how close the eigenvalues of $\boldsymbol{C}(q)$ are to the unit circle. These conclusions can directly be extended to multivariate ARMAV models. Based on an approach for reduction of the transient behaviour in the case of ARMAX

models, see Knudsen [64], a similar approach has been developed for univariate ARMA models in Andersen et al. [4]. In both these cases the approach is referred to as the backward forecasting initialization method. The method used in Andersen et al. [4] can directly be generalized to multivariate ARMAV models. The idea behind the method is simple; since the predictor in (7.7) is obtained as the conditional mean given all previous measurements, the zeroes used in the direct start initialization in (7.9) can be interpreted as the unconditional expectation for the corresponding noise samples. To remove the bad transient behaviour the predictor should instead be initialized by a conditional expectation. This conditional expectation is in fact a multi-step prediction of past unknown prediction errors given all available future data, see Andersen et al. [4]. It must be emphasized that the problems described in this section are no argument against the validity of the asymptotic properties of the PEM methods. It is only in case of a small number of data point that the transient behaviour has any influence. Clearly, as the number of data points tends to infinity the effects of the transient behaviour vanishes. The idea of initialization of the prediction filter is not restricted to the PEM methods. When using e.g. maximum likelihood estimation similar problems with bad transient behaviour occur. The elimination of these problems has been investigated in e.g. Box et al. [16] and Saric et al. [98].

7.2 Using MATLAB as Foundation for Identification Software

The system identification process is truly an iterative process, which can be characterized as a search for the optimal model among several candidate models. This implies that several models are identified and rejected before the optimal one is determined. Such a process can of course not be fully automated, which is why system identification in practice is best performed in an interactive environment, where human decisions are mixed with numerical calculations. An efficient system identification session requires good software as a basis. Such software must fulfil the users needs as best possible. In case of system identification of ambient excited civil engineering structures this means that the software as a basis must include:

- ☞ *Data handling, plotting, filtering, detrending, decimation, resampling.*
- ☞ *Non-parametric identification methods, such as FFT-based spectral density estimation.*
- ☞ *Parametric identification methods for various multivariate stochastic model structures.*
- ☞ *Presentation of model properties by e.g. plotting of spectral densities.*
- ☞ *Model validation procedures.*

The reason for performing system identification can be caused by different needs. One of the main reasons for performing it on ambient excited civil engineering structures is for modal analysis, as explained in chapter 6. Therefore, in addition the software must include some specialized features, such as:

- ☞ *Calculation of the modal parameters and estimation of their associated uncertainties.*
- ☞ *Procedures for identification of structural modes using e.g. various stability diagrams.*
- ☞ *Animation of the mode shapes of the structure.*

System identification software is as such more a toolbox consisting of different routines than a single program. Common to these routines is the requirement for some basic graphical and numerical features e.g. for manipulation of matrices, filtering etc. If the routines are implemented in a low-level programming language, these features must also be implemented. However, in this Ph.D. project the software implementation has been performed in the MATLAB™ language (MathWorks Inc.). This language is an interactive type of software. It is a high-performance numerical, computational and visualization software.

MATLAB integrates numerical analysis, matrix computation, signal processing, and graphics in an easy-to-use environment, where problems and solutions are expressed just as they are written mathematically. These features are exactly what is needed as foundation for efficient implementation of system identification software, and it is all accomplished without doing a single line of low-level programming. The name MATLAB stands for MATrix LABoratory, and it was originally written to provide easy access to the matrix software developed by the LINPACK and EISPACK projects, see Dongarra et al. [22] and Smith et al. [103]. Together these projects represent the state of the art in software for matrix computation. MATLAB is an interactive system whose basic data element is a matrix that does not require dimensioning, and the system uses complex arithmetic whenever it is necessary. MATLAB has evolved over a period of years with input from the users. In university environments, it has become the standard instructional tool for introductory courses in applied linear algebra, as well as advanced courses in other areas. In industrial settings, MATLAB is used for research and to solve practical engineering and mathematical problems. Typical uses include; general purpose numerical computation, algorithm prototyping, and special purpose problem solved with matrix formulations. MATLAB also features a family of application-specific solutions called toolboxes, which are very important to most users of MATLAB. A toolbox is a comprehensive collection of MATLAB functions (also known as m-files) that extend the MATLAB environment in order to solve a particular class of problems. Areas in which toolboxes are available include; signal processing, control systems design, dynamic systems simulation, system identification, and neural network.

For system identification several toolboxes exist. The *System Identification Toolbox*, see Ljung [72] features a flexible Graphical User Interface (GUI) as well as identification functions that implement both parametric and non-parametric identification techniques. The toolbox contains carefully implemented algorithms to ensure efficiency and reliable numerical results. All data sets and models created with the GUI are represented by icons. An entire session of data and models, along with relevant diaries, can be saved for reloading at a later time. The *Frequency Domain System Identification Toolbox*, see Kollár [66] and Kollár et al. [67], contains tools

for accurate modelling of linear systems with or without time-delay. The models are represented by transfer functions in the *z*-domain or *s*-domain. The procedures includes input design, data preprocessing, parameter estimation, graphical presentation of results, and model validation.

The two mentioned toolboxes are the official ones distributed by MathWorks Inc. Unofficially other system identification toolboxes exist and some of these will be mentioned in the following.

The *State Space IDentification (SSID) Toolbox*, see McKelvey [76] provides routines for system identification of multivariate state space models from input/output measurements. The main feature of this toolbox is that the models do not use the standard identifiable state space parameterization, which could e.g. be the observability canonical form, but uses balanced realization with a full parameterization of all state space matrices. This toolbox cannot work on its own. It depends on the presence of the System Identification Toolbox and the Control Toolbox, which is another official toolbox distributed by MathWorks Inc. Another unofficial system identification toolbox is the *SENSTOOLS toolbox*, see Knudsen [63]. This toolbox implements algorithms for direct identification of physical/continuous-time parameters in linear and nonlinear systems from input/output measurements. The routines of this toolbox covers features such as input design, parameter estimation, model validation, and sensitivity analysis. The *NNSYSID toolbox*, see Nørgaard [83], is a neural network based nonlinear system identification toolbox, which contains a large number of functions for training and evaluation of multilayer perceptron type neural networks. The main focus is on the use of neural networks as a generic model structure for the identification of nonlinear systems.

However, in order to use the above-mentioned toolboxes for identification of structural systems routines for estimation of modal parameters are missing. Recently, some MATLAB toolboxes made for solving structural engineering identification problems have been presented.

The purpose of the *Structural Dynamics Toolbox*, see Balmés [12], is to provide a low cost, modular, and versatile access to methods in experimental and analytical structural dynamic modelling. The toolbox includes functions for experimental modal analysis, Finite Element design and update. A GUI provides a layer of predefined operations for Frequency Response Function visualization, analysis, identification and 3-D deformation animation. The *X-modal* modal analysis software package, Philips et al. [91], was developed as a collaboration between the University of Cincinnatti and the industry. The software is based on the Unified Matrix Polynomial Approach (UMPA), see Fladung et al. [23]. The unified approach to modal parameter estimation in *X-modal* not only involves the formulation of the algorithms, but also the presentation of their controlling parameters. This software package was programmed using the MATLAB language, as well as the C language, and the X-windows system.

7.3 The Structural Time Domain Identification (STDI) Toolbox

The review given in the previous section of the various system identification toolboxes gives a clear motivation for using MATLAB. However, most of the above toolboxes are not especially suited for experimental modal and spectrum analysis of ambient excited civil engineering structures using stochastic time domain models. This is due to several reasons, where the most important are:

- ☞ *The stochastic part of a parametric model is typically considered as a nonphysical noise model.*
- ☞ *Most identification routines require measured input.*
- ☞ *Traditional experimental modal analysis tends to use non-parametric identification methods and to operate in frequency domain.*

As a part of this Ph.D. project, it has therefore been the intention to develop a MATLAB based toolbox for identification of especially ambient excited civil engineering structures using multivariate stochastic time domain models. For this reason the toolbox is called the *Structural Time Domain Identification (STDI) toolbox*. This toolbox has been developed in accordance with the theory and notation of this thesis. It has been the intention to make it completely independent of other official as well as unofficial toolboxes.

7.3.1 Organizing the Identification Results

One of the most important problems that must be solved by a good system identification software is, how to organize all the results that are generated through a system identification session. In this context, it is important to distinguish between primary and secondary results. Primary results are defined as results obtained directly from an identification algorithm. These results will typically be:

- ☞ *Model structure information.*
- ☞ *Estimated model parameters.*
- ☞ *Estimated uncertainties of the model parameters.*
- ☞ *Estimated innovation covariance matrix.*
- ☞ *Performance criteria.*

The model structure information will typically be a set of parameters describing the dimensions of the identified model, and which of the parameters of the model have been estimated and which of them have been held constant. Other primary results are the estimated uncertainties of the estimated model parameters. These uncertainties are based upon the Hessian of the criterion function and must therefore be estimated inside the identification algorithm. This also accounts for the estimated innovation covariance matrix, which is based upon the prediction errors calculated from the final parameter estimates and the measurements, and the performance measures such as the AIC and FPE criteria. Based on these primary results any secondary result can be calculated.

Typical secondary results are the modal parameters and their estimated uncertainties, and the spectral densities of the response of the model. The decisions whether the model is optimal or not can also be characterized as a secondary results, since this decision is based on analysis of different primary results. Common to all secondary results is that they can be calculated at any time if the primary results are available. It is therefore only necessary to store the primary results of an identification session. In the toolbox the storage of the primary results is performed by organizing these in one matrix structure, which is called a DDS structure matrix. DDS is an abbreviation of Data Dependent Systems, see Pandit [84], which is a synonym for system identification based on measured data. The contents of this matrix is presented in table 7.1.

The DDS Structure Matrix
Model structure information.
Index of adjustable parameters.
Adjustable and non-adjustable model parameters.
Estimated covariance matrix of adjustable model parameters.
Estimated innovation covariance matrix.
Number of measurement channels.
Number of measurements in each channel.
Scaling matrix of the measurements.
Sampling interval.
Final loss (Final value of the criterion function).
Akaike's Final Prediction Error Criterion (FPE).
ID number for the routine that created the structure.
Time and date for the creation of the structure.

Table 7.1: Information stored in the DDS structure matrix.

The creation of the structure is performed automatically by the identification routines, and the access to the different elements of the structure is provided by different functions. A call to a PEM identification routine in the MATLAB environment will typically look like

» dds = *function*(y, ddsinit)

where y is the measured output and ddsinit is an initial DDS structure containing initial model parameters and model structure information. The primary results of the identification are then returned in dds. On the basis of the primary results stored in dds the model validation routines are typically called by

» valid_result = *function*(dds, y)

and secondary results, such as the modal parameters, are typically obtained by

» second_result = *function*(dds)

7.3.2 An Overview of the Routines of the Toolbox

In this section the different routines in the toolbox will be presented. The purpose of the routines can be divided into the following categories that cover all part of a system identification session:

- ☞ *Information importing.*
- ☞ *Data preprocessing.*
- ☞ *Parameter estimation.*
- ☞ *Model validation.*
- ☞ *Structural mode selection.*
- ☞ *Assessment of uncertainties.*
- ☞ *Information exporting.*

The presentation of the different routines will include a short description of its purpose. A more complete description of the performance of the routines and how they are used can be found in the *Structural Time Domain Identification Users Manual*, see Andersen et al. [8]. The first step in a system identification session is to set up the bookkeeping involved in the identification process and to acquire data. Table 7.2 shows that the STDI toolbox offers functions for the project management and preprocessing of the geometry of the structure, which is necessary for e.g. animation of mode shapes. Finally, among the routines for information importing the toolbox also offers routines for communication with plug-in data acquisition cards and loading of ASCII data files.

Information Importing
Project bookkeeping (Create, load, save....). Structural geometry preprocessing. Data acquisition (Interface to Data Translation plug-in AD-boards, load/save data files acquired in other data acquistion enviroments).

Table 7.2: Information importing.

For preprocessing of the data the STDI toolbox contains a whole range of functions, see table 7.3. The goal of such a preprocessing is to make the measured signals suitable for system identification. Further, the preprocessing functions can be used to show and to estimate statistical characteristics of the measured data.

Preprocessing
Scaling, decimation and resampling. Show measured data and FFT-based spectral densities of it. Estimates of moments (mean, standard deviation ..). Low, high or bandpass filtering. Split the data into identification and validation data sets.

Table 7.3: Preprocessing of the measured data.

For the actual identification the toolbox implements algorithms for identifications of multivariate ARMAV models as well as multivariate stochastic state space systems. The essential algorithms are based on the PEM approach described in chapter 5. However, several other identification algorithms exist. These can either be used as standalone routines or to provide reasonable initial estimates for the PEM routines. In chapter 2, it was shown how to convert an ARMAV model to a stochastic state space realization, and how to convert a stochastic state space realization to an ARMAV model. These conversion schemes make it possible to initialize the ARMAV PEM routine with the parameters of a stochastic state space realization, or to initialize the PEM routine of a stochastic state space realization with initial parameters of an ARMAV model. The different ARMAV estimation routines are listed in table 7.4a, whereas the different stochastic state space realization estimators are listed in table 7.4b.

Parameter Estimation - ARMAV Models
Multi-Stage Least-Square estimation of an ARMAV model. Non-linear Least-Square (PEM) estimation of an ARMAV model. Least-Square (PEM) estimation of an ARV model.

Table 7.4a: Estimation of the parameters of ARMAV related models.

Parameter Estimation - Stochastic State Space Realizations
Non-linear Least-Square (PEM) estimation of a stochastic state space realization. An Eigensystem Realization Algorithm (ERA) for estimation of a stochastic state space realization. Estimation of a stochastic state space realization using a Numerical algorithm Subspace State Space System Identification (N4SID). Estimation of a stochastic state space realization using factorization of a block Hankel matrix of estimated covariance functions of the measurements (MBH factorization).

Table 7.4b: Estimation of the parameters of stochastic state space realizations.

The multistage least-square algorithm for estimation of the parameters of an ARMAV model is described in section 5.6.2. All PEM estimators are based on the theory and principles given in chapter 5, whereas the ERA realization estimator is based on Juang et al. [47]. The N4SID algorithm is based on Van Overschee et al. [107], and the MBH estimator on Aoki [11] and Hoen [38].

To validate or assess the quality of a set of identified models the toolbox offers several functions, see table 7.5. It is possible to obtain the AIC and FPE criteria of the models, see section 5.7.3, plot the poles and zeroes, plot the spectral densities and correlation functions of the prediction errors, and compare the predicted and measured response. Further, it is also possible to compare the FFT-based spectral densities of the measured response with the spectral densities obtained from the model.

Model Validation
FPE - Akaike's Final Prediction Error criterion. AIC - Akaike's Information theoretic Criterion. Plot spectral densities and correlation functions of prediction errors. Plot measured and predicted response. Plot poles and zeroes and their uncertainty elipsoides. Compare spectral densities of measured response with spectral densities of one or more identified models.

Table 7.5: Model validation functions.

To select the structural or fundamental modes of an optimal model the toolbox offers different mode selection functions, see table 7.6. One of the simplest ways to identify the structural modes is to compute the modal parameters and compare these with the spectral densities of the model and e.g. the MAC values of the mode shapes.
Another efficient way is to plot stability diagrams with different plotting criteria of the kinds described in section 6.4. This can of course only be accomplished if several identified models are available.

Structural Mode Selection
Compute the modal parameters. Compute the Modal Assurance Criterion (MAC). Plot the spectral densities of the model. Plot various stability diagrams.

Table 7.6: Mode selection functions.

When the optimal model has been selected, it is important to be able to quantify the uncertainties of the estimated model and modal parameters, see table 7.7. The estimation of the uncertainties of the estimated parameters is performed according to the principles explained in section 6.2.

Assessment of Uncertainties
Return estimated covariance matrix of estimated model parameters. Return estimated standard deviations of estimated modal parameters.

Table 7.7: Assessment of uncertainties.

Having finished an identification session, the results of it have to be presented and exported. The toolbox offers various functions especially made for these purposes. These functions make tables with all estimated modal parameters and their uncertainties, animate mode shapes, and generate a final report of the identification session, see table 7.8.

Information Exporting
Save tables with estimated modal parameters and their uncertainties to an ASCII file. Animate the estimated mode shapes. Generate a final documentation report of the identification session.

Table 7.8: Information exporting.

Besides implementing routines that solve the primary tasks presented in the tables the toolbox implements a variety of necessary auxiliary functions. These routines can be divided into the categories shown in table 7.9.

Auxiliary Functions
Manipulation of matrix polynomials (Multiplication, modal decomposition, filtering, stabilization etc.). Manipulation of state space system (Modal decomposition, filtering, balancing , solving Algebraic Riccati and Lyapunov equations etc.). Relating continuous-time to equivalent discrete-time multivariate systems (Zero-order hold and covariance equivalence techniques).

Table 7.9: Auxiliary functions of the toolbox.

7.3.3 Example 7.1 - An Identification Session

The best way to shown how to use the toolbox is to give a short example. The steps that will be shown in this example are:

- ☞ *Simulation of the response of a Gaussian white noise excited second-order system.*
- ☞ *Identification of an ARMAV model.*
- ☞ *Model validation by investigation of prediction errors and spectral density comparisons.*
- ☞ *Extraction of model parameters and their estimated standard deviations.*

Simulation of the Response of a Gaussian White Noise Excited Second-Order System

The identification session will be based on simulation of a Gaussian white noise excited second-order 2-DOF system. This system is described by the mass, damping and stiffness matrices

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} 0.4 & -0.2 \\ -0.2 & 0.6 \end{bmatrix}, \quad \boldsymbol{K} = \begin{bmatrix} 350 & -150 \\ -150 & 450 \end{bmatrix} \qquad \textbf{(10)}$$

The intensity matrix of the continuous-time Gaussian white noise is $\boldsymbol{W} = \boldsymbol{I}$, and the sampling interval is $T = 0.1$ seconds. An easy way to simulate this system is to calculate a covariance equivalent ARMAV(2,1) model, subject this model to a realization of an equivalent discrete-time Gaussian white noise. This ARMAV(2,1) model, which can be determined by the technique presented in section 4.2.1, is implemented in the routine armav21.m in the toolbox. The following command-line calls determine the covariance equivalent ARMAV model and

simulate 3000 samples in each channel. These simulated noise-free samples are returned in a data matrix y.

```
» M = eye( 2 );
» C = [ 0.4 -0.2;-0.2 0.6 ];
» K = [ 350 -150;-150 450 ];
» W = eye( 2 );
» T = 0.1;
» N = 3000;
» u = randn( N, 2 );
» dds21 = armav21( M, C, K, T, W );
» y = ddssim( dds21, u );
```

Gaussian white noise can then be added to the response of each of channels of y to simulate measurement noise. These two added noise sequences are assumed independent. Each of them is assumed to have a standard deviation equal to 10 % of the sampled standard deviations of the simulated responses. The following show how to add these noise terms and how to plot the spectral densities of the noise-contaminated response using the toolbox routine fftspec.m.

```
» R = 0.1*diag( std( y )' );
» v = ( R*randn( 2, N ) )';
» y = y + v;
» fftspec( y, 256, 128, T );
```

This last command-line call will then make the following plots of the spectral densities of the response using FFT. The resolution is 256 points and the segments overlap by 128 points.

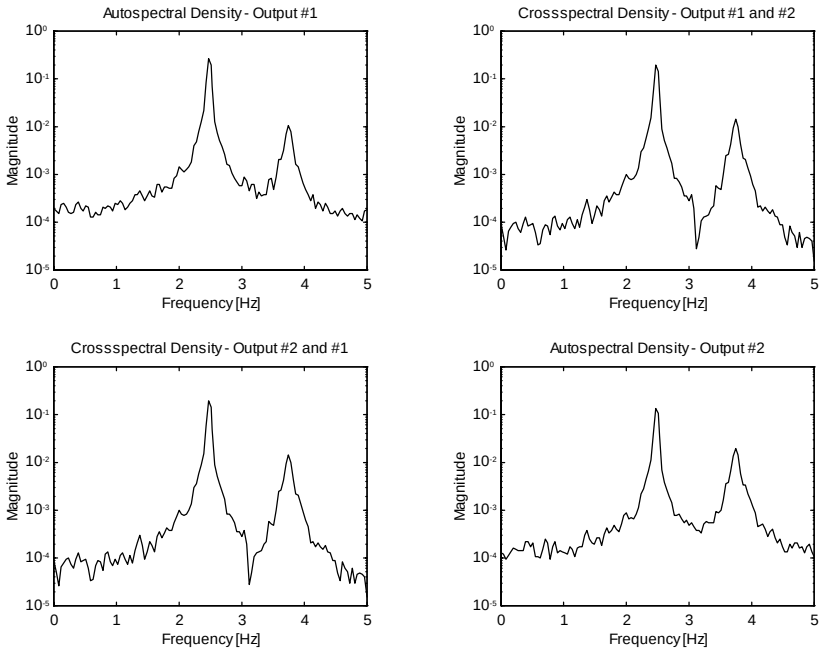

Figure 7.2: Spectral densities of noise-contaminated simulated response.

Identification of an ARMAV Model

The system can then be identified from the measurements. Since Gaussian white noise has been added to the true system, which was described by an ARMAV(2,1) model, the appropiate discrete-time model is an ARMAV(2,2) model. This model can be calibrated to the measurements using the identification routine armav.m that utilizes the nonlinear PEM estimator described in chapter 5. The following command-line calls shows how armav.m can be used and how the primary results of the identification can be viewed.

```
» dds22 = armav( y, [2 2] );
» dds22 = t2dds( dds22, T );
» showdds( dds22 );
```

This discrete-time 2-channel ARMAV(2,2) model was created by the command ARMAV on the 12/2 1997 at 13:07. It was based on 3000 samples in each channel, and a sampling interval of 0.1 seconds.
Loss function: 1.269e-007, Akaikes FPE: 1.283e-007
The parameters and their standard deviations given as imaginary parts are:

```
A =

  Columns 1 through 4

   1.0000         0          0.4405 + 0.0088i  -0.6517 + 0.0116i
   0         1.0000          -0.6679 + 0.0076i   0.8937 + 0.0098i

  Columns 5 through 6

   0.9606 + 0.0085i   0.0255 + 0.0111i
   0.0186 + 0.0073i   0.9427 + 0.0095i

C =

  Columns 1 through 4

   1.0000         0          0.4089 + 0.0211i  -0.2605 + 0.0238i
    0         1.0000          -0.2626 + 0.0194i   0.5834 + 0.0217i

  Columns 5 through 6

   0.1887 + 0.0205i  -0.0679 + 0.0226i
  -0.0477 + 0.0189i   0.1965 + 0.0208i

The innovation covariance matrix is:
Lam =
  1.0e-003 *

    0.4020    0.1167
    0.1167    0.3497

The scaling matrix of the response is:
Scale =

     1     0
     0     1
```

Model Validation by analysis of the Prediction Errors and Spectral Density Comparisons

If the estimated model contains the true system then the prediction errors should be a white noise sequence. This can be investigated by plotting the spectral densities and the correlation functions of the prediction errors. This is one of the purposes of the routine dds2pe.m in the toolbox. The spectral densities of a discrete-time white noise is constant in the frequency range from minus to plus the Nyquist frequency. The correlation function of the discrete-time white noise is described by a spike, a Kronecker delta, at the zero lag. Since only a sequence of the prediction errors is available and not the complete process, it is satisfactory if the correlation at other time lags is located inside a 95 % confidence interval. This confidence interval is also shown as a dotted line in the correlation plots obtained from dds2pe.m. The routine can be called from the MATLAB command-line as :

```
» dds2pe( dds22, y );
```

and the result is the following plots.

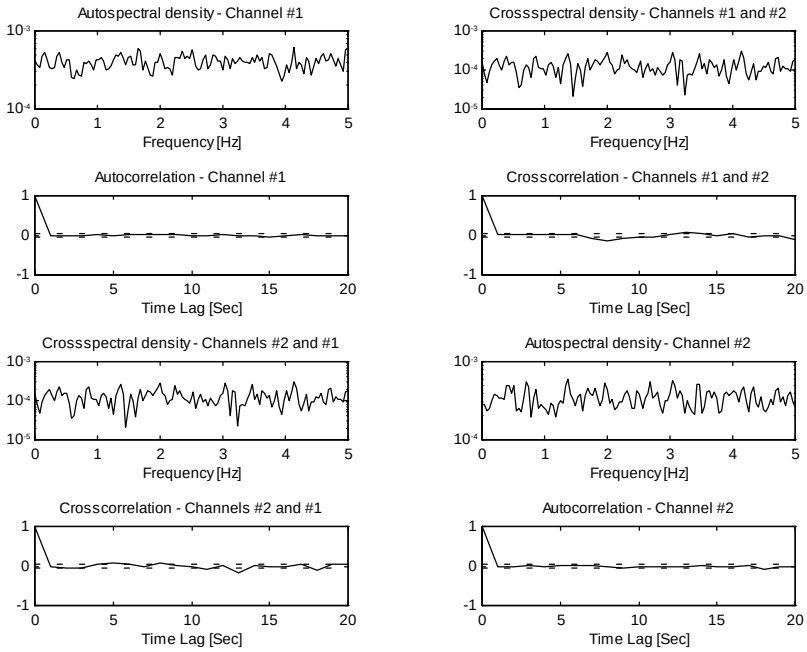

Figure 7.3: Spectral densities and correlation function estimates of the prediction errors.

On the basis of the plots of correlation functions the prediction error sequences can be assumed to be a white noise sequences. However, the sequences are not completely serially uncorrelated. The correspondence between the estimated model and the measured data can also be verified visually by comparing the spectral densities obtained directly from the measurements using FFT with the spectral densities obtained from the model. This can be accomplished by the following command-line call to the routine speccmp.m in the toolbox.

```
» speccmp( dds22, y );
```

and the result is the following plots.

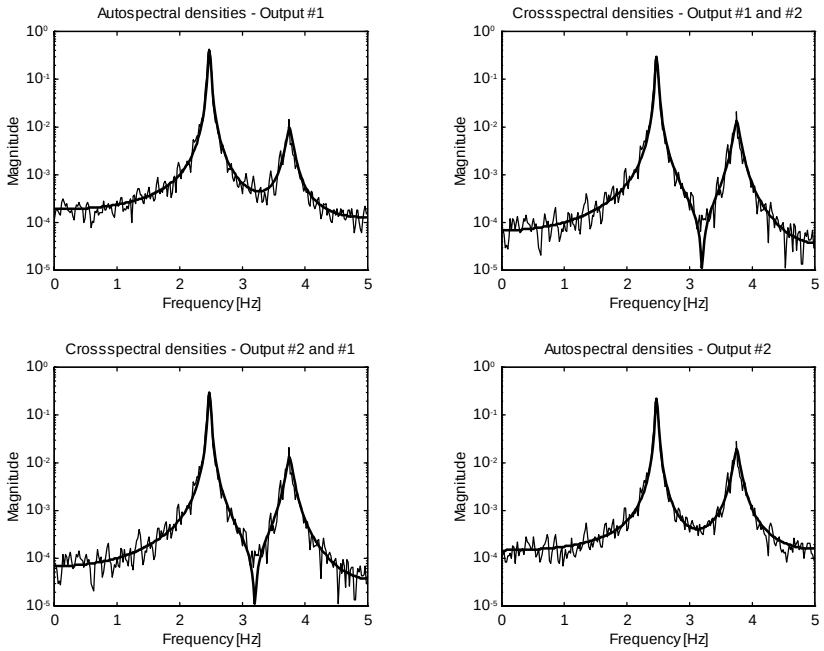

Figure 7.4: Comparison of spectral densities obtained directly from the measured response using FFT and obtained from the estimated model. The smooth lines are the spectral densities of the model.

Extraction of Modal Parameters and their Estimated Standard Deviations

As a final example of how to use the toolbox, it could be interesting to investigate the modal parameters of the system. The natural eigenfrequencies, damping ratios and scaled mode shapes of the estimated model can easily be obtained using the routine modal.m in the toolbox. In addition, it could be interesting to have an idea about the quality of these parameter estimates. The quality can be assessed by estimation of the standard deviations of the parameter, which can be obtained by calling the routine sdmodal.m in the toolbox. This estimation follows the principles explained in section 6.2. The results obtained from calling modal.m from the MATLAB command-line are shown below.

```
» [ phi, f, zeta ] = modal( dds22 )

phi =
  1.3839 + 0.0106i  -0.7071 - 0.0018i
  1.0000             1.0000

f =
   2.4755
   3.7533

zeta =
   0.0080
   0.0159
```

The columns of phi are the scaled and complex mode shapes. These mode shapes have been normalized with respect to the last coordinate of each mode shape. The vector f contains the

corresponding natural eigenfrequencies, and zeta the damping ratios. The estimated standard deviations are obtained from calling sdmodal.m as shown below.

```
» [ sdphi, sdf, sdzeta ] = sdmodal( dds22 )

sdphi =
   0.0105   0.0176
   0        0

sdf =
   0.0033
   0.0066

sdzeta =
   0.0013
   0.0017
```

The columns of sdphi are the estimated standard deviations of the mode shapes with respect to the chosen normalization. The vector sdf contains the estimated standard deviations of the natural eigenfrequencies, whereas the vector sdzeta contains the estimated standard deviations of the damping ratios.

As seen, it is easy to manage the data obtained during a system identification session. All primary results are hidden away in the DDS structure matrix and all secondary results are accessed through various routine calls having the structure matrix as input argument. All validation routines and routines that extract secondary results work regardless of whether a DDS structure describes an ARMAV model or a stochastic state space realization. The reason is that computational manipulations of the ARMAV models are performed by converting these models into balanced stochastic state space realization. □

7.4 Summary

In this chapter the numerical and computational aspects have been considered. It has been described how to prepare the measured data by prefiltering, resampling and detrending to make them suitable for system identification. The chapter has also concerned how to scale the measured data and balance the estimated models to improve the numerical accuracy of e.g. the modal decomposition of the identified system. If the measured data contains outliers the prediction filter used should account for this, since the presence of outliers can have significantly bad effect on the performance of the estimator. Further, if the amount of measured data is limited, and if the model order is relatively high then a bad transient behaviour of the prediction filter might occur. If this is the case, it should be eliminated since it also can have a significantly bad effect on the performance of the estimator. One way of eliminating this problem is to use the backward forecasting procedure for estimation of the unknown initial conditions of the prediction filter. As a part of the Ph.D. project on which this thesis is based a MATLAB toolbox for system identification of especially ambient excited civil engineering structures has been developed. This toolbox is called the Structural Time Domain Identification (STDI) toolbox. In this chapter, applications of this toolbox have been introduced. The use of the toolbox has been illustrated by an example based on simulated response of a Gaussian white noise excited second-order system.