NTNU – Trondheim
Norwegian University of
Science and Technology

Operational Modal Analysis of Large Bridges

Sindre Aavik Schanke

Civil and Environmental Engineering
Submission date: June 2015
Supervisor: Ole Andre Øiseth, KT

Norwegian University of Science and Technology
Department of Structural Engineering

Institutt for konstruksjonsteknikk

FAKULTET FOR INGENIØRVITENSKAP OG TEKNOLOGI
NTNU – Norges teknisk-naturvitenskapelige universitet

MASTEROPPGAVE 2015

for

Sindre Aavik Schanke

Modalanalyse av store brukonstruksjoner

Operational Modal Analysis of Large Bridges

I forbindelse med prosjektet ferjefri E39 blir Hardangerbrua instrumentert for å kartlegge nøyaktigheten til de metodene som benyttes til å beregne dynamisk respons av konstruksjoner utsatt for vindlaster. For å analysere denne typen målinger brukes teknikker innen systemidentifikasjon og modalanalyse. Denne oppgaven dreier seg om modalanalyse av store brokonstruksjoner.

Oppgaven bør inneholde følgende temaer:

- Grunnleggende teori for modalanalyse
- Implementering og grundig teoretisk fremstilling av en rekke teknikker for modalanalyse.
- Estimater av egenfrekvenser, demping og svingeformer for Hardangerbrua.
- Sammenligning av metodene og analyse av usikkerheter

Besvarelsen organiseres i henhold til gjeldende retningslinjer.

Veiledere: Ole Andre Øiseth, Knut Andreas Kvåle

NTNU, 16.3 2015

Ole Andre Øiseth
Faglærer

Abstract

As a part of the new coastal highway E39 which is being planned and built along the west coast of Norway by the Norwegian Public Roads Administration, state of the art methods of structural dynamics needs to be developed and used to cross the deep and wide fjords. Operational modal analysis aims to find the modal properties; natural frequencies, damping ratios and mode shapes of a structure while it is under operating conditions using its vibration data. Several methods of operational modal analysis are developed and they can broadly be divided into time-domain and frequency-domain. Theory of operational modal analysis and the methods chosen are presented in this thesis.

The main objective of this thesis was to develop MATLAB functions which implements several of these methods. The methods are performed on two case studies. A shear frame with low and high damping where the exact modal parameters are known, this is done to check that the methods work properly and to see how damping influences the accuracy of the methods. Then the methods are used on the Hardanger Bridge, a bridge in operating conditions. The modal properties are extracted for each of the methods and then compared on account of accuracy. The methods are also compared regarding ease of use and computational efficiency.

Most of the methods found the natural frequencies for the low damping shear frame, but for high damping some of the methods proved to be inadequate. For damping ratio and mode shapes only a few of the methods managed to give satisfactory results. For the Hardanger Bridge most of the methods managed to find the natural frequencies, but only a few managed to estimate the damping ratios and mode shapes. Overall, Cov-SSI proved to be the best method.

Sammendrag

Som en del av den nye motorveien E39 som er under planlegging og bygging langs vestkysten av Norge av Statens Vegvesen trengs siste nytt av metoder innen konstruksjonsdynamikk å bli utviklet og brukt for å krysse dype og brede fjorder. Modalanalyse prøver å finne dynamisk respons; egenfrekvenser, demping og svingeformer, av en konstruksjon imens den er i bruk ved å bruke konstruksjonens vibrasjonsdata. Flere metoder av modalanalyse har blitt utviklet og de kan grovt deles inn i tidsplanet og frekvensplanet. Teori om modalanalyse og metodene som er valgt vil bli presentert i denne oppgaven.

Hovedmålet med denne oppgaven var å utvikle MATLAB-funksjoner som implementerer flere av disse metodene. Metodene er utført på to sakstudier. En rammekonstruksjon med lav og høy demping hvor den eksakste dynamiske responsen er kjent, dette for å sjekke om metodene fungerer og for å se hvordan demping påvirker nøyaktigheten til metodene. Deretter blir metodene utført på Hardangerbrua, en bru i operativ tilstand. Den dynamiske responsen blir funnet for hver av metodene og sammenlignet med tanke på nøyaktighet. Metodene blir også sammenlignet når det gjelder brukervennelighet og dataeffektivitet.

De fleste metodene fant egenfrekvensene for rammekonstruksjonen med lav demping, men for høy demping var flere av metodene utilstrekkelige. For demping og svingeformer var det bare noen av metodene som ga tilfredsstillende resultat. For Hardangerbrua klarte de fleste metodene å finne egenfrekvensene, men bare noen klarte å estimere demping og svingeformene. Cov-SSI viste seg å være den beste metoden alt i alt.

Preface

This master thesis is the result of 20 weeks of work in spring 2015 carried out for the Department of Structural Engineering. It concludes the five year civil engineering Master's degree study at the Norwegian University of Science and Technology (NTNU).

I would like to give thanks to my supervisors Associate Professor Ole Andre Øiseth and PhD Candidate Knut Andreas Kvåle. Their input have been prompt and necessary to help solve some of the obstacles experienced during my work. My results would not have been possible without their generous support and supply of data used for the calculations.

Sindre A. Schanke

Trondheim, 2015-06-03

Contents

Abbreviations

BSS	Blind source separation
Cov-SSI	Covariance-driven stochastic subspace identification
CPSD	Cross power spectral density
DD-SSI	Data-driven stochastic subspace identification
DFT	Discrete fourier transformation
DOF	Degree of freedom
EMA	Experimental modal analysis
FDD	Frequency domain decomposition
FFT	Fast fourier transform
FRF	Frequency response function
JAD	Joint approximation diagonalization
LSCF	Least squares complex frequency method
MDOF	Multi-degree of freedom
MFD	Matrix fraction description
NFFT	Number of samples in FFT
OMA	Operational modal analysis
pLSCF	Poly-reference least squares complex frequency method
PSD	Power spectral density
RMFD	Right matrix fraction description
SDOF	Single-degree of freedom
SOBI	Second order blind identification
SSI	Stochastic subspace identification
SVD	Singular value decomposition

ZOH Zero order hold

Nomenclature

Symbols

$[]$	Matrix
$[]^T$	Matrix transpose
$[]^H$	Hermitian transpose and complex conjugate transpose
$[]^{-1}$	Matrix inverse
$[]^+$	Matrix pseudo-inverse
$\dot{}$	Time derivative $\frac{d}{dt}$
$Re()$	Real part of a complex number
$Im()$	Imaginary part of a complex number
i	Imaginary unit
$E()$	Expectation value
$\otimes$	Kronecker product

Variables

m, c, k	Mass, damping and stiffness coefficients
$[M], [C], [K]$	Mass, damping and stiffness matrices
$[\ddot{y}(t)], [\dot{y}(t)], [y(t)]$	Acceleration, velocity and displacement vectors
$[f(t)]$	Force vector
ω_n	Natural frequency $\frac{rad}{sec}$
ω_d	Damped modal frequency $\frac{rad}{sec}$
ζ	Damping ratio

λ	Eigenvalue in continuous time
$[q]$	Eigenvector, in solution of differential equation $[q]e^{\lambda t}$
ω	Load frequency $\frac{rad}{sec}$
$[Y(\omega)]$	Fourier transform of $[y(t)]$
$[F(\omega)]$	Fourier transform of $[f(t)]$
$H(\omega)$	FRF
$[\bar{B}]$	Location of inputs
$[u(t)]$	Time variation
$[s(t)], [\dot{s}(t)]$	State vector and its derivative
$[A_c]$	State matrix in continuous time
$[B_c]$	Input influence matrix in continuous time
$[C_a]$	Output location matrix for acceleration
$[C_v]$	Output location matrix for velocity
$[C_d]$	Output location matrix for displacement
$[C_c]$	Output influence matrix in continuous time
$[D_c]$	Direct transmission matrix in continuous time
$[s_k]$	Discrete-time state vector
$[u_k]$	Sampled input
$[y_k]$	Sampled output
$[A]$	State matrix in discrete time
$[B]$	Input influence matrix in discrete time
$[C]$	Output influence matrix in discrete time
$[D]$	Direct transmission matrix in discrete time
$[w_k]$	Process noise
$[v_k]$	Measurement noise
$[R]$	Output covariance matrix
$[G]$	Next state-output covariance matrix
N	Number of samples
Δt	Time step
T	Finite sampling period

$R_y(\tau)$	Correlation function
$S_y(\omega)$	PSD
$R_{xy}(\tau)$	Cross-correlation function
$S_{xy}(\omega)$	CPSD
$\lambda^2_{xy}(\omega)$	Coherence function
$S_f(\omega)$	PSD of the input
$x(t)$	Sample record
$v(t)$	Unlimited record
$w(t)$	Time window
$[Y]$	Data matrix
l	Number of measurement channels
n	System order
i	Number of block rows
x	Magnitude of block rows
$[T_{1\|i}]$	Toeplitz matrix
$[U], [\Sigma], [V]$	Products of SVD
$[O_i]$	Observability matrix
$[\Gamma_i]$	Reversed controllability matrix
$[\mu]$	Eigenvalues in discrete time
$[\Phi]$	Eigenvectors
N_f	Number of frequency lines
j	Number of columns in Hankel matrix
$[H]$	Hankel matrix
$[L]$	Lower triangular decomposition of Hankel matrix
$[Q]$	Orthonormal decomposition of Hankel matrix
$[P_i], [P_{i-1}]$	Projections
$[Y_{i\|i}]$	Output sequence
$[\hat{S}_i], [\hat{S}_{i+1}]$	Kalman filter state sequence
$[Y_c]$	Data matrix with zero mean
$[W]$	Whitening matrix

$[z]$	Whitened data
$[\tilde{A}']$	Unitary matrix
p	Number of time lags
t	Threshold
$[A_m]$	Mixing matrix
$[s_o(t)]$	Sources
$p(t), [p(t)]$	Modal coordinate and modal coordinate vector
$[\theta_d]$	Vector of denominator coefficients
z_f	Generalized transform variable
$[\alpha]$	Matrix of denominator coefficients
u_0, v_o, a_o	Starting conditions for displacement, velocity and acceleration

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation for Research

Dynamic behavior is an important part of many civil engineering structures. Only trough understanding vibrations, an optimization of design can be achieved. To gain knowledge of the dynamic response of a civil structure experimental tests have been used, a practice dating back to the middle of the twentieth century [18]. Experimental modal analysis (EMA) identifies the dynamic response from measurements of the applied force and the vibration response. It is a classic input-output method where input need to be applied, controlled and measured. The output, vibration response, also need to be measured. EMA is a method widely used in different fields of engineering [19]. For civil structures EMA techniques become more challenging, due to their large size and low frequency range [18]. For large buildings, bridges etc. it can be expensive or disruptive to carry out EMA. Therefore another method, operational modal analysis (OMA) have been developed which takes advantage of the naturally occurring loads (e.g. wind or traffic).

OMA is defined as the modal testing procedure that allows the experimental estimation of the modal parameters of the structure from measurements of the vibration response only [18]. Most of the OMA methods have been derived from EMA procedures, because they share a common theoretical background. The main difference is regarding the input, for EMA it is known and measured, while for OMA it is random and not measured. For civil structures the

advantages of OMA over EMA is many, mainly the reduction in cost and the possibility to use the structure as normal while data is collected. The disadvantages are more complicated methods and lower accuracy [24].

1.2 Scope of Thesis

This thesis aims to implement different methods of OMA into MATLAB. They are used on two case studies. The first is a simple shear frame where the exact solution is known to check if the methods work sufficiently. The methods will be tested on a shear frame with low damping and on a shear frame with high damping, this is done to check how the methods react to the degree of damping. The second is on the Hardanger Bridge, using time series from measurements. The methods aim to find the modal properties; natural frequency, damping ratios and mode shapes for the problems. The accuracy and efficiency of the methods are then compared.

This thesis will derive and present a comprehensive theoretical basis for the methods implemented. The methods to be presented are based on the work of others, but the implementation in MATLAB is new work. The reader is assumed to already be familiar with conventional mechanics and structural dynamics. However a small introduction to a MDOF system will be given. The reader is also assumed to be familiar with basic mathematics and statistics.

1.3 Structure of the Report

Chapter 2: Comprehensive theoretical basis for OMA and the methods implemented.

Chapter 3: Case study on a shear frame.

Chapter 4: Case study on the Hardanger Bridge.

Chapter 5: Conclusions and remarks about possible future work.

Chapter 2

Theory

2.1 Structural Dynamics and Modal Analysis

The dynamic behavior of a structure can be represented in either the time domain or the frequency domain. In the time domain it is a set of differential equations, while in the frequency domain it is a set of algebraic equations.

2.1.1 Time Domain

The equation of motion is traditionally given in the time domain. For a general multi-degree of freedom (MDOF) system the equation of motion is:

$$[M][\ddot{y}(t)] + [C][\dot{y}(t)] + [K][y(t)] = [f(t)] \tag{2.1.1}$$

Where [M], [C] and [K] symbolizes the constant mass, damping and stiffness matrix for the system respectively. [$\ddot{y}$(t)], [$\dot{y}$(t)] and [y(t)] symbolizes the acceleration, velocity and displacement respectively. [f(t)] symbolizes the force vector. The solution for the dynamic parameters of a MDOF system is represented by different modes, which relates to the relative displacement of the system's degrees of freedom. Each mode have a natural frequency (ω_n) and a damping ratio (ζ). One way to extract these dynamic parameters is to solve the differential Equation 2.1.1. This differential equation is assumed to have a solution on the form [4]:

$$[y(t)] = [q]e^{\lambda t} \tag{2.1.2}$$

Derivative of the displacement gives:

$$[\dot{y}(t)] = \lambda[q]e^{\lambda t} \tag{2.1.3}$$

$$[\ddot{y}(t)] = \lambda^2[q]e^{\lambda t} \tag{2.1.4}$$

Putting Equation 2.1.2, Equation 2.1.3 and Equation 2.1.4 into Equation 2.1.1 and assuming the system is unloaded (f(t)=0) gives:

$$(\lambda^2[M] + \lambda[C] + [K])q = [0] \tag{2.1.5}$$

The solution of Equation 2.1.5 depends on the damping of the system. For most civil structures the damping of the system is underdamped which means that the solution for λ is a complex conjugate pair. One of the solutions are:

$$\lambda_k = -\zeta\omega_n \pm \sqrt{1-\zeta^2}\omega_n i \tag{2.1.6}$$

Which gives:

$$\omega_n = |\lambda_k| \tag{2.1.7}$$

$$\omega_d = Im(\lambda_k) \tag{2.1.8}$$

$$\zeta = -\frac{Re(\lambda_k)}{|\lambda_k|} \tag{2.1.9}$$

2.1.2 Frequency Domain

By applying the Fourier transform, which will be explained in Section 2.4.1, to Equation 2.1.1 it becomes a set of linear algebraic equations [18]:

$$(-\omega^2[M] + i\omega[C] + [K])[Y(\omega)] = [F(\omega)] \tag{2.1.10}$$

Where [Y(ω)] and [F(ω)] are the Fourier transforms of [y(t)] and [f(t)] respectively and i is the imaginary unit. By introducing the frequency response function (FRF) $H(\omega)$:

$$[H(\omega)]^{-1} = -\omega^2[M] + i\omega[C] + [K] \tag{2.1.11}$$

Substituting Equation 2.1.11 into Equation 2.1.10 gives:

$$[H(\omega)]^{-1}[Y(\omega)] = [F(\omega)] \tag{2.1.12}$$

Which means that the FRF represents the ratio between the Fourier transforms of the input and the output:

$$[H(\omega)] = \frac{[Y(\omega)]}{[F(\omega)]} \tag{2.1.13}$$

2.2 Operational Modal Analysis

OMA takes in measured data in form of a signal. A signal is a physical quantity varying with respect to one or more independent variables and associated to information of interest[18]. The signal can be in different domains (time or frequency) and can be converted from one to another. A system converts an input signal into an output signal. Finding response to a known system and given input is called a forward problem. While an inverse problem is where output is known, but either input or system characteristics are unknown. Noise is undesired signal superimposed on the signal of interest. Since input is not controlled for OMA some assumptions need to be made. If a structure is excited by white noise, a Gaussian distributed, statistically independent value with a constant input spectrum, then all the modes are equally

excited and the output spectrum contains full information about the structure [18]. However the naturally occurring loads (wind, traffic etc.) are uncontrollable and immeasurable and noise is likely to occur during a measurement. Therefore in OMA the structure is assumed to be excited by unknown forces, which are the output of the excitation system loaded by white noise as shown in Figure 2.1.

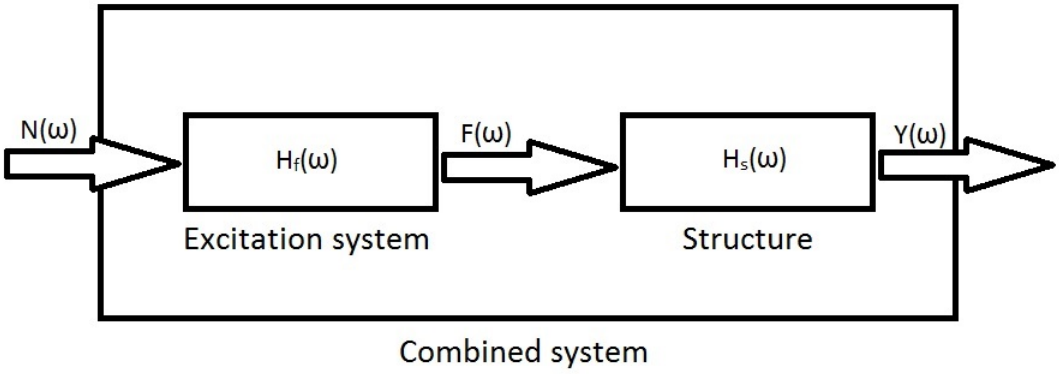

Figure 2.1: Combined system

Where N(ω) is the Fourier transformation of the white noise input into the excitation system given by its FRF $H_f(\omega)$. $F(\omega)$ is the Fourier transform of the excitation system output and $H_s(\omega)$ is the FRF of the structure. $Y(\omega)$ is the output from the structure, which is the value measured in OMA. Given by the following equations:

$$[F(\omega)] = [H_f(\omega)][N(\omega)] \tag{2.2.1}$$

$$[Y(\omega)] = [H_s(\omega)][F(\omega)] \tag{2.2.2}$$

2.3 State Space Models

State space models are used to convert a second order differential equation into two first order differential equations defined by the so-called state equation and the observation equation. We can obtain the state equation from Equation 2.1.1. If we factorize $[f(t)]$ into $[\bar{B}]$ and $[u(t)]$. Where the matrix $[\bar{B}]$ defines the location of inputs and the vector $[u(t)]$ describes

the time variation [18]. The second order differential equation given by Equation 2.1.1 can be rewritten as:

$$[M][\ddot{y}(t)] + [C][\dot{y}(t)] + [K][y(t)] = [\bar{B}][u(t)] \tag{2.3.1}$$

Dividing by [M]:

$$[\ddot{y}(t)] + [M]^{-1}[C][\dot{y}(t)] + [M]^{-1}[K][y(t)] = [M]^{-1}[\bar{B}][u(t)] \tag{2.3.2}$$

Introducing the state vector, defined by:

$$[s(t)] = \begin{bmatrix} \dot{y}(t) \\ y(t) \end{bmatrix} \tag{2.3.3}$$

And the derivative of the state vector:

$$[\dot{s}(t)] = \begin{bmatrix} \ddot{y}(t) \\ \dot{y}(t) \end{bmatrix} \tag{2.3.4}$$

Equation 2.3.3 into Equation 2.3.2 and rearranging gives:

$$[\ddot{y}(t)] = \begin{bmatrix} -[M]^{-1}[C] & -[M]^{-1}[K] \end{bmatrix} [s(t)] + \begin{bmatrix} [M]^{-1}[\bar{B}]] \end{bmatrix} [u(t)] \tag{2.3.5}$$

And it can easily be seen from Equation 2.3.3 that, where $[I]$ is the identity matrix:

$$[\dot{y}(t)] = \begin{bmatrix} [I] & [0] \end{bmatrix} [s(t)] \tag{2.3.6}$$

Equation 2.3.5 and Equation 2.3.6 into Equation 2.3.4 gives:

$$[\dot{s}(t)] = \begin{bmatrix} -[M]^{-1}[C] & -[M]^{-1}[K] \\ [I] & [0] \end{bmatrix} [s(t)] + \begin{bmatrix} [M]^{-1}[\bar{B}] \\ [0] \end{bmatrix} [u(t)] \tag{2.3.7}$$

From Equation 2.3.7 the State matrix $[A_c]$ and the input influence matrix $[B_c]$ can be defined:

$$[A_c] = \begin{bmatrix} -[M]^{-1}[C] & -[M]^{-1}[K] \\ [I] & [0] \end{bmatrix} \tag{2.3.8}$$

$$[B_c] = \begin{bmatrix} [M]^{-1}[\bar{B}] \\ [0] \end{bmatrix} \tag{2.3.9}$$

Together they yield the state equation:

$$[\dot{s}(t)] = [A_c][s(t)] + [B_c][u(t)] \tag{2.3.10}$$

If we assume that the structural response measurements are taken at l locations and the sensor are accelerometers (first term), velocimeters (second term) and displacement transducers (third term) the observation equation can be written as [18]:

$$[y_l(t)] = [C_a][\ddot{y}(t))] + [C_v][\dot{y}(t)] + [C_d][y(t)] \tag{2.3.11}$$

Substituting Equation 2.3.3 and Equation 2.3.5 into Equation 2.3.11:

$$[y_l(t)] = \begin{bmatrix} [C_v] - [C_a][M]^{-1}[C] & [C_d] - [C_a][M]^{-1}[K] \end{bmatrix} [s(t)] + \begin{bmatrix} [C_a][M]^{-1}[\bar{B}]] \end{bmatrix} [u(t)] \tag{2.3.12}$$

From Equation 2.3.12 the output influence matrix $[C_c]$ and the direct transmission matrix $[D_c]$ can be defined:

$$[C_c] = \begin{bmatrix} [C_v] - [C_a][M]^{-1}[C] & [C_d] - [C_a][M]^{-1}[K] \end{bmatrix} \tag{2.3.13}$$

$$[D_c] = \begin{bmatrix} [C_a][M]^{-1}[\bar{B}]] \end{bmatrix} \tag{2.3.14}$$

Together they yield the observation equation:

$$[y(t)] = [C_c][s(t)] + [D_c][u(t)] \tag{2.3.15}$$

Measurements from experiments are taken at discrete time instants, and therefore Equation 2.3.10 and Equation 2.3.15 has to be converted to discrete time. For a given sampling period Δt, the discrete time instants are $t_k = k\Delta t$. Zero order hold (ZOH) states that the input

is piecewise constant over the sampling period [18]. If we assume this, Equation 2.3.10 and Equation 2.3.15 can be converted to the discrete-time state space model:

$$[s_{k+1}] = [A][s_k] + [B][u_k] \tag{2.3.16}$$

$$[y_k] = [C][s_k] + [D][u_k] \tag{2.3.17}$$

Where $[A]$ is the discrete state matrix, $[B]$ is the discrete input matrix, $[C]$ is the discrete output matrix and $[D]$ is the direct transmission matrix [18]. $[s_k] = [s(k\Delta t)]$ is the discrete-time state vector which yields the sampled displacements and velocities. $[u_k]$ is the sampled input, while $[y_k]$ is the sampled output. The relations between the matrices $[A]$, $[B]$, $[C]$ and $[D]$ in continuous time and in discrete time are [18]:

$$[A] = e^{|A_c|\Delta t} \tag{2.3.18}$$

$$[B] = ([A] - [I])[A_c]^{-1}[B_c] \tag{2.3.19}$$

$$[C] = [C_c] \tag{2.3.20}$$

$$[D] = [D_c] \tag{2.3.21}$$

The discrete-time state space model shown in Equation 2.3.16 and Equation 2.3.17 is a deterministic model, since the input is deterministic. To describe actual measurement data you need a stochastic component, and therefore process noise $[w_k]$ and measurement noise $[v_k]$ is introduced. $[w_k]$ and $[v_k]$ are two stochastic processes, and the discrete-time combined deterministic-stochastic state-space model is obtained:

$$[s_{k+1}] = [A][s_k] + [B][u_k] + [w_k] \tag{2.3.22}$$

$$[y_k] = [C][s_k] + [D][u_k] + [v_k] \tag{2.3.23}$$

The process noise $[w_k]$ is due to disturbances and model inaccuracies, while the measurement noise $[v_k]$ is due to sensor inaccuracies [18].

For OMA, the input of the structure is immeasurable. $[u_k]$ is not available and therefore the system output $[y_k]$ is generated only by $[w_k]$ and $[v_k]$. This gives the discrete-time stochastic state-space model:

$$[s_{k+1}] = [A][s_k] + [w_k] \tag{2.3.24}$$

$$[y_k] = [C][s_k] + [v_k] \tag{2.3.25}$$

The goal of the stochastic subspace identification (SSI) methods of OMA is to measure a large amount of output $[y_k]$ to be able to find $[A]$ and $[C]$. $[w_k]$ and $[v_k]$ is immeasurable but are assumed to be zero mean white noise processes with the following covariance matrices if p = q:

$$E\left[\begin{bmatrix}[w_p]\\ [v_p]\end{bmatrix}\quad\begin{bmatrix}[w_q]^T & [v_q]^T\end{bmatrix}\right] = \begin{bmatrix}[Q^{ww}] & [S^{wv}]\\ [S^{wv}]^T & [R^{vv}]\end{bmatrix} \tag{2.3.26}$$

And if p $\neq$ q:

$$E\left[\begin{bmatrix}[w_p]\\ [v_p]\end{bmatrix}\quad\begin{bmatrix}[w_q]^T & [v_q]^T\end{bmatrix}\right] = [0] \tag{2.3.27}$$

Where p and q are two arbitrary time instants.

With this assumption the system response in the state-space model is also represented by a zero mean Gaussian process. Then the covariance of the output, the output covariance matrix $[R]$, is given by:

$$[R_i] = E([y_{k+i}][y_k]^T) \tag{2.3.28}$$

The discrete-time state vector $[s_k]$ is also a zero mean Gaussian process. And the covariance of the state vector is:

$$[\Sigma] = E([s_k][s_k]^T) \tag{2.3.29}$$

And $[s_k]$ is uncorrelated with $[w_k]$ and $[v_k]$:

$$E([s_k][w_k]^T) = [0] \tag{2.3.30}$$

$$E([s_k][v_k]^T) = [0] \tag{2.3.31}$$

Taking these realizations into account mathematical manipulations give [18]:

$$[\Sigma] = [A][\Sigma][A]^T + [Q^{ww}] \tag{2.3.32}$$

$$[R_0] = [C][\Sigma][C]^T + [R^{vv}] \tag{2.3.33}$$

$$[G] = [A][\Sigma][C]^T + [S^{wv}] \tag{2.3.34}$$

$$[R_i] = [C][A]^{i-1}[G] \tag{2.3.35}$$

Where the next state-output covariance matrix $[G]$ is given by:

$$[G] = E([s_{k+i}][y_k]^T) \tag{2.3.36}$$

2.4 Frequency Response

2.4.1 Fourier Transform

A signal $y(t)$ can be described as the sum of harmonic signals. To divide the signal into the harmonic components Fourier transformation is used. It changes the time-domain signal $y(t)$ to the frequency-domain function $Y(\omega)$. For a non-periodic signal $y(t)$ it is given by [18]:

$$Y(\omega) = \int_{-\infty}^{\infty} y(t) e^{-i2\pi ft} dt \tag{2.4.1}$$

$$y(t) = \int_{-\infty}^{\infty} Y(\omega) e^{-i2\pi ft} d\omega \tag{2.4.2}$$

In practical applications the measurements are done over a limited sampling period, T, and therefore you need to take the discrete Fourier transformation (DFT) instead of the continuous Fourier transformation. Which yields [14]:

$$Y_k = \frac{1}{N} \sum_{r=0}^{N-1} y_r e^{-i2\pi \frac{kr}{N}} \tag{2.4.3}$$

$$y_r = \sum_{k=0}^{N-1} Y_k e^{i2\pi \frac{kr}{N}} \tag{2.4.4}$$

Where N is the total number of measurements $N = T\Delta t$. And the N measurement taken at equally spaced time instants Δt are numbered $r = 0, 1, 2, ..., N-1$. And the frequency values are numbered $k = 0, 1, 2, ..., N-1$. The evaluation of Equation 2.4.3 requires N^2 operations [18]. As a consequence the fast Fourier transform (FFT) was developed. Provided that the number of data points equals a power of 2, the number of operations is reduced to $Nlog_2N$ in FFT [5].

2.4.2 Spectrum

Power Spectral Density

The power spectral density (PSD) $[S_y(\omega)]$ gives a representation of how the power of a signal is distributed over its frequencies. For a stochastic signal $y(t)$ the correlation function R_y is defined as [14]:

$$R_y(\tau) = E[y(t)y(t+\tau)] \tag{2.4.5}$$

The PSD and the correlation function are a Fourier transform pair [14]:

$$S_y(\omega) = \int_{-\infty}^{\infty} R_y(\tau) e^{-i2\pi ft} dt \tag{2.4.6}$$

$$R_y(\tau) = \int_{-\infty}^{\infty} S_y(\omega) e^{-i2\pi ft} d\omega \tag{2.4.7}$$

The PSDs are real valued functions.

Cross Power Spectral Density

The cross power spectral density (CPSD) $[S_{xy}(\omega)]$ gives a representation of how the power of the covariance between two signals is distributed over their frequencies. For two stochastic signals $x(t)$ and $y(t)$ the cross-correlation function R_{xy} is defined as [14]:

$$R_{xy}(\tau) = E[x(t)y(t+\tau)] \tag{2.4.8}$$

The CPSD and the cross-correlation function are a Fourier transform pair [14]:

$$S_{xy}(\omega) = \int_{-\infty}^{\infty} R_{xy}(\tau) e^{-i2\pi ft} dt \tag{2.4.9}$$

$$R_{xy}(\tau) = \int_{-\infty}^{\infty} S_{xy}(\omega) e^{-i2\pi ft} d\omega \tag{2.4.10}$$

The CPSDs are complex valued functions.

Coherence

The coherence function is defined as [18]:

$$\gamma_{xy}^2(\omega) = \frac{|S_{xy}(\omega)|^2}{S_x(\omega)S_y(\omega)} \tag{2.4.11}$$

With the constraints:

$$0 \leq \gamma_{xy}^2(\omega) \leq 1 \tag{2.4.12}$$

2.4.3 Frequency Response Function

Taking the realization found earlier (Equation 2.1.13), the definition of the PSD and the properties of transpose, the following relation between the PSD of the input, the PSD of the output and the FRF matrix can be obtained [18]:

$$[S_y(\omega)] = [H(\omega)]^H [S_f(\omega)][H(\omega)]^T \tag{2.4.13}$$

Where ^{H} denotes the complex conjugate transpose or the Hermitian adjoint. So if the PSD of the input $S_f(\omega)$ is constant as assumed in Section 2.2, since the input to the combined system is a stationary, zero mean Gaussian white noise, the output PSD carries the same information as the FRF. This is a realization that is used later for the frequency domain methods.

FRF is a special case of matrix fraction description (MFD) represented by [18]:

$$[H(\omega)] = \frac{[B(\omega)]}{A(\omega)} \tag{2.4.14}$$

Where the numerator $[B(\omega)]$ is a matrix polynomial:

$$[B(\omega)] = \sum_{j=0}^{n} [B_j(\omega)] z^j(\omega) \tag{2.4.15}$$

And the denominator $A(\omega)$ is a polynomial characterized by scalar coefficients:

$$A(\omega) = \sum_{j=0}^{n} a_j(\omega) z^j(\omega) \tag{2.4.16}$$

Another version of MFD is the right matrix fraction description (RMFD) [3]:

$$[H(\omega)] = [B_R(\omega)][A_R(\omega)]^{-1} \tag{2.4.17}$$

Both FRF and RMFD are used later for the frequency domain methods.

2.5 Welch's Method

In mpsd.m Welch's method is used to estimate the PSD and CPSD for each measurement channel. Welch's method is carried out by dividing the time series into K short sections [23]. Then the periodogram, estimate of the spectral density, of that section is calculated:

$$p_k = \frac{1}{M} Y(\omega) Y(\omega)^H \tag{2.5.1}$$

Where $Y(\omega)$ is the FFT of the time series of that section and M is the number of samples in FFT (NFFT).

Then the PSD is given as an average of the periodograms across the period [23]:

$$S_y(\omega) = \frac{1}{K} \sum_{k=1}^{K} p_k \tag{2.5.2}$$

In the same way the CPSD can be calculated:

$$p_k = \frac{1}{M} X(\omega) Y(\omega)^H \tag{2.5.3}$$

$$S_{xy}(\omega) = \frac{1}{K} \sum_{k=1}^{K} p_k \tag{2.5.4}$$

If the signal is periodic the sectioning of the time series should be done equally to the wave length of the signal. If not, the time series will get a discontinuity at each change of section. For a non-periodic signal you will get this discontinuity no matter what length you choose for the section. And therefore a windowing function is introduced. It is a mathematical function that is zero-valued outside the chosen interval. A sample record $x(t)$ can be represented as an unlimited record $v(t)$ multiplied with a time window w(t). The simplest window function is a rectangular window given as:

$$x(t) = w(t)v(t) \qquad w(t) = \begin{cases} 1 & 0 \leq t \leq T \\ 0 & \text{elsewhere} \end{cases} \tag{2.5.5}$$

This will however not remove the problem with the discontinuity as the value at the edge of the section is multiplied with one. Figure 2.2 shows the windowing function and its effect on the FFT with 25 samples.

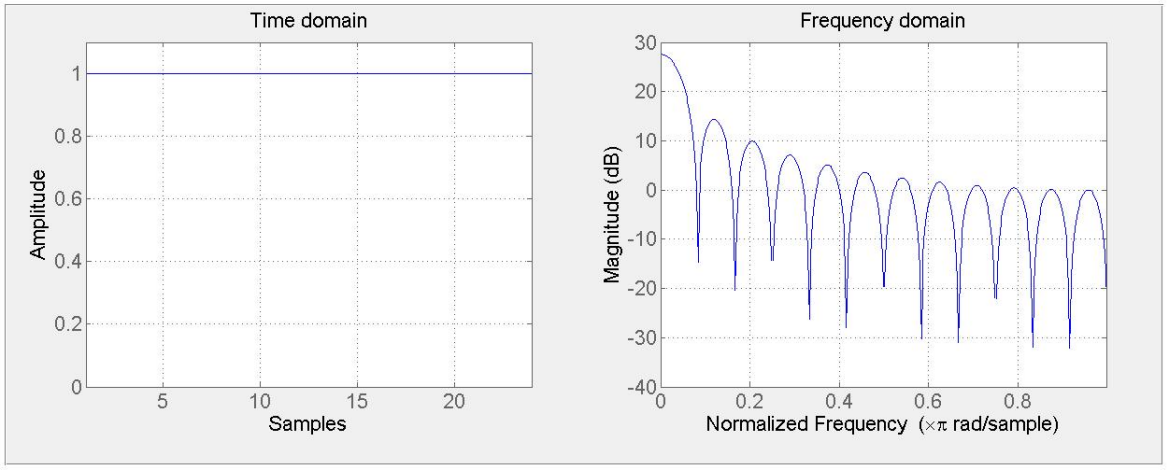

Figure 2.2: Rectangular window

A triangular window function would remove the problem with discontinuity as it is zero at the edges as show in Figure 2.3.

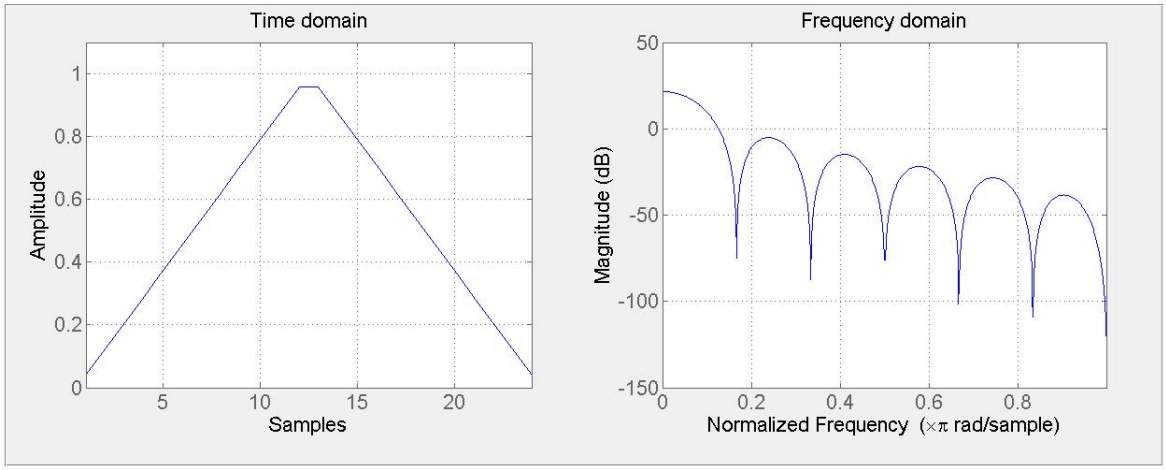

Figure 2.3: Triangular window

mpsd.m takes the window function as an input and the user is free to use whatever windowing function he or she prefer as they have different advantages and disadvantages . In this thesis the Hanning window is used:

$$x(t) = w(t)v(t) \qquad w(t) = \begin{cases} 1 - cos^2(\frac{\pi t}{T}) & 0 \leq t \leq T \\ 0 & \text{elsewhere} \end{cases} \tag{2.5.6}$$

The effect is shown in Figure 2.4.

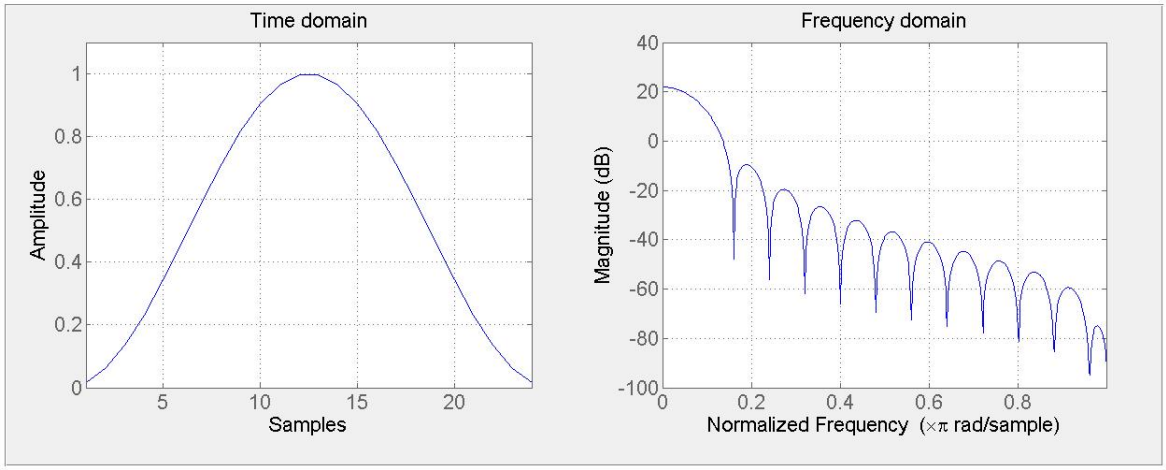

Figure 2.4: Hanning window

For the triangular window and the Hanning window the edges of the window suffers a "loss", therefore overlapping is used to compensate. For instance if 50% overlap is chosen, the second

half of the first section is the same as the first half of the second section. Since their scaling factor, from the windowing function, will be ”opposite” at a time t in the time series this will reduce the loss at the edges. This can be seen in Figure 2.4.

2.6 Time Domain Methods

2.6.1 Covariance-Driven Stochastic Subspace Identification

Covariance-driven stochastic subspace identification (Cov-SSI) identifies a stochastic state-space model from output-only data. It is a time-domain, parametric, covariance driven procedure for OMA [18]. The goal of this method is to use the output measurements to obtain [A],[C] and [G] as shown in Section 2.3, from which the dynamic response is found.

Input

A system of order n is possible to observe only if the so called observability matrix and controllability matrix is of rank n. They will be introduced later in this section. However, as the system order is usually unknown for complicated problems, a conservative approach is to overestimate the order of the system. Due to the fact that the system order is overestimated additional nonphysical poles, mathematical poles, will occur next to the physical poles [18]. To be able to distinguish them and find the correct poles a stabilization diagram has to be used. It is further discussed in Section 2.8. This overestimated max order, n_{max}, is used as an input for the problem and therefore some experience is needed to choose an appropriate value. If the max order is chosen to be smaller than the correct system order you will not get any correct results. If you however assume this value too high you will get too many nonphysical modes and it will be harder to derive which modes that are the actual correct physical modes, in addition the computational time greatly increases.

Another input required for Cov-SSI is the data matrix. It can either be measured deformation, velocity or acceleration under the influence of environmental loads. The data matrix Y have dimensions $l*N$ where l is the number of measurement channels and N is the number of

measurements. The time between each measurement is the time step. The last input is the magnitude of block rows which will be explained shortly.

Initial Calculations

The first step of this method is to calculate the output correlations (Equation 2.3.28). $[R_i]$ denotes the unbiased estimate of the correlation matrix at time lag i based on a finite number of data [18]:

$$[R_i] = \frac{1}{N-i}[Y_{(1:N-i)}][Y_{(i:N)}]^T \tag{2.6.1}$$

Where $[Y_{(1:N-i)}]$ is the data matrix Y with the last block rows i removed. And $[Y_{(i:N)}]^T$ is the transpose data matrix with the first block rows i removed. Therefore each $[R_i]$ matrix get dimensions *l*l*. The estimated correlations at different time lags (Equation 2.6.1) are then gathered into a matrix called the block Toeplitz matrix:

$$[T_{1|i}] = \begin{bmatrix} [R_i] & [R_{i-1}] & \cdots & [R_1] \\ [R_{i+1}] & [R_i] & \cdots & [R_2] \\ \vdots & \vdots & \ddots & \vdots \\ [R_{2i-1}] & [R_{2i-2}] & \cdots & [R_i] \end{bmatrix} \tag{2.6.2}$$

The Toeplitz matrix contains *i*i* $[R_i]$ matrices and is therefore of dimensions *li*li*. For the identification of the modal parameters of a system of order n, the Toeplitz matrix need to be n*n. Therefore the following need to be true for the number of block rows i:

$$li \geq n \tag{2.6.3}$$

Where n is the system order. However as mentioned earlier the system order is usually unknown and the following should be fulfilled:

$$i_{min} = \frac{n_{max}}{l} \tag{2.6.4}$$

For complicated structures the number of block rows should be higher than this minimum criteria for better results. The magnitude x of this depends on the problem and should be

chosen as an input by the user:

$$i_{min} = x\frac{n_{max}}{l} \tag{2.6.5}$$

Then the singular value decomposition (SVD) of the block Toeplitz is calculated. It gives the unitary matrices [U] and [V], with dimensions *li*n* and *n*li* respectively. And the positive singular values (which equals its rank) in descending order in the diagonal matrix [Σ] [22].

$$[T_{1|i}] = [U_1][\Sigma_1][V_1]^T \tag{2.6.6}$$

During this step the one-lag shifted Toeplitz matrix is also calculated:

$$[T_{2|i+1}] = \begin{bmatrix} [R_{i+1}] & [R_i] & \cdots & [R_2] \\ [R_{i+2}] & [R_{i+1}] & \cdots & [R_3] \\ \vdots & \vdots & \ddots & \vdots \\ [R_{2i}] & [R_{2i-1}] & \cdots & [R_{i+1}] \end{bmatrix} \tag{2.6.7}$$

State Matrix

To extract the dynamic response you need to obtain the state matrix [A] (Equation 2.3.18). This is done for each order from 1 to n_{max}. The first thing needed is to find the observability matrix $[O_i]$ and the reversed controllability matrix $[\Gamma_i]$ which is found by the factorization of $[T_{1|i}]$. This factorization comes from the connection given by Equation 2.3.35:

$$[T_{1|i}] = [O_i][\Gamma_i] \tag{2.6.8}$$

Where the observability matrix $[O_i]$ with dimensions *li*n* is given by:

$$[O_i] = \begin{bmatrix} [C] \\ [C][A] \\ \vdots \\ [C][A]^{i-1} \end{bmatrix} \tag{2.6.9}$$

And the reversed controllability matrix $[\Gamma_i]$ with dimensions *n*li* is given by:

$$[\Gamma_i] = \begin{bmatrix} [A]^{i-1}[G] & \cdots & [A][G] & [G] \end{bmatrix} \tag{2.6.10}$$

The SVD of $[T_{1|i}]$ is already computed in Equation 2.6.6 and we can use the result to find $[O_i]$ and $[\Gamma_i]$. By splitting the SVD in two parts and using the identity matrix [I]:

$$[O_i] = [U_1][\Sigma_1]^{1/2}[I]^T \tag{2.6.11}$$

$$[\Gamma_i] = [I]^{-1}[\Sigma_1]^{1/2}[V_1]^T \tag{2.6.12}$$

Now that we have obtained $[O_i]$ and $[\Gamma_i]$ we can find the output influence matrix [C] and the next state-output covariance matrix [G]. [C] is obtained from the first l rows of $[O_i]$ as seen in Equation 2.6.9. And in the same way [G] is obtained from the last l columns of $[\Gamma_i]$ as seen in Equation 2.6.10.

The one-lag shifted Toeplitz matrix (Equation 2.6.7) can be factorized in the same way as the normal Toeplitz matrix, which yields:

$$[T_{2|i+1}] = [O_i][A][\Gamma_i] \tag{2.6.13}$$

Introducing Equation 2.6.11 and Equation 2.6.12 into Equation 2.6.13 and then solving for [A] gives us the $n*n$ state matrix:

$$[A] = [\Sigma_1]^{-1/2}[U_1]^T[T_{2|i+1}][V_1][\Sigma_1]^{-1/2} \tag{2.6.14}$$

Modal Parameters

Now that the state matrix [A] and the output influence matrix [C] is known, modal parameters can be extracted [18]. Solving the eigenvalue problem for [A] yields the diagonal matrix [M] which contains the eigenvalues μ and the eigenvectors $[\Psi]$ [4]:

$$[A] = [\Psi][M][\Psi]^{-1} \tag{2.6.15}$$

The mode shapes of the system [Φ] is obtained from [Ψ] and [C]:

$$[\Phi] = [C][\Psi] \tag{2.6.16}$$

The rest of the modal parameters are obtained from the eigenvalues μ. They are found in the diagonal matrix [M]. For each mode, μ_m are found, they are obtained in discrete time and need to be converted to continuous time. This is done according to Equation 2.3.18, which yields:

$$\lambda_m = \frac{ln(\mu_m)}{\Delta t} \tag{2.6.17}$$

And we obtain the complex λ_m as we found in Equation 2.1.6. λ_m contains the continuous time eigenvalues of each mode for the current order. It can be used to find the natural frequencies (ω_n), damped modal frequencies (ω_d) and the damping ratio (ζ) for the r-th mode as shown in Equation 2.1.7, Equation 2.1.8 and Equation 2.1.9:

$$\omega_{n,r} = |\lambda_{m,r}| \tag{2.6.18}$$

$$\omega_{d,r} = Im(\lambda_{m,r}) \tag{2.6.19}$$

$$\zeta_r = -\frac{Re(\lambda_{m,r})}{|\lambda_{m,r}|} \tag{2.6.20}$$

When the modal parameters are found for the current order, the process of finding the state matrix and the modal parameters are repeated for each order up until n_{max}. Then all the parameters are plotted in a stabilization diagram, which will be discussed in Section 2.8.

2.6.2 Data-Driven Stochastic Subspace Identification

Data-driven stochastic subspace identification (DD-SSI) is similar to Cov-SSI, but identifies the state sequence before the estimation of the state-space matrices. It is a numerically very efficient method, as it uses well known and robust linear algebra to identify the state-space

matrices. The input of DD-SSI is identical to the input of Cov-SSI.

Initial Calculations

Firstly the number of block rows i is set equal to x times the maximum system order (n_{max}) divided by the number of measurement channels l:

$$i = x\frac{n_{max}}{l} \tag{2.6.21}$$

Where x is the magnitude of block rows as for Cov-SSI.

Then the Hankel matrix is constructed. The number of columns j of the Hankel matrix is assumed to be ∞ for the statistical proof of the method, and therefore j needs to be large. In practical applications it is set as $N - 2i + 1$ so that all given data samples are used under the construction of the Hankel matrix [18].

$$j = N - 2i + 1 \tag{2.6.22}$$

The Hankel matrix is constructed directly from the measured data $Y(1:N)$:

$$[H_{0|2i-1}] = \frac{1}{\sqrt{j}} \left[\begin{array}{cccc} [y_0] & [y_1] & \cdots & [y_{j-1}] \\ [y_1] & [y_2] & \cdots & [y_j] \\ \vdots & \vdots & \ddots & \vdots \\ [y_{i-1}] & [y_i] & \cdots & [y_{i+j-2}] \\ \hline [y_i] & [y_{i+1}] & \cdots & [y_{i+j-1}] \\ [y_{i+1}] & [y_{i+2}] & \cdots & [y_{i+j}] \\ \vdots & \vdots & \ddots & \vdots \\ [y_{2i-1}] & [y_{2i}] & \cdots & [y_{2i+j-2}] \end{array}\right] \tag{2.6.23}$$

Then the LQ factorization of the Hankel matrix is done [11]:

$$[H_{0|2i-1}] = [L][Q] \tag{2.6.24}$$

Which means that the Hankel matrix is expressed as a product of a lower triangular matrix $[L]$ and an orthonormal matrix $[Q]$ [8]:

$$[L] = \begin{array}{c} \\ li \\ l \\ l(i-1) \end{array} \begin{array}{c} \begin{array}{ccc} li & l & l(i-1) \\ \leftrightarrow & \leftrightarrow & \leftrightarrow \end{array} \\ \updownarrow \begin{bmatrix} [L_{11}] & [0] & [0] \\ [L_{21}] & [L_{22}] & [0] \\ [L_{31}] & [L_{32}] & [L_{33}] \end{bmatrix} \end{array} \tag{2.6.25}$$

$$[Q] = \begin{array}{c} j \\ \leftrightarrow \\ \begin{bmatrix} [Q_1]^T \\ [Q_2]^T \\ [Q_3]^T \end{bmatrix} \end{array} \tag{2.6.26}$$

The projections $[P_i]$ and $[P_{i-1}]$ can be obtained from the LQ decomposition:

$$[P_i] = \begin{bmatrix} [L_{21}] \\ [L_{31}] \end{bmatrix} [Q_1]^T \tag{2.6.27}$$

$$[P_{i-1}] = \begin{bmatrix} [L_{31}] & [L_{32}] \end{bmatrix} \begin{bmatrix} [Q_1]^T \\ [Q_2]^T \end{bmatrix} \tag{2.6.28}$$

So can the output sequence $[Y_{i|i}]$:

$$[Y_{i|i}] = \begin{bmatrix} [L_{21}] & [L_{22}] \end{bmatrix} \begin{bmatrix} [Q_1]^T \\ [Q_2]^T \end{bmatrix} \tag{2.6.29}$$

And finally the SVD of $[P_i]$ is calculated:

$$[P_i] = [U][\Sigma][V]^T \tag{2.6.30}$$

State Matrix

As for Cov-SSI these steps are repeated for each order from 1 to n_{max}. First the observability matrix $[O_i]$ and the Kalman filter state sequence $[\hat{S}_i]$ is found from the factorization of $[P_i]$:

$$[P_i] = [O_i][\hat{S}_i] \tag{2.6.31}$$

The SVD of $[P_i]$ is already calculated (Equation 2.6.30) and can be used to compute the observability matrix in the same way as for Cov-SSI:

$$[O_i] = [U][\Sigma]^{1/2}[I]^T \tag{2.6.32}$$

From Equation 2.6.31 the Kalman filter state sequence can be computed as:

$$[\hat{S}_i] = [O_i]^+[P_i] \tag{2.6.33}$$

Where $[O_i]^+$ means the pseudo-inverse of $[O_i]$.

With a factorization similar to Equation 2.6.31 of $[P_{i-1}]$:

$$[P_{i-1}] = [O_i^{\uparrow}][\hat{S}_i] \tag{2.6.34}$$

One can find the Kalman state sequence $[\hat{S}_{i+1}]$ in the same way as Equation 2.6.33:

$$[\hat{S}_{i+1}] = [O_i^{\uparrow}]^+[P_i] \tag{2.6.35}$$

Where $[O_i^{\uparrow}]$ is obtained from $[O_i]$ by deleting the last l rows.

Now that $[\hat{S}_{i+1}]$, $[\hat{S}_i]$ and $[Y_{i|i}]$ is found one can find the asymptotically unbiased least squares estimate of $[A]$ and $[C]$ [21]:

$$\begin{bmatrix}[A]\\[C]\end{bmatrix} = \begin{bmatrix}[\hat{S}_{i+1}]\\[Y_{i|i}]\end{bmatrix}[\hat{S}_i]^+ \tag{2.6.36}$$

When the state matrix [A] and the output influence matrix [C] is known the modal parameters are extracted in the same way as for Cov-SSI. The process is then repeated for each system order and then plotted in a stabilization diagram (Section 2.8).

2.6.3 Second Order Blind Identification

Second order blind identification (SOBI) is identified as a time-domain, nonparametric method which uses blind source separation (BSS) to extract a set of signals, called *sources*, from observation of their mixtures [18]. The only input required is the data matrix Y and the time step Δt.

Initial Calculations

The first part of the SOBI algorithm is centering (make zero mean) and whitening the data [25].

$$[z] = [W][Y_c] \tag{2.6.37}$$

Where $[Y_c]$ is the centered data matrix, $[W]$ is the whitening matrix and $[z]$ is the whitened data.

By taking the SVD of the centered data $[Y_c]$:

$$[Y_c] = [U][\Sigma][V]^T \tag{2.6.38}$$

The whitening matrix $[W]$ can be found:

$$[W] = [\Sigma]^+[V]^T \tag{2.6.39}$$

A number of time-shifted covariance matrices have to be computed:

$$[R_z(\tau_k)] \tag{2.6.40}$$

Where $k = 1, 2, ..., p$.

Joint Approximation Diagonalization

A joint approximation diagonalization (JAD) technique is used on the time-shifted covariance matrices (Equation 2.6.40), where the goal is to find the unitary matrix $[\tilde{A}']$. This is a minimization problem where you want to find the matrix $[\tilde{A}']$ which minimizes the sum of the off diagonal terms of $([\tilde{A}']^T[R_z(\tau_k)][\tilde{A}'])$ as seen in Equation 2.6.41 [18]:

$$min \sum_{k=1}^{n} ([\tilde{A}']^T[R_z(\tau_k)][\tilde{A}']) \tag{2.6.41}$$

The solution of this problem is done by a numerical algorithm based on the Jacobi rotation technique [2]. Two parameters have to be set, the number of time lags p and the threshold t. The performance of the diagonalization improves with an increasing p, but seems to rapidly converge [18], therefore these values have been chosen initially:

$$p = min(100, N/3) \tag{2.6.42}$$

As for the threshold, the value where the JAD should stop, it is usually not necessary to have very small values, as the diagonalization already only is an approximation [18]. This value has been chosen initially:

$$t = \frac{1}{\sqrt{N}} \tag{2.6.43}$$

Once the unitary matrix $[\tilde{A}']$ has been found [6] the mixing matrix $[A_m]$ can be computed:

$$[A_m] = [W]^+[\tilde{A}'] \tag{2.6.44}$$

And the sources $[s_o(t)]$:

$$[s_o(t)] = [\tilde{A}']^T * [z(t)] \tag{2.6.45}$$

Modal Parameters

The mode shapes can be found right from the columns of the mixing matrix. While the natural frequency, damped modal frequency and damping ratio is obtained with a single-degree

of freedom (SDOF) curve fitting estimator. It is based on FFT and then picking one peak (therefore SDOF). The process is repeated for the number of measurement channels, and therefore the number of modal parameters found by SOBI is the same as the number of measurement channels. The SOBI algorithm is not especially sophisticated, and damping ratio estimates are not particularly good [18], SOBI will therefore not be used to estimate damping ratios in this thesis, but extraction for damping ratios is implemented in the MATLAB function.

2.7 Frequency Domain Methods

2.7.1 Peak Picking

Peak picking is the most basic OMA method. It has been widely used in the past and is still used due to its efficiency and simplicity. The input is the PSDs and CPSDs between each measurement channel, as well as the corresponding frequency vector. It can be classified as a SDOF method, because it is based on the assumption that, around a resonance, only one mode is dominant [18]. Therefore, it is not a good method to find closely spaced modes. The theoretical proof of the method is based on the fact that the structural response is approximately equal to the modal response if only one mode is dominant [15]:

$$[y(t)] \approx [\Phi]p(t) \tag{2.7.1}$$

Where $[\Phi]$ is the modal vector and $p(t)$ is the modal coordinate. The correlation matrix $[R_y]$ is approximately:

$$[R_y(\tau)] = E\left[[y(t+\tau)][y(t)]^T\right] = R_p(\tau)[\Phi][\Phi]^T \tag{2.7.2}$$

Where the modal auto-correlation function $R_p(\tau)$ is given by:

$$R_p(\tau) = E\left[[p(t+\tau)]p(t)\right] \tag{2.7.3}$$

By taking the Fourier transform of Equation 2.7.2 you'll get the PSD:

$$[S_y(\omega)] = S_p(\omega)[\Phi][\Phi]^H \tag{2.7.4}$$

Where $S_p(\omega)$ is the PSD of the modal coordinate, and therefore $[S_y(\omega)]$ must have rank one. So at resonance any column of $[S_y(\omega)]$ will give an estimate of the mode shape up to a scaling factor [18].

Plotting

From plotting the PSD of each measurement channel you can pick the peaks to identify the natural frequencies. Then, the corresponding mode shapes can be found. Ideally one measurement channel could find all the structural modes of the system, but often in practical applications several measurement channels are required to find all the modes. Especially a measurement channel in x-direction can sometimes only find the horizontal modes, and then a measurement channel in y-direction is required to find the vertical modes. Usually both the x-direction and y-direction channels can find the torsion modes. In addition two sensors placed on opposite parts of a bridge will also show different impact on the different modes.

You would also want to plot the CPSD between the measurement channels as shown in Figure 2.5 to help picking the correct peaks. As an alternative you could plot the coherence between the channels as shown in Figure 2.6. At an actual mode the coherence is close to 1, while at fake peaks due to disturbances in the PSD, the coherence is not close to 1 and picking these fake peaks can therefore be avoided.

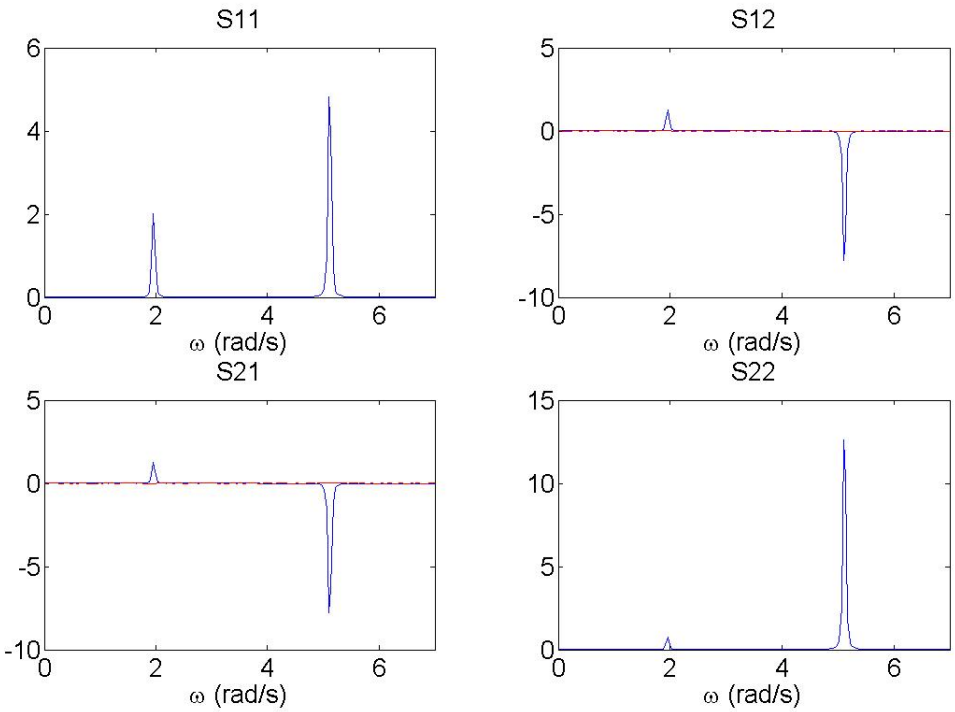

Figure 2.5: Example of PSD and CPSD plots

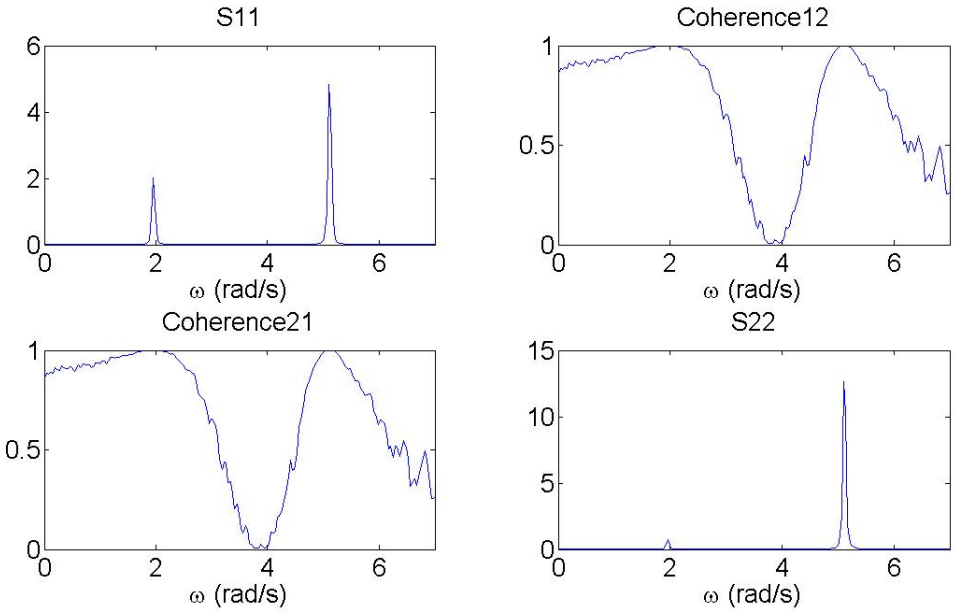

Figure 2.6: Example of PSD and coherence plots

2.7.2 Frequency Domain Decomposition

Since peak picking had problems with identifying closely spaced modes, another method was introduced, the frequency domain decomposition (FDD) method. The input required for the FDD method is the same as for the peak picking method, the PSDs and CPSDs, as well as the frequency vector.

Calculations

The theoretical proof of the method is based on the modal expansion of the structural response [18]:

$$[y(t)] = [\Phi][p(t)] \tag{2.7.5}$$

Where $[\Phi]$ is the modal matrix and $[p(t)]$ is the vector of modal coordinates. The correlation matrix $[R_y]$ of the response is:

$$[R_y(\tau)] = E\left[[y(t+\tau)][y(t)]^T\right] = [\Phi][R_p(\tau)][\Phi]^T \tag{2.7.6}$$

By taking the Fourier transform you'll get the PSD:

$$[S_y(\omega)] = [\Phi][S_p(\omega)][\Phi]^H \tag{2.7.7}$$

As shown in Equation 2.7.7 one needs to gather the PSDs and CPSDs for each frequency line. This results in a $l * l$ matrix with the values for each frequency line from $1, 2, ..., N_f$. Then a SVD is computed for each frequency line:

$$[S_y(\omega)] = [U][\Sigma][V]^H \tag{2.7.8}$$

The PSDs and CPSDs gathered for each frequency line is a Hermitian and positive definite matrix, meaning that its real values is symmetric and its complex values is equal to its own conjugate transpose. Therefore $[U] = [V]$ and Equation 2.7.8 can be rewritten:

$$[S_y(\omega)] = [U][\Sigma][U]^H \tag{2.7.9}$$

The similarities between Equation 2.7.7 and Equation 2.7.9 suggests that one can find the mode shapes [Φ] from [U] and the Singular values [S] corresponds to the modal response. Since they are sorted in a descending order, the first singular value will peak close to a resonance. And then one can find the mode shape from the first row of [U] for that singular value. A logarithmic plot of the singular values will show which modes are dominant.

From the Singular value plots one can pick the peaks. A simple example is shown in Figure 2.7. One can easily pick the two peaks which relates to the two modes of the system, the mode shapes can then be extracted.

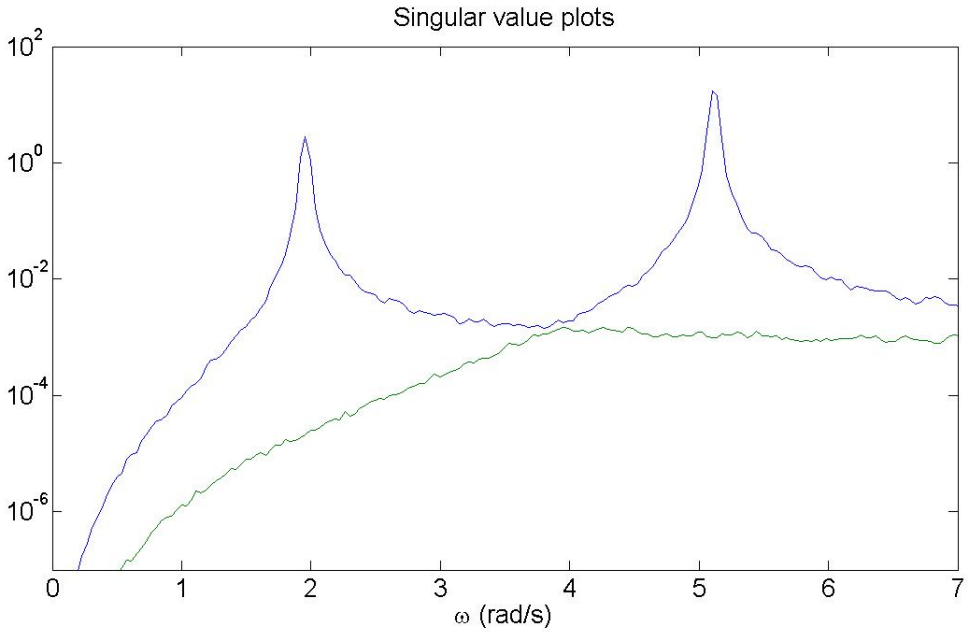

Figure 2.7: Example of singular value plots

2.7.3 Least Squares Complex Frequency Method

Least squares complex frequency method (LSCF) takes advantage of the global character of the structural poles and common-denominator model to identify the modal parameters [18]. It is a parametric, frequency domain modal identification procedure [18].

LSCF input is the same as for the other frequency domain methods. The PSDs and CPSDs computed for all the measurement channels, as well as the corresponding frequency vector.

It also needs the time step Δt and a n_{max} for the plotting of the stabilization diagram.

Intial Calculations

LSCF is based on FRF as shown in Section 2.4.3 and Equation 2.4.14. For the l measurement channels you obtain $l * l$ PSDs and CPSDs. They are numbered as $k = 1, 2, ..., l * l$ and arranged, as for FDD, by each frequency line $f = 1, 2, ..., N_f$.

$$S_k(\omega_f) = \frac{N_k(z_f, [\theta])}{d(z_f, [\theta])} \tag{2.7.10}$$

Where the numerator polynomial N_k is:

$$N_k(z_f, [\theta]) = \sum_{j=0}^{n} N_{k,j} z_f^j \tag{2.7.11}$$

And the common-denominator polynomial d is:

$$d(z_f, [\theta] = \sum_{j=0}^{n} d_j z_f^j \tag{2.7.12}$$

The coefficients d_j are gathered in θ_d, which are the complex-valued parameters to be estimated:

$$[\theta_d] = \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} \tag{2.7.13}$$

Where one of the denominator coefficients should be equal to 1 for mathematical reasons [18]. d_n is set equal to 1, since this choice simplifies the identification of the structural pole [18].

z_f is the generalized transform variable at frequency line f. Different choices for z_f is possible, but here the following is chosen [3]:

$$z_f^j = e^{i\omega_f \Delta t} \tag{2.7.14}$$

Calculations

These calculations are repeated for $n = 1, 2, ..., n_{max}$ and then plotted in a stabilization diagram for each n. Γ_k and Υ_k are calculated for each k:

$$[\Gamma_k] = \begin{bmatrix} z_1^0 & z_1^1 & \cdots & z_1^n \\ z_2^0 & z_2^1 & \cdots & z_2^n \\ \vdots & \vdots & \ddots & \vdots \\ z_{N_f}^0 & z_{N_f}^1 & \cdots & z_{N_f}^n \end{bmatrix} \tag{2.7.15}$$

$$[\Upsilon_k] = \begin{bmatrix} -S_k(\omega_1)z_1^0 & -S_k(\omega_1)z_1^1 & \cdots & -S_k(\omega_1)z_1^n \\ -S_k(\omega_2)z_2^0 & -S_k(\omega_2)z_2^1 & \cdots & -S_k(\omega_2)z_2^n \\ \vdots & \vdots & \ddots & \vdots \\ -S_k(\omega_{N_f})z_{N_f}^0 & -S_k(\omega_{N_f})z_{N_f}^1 & \cdots & -S_k(\omega_{N_f})z_{N_f}^n \end{bmatrix} \tag{2.7.16}$$

Where z_f^j is as in Equation 2.7.14 and $S_k(\omega_f)$ as Equation 2.7.10.

Some matrices are defined. It can be shown [10] that using only the real parts of these matrices in the following gives a good result with lower computational cost:

$$[R_k] = Re([\Gamma_k]^H[\Gamma_k]) \tag{2.7.17}$$

$$[S_k] = Re([\Gamma_k]^H[\Upsilon_k]) \tag{2.7.18}$$

$$[T_k] = Re([\Upsilon_k]^H[\Upsilon_k]) \tag{2.7.19}$$

The reduced normal equations are given as [3]:

$$[M][\theta_d] \approx [0] \tag{2.7.20}$$

Where $[M]$ is given as [10]:

$$[M] = 2\sum_{k=1}^{l*l}([T_k] - [S_k]^H[R_k]^{-1}[S_k]) \tag{2.7.21}$$

When $[M]$ is calculated it can be used to find $[\theta_d]$ as given by Equation 2.7.13:

$$[\theta_d] = \begin{bmatrix} -[M(1:n,1:n)]^{-1}[M(1:n,n+1)] \\ 1 \end{bmatrix} \tag{2.7.22}$$

Where $[M(1:n,1:n)]$ is the submatrix of $[M]$ made by the first n rows and n columns. And $[M(1:n,n+1)]$ is the submatrix of $[M]$ made by the first n rows and the last column since $[M]$ has dimensions $(n+1)*(n+1)$.

Once $[\theta_d]$ is known, $[z_f]$ can be found from Equation 2.7.12 by solving the roots. Then the natural frequencies and damping ratio can be found by first using Equation 2.6.17 , then Equation 2.6.18, Equation 2.6.19 and Equation 2.6.20.

2.7.4 Poly-Reference Least Squares Complex Frequency Method

Poly-reference least squares complex frequency method (pLSCF), also called PolyMax, works in much the same way as LSCF, but is based on RMFD as shown in Section 2.4.3 and Equation 2.4.17. It improves the method especially concerning the identification of closely spaced modes [16]. The input is the same as for LSCF.

Initial Calculations

For each frequency line $f = 1, 2, ..., N_f$ the PSD and CPSD vector is:

$$[S_y(\omega_f)] = [B(z_f,[\theta])][A(z_f,\theta]^{-1} \tag{2.7.23}$$

Where the numerator matrix polynomial $[B_o(z_f,[\theta])]$ for each measurement channel $o = 1, 2, ...l$ is:

$$[B_o(z_f, [\theta])] = \sum_{j=1}^{n} [B_{o,j}] z_f^j \tag{2.7.24}$$

And the denominator matrix polynomial $[A(z_f, [\Phi]]$ is:

$$[A(z_f, [\theta]] = \sum_{j=0}^{n} [A_j] z_f^j \tag{2.7.25}$$

The coefficients $[A_j]$ is gathered in α:

$$[\alpha] = \begin{bmatrix} [A_0] \\ [A_1] \\ [A_2] \\ \vdots \\ [A_n] \end{bmatrix} \tag{2.7.26}$$

As for LSCF the last coefficient $[A_n]$ is set equal to the identity matrix with dimensions $l * l$.

z_f is chosen as for LSCF:

$$z_f^j = e^{i\omega_f \Delta t} \tag{2.7.27}$$

Calculations

The calculations are repeated for $n = 1, 2, ...n_{max}$ and plotted in a stabilization diagram. Γ_o and Υ_o are calculated for each o [10]:

$$[\Gamma_o] = \begin{bmatrix} z_1^0 & z_1^1 & \cdots & z_1^n \\ z_2^0 & z_2^1 & \cdots & z_2^n \\ \vdots & \vdots & \ddots & \vdots \\ z_{N_f}^0 & z_{N_f}^1 & \cdots & z_{N_f}^n \end{bmatrix} \tag{2.7.28}$$

$$[\Upsilon_o] = \begin{bmatrix} -\begin{pmatrix} z_1^0 & z_1^1 & \cdots & z_1^n \end{pmatrix} \otimes S_o(\omega_1) \\ -\begin{pmatrix} z_2^0 & z_2^1 & \cdots & z_2^n \end{pmatrix} \otimes S_o(\omega_2) \\ \vdots \\ -\begin{pmatrix} z_{N_f}^0 & z_{N_f}^1 & \cdots & z_{N_f}^n \end{pmatrix} \otimes S_o(\omega_{N_f}) \end{bmatrix} \tag{2.7.29}$$

Where z_f^j is as in Equation 2.7.27 and $S_o(\omega_f)$ as in Equation 2.7.23. $\otimes$ denotes the Kronecker product. Since $S_o(\omega_f)$ have dimensions $1 * l$, Υ_o will have dimensions $N_f * (n+1)l$ while Γ_o have dimensions $N_f * (n+1)$.

As for LSCF some matrices are defined. It can be shown [17] that they can be substituted by its real part, which reduces computational cost:

$$[R_o] = Re([\Gamma_o]^H[\Gamma_o]) \tag{2.7.30}$$

$$[S_o] = Re([\Gamma_o]^H[\Upsilon_o]) \tag{2.7.31}$$

$$[T_o] = Re([\Upsilon_o]^H[\Upsilon_o]) \tag{2.7.32}$$

And M is calculated [17]:

$$[M] = 2\sum_{o=1}^{l}([T_o] - [S_o]^H[R_o]^{-1}[S_o]) \tag{2.7.33}$$

Since [18]:

$$[M][\alpha] \approx [0] \tag{2.7.34}$$

$[\alpha]$ can now be calculated:

$$[\alpha] = \begin{bmatrix} -[M(1:nl, 1:nl)]^{-1}[M(1:nl, (nl+1):(n+1)l)] \\ I_l \end{bmatrix} \tag{2.7.35}$$

Where $[M(1:nl, 1:nl)]$ is the submatrix of $[M]$ made by the first $n*l$ rows and $n*l$ columns. And $[M(1:nl, (nl+1):(n+1)l)]$ is the submatrix of $[M]$ made by the first $n*l$ rows and the last l columns.

Now that $[\alpha]$ is calculated, the roots z_f of the denominator polynomial $[A(z_f, [\Phi]]$ can be found as the eigenvalues of the following companion matrix [9]:

$$[A] = \begin{bmatrix} [0] & [I] & [0] & \cdots & [0] & [0] \\ [0] & [0] & [I] & \cdots & [0] & [0] \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ [0] & [0] & [0] & \cdots & [0] & [I] \\ -[A_0]^T & -[A_1]^T & -[A_2]^T & \cdots & -[A_{n-2}]^T & -[A_{n-1}]^T \end{bmatrix} \tag{2.7.36}$$

When the eigenvalues have been found they need to be converted to continuous time as seen in Equation 2.6.17 and then the modal parameters can be extracted as shown in Equation 2.6.18, Equation 2.6.19 and Equation 2.6.20.

2.8 Stabilization Diagram

When the modal parameters are found the stabilization diagram can be constructed. Here the natural frequencies obtained are plotted for each order. Since the order of the system is overestimated the plot will contain noise modes and mathematical modes. The noise modes are due to physical reasons, whereas the mathematical modes are created to ensure mathematical description of the measured data. The aim of the stabilization diagram is to separate the physical poles from the mathematical poles. The mathematical poles tends to be more scattered and typically do not stabilize [18]. Therefore physical modes can be determined from an alignment of stable poles. To find these alignments you need to separate the stable poles from the unstable ones. This is based on the comparison of the poles associated to a given model order with those obtained from a one-order lower model [18].

The natural frequencies and damping ratio of poles from two orders are compared:

$$\frac{|f(n-1) - f(n)|}{f(n-1)} < x \tag{2.8.1}$$

$$\frac{|\zeta(n-1) - \zeta(n)|}{\zeta(n-1)} < y \tag{2.8.2}$$

Only the poles that fulfill a stabilization criteria defined by the user (x and y) are labeled as stable. The size of these depends on several factors, among them the structure complexity and the measurements accuracy. For natural frequency the values should coincide well and a low stability requirement should be used. However for damping ratios the values can vary more. Especially for lightly damped modes where their percentage variation could be relatively large. The value should initially be chosen relatively small and then increased if needed.

Chapter 3

Case: Shear Frame

3.1 System Description

Two different shear frames will be analyzed, one with low damping and one with high damping. This is done to check that the methods work probably and how the different methods react to damping.

3.1.1 Low Damping

The different MATLAB functions will be tested on this shear frame. The shear frame is a 10 story high building with one degree of freedom (DOF) per story as shown in Figure 3.1. Each story have the same stiffness k, damping c and the same lumped mass m. The values are chosen as:

$$k = 50\frac{N}{m} \qquad c = 0.1\frac{Ns}{m} \qquad m = 5kg$$

To construct this problem two functions created by Knut Andreas Kvåle were used: shearFrame.m and newmark.m. shearFrame.m takes k, c and m and the number of story's to establish $[K]$, $[C]$ and $[M]$. Then MATLABs randn.m function is used to create the force vector $[P]$ with white noise. The time step Δt and the total number of samples N is chosen as:

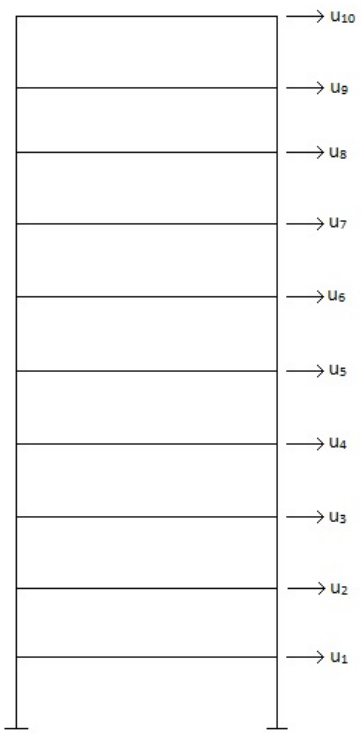

Figure 3.1: Shear frame

$$\Delta t = 0.01s \qquad N = 1000000$$

The starting conditions are taken as:

$$u_0 = 0m \qquad v_0 = 0\frac{m}{s} \qquad a_0 = 0\frac{m}{s^2}$$

$[P]$, $[K]$, $[C]$, $[M]$, Δt, u_o, v_0 and a_0 are used as input into newmark.m to get the time series for displacement u, velocity v and acceleration a.

3.1.2 High Damping

Everything is exactly the same in this analysis, except the damping which is:

$$c = 3.5\frac{Ns}{m}$$

3.2 Analysis

MATLABs function polyeig.m is used on $[K]$, $[C]$ and $[M]$ to find the exact values for natural frequencies, damping ratios and mode shapes. They are compared to the values found for each method in the following. Mode shapes are only estimated for Cov-SSI with low damping as it is space-demanding.

In StabDiag.m the poles that are found stable is given a blue color and the unstable poles are given red. For natural frequencies a circle is used and for damping ratio a cross is used. The following criteria were used:

$$\frac{|f(n-1) - f(n)|}{f(n-1)} < 0.005$$

$$\frac{|\zeta(n-1) - \zeta(n)|}{\zeta(n-1)} < 0.01$$

3.2.1 Covariance-Driven Stochastic Subspace Identification

First the analysis is carried out using Cov-SSI with SSICov.m. The input used was the displacement u time series, $n_{max} = 50$ and the magnitude of block rows = 1 for this simple analysis.

Low Damping

The stabilization diagram is shown in Figure 3.2. The stable modes are very clear and the natural frequencies can be picked easily. The natural frequencies are gathered in Table 3.1. To get a good estimate of the damping ratios, they are plotted in Figure 3.3. The damping ratio of the first nine modes are pretty stable, but the tenth mode did not stabilize. A mean value for the damping ratio has been chosen and gathered in Table 3.2. The mode shapes are found during the analysis and then organized and plotted using ModeShapes.m. Figure 3.6 shows the modes found, which can be compared to the exact modes shown in Figure 3.7.

High Damping

For the case with high damping the stabilization diagram is shown in Figure 3.4. The stable modes are very clear for the high damping as well and the natural frequencies are found, then gathered in Table 3.1. As for the low damping the damping ratios are plotted in Figure 3.5. The plot looks pretty similar to that of the low damping case. A mean value for the damping ratio has been chosen and gathered in Table 3.3.

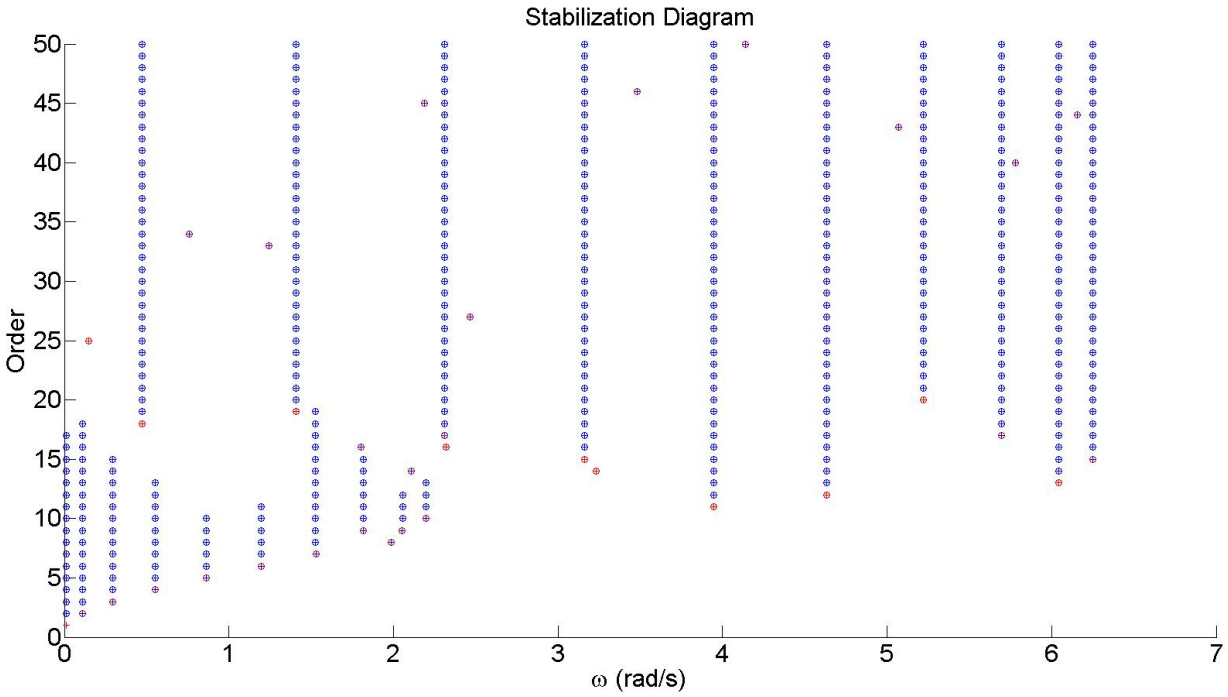

Figure 3.2: Stabilization diagram with Cov-SSI low damping

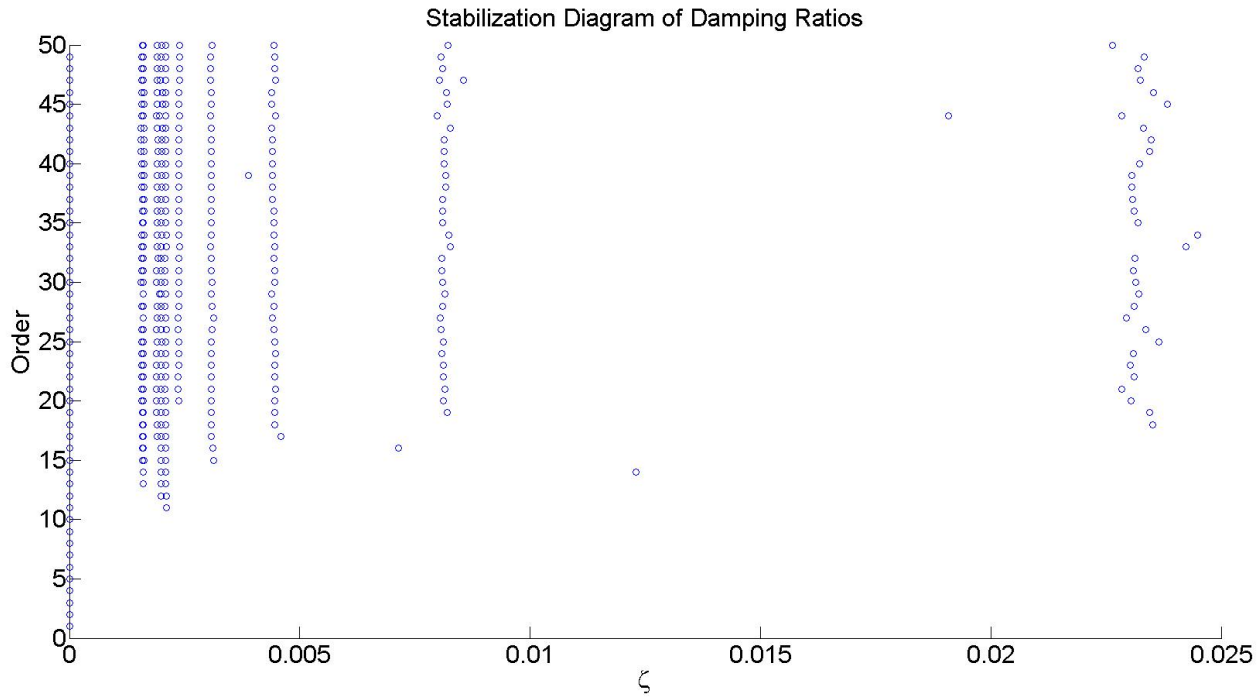

Figure 3.3: Damping ratios with Cov-SSI low damping

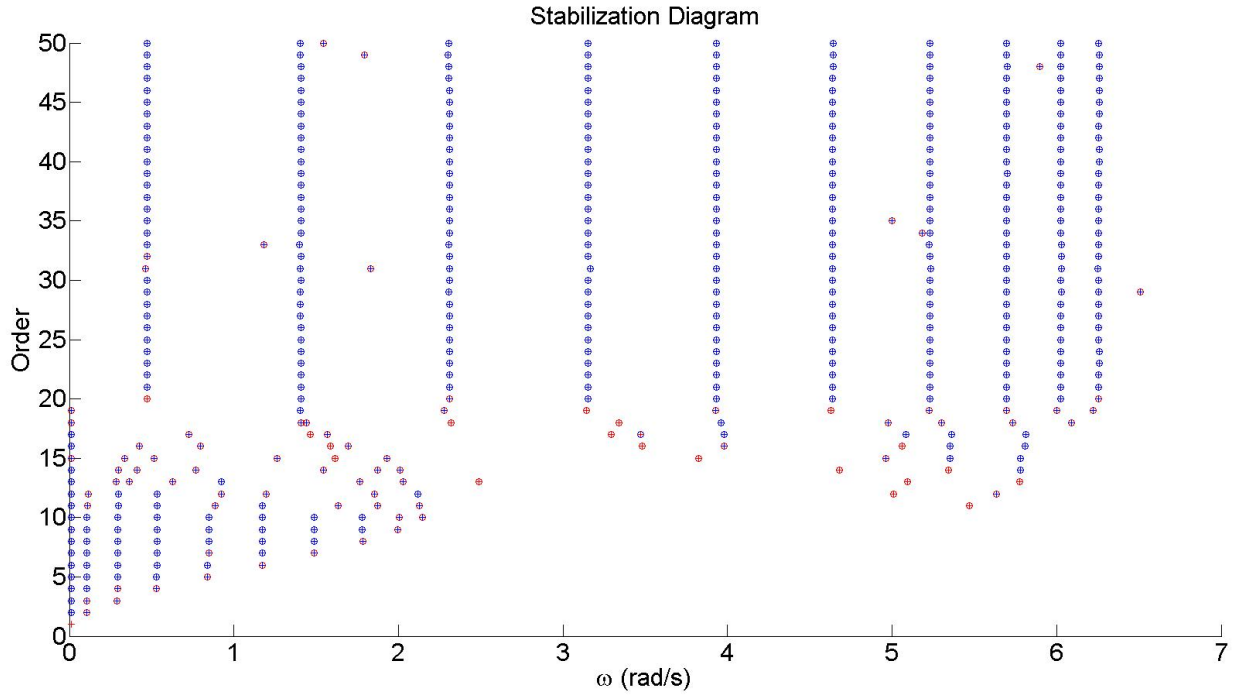

Figure 3.4: Stabilization diagram with Cov-SSI high damping

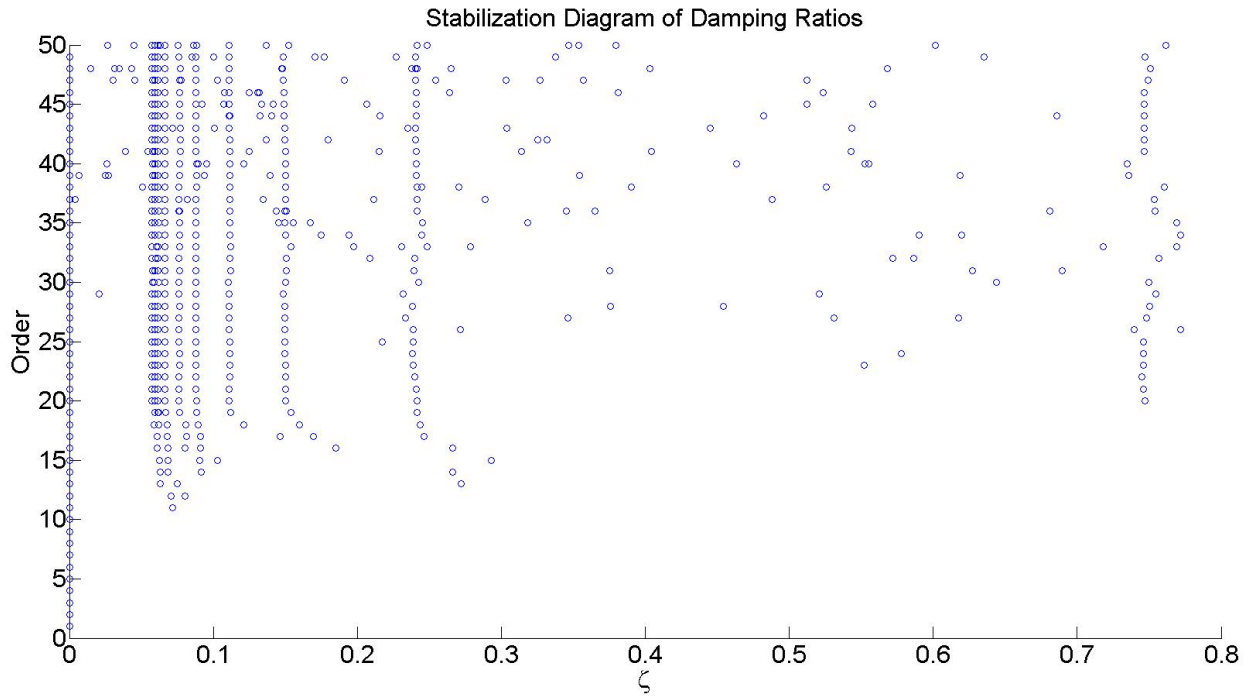

Figure 3.5: Damping ratios with Cov-SSI high damping

Table 3.1: Natural frequencies for Cov-SSI

Mode number n	Exact $\omega_n(\frac{rad}{s})$	Cov-SSI (low damping) $\omega_n(\frac{rad}{s})$	Cov-SSI (high damping) $\omega_n(\frac{rad}{s})$
1	0.473	+0.002	0
2	1.407	0	+.0.001
3	2.311	0	+0.003
4	3.162	0	0
5	3.943	+0.001	+0.001
6	4.636	-0.001	0
7	5.226	-0.001	+0.001
8	5.698	+0.001	+0.001
9	6.044	-0.002	-0.001
10	6.254	+0.001	0

Table 3.2: Damping ratios for Cov-SSI low damping

Mode number n	Exact ζ	Cov-SSI ζ
1	0.02116	+0.00014
2	0.00711	-0.00139
3	0.00433	-0.00015
4	0.00316	+0.00004
5	0.00254	-0.00004
6	0.00216	0
7	0.00191	-0.00016
8	0.00175	-0.00005
9	0.00165	-0.00003
10	0.00160	0

Table 3.3: Damping ratios for Cov-SSI high damping

Mode number n	Exact ζ	Cov-SSI ζ
1	0.7405	+0.0019
2	0.2487	-0.0029
3	0.1515	+0.0048
4	0.1107	-0.0012
5	0.0888	+0.0009
6	0.0755	+0.0003
7	0.0670	+0.0004
8	0.0614	-0.0011
9	0.0579	+0.0013
10	0.0560	+0.0021

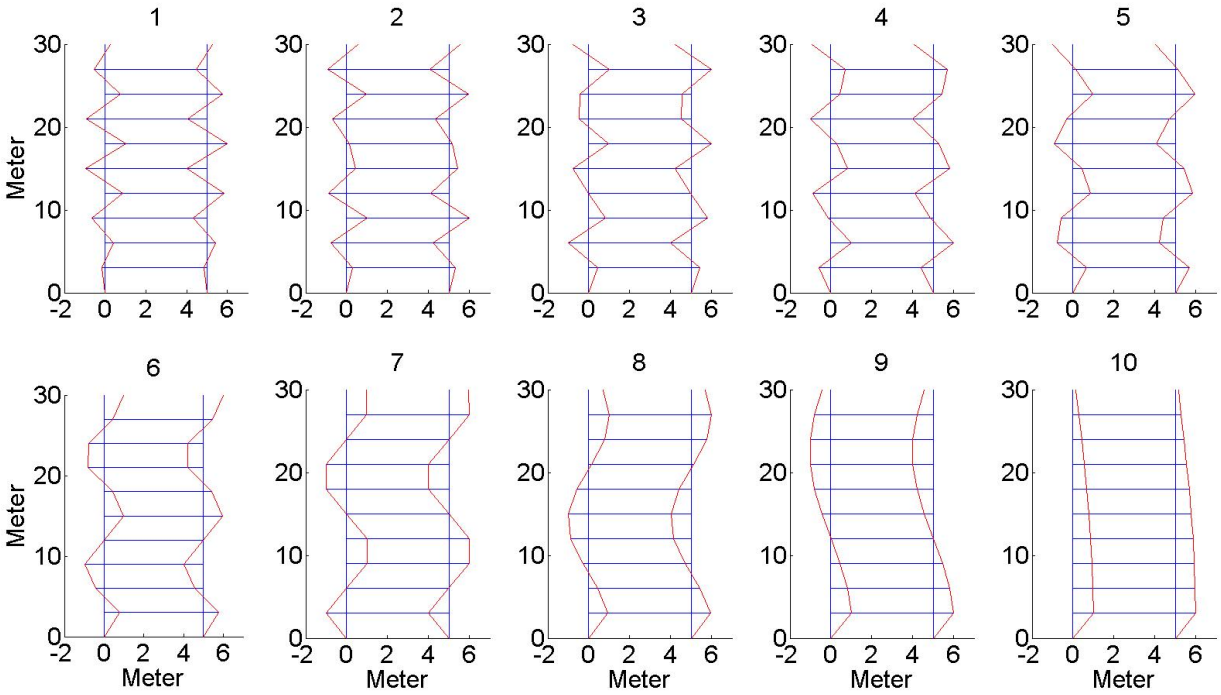

Figure 3.6: Mode shapes found with Cov-SSI

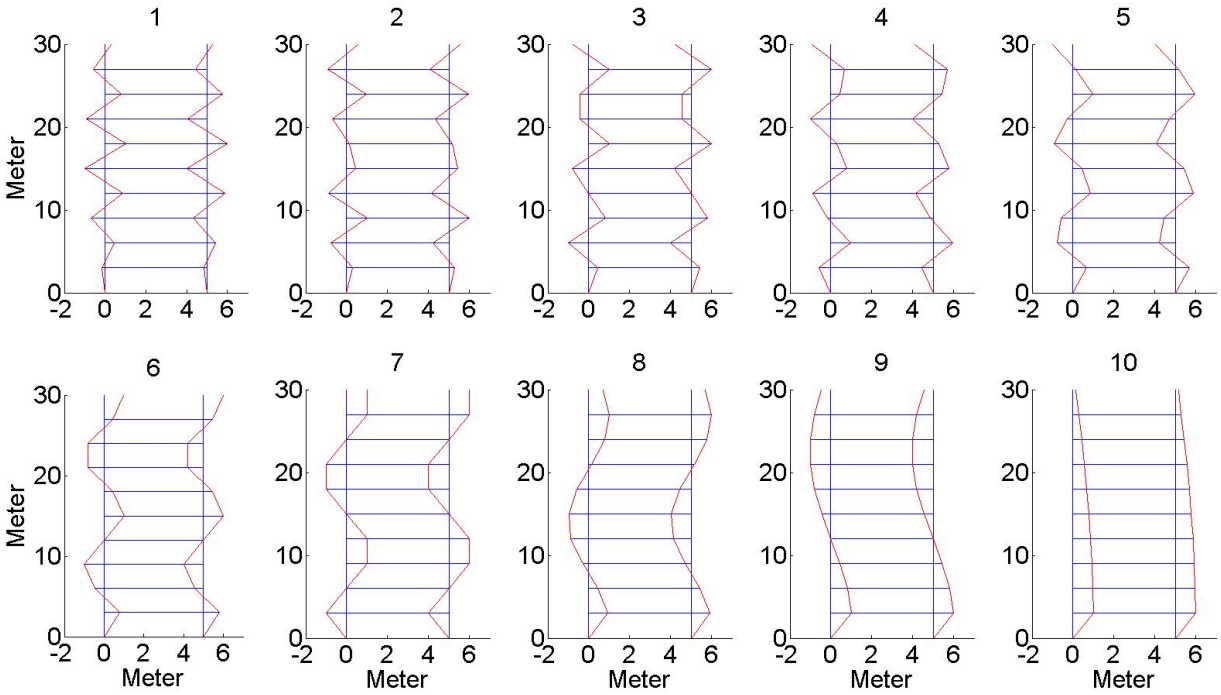

Figure 3.7: Exact mode shapes

Results

The natural frequencies found by Cov-SSI are very close to the exact values for both low and high damping. For the case with the low damping most damping ratios are accurate, except for the second mode. For the high damping case all the damping ratios are accurate. The mode shapes found are almost identical with the exact values. The numerical error is less then 1% and the difference can hardly be seen in the figures.

3.2.2 Data-Driven Stochastic Subspace Identification

Then the analysis is carried out using DD-SSI with SSIData.m. The input used was the displacement u time series and $n_{max} = 50$.

Low Damping

The stabilization diagram is shown in Figure 3.8. The stable modes are very clear and the natural frequencies can be picked easily and gathered in Table 3.4. To get a good estimate of the damping ratios, they are plotted in Figure 3.9. As for Cov-SSI an average value is chosen and gathered in Table 3.5.

High Damping

The stabilization diagram is shown in Figure 3.10. They are not as stable as for the low damping case, but are still easy to pick. When that is done they are gathered in Table 3.4. The damping ratios are plotted in Figure 3.11 and found in the same way as for low damping, then they are gathered in Table 3.6.

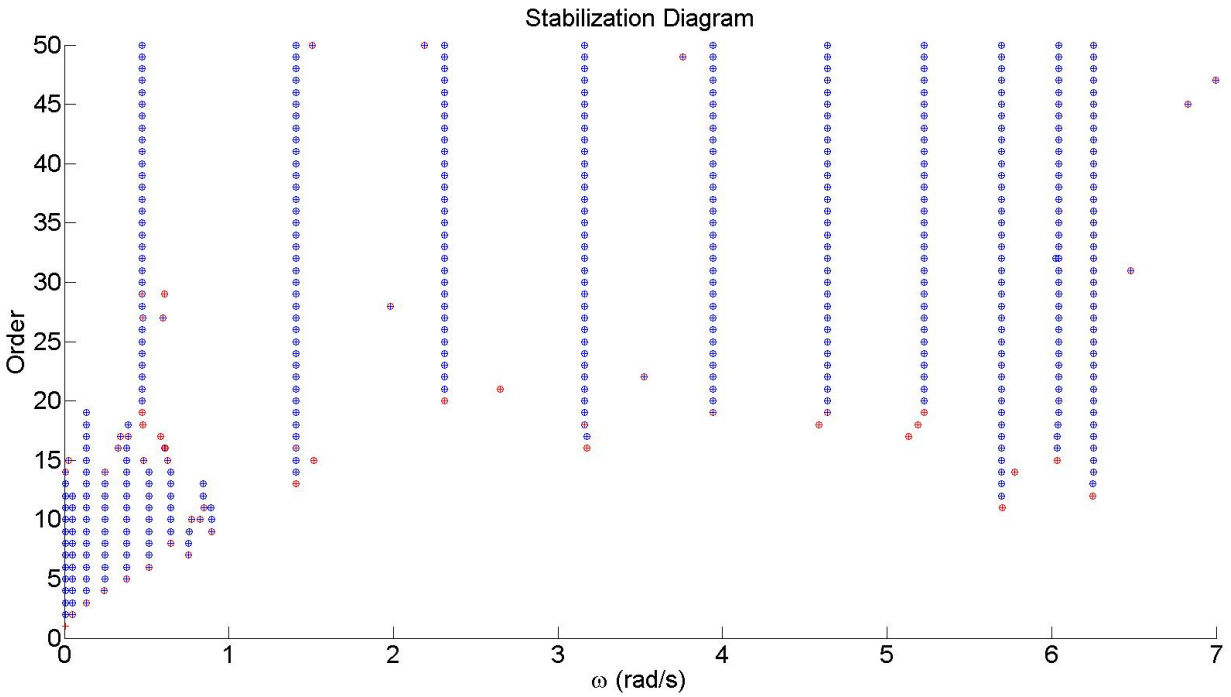

Figure 3.8: Stabilization diagram with DD-SSI low damping

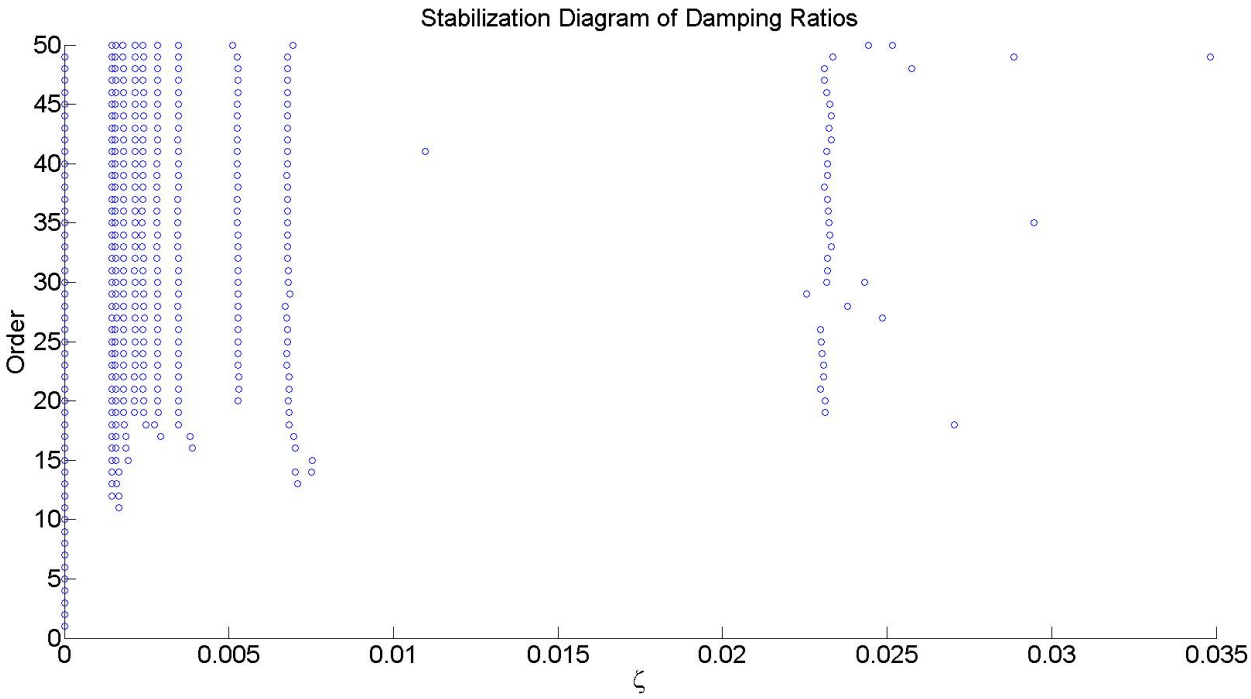

Figure 3.9: Damping ratios with DD-SSI low damping

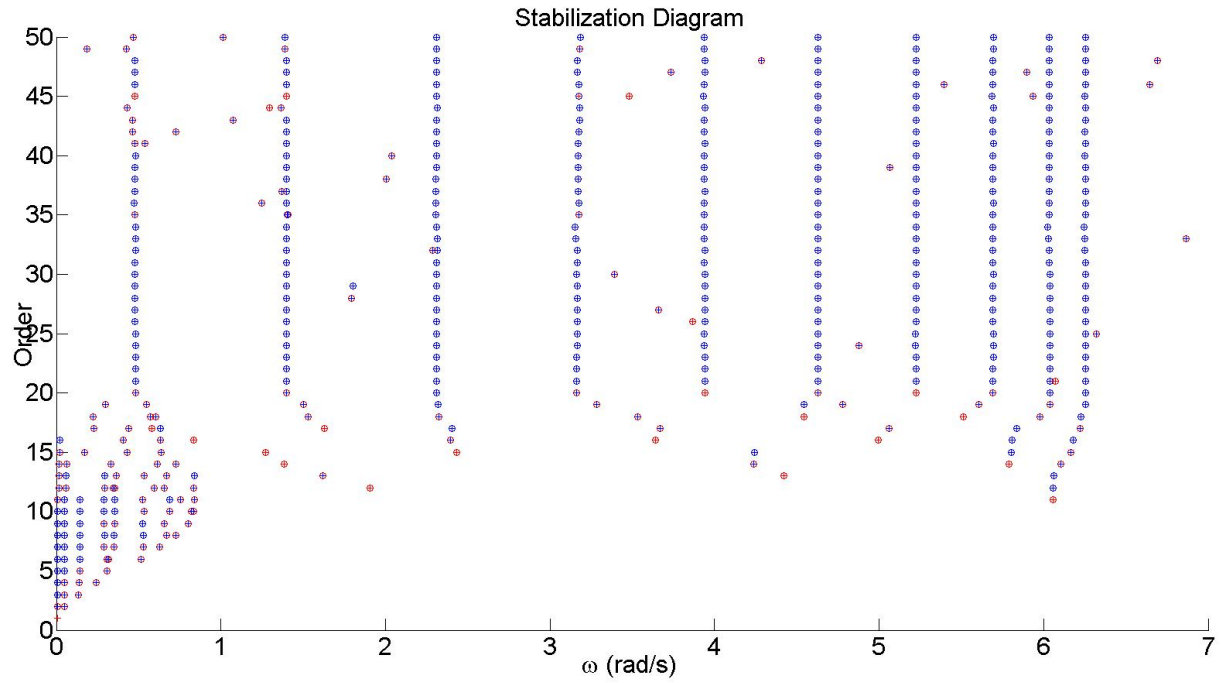

Figure 3.10: Stabilization diagram with DD-SSI high damping

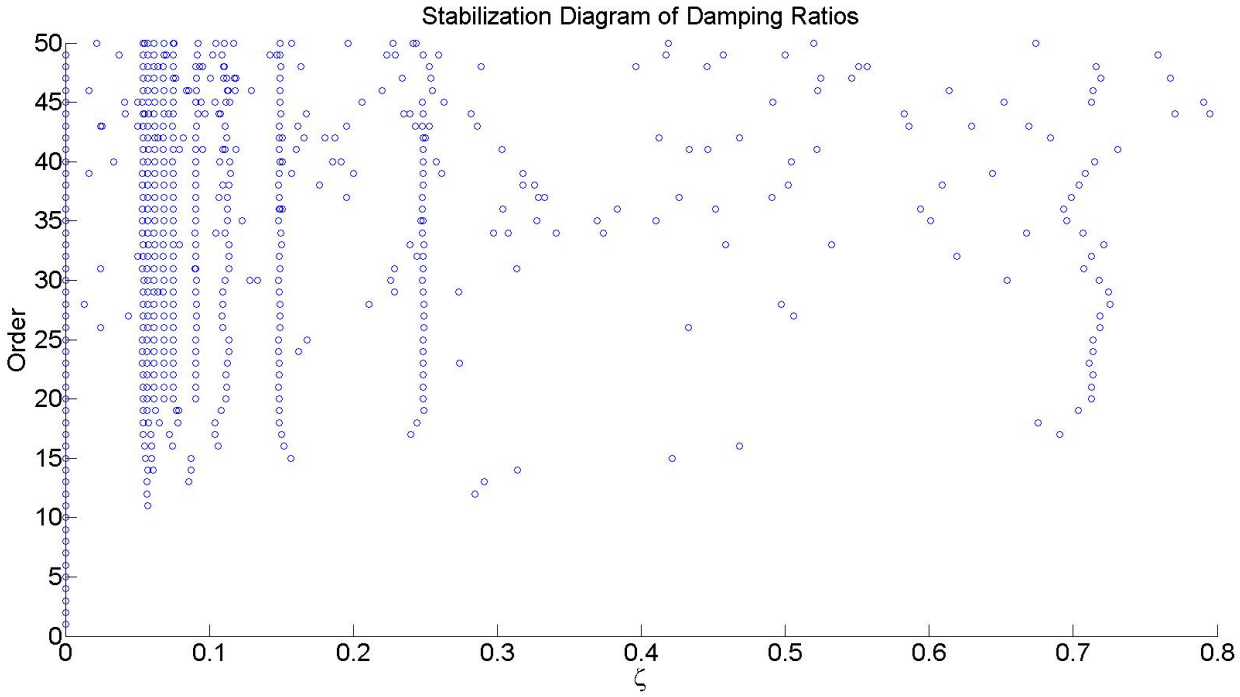

Figure 3.11: Damping ratios with DD-SSI high damping

Table 3.4: Natural frequencies for DD-SSI

Mode number n	Exact $\omega_n(\frac{rad}{s})$	DD-SSI(low damping) $\omega_n(\frac{rad}{s})$	DD-SSI(high damping) $\omega_n(\frac{rad}{s})$
1	0.473	-0.004	+0.006
2	1.407	+0.001	-0.008
3	2.311	-0.002	-0.002
4	3.162	+0.003	+0.006
5	3.943	+0.001	-0.002
6	4.636	+0.001	-0.008
7	5.226	+0.001	-0.002
8	5.698	+0.001	-0.005
9	6.044	-0.001	-0.007
10	6.254	+0.004	-0.001

Table 3.5: Damping ratios for DD-SSI low damping

Mode number n	Exact ζ	DD-SSI ζ
1	0.02116	+0.01079
2	0.00711	+0.00341
3	0.00433	+0.00260
4	0.00316	+0.00088
5	0.00254	+0.00007
6	0.00216	-0.00023
7	0.00191	-0.00018
8	0.00175	-0.00005
9	0.00165	-0.00028
10	0.00160	-0.00069

Table 3.6: Damping ratios for DD-SSI high damping

Mode number n	Exact ζ	DD-SSI ζ
1	0.7405	-0.0266
2	0.2487	0
3	0.1515	-0.0027
4	0.1107	+0.0020
5	0.0888	+.0.0016
6	0.0755	-0.0007
7	0.0670	+0.0011
8	0.0614	+0.0001
9	0.0579	-0.0010
10	0.0560	-0.0020

Results

The natural frequencies found by DD-SSI are very accurate for low damping. For high damping they are a bit less accurate, but still good. The damping ratios extracted are not that good for low damping, but pretty accurate for high damping, except for the first mode.

3.2.3 Second Order Blind Identification

For the SOBI algorithm sobi.m was used with the displacement u time series to find the mixing matrix and the sources. And then sobifind.m was used to estimate the natural frequencies, which then were gathered in Table 3.7.

Table 3.7: Natural frequencies for SOBI

Mode number n	Exact $\omega_n(\frac{rad}{s})$	SOBI(low damping) $\omega_n(\frac{rad}{s})$	SOBI(high damping) $\omega_n(\frac{rad}{s})$
1	0.473	-0.004	-0.325
2	1.407	0	+0.037
3	2.311	+0.005	-0.143
4	3.162	0	-0.226
5	3.943	-0.003	0.114
6	4.636	-0.003	0.110
7	5.226	-0.001	+0.024
8	5.698	-0.004	+0.137
9	6.044	-0.009	+0.104
10	6.254	-0.002	-0.026

Results

All the natural frequencies where found. For low damping they are close to the accurate values. However for high damping the SOBI algorithm is not accurate at all. One of the reasons that the values you find using SOBI for high damping are not accurate is because the values you find are not really the undamped natural frequencies. The reason for this is that a SDOF estimator have been used which is based on FFT as mentioned in Chapter 2. The values of the peaks will not be the undamped natural frequencies and some sort of transfer function should be used.

3.2.4 Peak Picking

Low Damping

Using mpsd.m with the velocity v time series, Hanning window with 65536 number of samples in FFT and 50% overlap gave the PSDs and CPSDs. PSDplot.m was then used on all the measurement channels to plot the PSDs and CPSDs. Only the first four measurement channels are shown in Figure 3.12 for convenience. PSDplot2.m was also tested on all the measurement channels to plot the PSDs and the coherence functions as shown in Figure 3.13. From both plots it is easy to pick the correct modes. The coherence plots are the easiest way to separate real eigenvalues and peaks due to disturbance and is therefore prioritized for the rest of this thesis. The natural frequencies found by studying the plots for all the measurement channels are gathered in Table 3.8.

High Damping

For the case with high damping, mpsd.m was used with the velocity v time series, Hanning window with 16384 number of samples in FFT and 0% overlap. The PSDs and coherence function is plotted using PSDplot2.m. The plot of the first 4 measurement channels is shown in Figure 3.14. It is at once obvious that the peak picking method experience trouble with high damping. That is also the reason the number of samples in FFT had to be reduced compared to the case with low damping. Picking the modes is much harder, and not all the modes can be clearly found. Studying all the measurement channels and coherence functions gives some values which are gathered in Table 3.8.

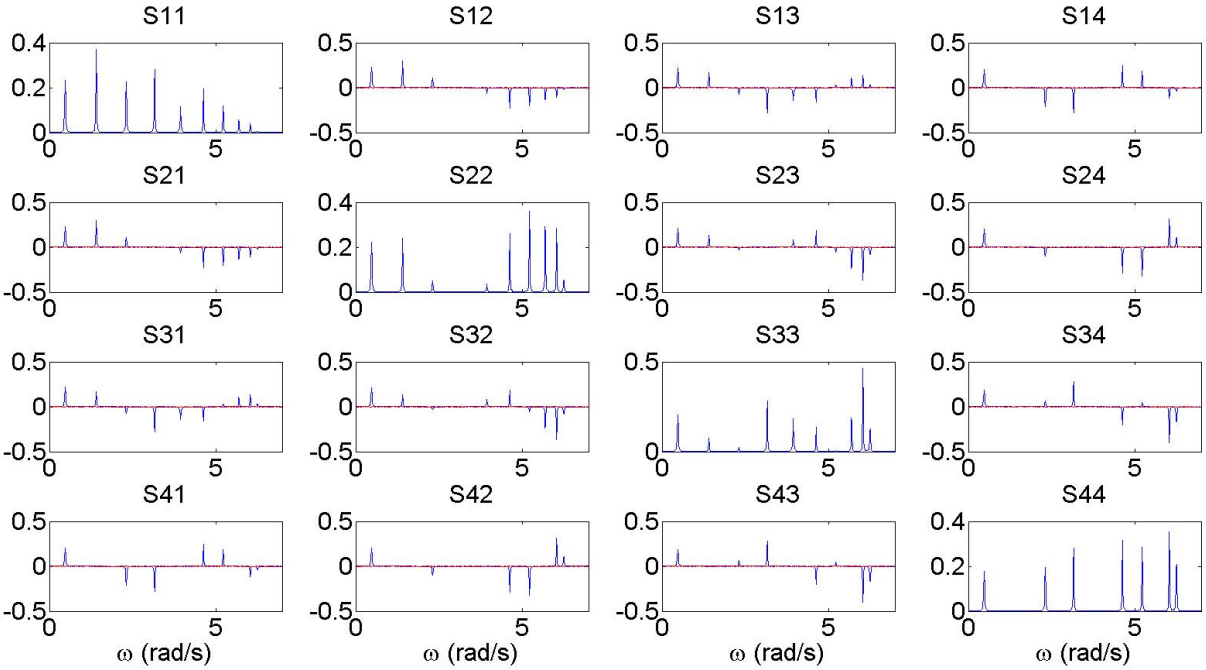

Figure 3.12: PSD and CPSD for the first 4 measurement channels low damping

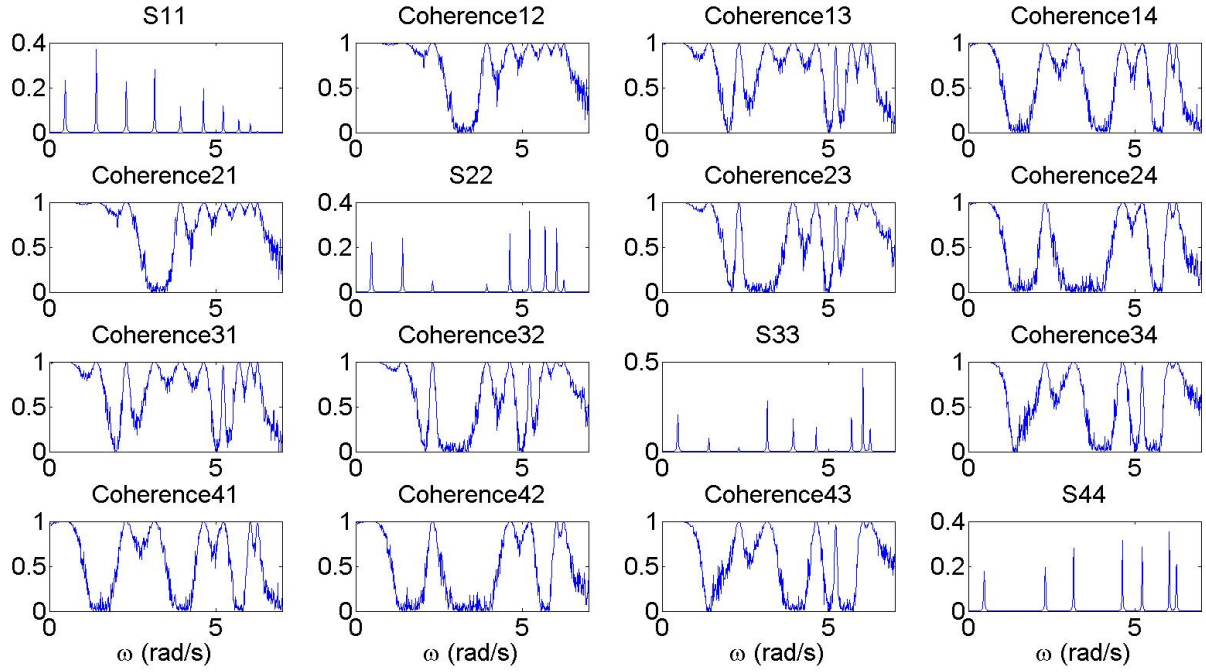

Figure 3.13: PSD and coherence for the first 4 measurement channels low damping

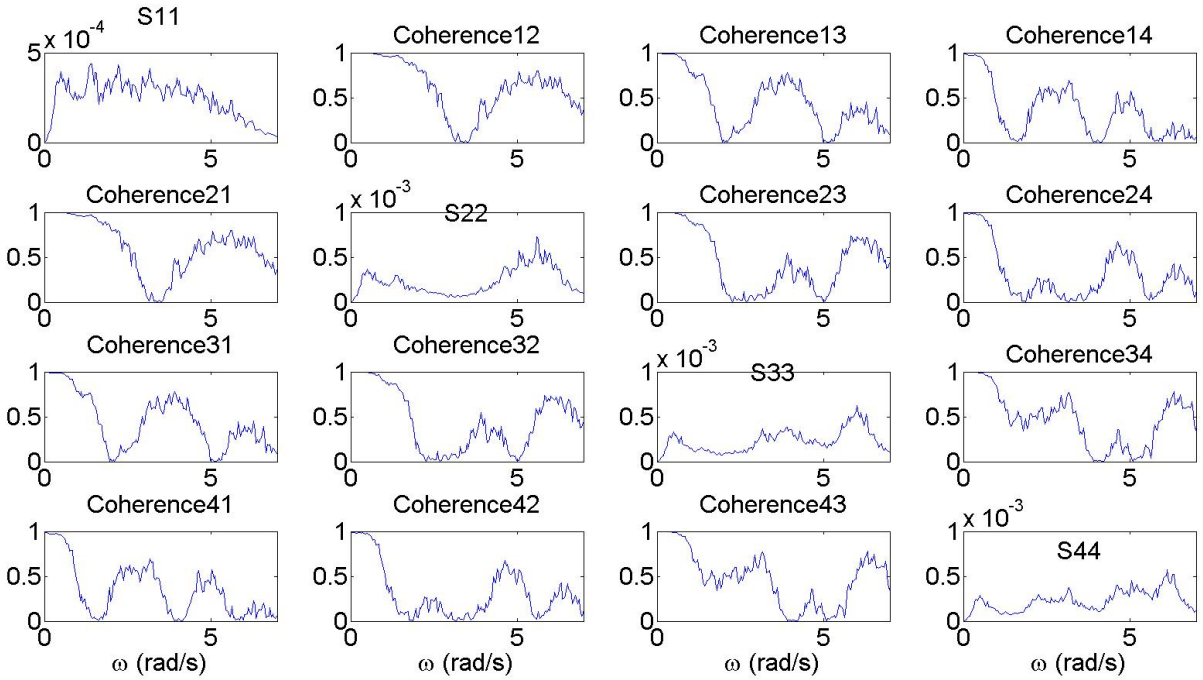

Figure 3.14: PSD and coherence for the first 4 measurement channels high damping

Table 3.8: Natural frequencies for peak picking

Mode number n	Exact $\omega_n(\frac{rad}{s})$	Peak picking(low damping) $\omega_n(\frac{rad}{s})$	Peak picking(high damping) $\omega_n(\frac{rad}{s})$
1	0.473	-0.003	-0.089
2	1.407	+0.002	+0.012
3	2.311	0	-0.048
4	3.162	+0.002	+0.021
5	3.943	-0.003	-0.146
6	4.636	+0.004	-0.034
7	5.226	-0.001	-0.010
8	5.698	+0.001	+0.169
9	6.044	+0.001	+0.054
10	6.254	-0.003	+0.035

Results

From Figure 3.13 and Table 3.8 you can see that peak picking is an accurate and efficient method for low damping. But for high damping it is clear that peak picking is not sufficient. If there was no other results to compare it and the number of modes were unknown this

method would be useless. As for SOBI, peak picking is based on FFT and therefore a transfer function should be used to find the actual natural frequencies. Since this is a MDOF problem, this function would also be influenced by the other modes.

3.2.5 Frequency Domain Decomposition

Low Damping

Using mpsd.m with the velocity v time series, Hanning window with 65536 number of samples in FFT and 50% overlap gave the PSDs and CPSDs. They were used as input in FDD.m to get the singular values. Only the three highest singular values are plotted in Figure 3.15. It is easy to pick the correct natural frequencies and they are gathered in Table 3.9.

High Damping

For the case with high damping, mpsd.m was used with the velocity v time series, Hanning window with 8192 number of samples in FFT and 50% overlap. Then the singular value plot looks like Figure B.1. The peaks are almost impossible to pick accurately. Therefore a non-logarithmic plot was used, as shown in Figure 3.16. From this plot values for the natural frequencies were picked and gathered in Table 3.9.

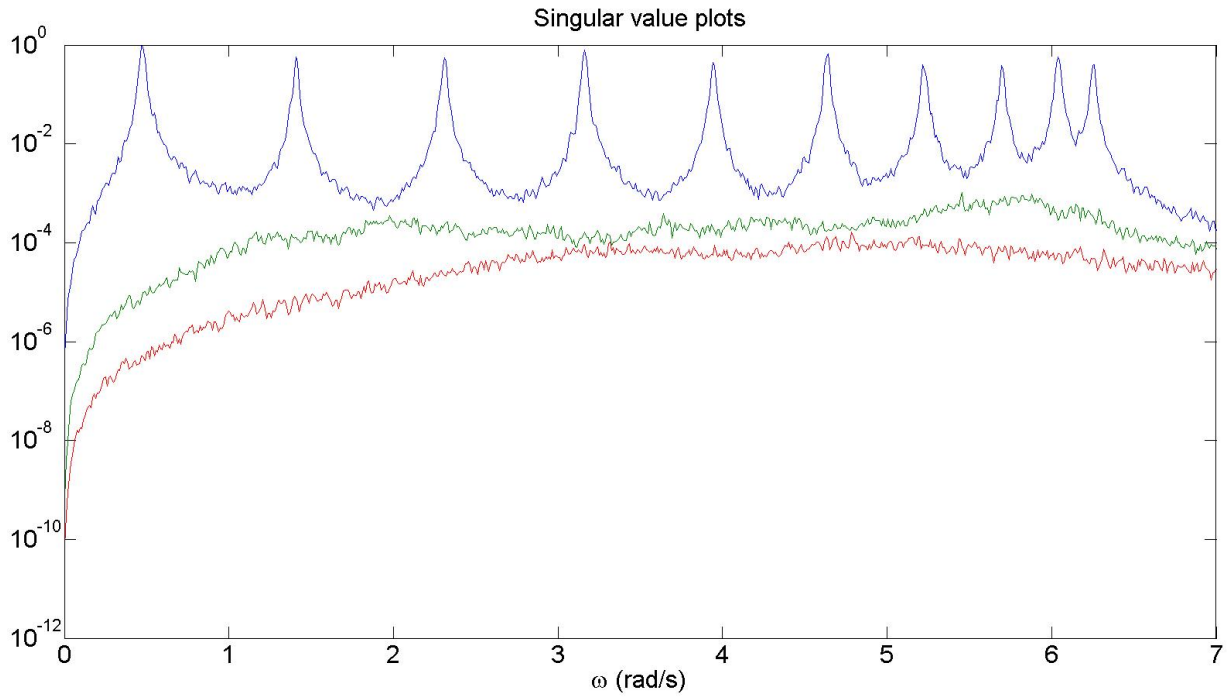

Figure 3.15: Singular value plots low damping

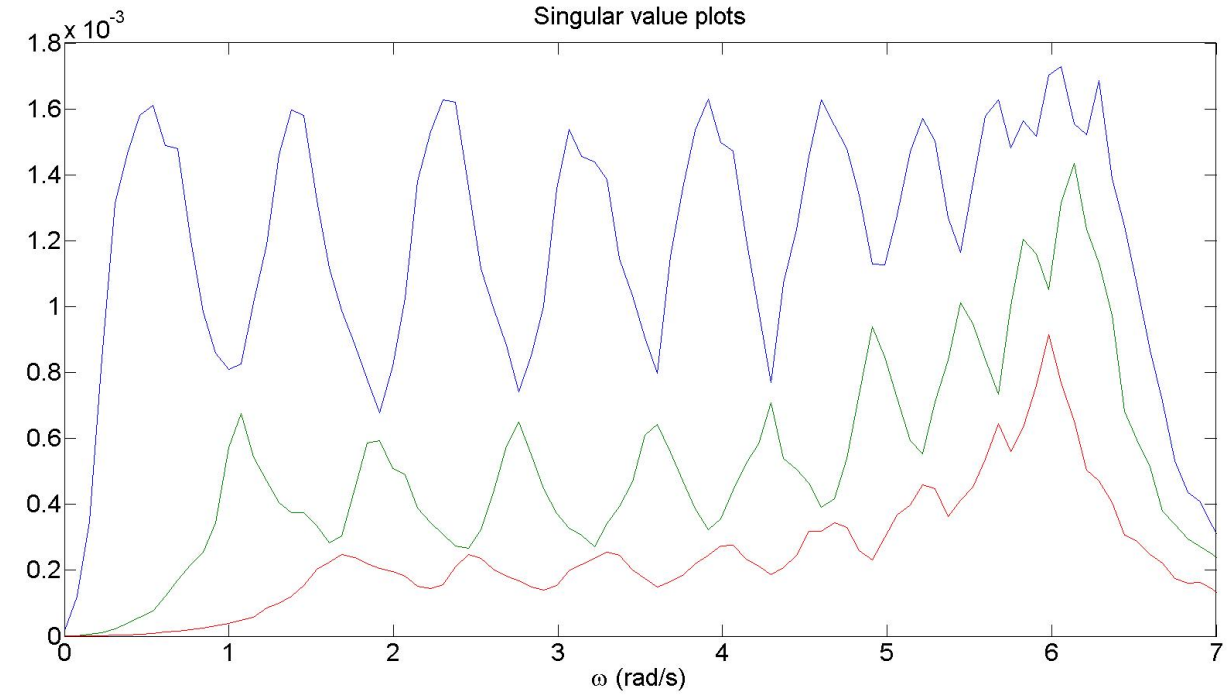

Figure 3.16: Singular value plots high damping non-logarithmic

Table 3.9: Natural frequencies for FDD

Mode number n	Exact $\omega_n(\frac{rad}{s})$	FDD(low damping) $\omega_n(\frac{rad}{s})$	FDD(high damping) $\omega_n(\frac{rad}{s})$
1	0.473	-0.003	+0.064
2	1.407	+0.002	-0.103
3	2.311	0	+0.067
4	3.162	+0.002	-0.094
5	3.943	-0.003	-0.031
6	4.636	+0.004	-0.034
7	5.226	-0.001	+0.066
8	5.698	+0.001	+0.054
9	6.044	+0.001	+0.015
10	6.254	-0.003	+0.035

Results

For the low damping the values are identical to the ones obtained in peak picking. For the case with high damping the FDD performs approximately as peak picking, both are way off. But for FDD it is easy to pick the number of modes, and the results would be much better than peak picking if the exact values were unknown. Here aswell, the values for high damping won't be correct because of the lack of a transfer function as mentioned for peak picking.

3.2.6 Least Squares Complex Frequency Method

Using mpsd.m with the acceleration a time series, Hanning window with 8192 number of samples in FFT and 50% overlap gave the PSDs and CPSDs. They were used as input in LSCF.m along with a n_{max}.

Low Damping

$n_{max} = 400$ gave the stabilization diagram in Figure B.2. Only some modes are found, so $n_{max} = 500$ is tried and the result can be seen in Figure 3.17. The modes that can be found are gathered in Table 3.10.

High Damping

$n_{max} = 500$ gave the stabilization diagram in Figure 3.18. Even fewer of the modes are found for high damping, those that are found are gathered in Table 3.10.

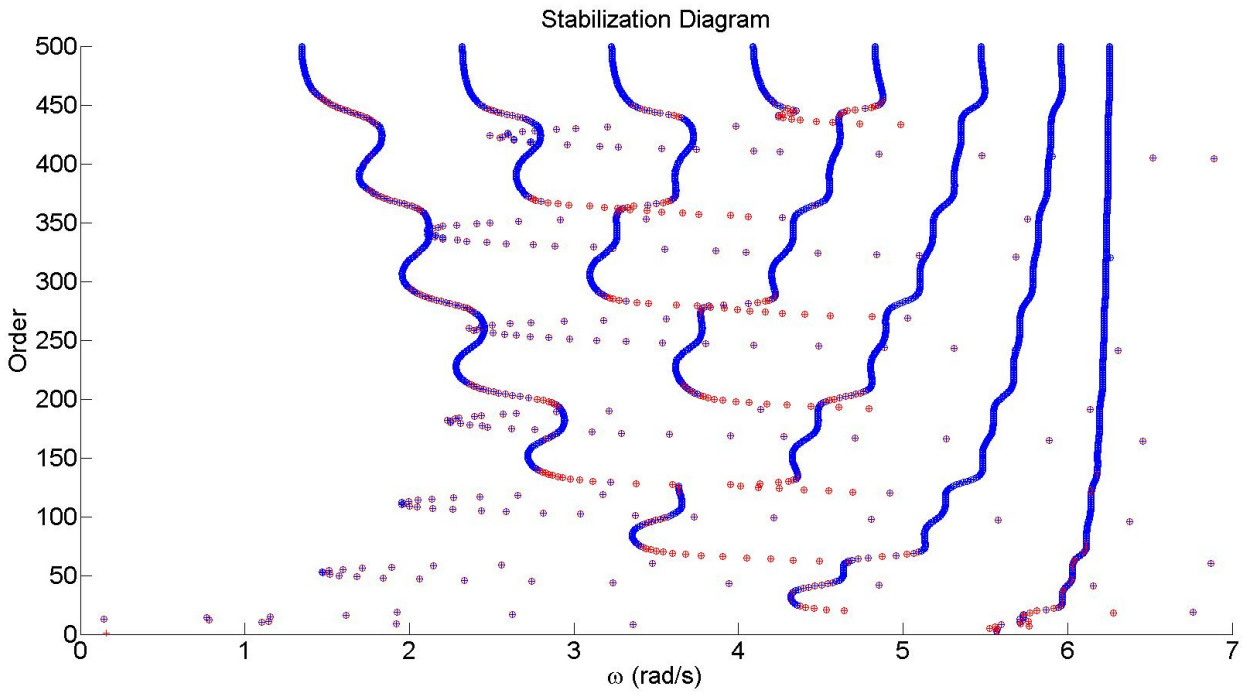

Figure 3.17: Stabilization diagram with LSCF low damping $n_{max} = 500$

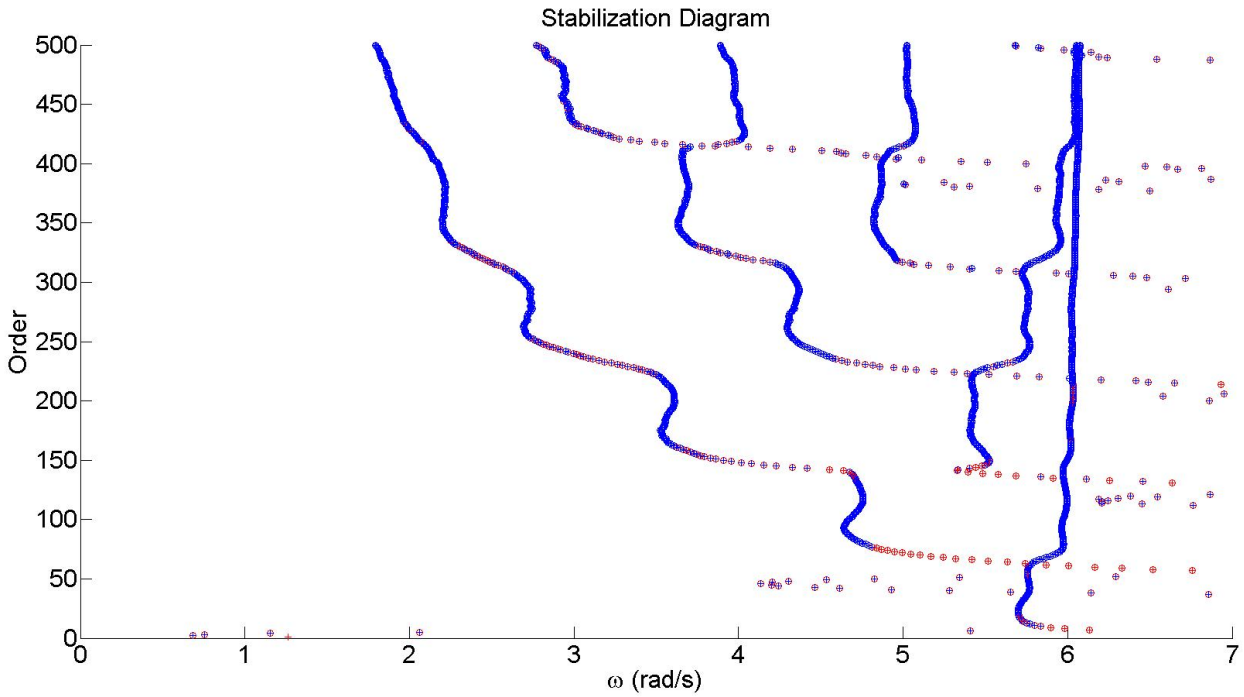

Figure 3.18: Stabilization diagram with LSCF high damping $n_{max} = 500$

Table 3.10: Natural frequencies for LSCF

Mode number n	Exact $\omega_n(\frac{rad}{s})$	LSCF (low damping) $\omega_n(\frac{rad}{s})$	LSCF (high damping) $\omega_n(\frac{rad}{s})$
1	0.473		
2	1.407	-0.069	+0.381
3	2.311	+0.001	
4	3.162	+0.054	-0.209
5	3.943	+0.124	+0.257
6	4.636	+0.159	
7	5.226	+0.213	-0.226
8	5.698		
9	6.044	-0.126	-0.259
10	6.254	-0.009	-0.283

Results

The results from LSCF are not good for either low or high damping. Many modes are missing and most of the estimates are not accurate.

3.2.7 Poly-Reference Least Squares Complex Frequency Method

Using mpsd.m with the acceleration a time series, Hanning window with 16384 number of samples in FFT and 50% overlap gave the PSDs and CPSDs. They were used as input in pLSCF.m along with a n_{max}.

Low Damping

$n_{max} = 50$ gave the stabilization diagram in Figure 3.19. The modes stabilizes very fast and are easy to pick, except for the first which does not stabilize, they are then gathered in Table 3.11. The damping ratios are labeled as unstable for most orders, a plot of the damping ratios in Figure B.3 shows why, they do not stabilize at all. Damping ratios can therefore not be picked.

High Damping

$n_{max} = 50$ gave the stabilization diagram in Figure B.4. The 6 last modes are found, but it doesn't manage to find the modes with low natural frequencies at all. Therefore $n_{max} = 100$ had to be tried as shown in Figure B.5. Still only the last 6 modes are completely stable, but two more modes are starting to look stable. n_{max} is then taken as 150 as shown in Figure 3.20, but still not all the modes are stable. The natural frequencies that are stable are found, and gathered in Table 3.11. A plot of the damping ratios in Figure B.6 shows that the damping ratios doesn't stabilize at all. And they can therefore not be gathered for high damping either.

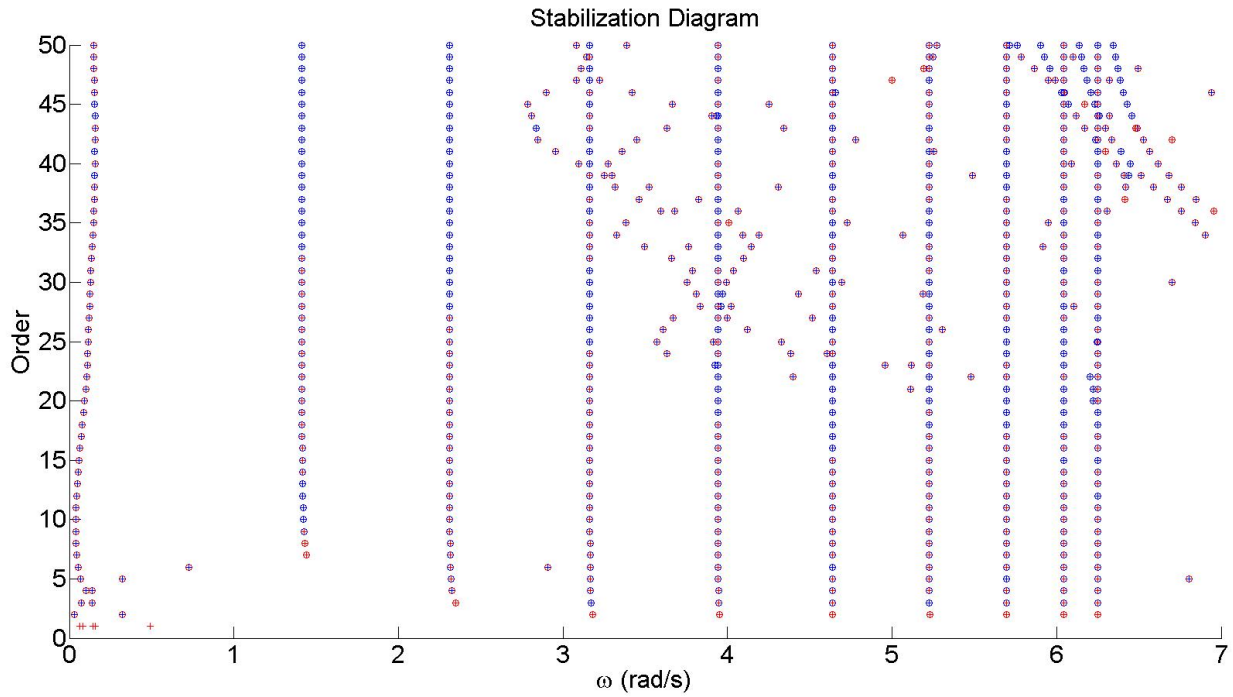

Figure 3.19: Stabilization diagram with pLSCF low damping

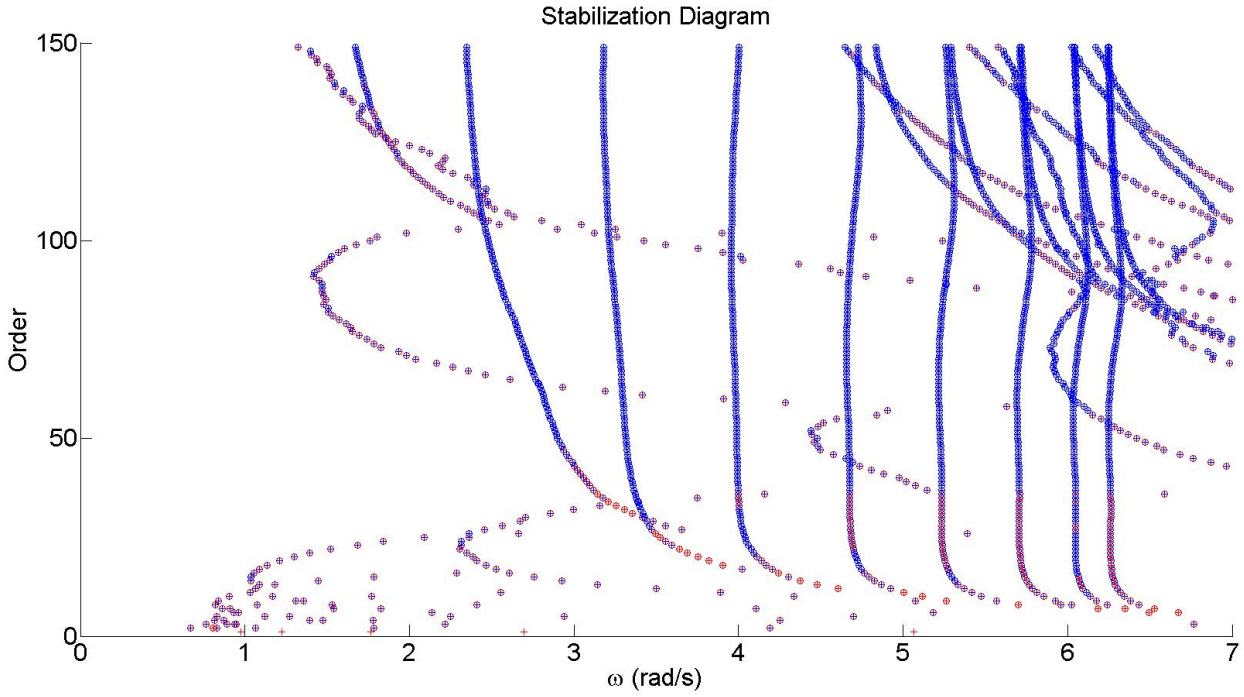

Figure 3.20: Stabilization diagram with pLSCF high damping $n_{max} = 150$

Table 3.11: Natural frequencies for pLSCF

Mode number n	Exact $\omega_n(\frac{rad}{s})$	pLSCF (low damping) $\omega_n(\frac{rad}{s})$	pLSCF (high damping) $\omega_n(\frac{rad}{s})$
1	0.473	-0.304	
2	1.407	+0.003	+0.305
3	2.311	-0.002	+0.054
4	3.162	-0.001	+0.041
5	3.943	-0.001	+0.032
6	4.636	+0.001	+0.034
7	5.226	-0.004	+0.010
8	5.698	-0.001	-0.003
9	6.044	-0.001	-0.002
10	6.254	-0.002	-0.006

Results

The natural frequencies found by pLSCF are very close to the exact values for low damping, except for the first mode. The high damping case required a high n_{max} to find the natural frequencies, but even then they were not particularly accurate. The first mode was not found at all and except for the three last modes the rest were not very close. The damping ratios were not stable for either low or high damping and could not be extracted.

3.3 Discussion of Results

Low Damping

For low damping all the methods managed to find all modes except LSCF. Most methods are accurate, with the exception that pLSCF is not accurate for the first mode. As for the methods that find the natural frequencies SOBI is the most effective. It is computational inexpensive and gives accurate estimations of all the modes. However SOBI has the huge drawback that the method just gives the answer, with no option to verify that the modes are found correctly. As a verification of other results it is extremely efficient and easy to use, but it is not a good tool on its own.

Peak picking and FDD are also very efficient methods, but requires the user to manually find

the values. As shown in the plots that is an easy job for the low damping and the values found are very accurate. Both these methods require some knowledge of modal analysis to choose the different input parameters to find the PSDs and CPSDs. However for an easy case like this most reasonable input parameters would have given sufficient results. One of the advantages of FDD instead of peak picking is the ability to separate closely spaced modes. For this shear frame none of the modes are closely spaced, and therefore FDD doesn't show any particular advantages over peak picking.

Cov-SSI and DD-SSI give very clear stabilization diagrams, both are easy to pick and the values are accurate. They also gives estimates of the damping ratio, as the only methods, and both looks pretty stable. However the values for Cov-SSI is better than DD-SSI. In addition Cov-SSI is computationally more efficient than DD-SSI. Since Cov-SSI proved to be the best overall method it was chosen to extract the mode shapes which where very accurate.

LSCF doesn't manage to find all the modes, and most estimations are not accurate either. The method is also computational expensive. pLSCF find all the modes accurately, except for the first one. The poles look stable already at a very low order, but the calculations are quite expensive per order and not comparable to the SSI methods. The damping ratio estimates are very bad and not pickable.

High Damping

For high damping several of the methods are starting to have problems. SOBI still manages to find all modes and is very computational inefficient, but gives very rough estimates for the natural frequencies. Peak picking manages to find all the modes, but, as previously mentioned, the method is useless on its own as it would be impossible to pick the correct modes if the answers were not already known. The answers obtained are, as for SOBI, only rough estimates. FDD also gives rough estimates, but compared to peak picking it is much better as it manage to find all the modes on it own.

Cov-SSI is still very accurate and gives clear stabilization diagrams. DD-SSI also gives quite

clear stabilization diagrams and good estimates, but not as good as Cov-SSI. For damping ratio both the methods achieve better accuracy than for low damping and they are both equally good. Still Cov-SSI is computationally more efficient than DD-SSI.

LSCF is as bad as for the case with low damping. pLSCF requires a very high n_{max} to find the natural frequencies and still only some of them are found accurately. With the high system order the analysis takes a long time and the results are not impressive. pLSCF doesn't manage to find the damping ratios for high damping either.

Chapter 4

Case: The Hardanger Bridge

4.1 System Description

With the new coastal highway route E39 along the west coast of Norway, running almost 1100 km from Kristiansand in the south to Trondheim in Central Norway, the Norwegian Public Road Administration aims to remove ferry crossings. Today there is several ferry crossings and the new highway will reduce transportation time with 7-9 hours [7]. The Hardanger Bridge crossing the Hardangerfjord was finished in 2013 and be can used as a study example [1]. It is a suspension bridge with 1310 meter main span and 70 meter side span. Tower height is 200 meter and sailing height under the bridge is 55 meter [1]. It connects Ulvik and Bu and eliminate the need for a ferry crossing.

The Hardanger Bridge have been instrumented with accelerometers and anemometers which measures acceleration over time and changes in wind velocity and wind direction, respectively. The position of the 20 accelerometers can be seen in Figure 4.1 [13]. 16 are placed over the bridge span, while the remaining four are placed at the top of the towers. Except for position 3 and 14 (marked in red color) they are all in pairs.

The length of one time series is approximately 30 minutes and the measurements are taken at 200 Hz. The accelerometers data is measured in g, where $g = 9.81 \frac{m}{s^2}$.

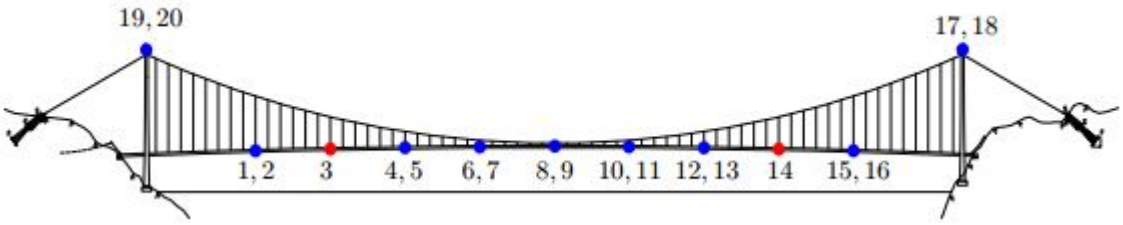

Figure 4.1: Accelerometer positions

4.2 Time Series

For this analysis only one time series have been used, it was measured on 03.02.2014. The length of the data matrix is 371998 measurements, which means:

$$\frac{371998}{200}\ \frac{\text{measurements}}{Hz} = 1860\ \text{seconds} = 31\ \text{minutes}$$

Each of the accelerometers have three outputs, x-direction, y-direction and z-direction, which means:

$$l = 20 * 3 = 60\ \text{measurement channels}$$

First the data is detrended in Matlab, which means removing the mean value or linear trend from the time series. Then the sampling frequency is evaluated. Originally it is 200 Hz which is unnecessary big and requires a lot of time to process. Therefore the data matrix is resampled to 20 Hz, which gives the following dimensions for the data matrix Y:

$$l = 60\ \text{measurement channels} \qquad N = 37200\ \text{samples} \qquad \Delta t = 0.05\ \text{seconds}$$

4.3 Analysis

The modal parameters found during these analyses are compared to the ones found analytically by an ABAQUS model [20]. The stabilization diagram uses the same values as for Chapter 3.

4.3.1 Covariance-Driven Stochastic Subspace Identification

For the analysis with Cov-SSI a n_{max} of 400 has been chosen. Different values for magnitude of block rows have been tried and the results can be seen in Figure C.1, Figure C.2, Figure 4.2 and Figure C.3. The first is obviously not good enough, for $x = 5$ most poles seems to stabilize. But problems occur for low frequencies and for closely spaced modes. For $x = 8$ all poles seem stable, but for $x = 10$ some of the mathematical poles are beginning to look stable. Therefore $x = 8$ is chosen to find the natural frequencies, then they are gathered in Table 4.1.

For the estimation of damping ratios they were also plotted for different values of x and the results can be seen in Figure C.4, Figure C.5, Figure 4.3 and Figure C.6. For a low x there is no alignments. As the magnitude of block rows increases they seem to stabilize more, but for $x = 10$ it starts to look chaotic. Therefore $x = 8$ is chosen and the damping ratios are found and gathered in Table 4.2.

The mode shapes are extracted and plotted in Figure 4.4 to Figure 4.18. The plots show both the horizontal and the vertical plane, from this we can find which modes are related to the different planes. This is added to Table 4.1. On the vertical plane two deformed nodes can be seen on many of the deformations, this relates to the different deflection at each side of the bridge (where sensors are located at each side). This is particularly important for the torsional modes.

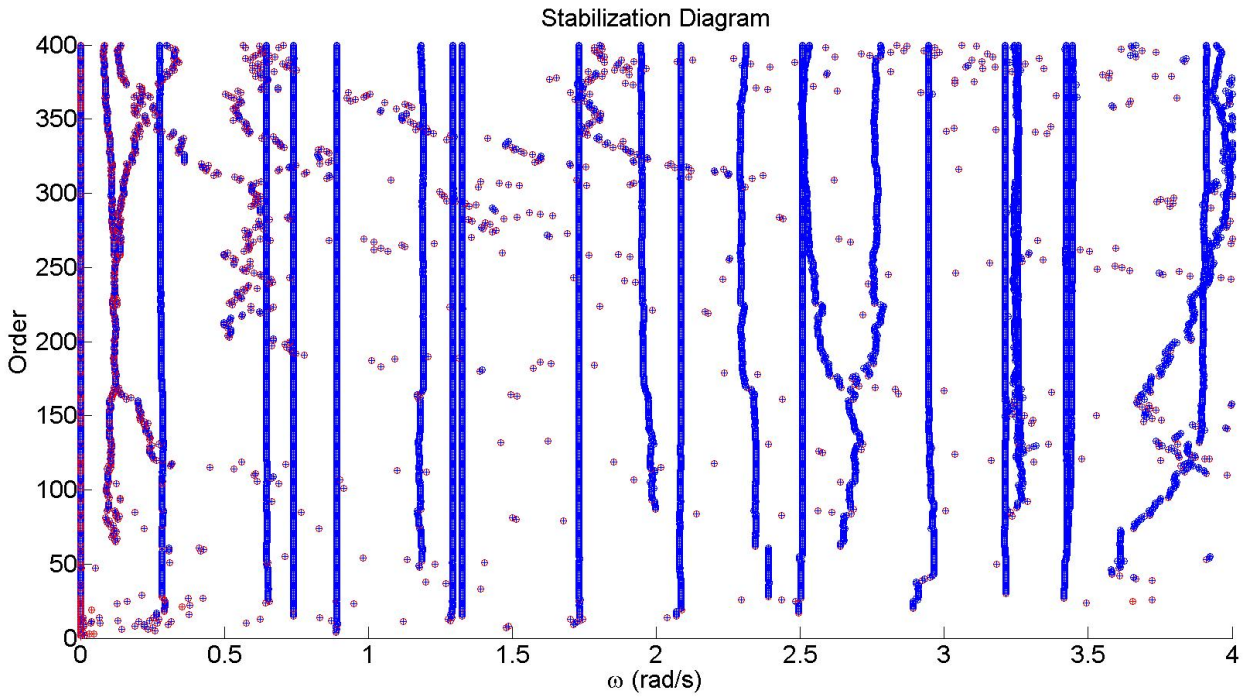

Figure 4.2: Stabilization diagram using Cov-SSI with $x = 8$

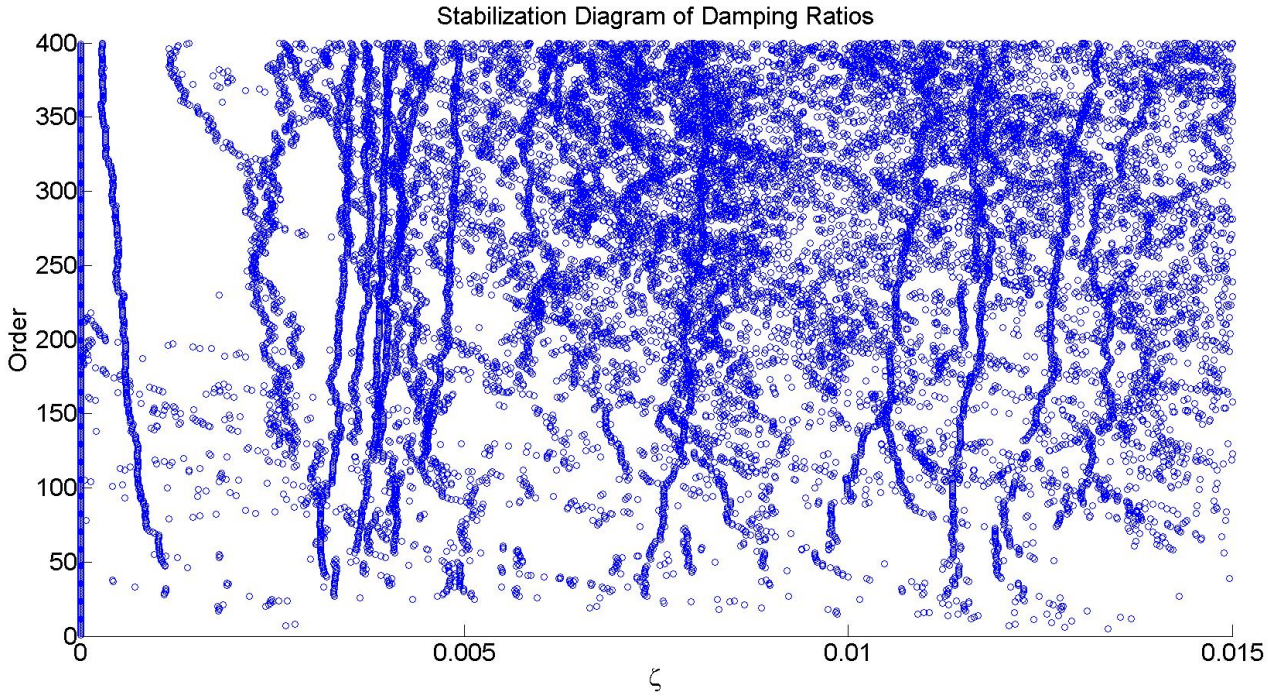

Figure 4.3: Damping ratios using Cov-SSI with $x = 8$

Table 4.1: Natural frequencies for Cov-SSI

Mode number n	Analytical $\omega_n(\frac{rad}{s})$	Cov-SSI $\omega_n(\frac{rad}{s})$	Mode shape
1	0.37	0.29	Horizontal
2	0.71	0.65	Horizontal
3	0.79	0.74	Vertical
4	0.89	0.89	Vertical
5	1.27	1.15	Horizontal
6	1.33	1.29	Vertical
7	1.34	1.33	Vertical
8	1.74	1.73	Vertical
9	2.08	1.96	Horizontal
10	2.15	2.09	Vertical
11	2.33	2.34	Torsional
12	2.52	2.51	Vertical
13	2.92	2.84	Horizontal
14	3.02	2.95	Vertical
15	3.41	3.44	Torsional

Table 4.2: Damping ratios for Cov-SSI

Mode number n	Cov-SSI ζ
1	0.01262
2	0.01193
3	0.00807
4	0.00709
5	0.00461
6	0.00425
7	0.00401
8	0.00381
9	0.00378
10	0.00349
11	0.00320
12	0.00246
13	0.00171
14	0.00077
15	0.00046

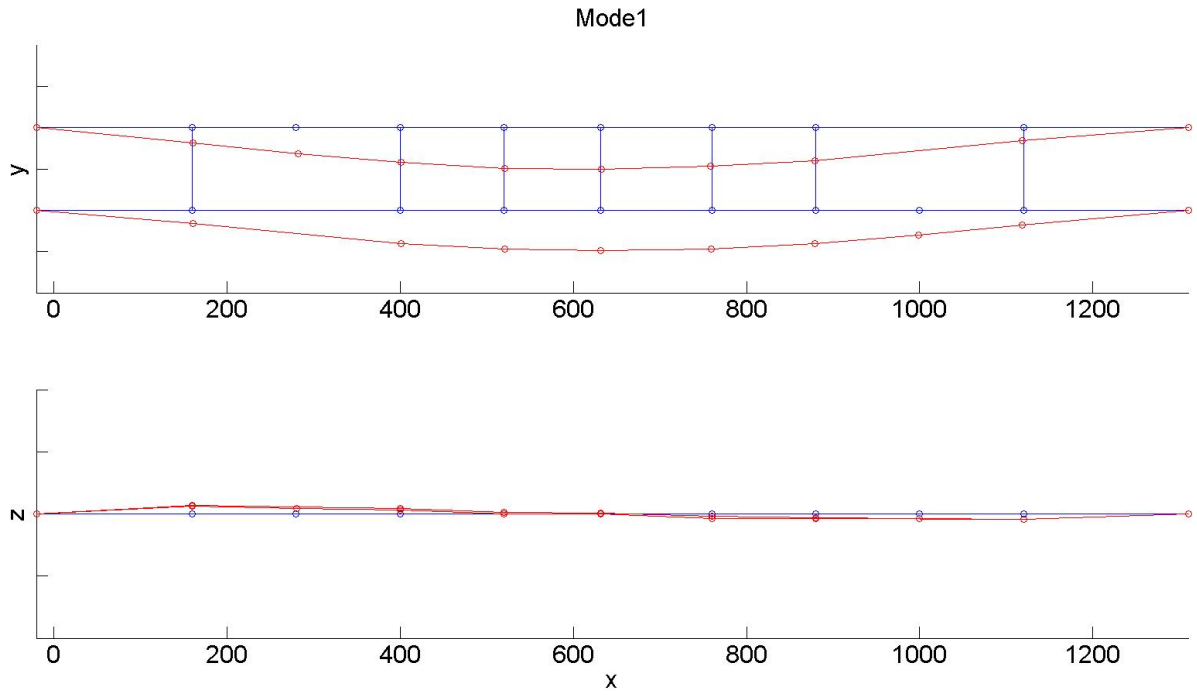

Figure 4.4: Mode 1 horizontal

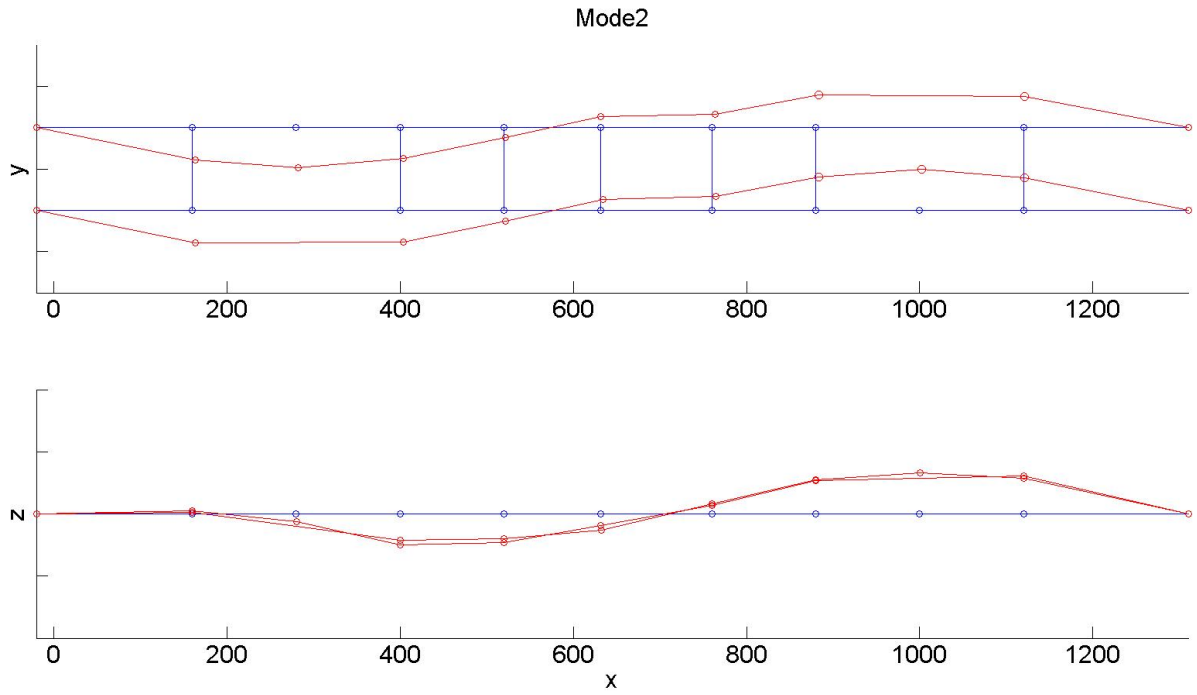

Figure 4.5: Mode 2 horizontal

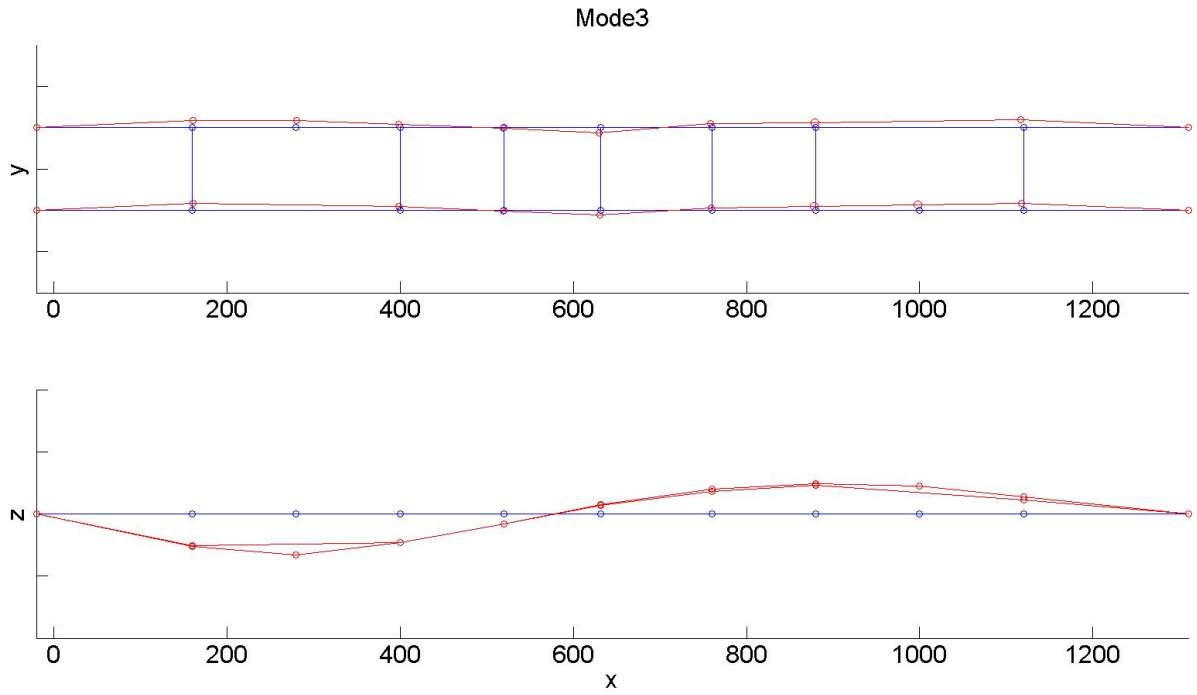

Figure 4.6: Mode 3 vertical

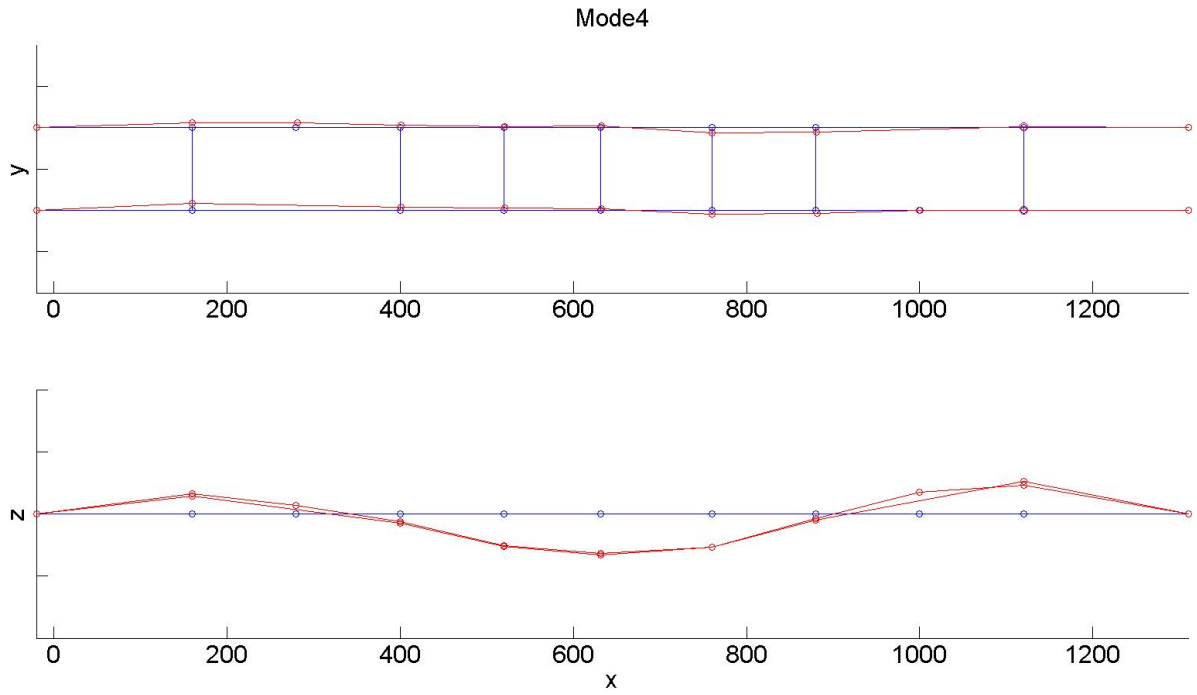

Figure 4.7: Mode 4 vertical

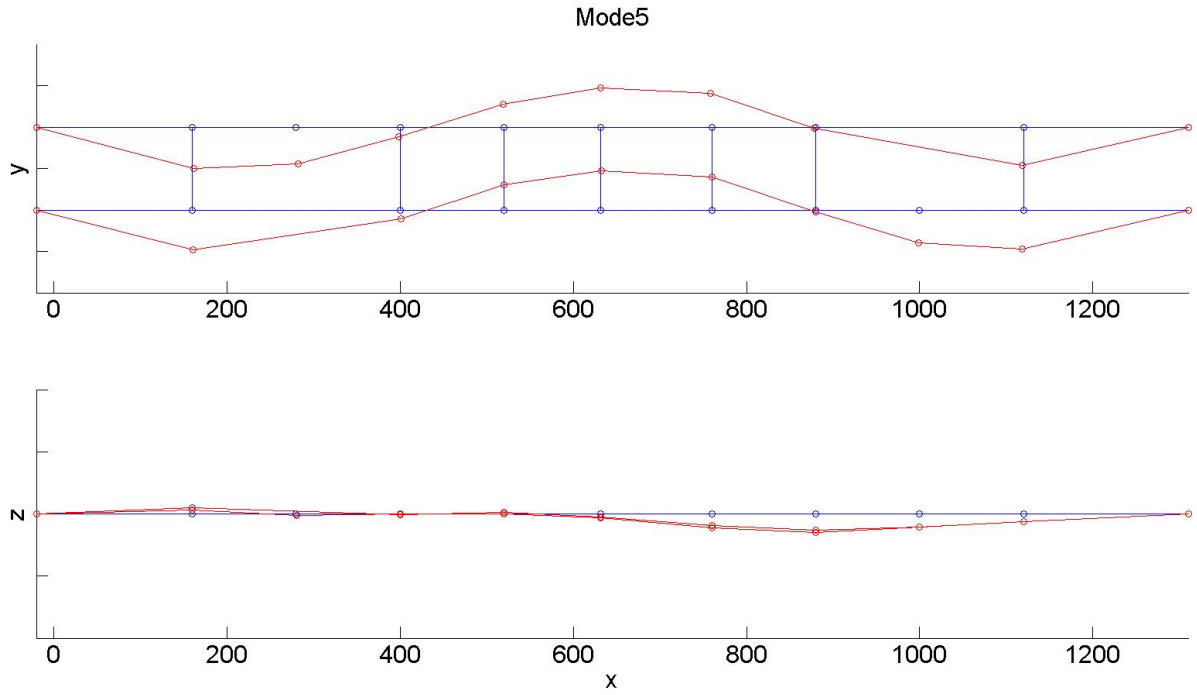

Figure 4.8: Mode 5 horizontal

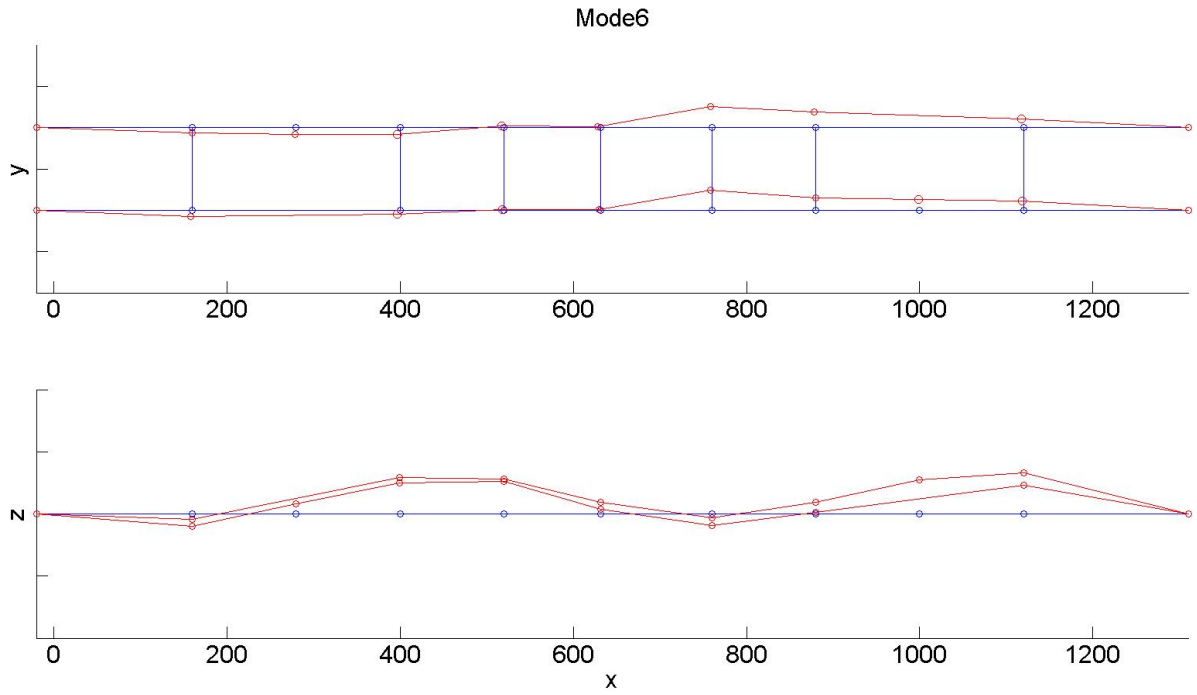

Figure 4.9: Mode 6 vertical

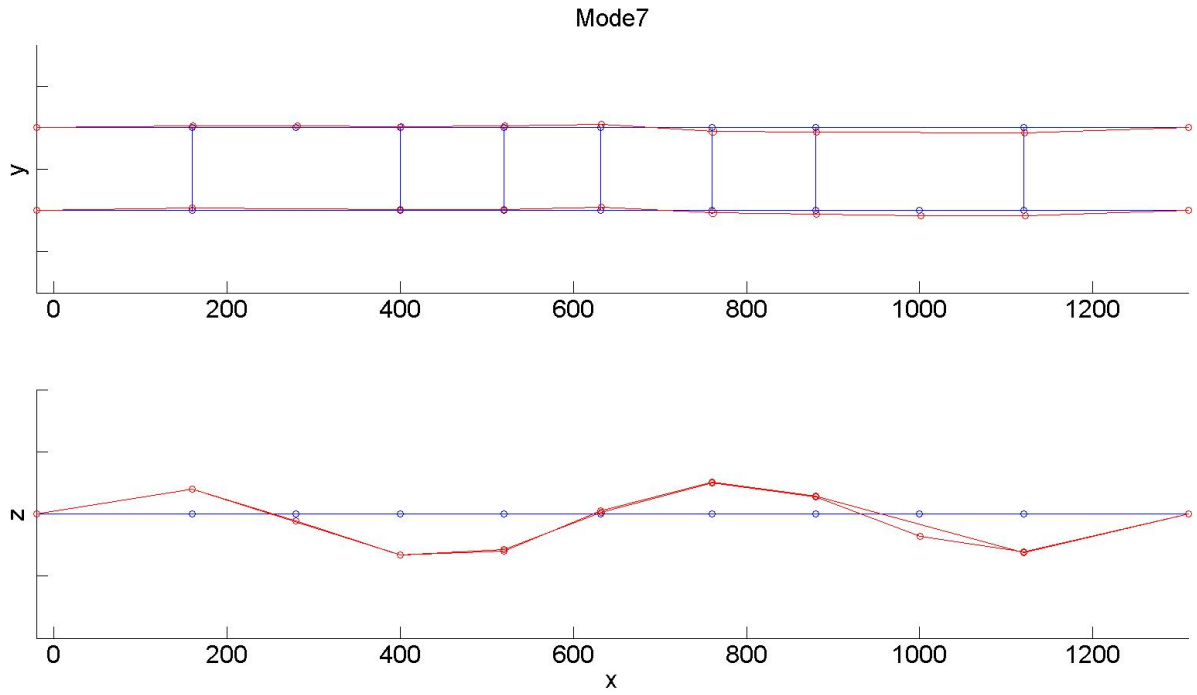

Figure 4.10: Mode 7 vertical

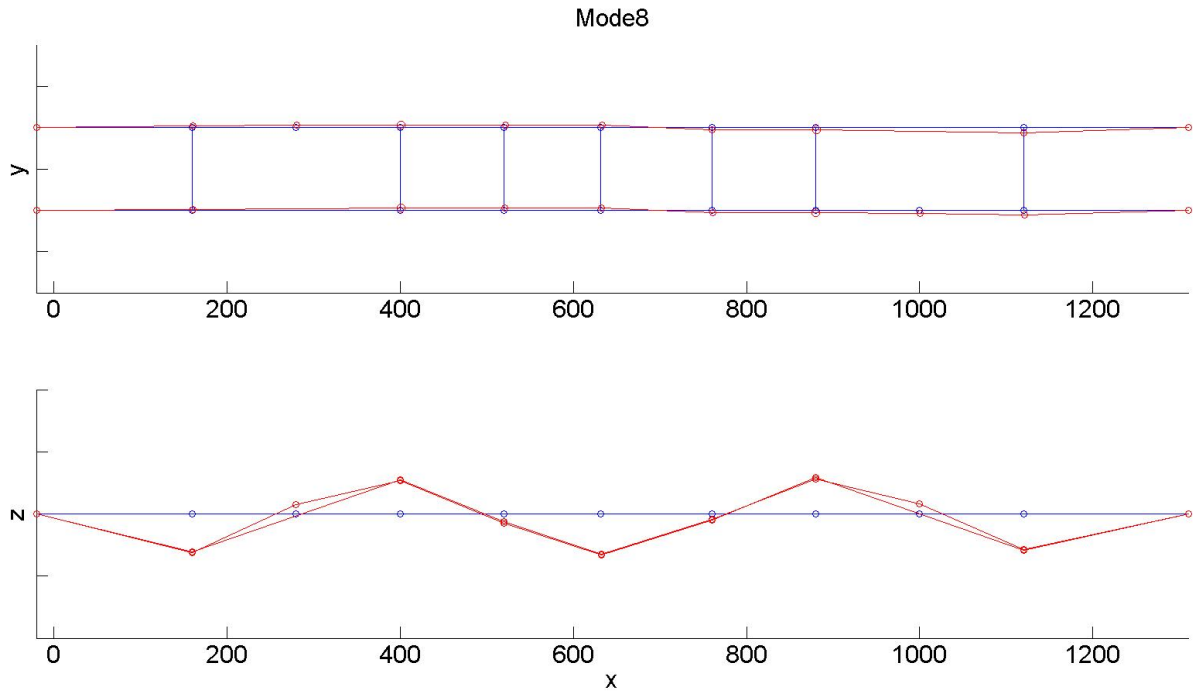

Figure 4.11: Mode 8 vertical

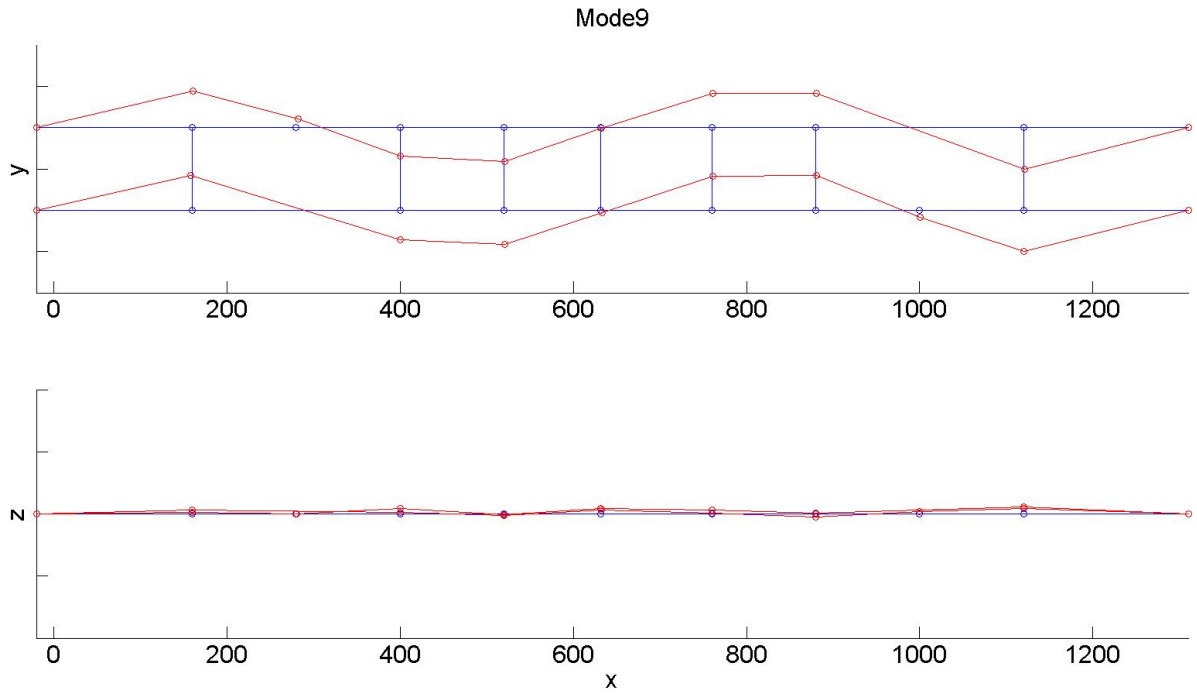

Figure 4.12: Mode 9 horizontal

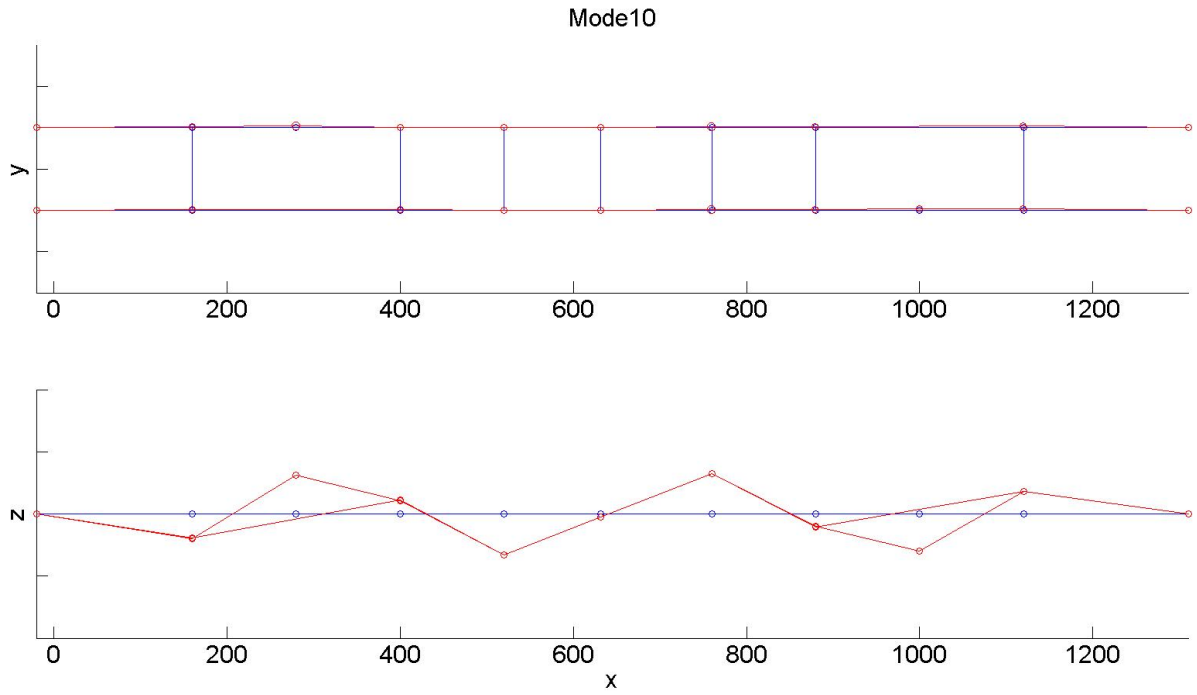

Figure 4.13: Mode 10 vertical

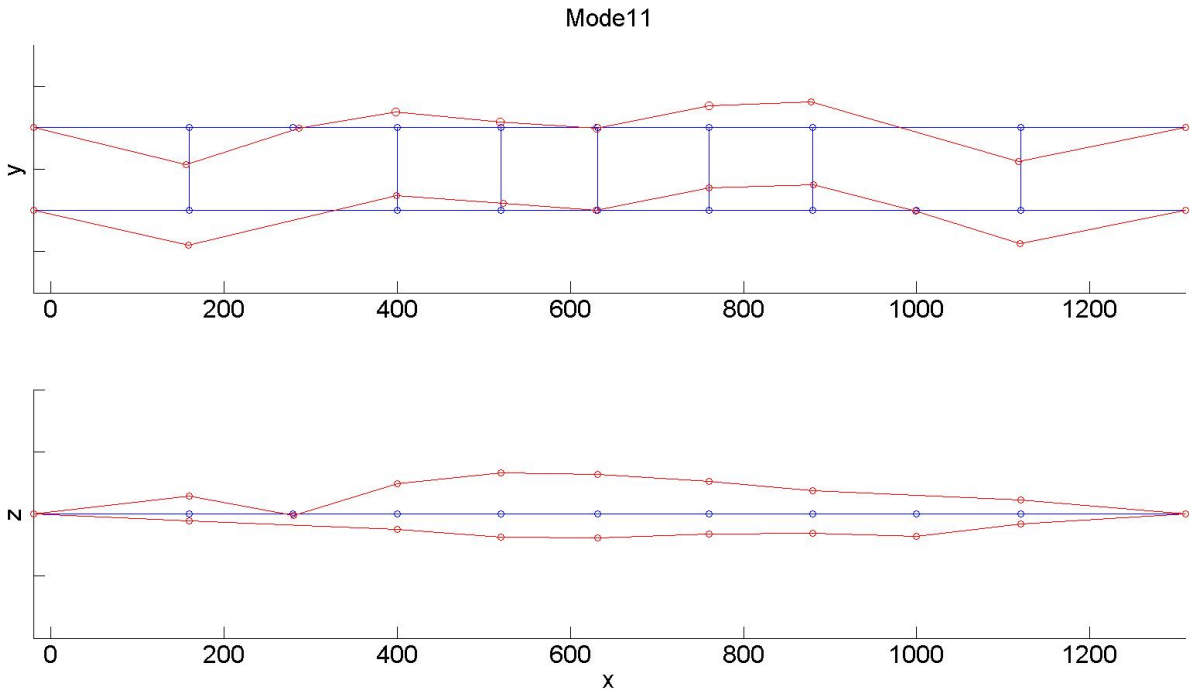

Figure 4.14: Mode 11 torsional

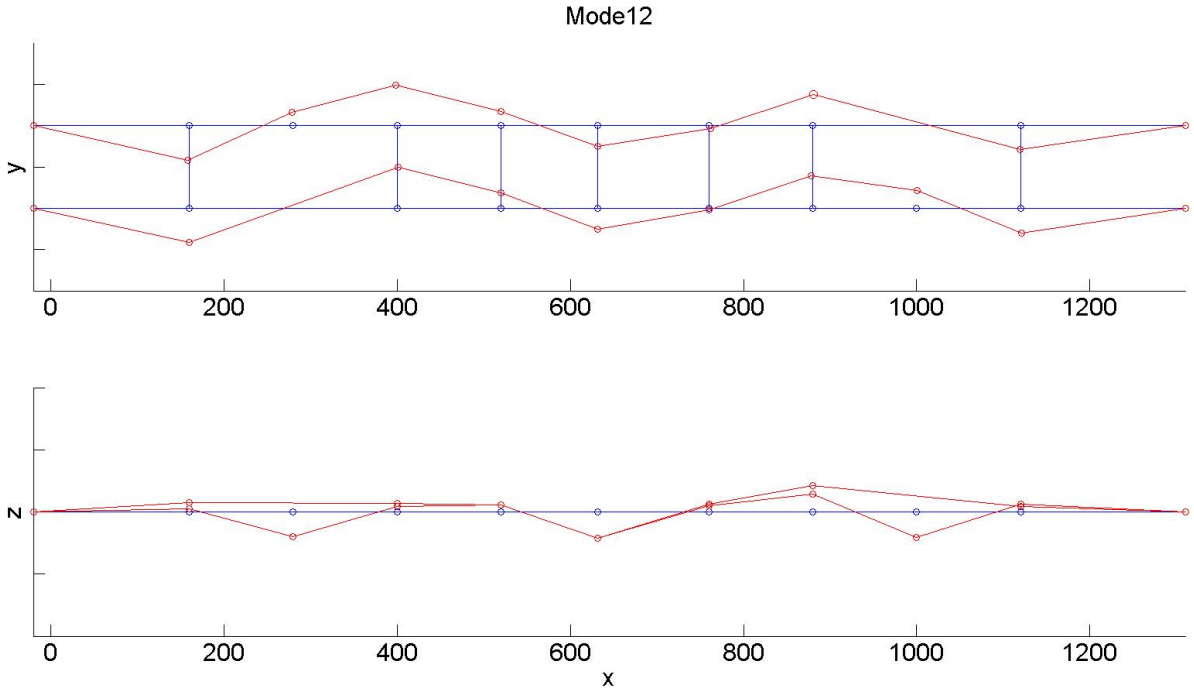

Figure 4.15: Mode 12 vertical

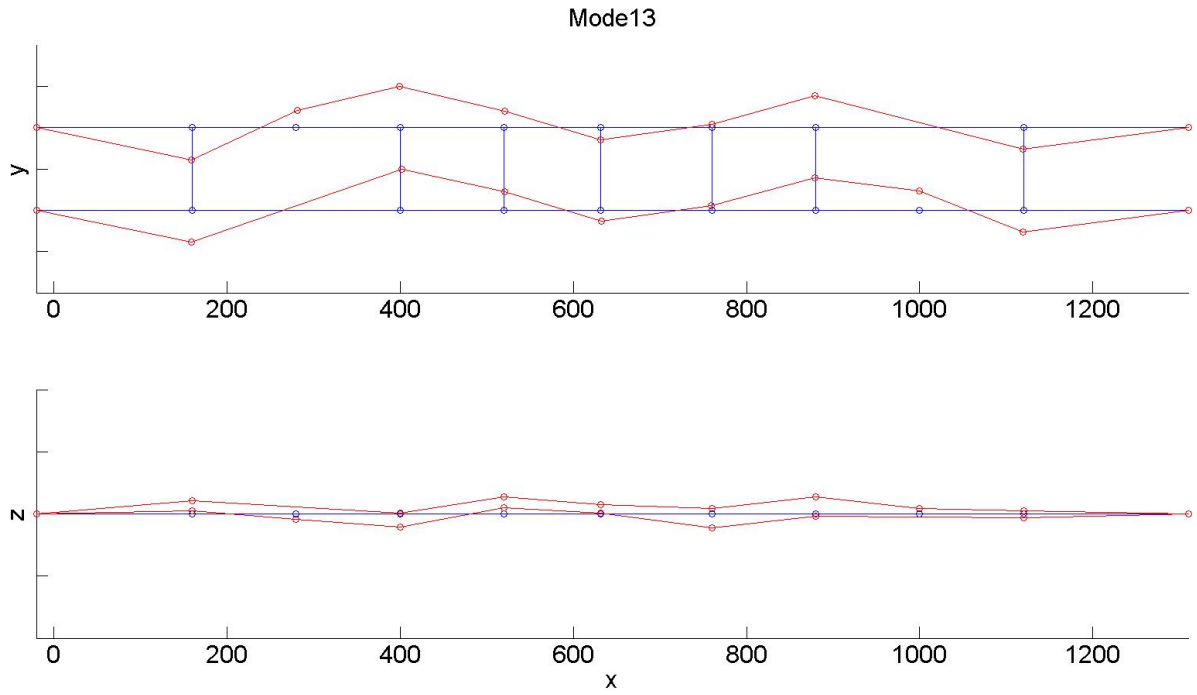

Figure 4.16: Mode 13 horizontal

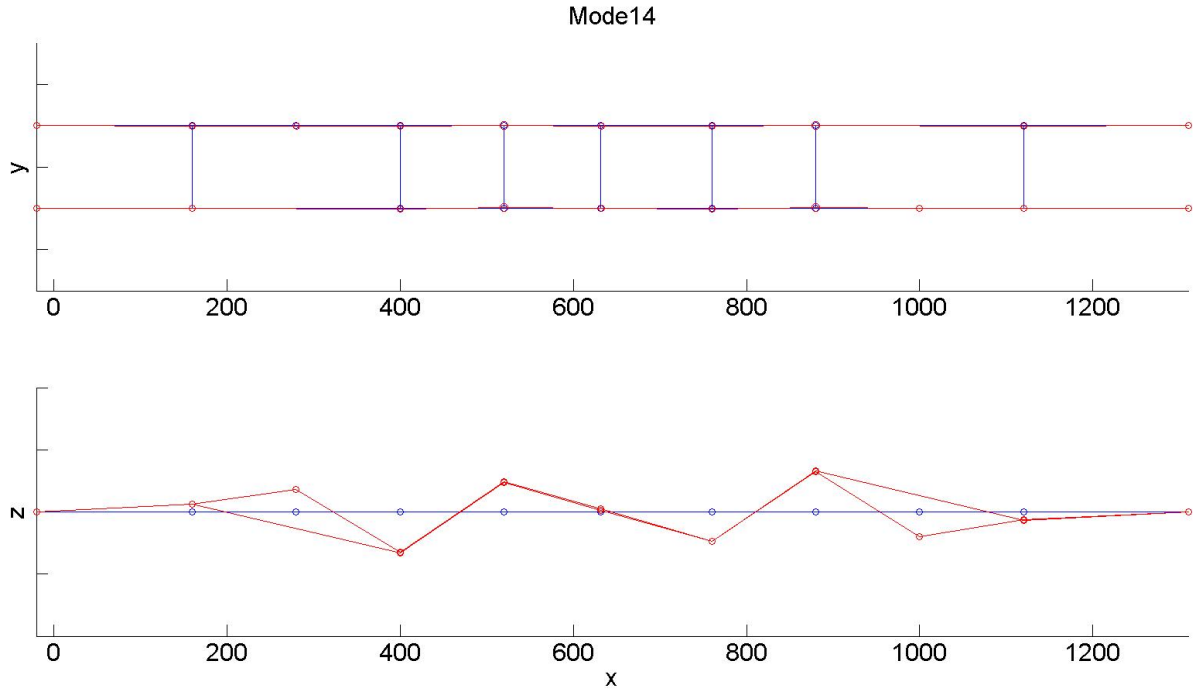

Figure 4.17: Mode 14 vertical

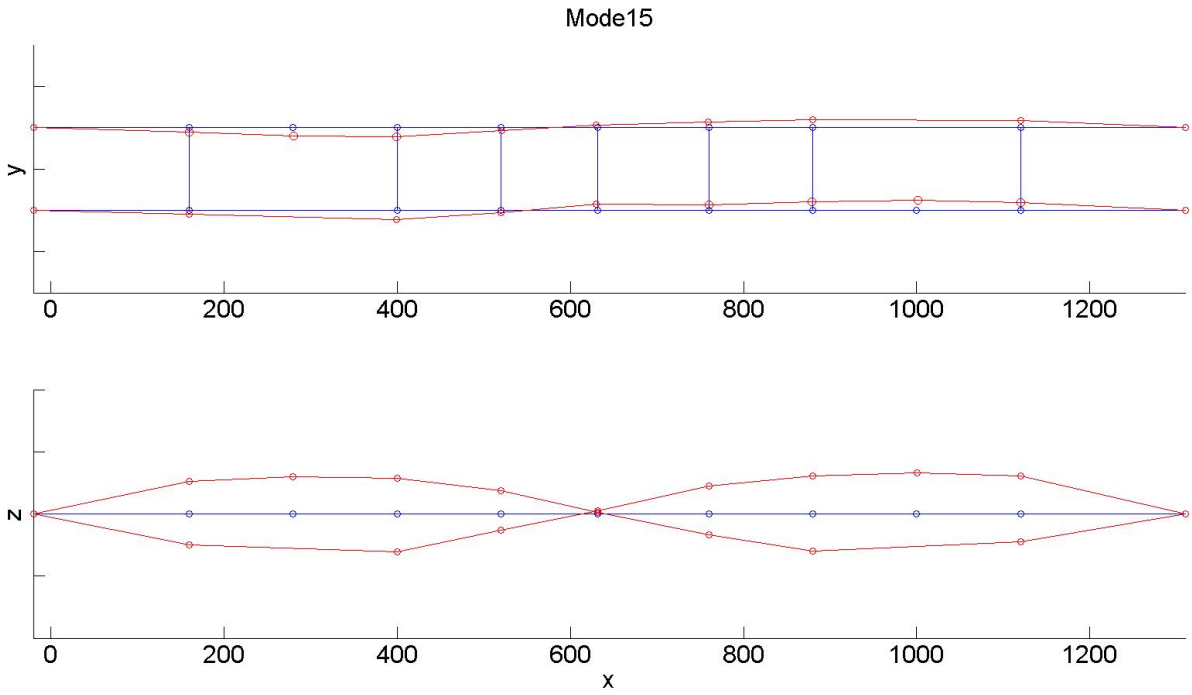

Figure 4.18: Mode 15 torsional

Results

The natural frequencies obtained looks very stable and are easily extracted. The natural frequencies are probably quite accurate. The damping ratios are much harder, and the values obtained should only be used as estimates. The mode shapes seem accurate and logical. A few of the nodes seems to be a bit off on some of the modes, but still the shape of each mode can easily be seen.

4.3.2 Data-Driven Stochastic Subspace Identification

DD-SSI require a n_{max} that is dividable with l, so therefore a value of 420 has been chosen. As for Cov-SSI different values for x have been tried and the results can be seen in Figure C.7, Figure C.8 and Figure 4.19. For $x = 2$ many of the poles have started to stabilize, but it is not able to separate closely spaced modes. For $x = 4$ you can see that the closely spaced modes (e.g around $\omega_n = 1.3$) are starting to split up, however there still seems to be trouble for the modes with a low natural frequency. When $x = 8$ most poles seems to stabilize.

Therefore 8 is chosen to find the natural frequencies, then they are gathered in Table 4.3.

For damping ratio estimation they were also plotted for different values of x and the results can be seen in Figure C.9, Figure C.10 and Figure 4.20. For a low x there is no alignments. As the magnitude of block rows increases they seem to stabilize more and therefore it is possible to pick the damping ratios with $x = 8$ which are gathered in Table 4.4.

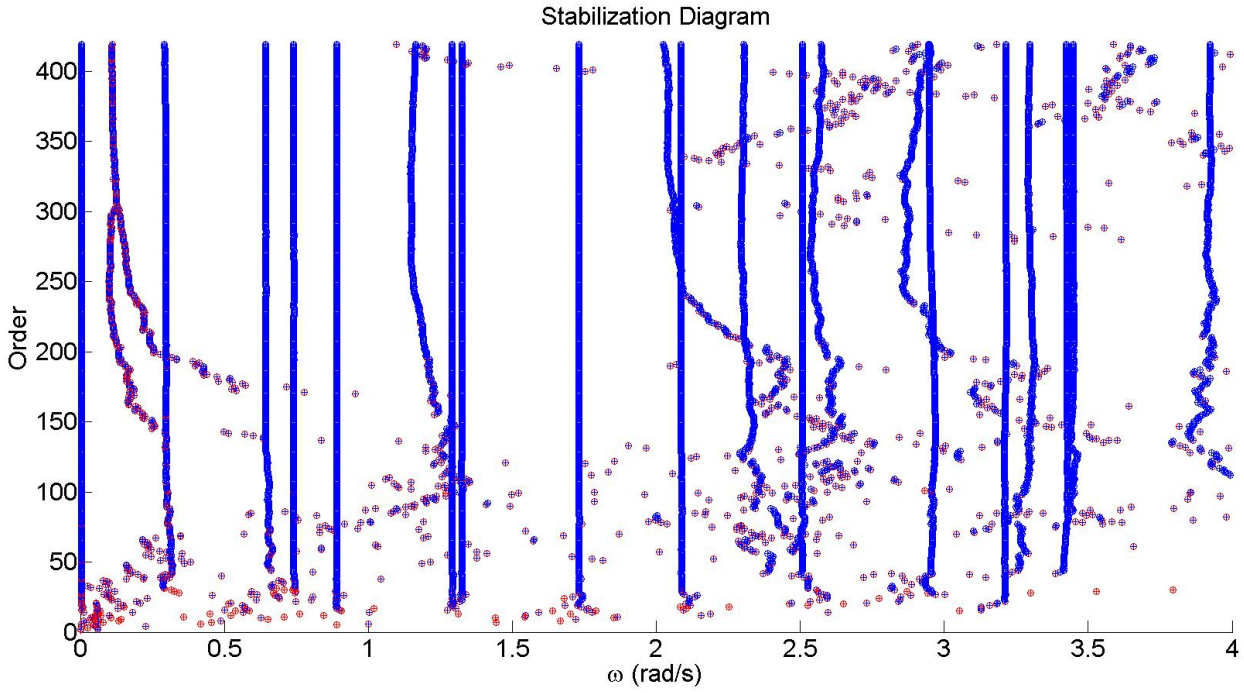

Figure 4.19: Stabilization diagram using DD-SSI with $x = 8$

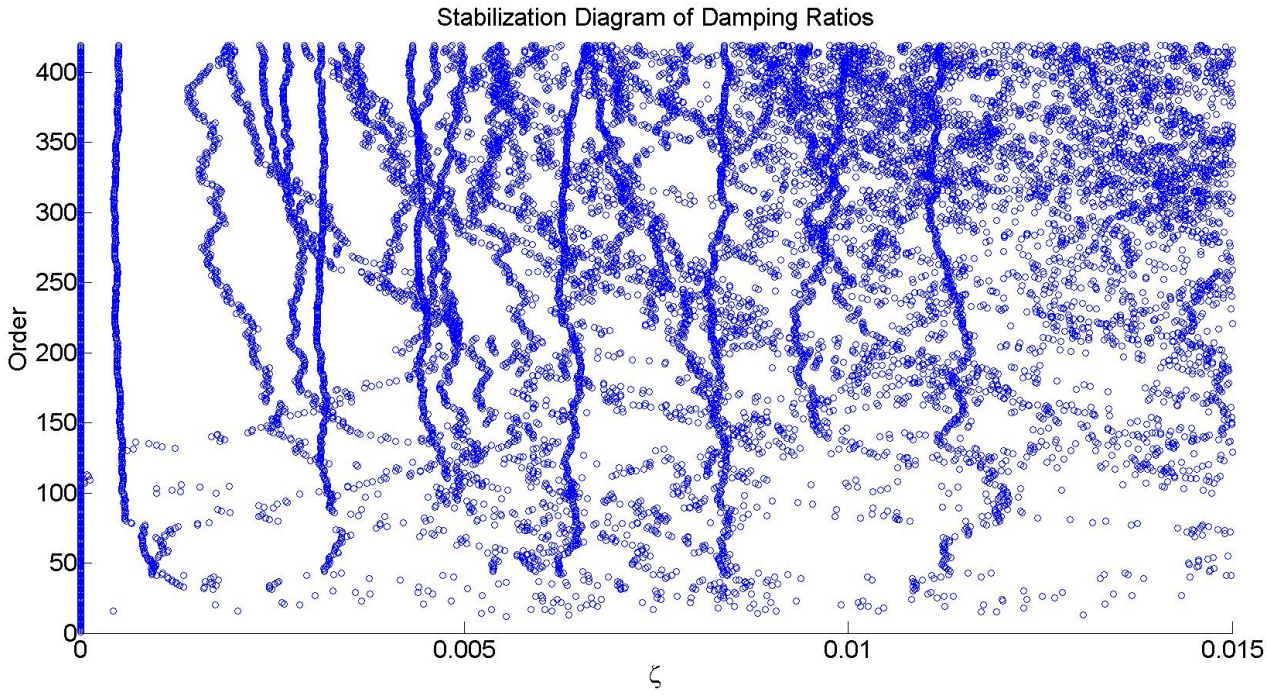

Figure 4.20: Damping ratios using DD-SSI with $x = 8$

Table 4.3: Natural frequencies for DD-SSI

Mode number n	Analytical $\omega_n(\frac{rad}{s})$	DD-SSI $\omega_n(\frac{rad}{s})$
1	0.37	0.30
2	0.71	0.65
3	0.79	0.74
4	0.89	0.89
5	1.27	1.15
6	1.33	1.29
7	1.34	1.33
8	1.74	1.73
9	2.08	2.02
10	2.15	2.09
11	2.33	2.30
12	2.52	2.51
13	2.92	2.91
14	3.02	2.95
15	3.41	3.42

Table 4.4: Damping ratios for DD-SSI

Mode number n	DD-SSI ζ
1	0.01115
2	0.00993
3	0.00833
4	0.00629
5	0.00551
6	0.00490
7	0.00459
8	0.00428
9	0.00366
10	0.00318
11	0.00271
12	0.00237
13	0.00196
14	0.00159
15	0.00047

Results

The natural frequencies found for DD-SSI are very good, and all the modes are easy to pick. For mode number 9, it could be preferable with a higher system order to see that it stabilizes completely. The damping ratios were harder to pick, but alignments could be found. As for Cov-SSI these damping ratios are only estimates.

4.3.3 Second Order Blind Identification

For the SOBI algorithm you will get as many natural frequencies as there are measurement channels. So for this analysis you will get 60 frequencies, but most of them are very high, so therefore it is easy to find the correct values. sobi.m doesn't necessary find all the correct modes at the first analysis(here 12 of 15 was found during the first analysis), therefore the analysis was repeated with different values of number of time lags p, different threshold for the JAD algorithm and different number of peaks and number of samples in FFT for the SDOF estimator.

Table 4.5: Natural frequencies for SOBI

Mode number n	Analytical $\omega_n(\frac{rad}{s})$	SOBI $\omega_n(\frac{rad}{s})$
1	0.37	0.32
2	0.71	0.66
3	0.79	0.73
4	0.89	0.89
5	1.27	1.13
6	1.33	1.28
7	1.34	1.33
8	1.74	1.73
9	2.08	2.09
10	2.15	2.11
11	2.33	2.31
12	2.52	2.51
13	2.92	2.94
14	3.02	3.21
15	3.41	3.44

Results

Although good estimates are found at last, a big drawback with this method is that not all modes are found at first try. Another issue is the lack of insurance that the modes you pick are correct, compared to e.g. the SSI methods with a stabilization diagram.

4.3.4 Peak Picking

The analysis was run with Hanning window and 50% overlap. Different number of samples in FFT was tried as shown in Figure C.11, Figure 4.21 and Figure C.12, where the first four measurement channels are shown. For a low NFFT some of the peaks is hard to pick accurately as they might be between two points because of the low resolution of the frequency vector. For a too high NFFT some of the peaks become more dominant, which makes it harder to pick the other peaks. In addition the coherence plot gets more chaotic. Therefore it is easiest to pick the peaks for a medium value of NFFT, here 8192 is chosen.

There is three measurement channels per sensor. For the one in the axial direction the poles are not very clear, as can be seen for the first and last PSD on Figure 4.21. For the two other

directions they differ between the vertical and the horizontal modes. As shown in Figure 4.22 where the horizontal poles can easily be found and in Figure 4.23 where the vertical poles can easily be found. Both directions manage to find the torsional modes. The values found are gathered in Table 4.6.

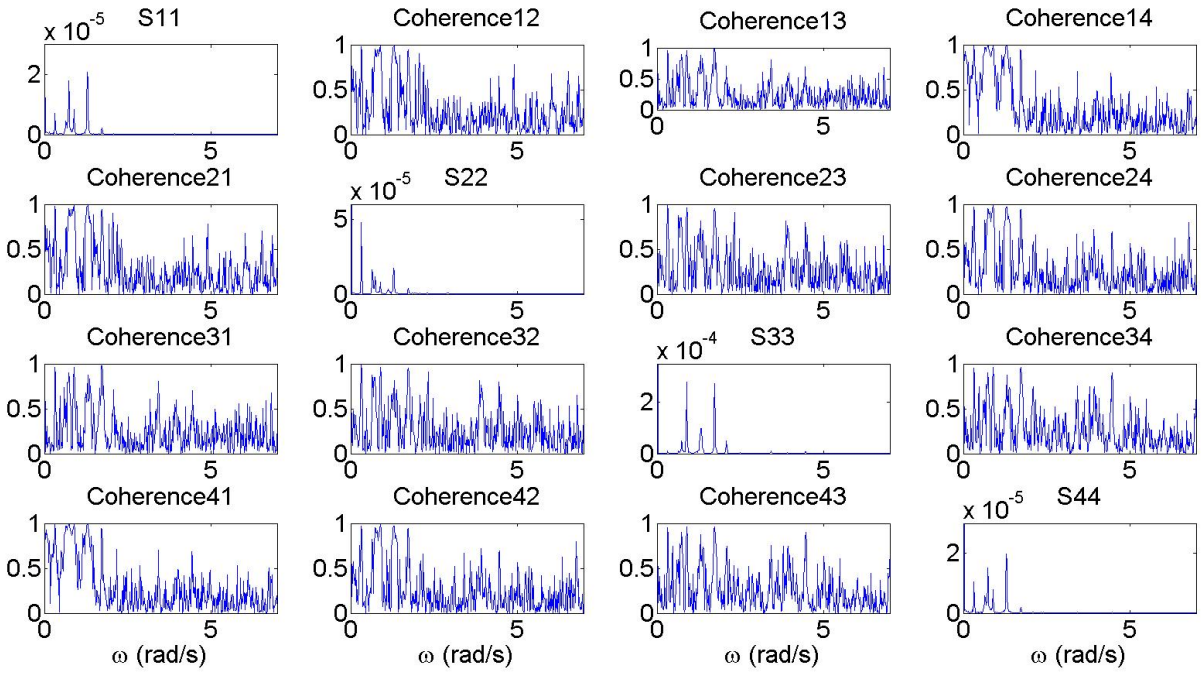

Figure 4.21: PSD and Coherence for $NFFT = 8192$

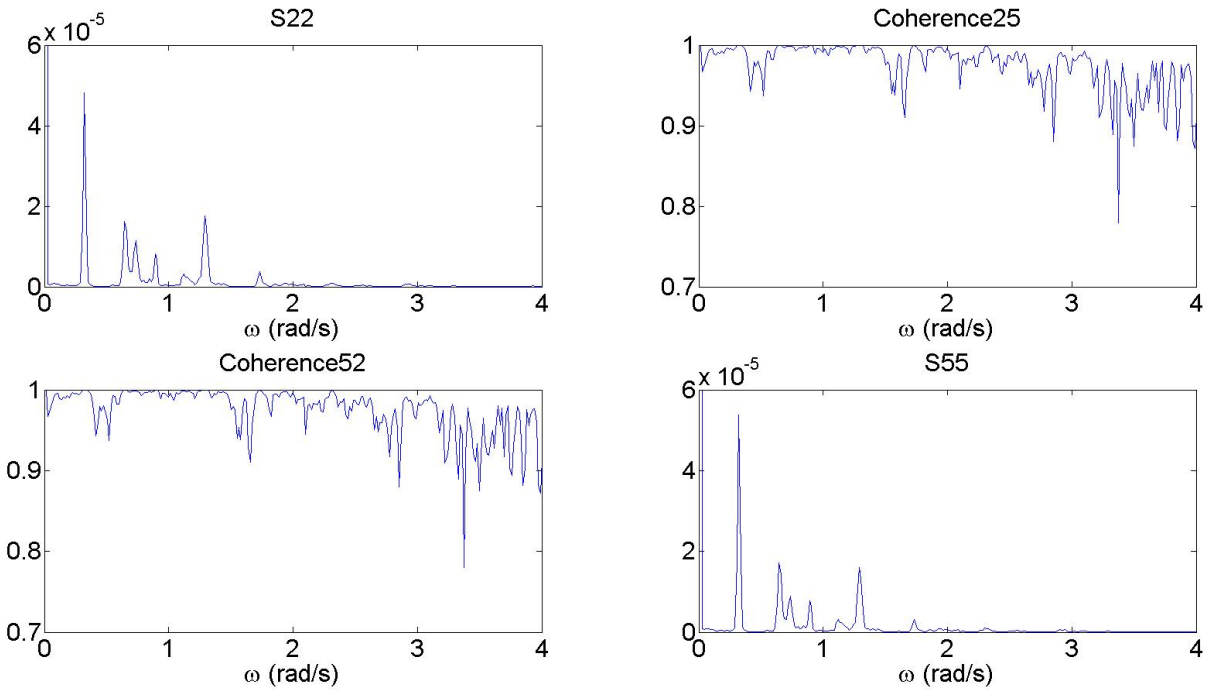

Figure 4.22: PSD and Coherence for $NFFT = 8192$ horizontal modes

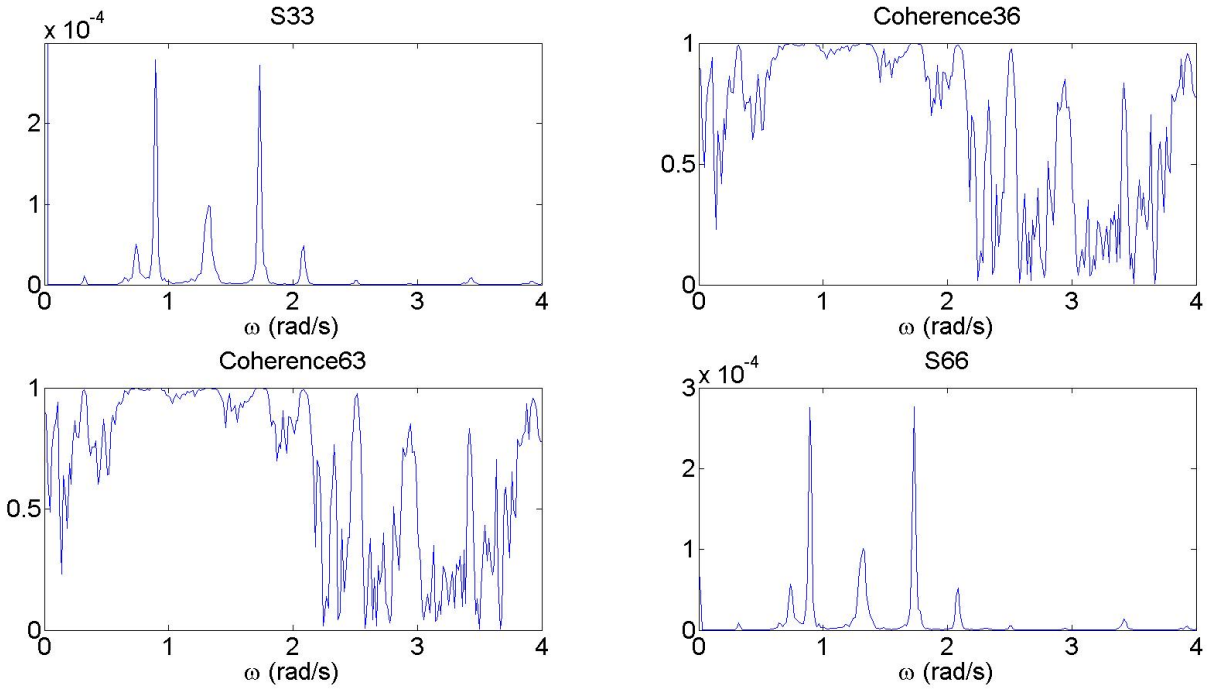

Figure 4.23: PSD and Coherence for $NFFT = 8192$ vertical modes

Table 4.6: Natural frequencies for peak picking

Mode number n	Analytical $\omega_n(\frac{rad}{s})$	Peak picking $\omega_n(\frac{rad}{s})$
1	0.37	0.32
2	0.71	0.65
3	0.79	0.74
4	0.89	0.89
5	1.27	1.10
6	1.33	1.29
7	1.34	
8	1.74	1.73
9	2.08	1.97
10	2.15	2.09
11	2.33	2.36
12	2.52	2.50
13	2.92	2.92
14	3.02	2.95
15	3.41	3.44

Results

With the peak picking method, several of the poles were hard to pick. Especially for high frequencies and the horizontal modes. As mentioned earlier one of the disadvantages of the peak picking method is the ability to separate closely spaced modes, here one mode couldn't be found. The values that are found are quite accurate.

4.3.5 Frequency Domain Decomposition

The analysis was run with Hanning window and 50% overlap. Different number of samples in FFT was tried as shown in Figure C.13, Figure 4.24, Figure C.14 and Figure C.15. For a low NFFT some of the peaks are hard to pick accurately as they might be between two points because of the low resolution of the frequency vector. For a too high NFFT the peaks are not as easily separateable. Therefore it is easiest to pick the peaks from a medium value of NFFT, here 8192 is chosen. The values found are gathered in Table 4.7.

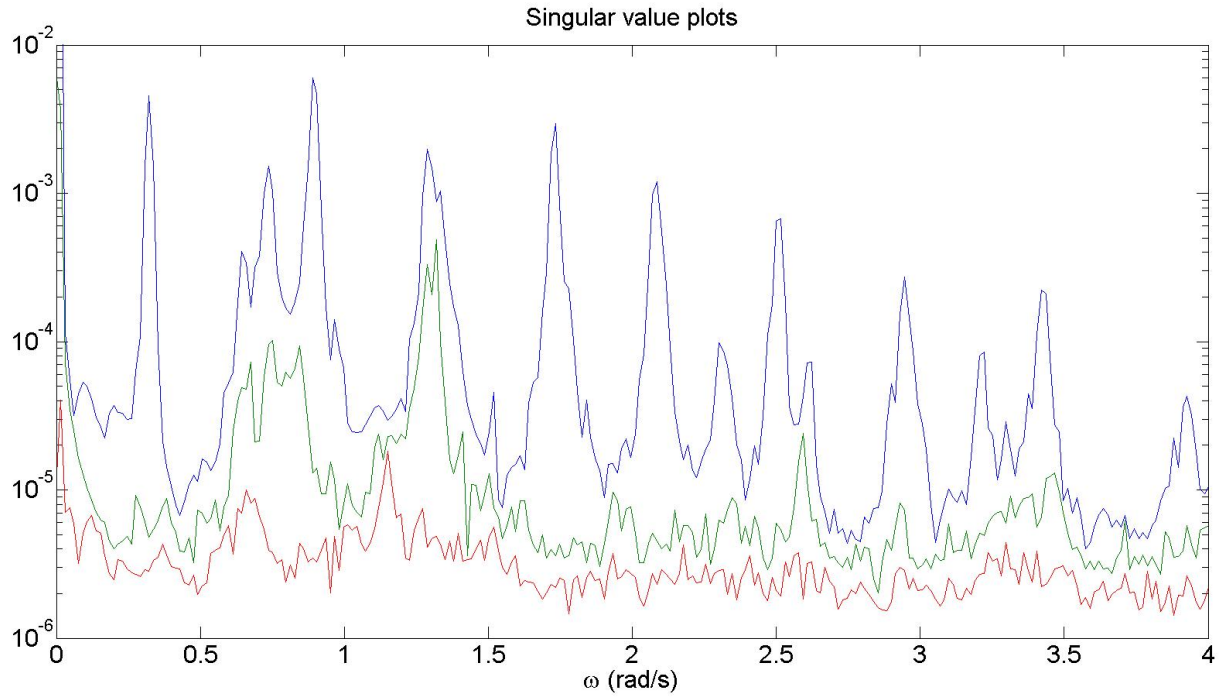

Figure 4.24: Singular value plots for $NFFT = 8192$

Table 4.7: Natural frequencies for FDD

Mode number n	Analytical $\omega_n(\frac{rad}{s})$	FDD $\omega_n(\frac{rad}{s})$
1	0.37	0.32
2	0.71	0.64
3	0.79	0.74
4	0.89	0.89
5	1.27	1.15
6	1.33	1.29
7	1.34	1.32
8	1.74	1.73
9	2.08	1.98
10	2.15	2.09
11	2.33	2.30
12	2.52	2.52
13	2.92	2.90
14	3.02	2.95
15	3.41	3.42

Results

The fifth and seventh peak is not that prominent, but except for that all the peaks are very easy to pick and the values found are quite accurate.

4.3.6 Least Squares Complex Frequency Method

The analysis was run with Hanning window, 50% overlap, $NFFT = 8192$ and $n_{max} = 300$. The stabilization diagram obtained is shown in Figure C.16. Only 8 of the modes are findable, but those are very clear. $n_{max} = 400$ is tried and the result can be seen in Figure C.17. Now 9 of the modes can be found, but still 6 is missing. Therefore $n_{max} = 500$ and $n_{max} = 600$ is tried as shown in Figure C.18 and Figure 4.25. From $n_{max} = 600$ values where found and gathered in Table 4.8.

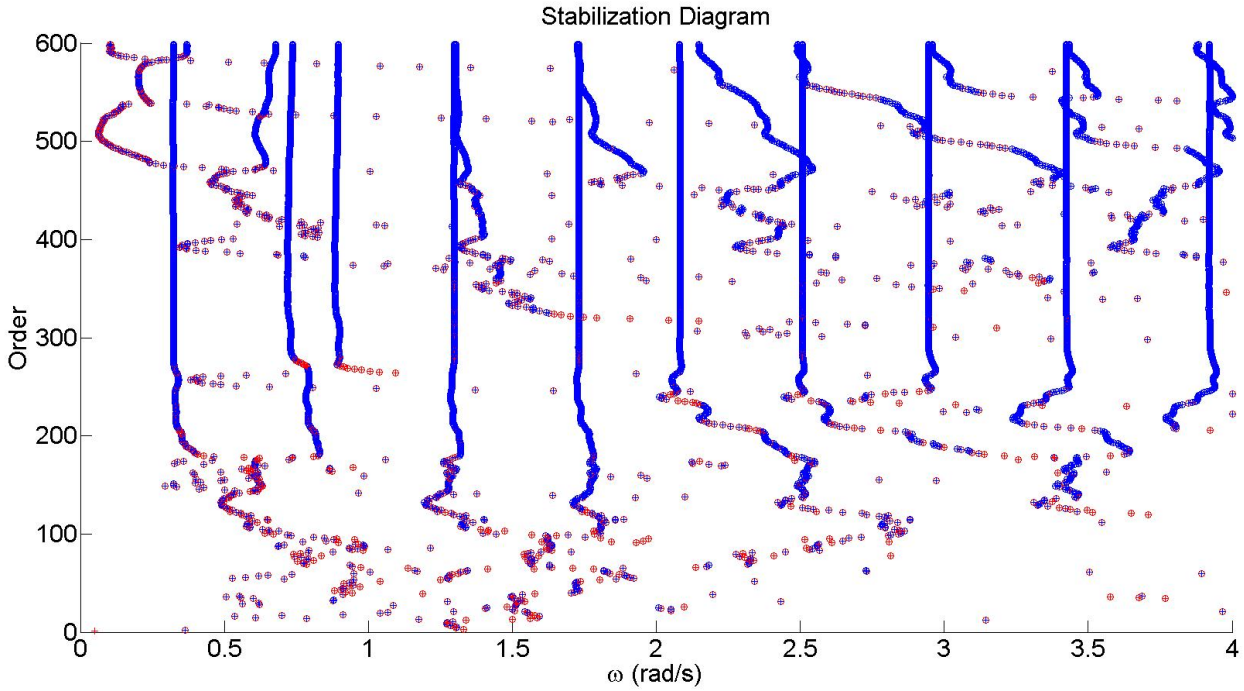

Figure 4.25: Stabilization diagram using LSCF $n_{max} = 600$

Table 4.8: Natural frequencies for LSCF

Mode number n	Analytical $\omega_n(\frac{rad}{s})$	LSCF $\omega_n(\frac{rad}{s})$
1	0.37	0.32
2	0.71	0.68
3	0.79	0.73
4	0.89	0.90
5	1.27	
6	1.33	1.30
7	1.34	1.31
8	1.74	1.73
9	2.08	
10	2.15	2.09
11	2.33	2.36
12	2.52	2.51
13	2.92	
14	3.02	2.95
15	3.41	3.43

Results

Even with the $n_{max} = 600$ LSCF doesn't manage to find all modes and such a large system order is very computational expensive. The modes that are found are quite accurate.

4.3.7 Poly-Reference Least Squares Complex Frequency Method

The analysis was run with Hanning window, 50% overlap, $NFFT = 8192$ and $n_{max} = 50$. The stabilization diagram obtained is shown in Figure C.19. Since this plot is a bit chaotic, a plot where only the poles that was classified as stable regarding natural frequencies were plotted, as shown in Figure 4.26. From here all the modes look stable and the values found are gathered in Table 4.9. As seen on the first figure, not many of the damping ratios were stable. This can also be seen in Figure C.20 where the damping ratios are plotted. Some of the first damping ratios look stable, but for higher modes they are not stable at all and can therefore not be gathered.

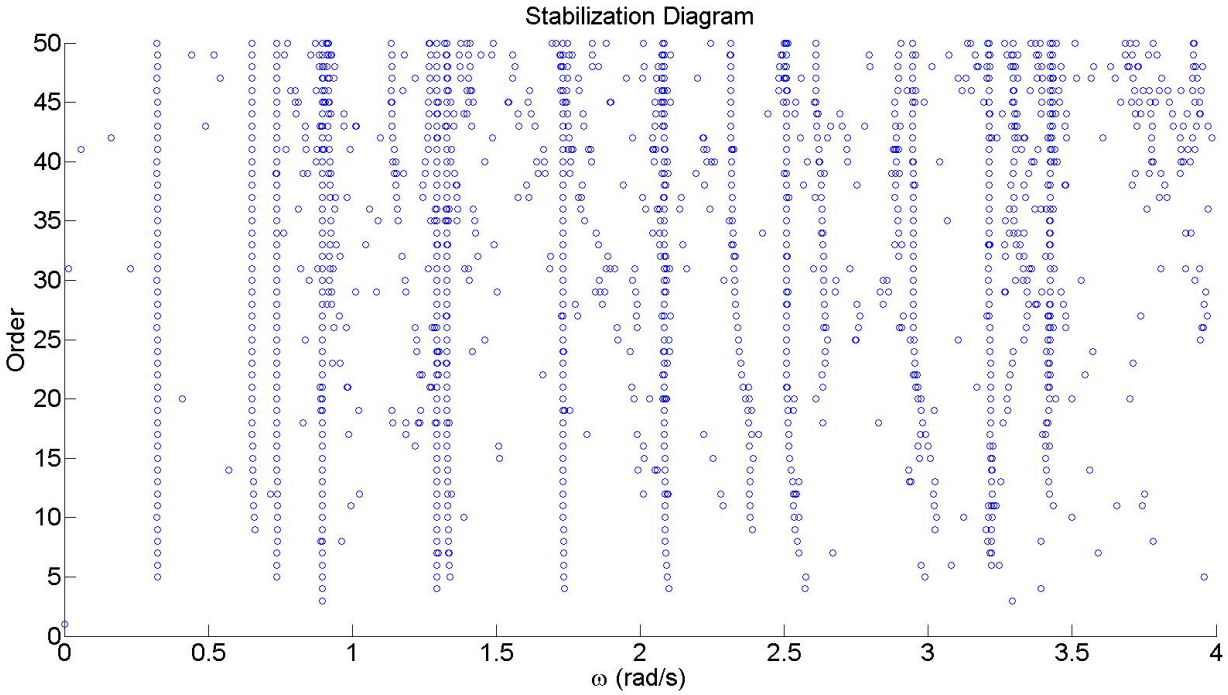

Figure 4.26: Stabilization diagram using pLSCF with only stable poles

Table 4.9: Natural frequencies for pLSCF

Mode number n	Analytical $\omega_n(\frac{rad}{s})$	pLSCF $\omega_n(\frac{rad}{s})$
1	0.37	0.32
2	0.71	0.65
3	0.79	0.74
4	0.89	0.89
5	1.27	1.14
6	1.33	1.29
7	1.34	1.33
8	1.74	1.73
9	2.08	2.00
10	2.15	2.09
11	2.33	2.31
12	2.52	2.51
13	2.92	2.91
14	3.02	2.95
15	3.41	3.42

Results

The natural frequencies found are quite accurate and easy to pick.

4.4 Discussion of Results

The methods managed to find all the modes, except for peak picking and LSCF. SOBI ended up finding all the modes with good accuracy, but since the method doesn't give all the correct modes on the first try and even find some natural frequencies that does not correspond to actual physical modes it is a dangerous method. It should only be used as a verification of other results as discussed for the shear frame.

For peak picking it is hard to find the modes for higher frequencies. Also it did not manage to separate between the two closely spaced modes 6 and 7. Even though the analysis is very efficient it takes some time to manually find the modes and try different parameters. FDD also requires the user to try different parameters and find the modes manually, but picking them is very easy for most of the modes. They are also more accurate than for peak picking.

Both Cov-SSI and DD-SSI gives clear stabilization diagrams and accurate values. The damping ratios are easier to pick for DD-SSI, but, as for the shear frame, Cov-SSI is more computational efficient than DD-SSI. As for the shear frame mode shapes were only plotted using Cov-SSI, as it has proven to be the most efficient method.

LSCF only managed to find 12 of the modes. The stabilization diagram is actually the easiest to pick the modes from, but since the system order need to be so high it is computational expensive and the analysis takes a very long time. pLSCF doesn't require a high system order and all the modes can be found easily and accurately, but still it takes longer time to do the analysis then the SSI methods and it is not able to find the damping ratios.

Chapter 5

Conclusion

5.1 Main Results and Research Findings

This thesis aimed to implement several OMA methods in MATLAB and compare them. They were tested on a shear frame with low and high damping influenced by white noise. Then they were tested on the Hardanger Bridge with actual measurement data. Both the basic methods (SOBI, peak picking and FDD) and the more advanced methods (Cov-SSI, DD-SSI, LSCF and pLSCF) proved to give mostly good estimates. Although they were effected differently by the amount of damping and noise on the measured data.

Cov-SSI was the best method. It is the computational most efficient of the advanced methods. It gave among the clearest stabilization diagrams and generally gave the best estimates of natural frequencies. As one of only two methods it managed to find good estimates of the damping ratios. Cov-SSI was also chosen as the method to find the mode shapes, because of this. DD-SSI also gave clear stabilization diagrams, and almost as good estimates as Cov-SSI. However, it was much more computational demanding. LSCF was computational demanding and did a poor job in estimating the natural frequencies for all three cases. pLSCF was better, but was also computational demanding and the estimates for damping ratio were not good. The comments about computational efficiency is based on the implementation done in this thesis, and may not be representative for the method, but rather the implementations.

For the basic methods FDD proved to give the best estimates. SOBI is perhaps the easiest method to use and very computationally efficient, but because of the previously mentioned drawbacks regarding the safety of the extracted values it is not recommended. Peak picking is not easier to use than FDD and not more efficient. The natural frequencies found by FDD were very accurate for the shear frame with low damping and the Hardanger Bridge, but a bit off for the shear frame with high damping. Still, it was actually one of the best estimates for natural frequencies on the shear frame with high damping, only beaten by the SSI methods. The basic methods were not used to estimate damping ratio and mode shapes in this thesis.

5.2 Future Work

Some suggestions for further work is:

- The application of weighting matrices for LSFC and pLSFC [3] [10], and for DD-SSI [18] and how they would influence the accuracy of the methods.
- The implementation of a fast algorithm for DD-SSI [12].
- Further work on the MATLAB functions to reduce computational time, especially LSCF and pLSCF.
- Use other programs (e.g. LabVIEW) to compare results.

Bibliography

[1] Brekke, G. (2011). Hardanger bridge information. *Norwegian Public Roads Administration*.

[2] Cardoso, J. and Souloumiac, A. (1996). Jacobi angles for simultaneous diagonalization. *SIAM Journal on Matrix Analysis and Applications*, 17(1):161–164.

[3] Cauberghe, B. (2004). *Applied frequency-domain system identification in the field of experimental and operational modal analysis*. PhD thesis, Vrije Universiteit Brussels, Brussels.

[4] Chopra, A. K. (2011). *Dynamics of Strucutres - Theory and Applications to Earthquake Engineering*. Pearson Prentice Hall, 4 edition.

[5] Cooley, J. W. and Tukey, J. W. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301.

[6] Degeringe, S. and Kane, E. (2007). A comparative study of approximate joint diagonalization algorithms for blind source separation in presence of additive noise. *Signal Processing, IEEE Transactions on*, 55(6):3022–3031.

[7] Ellevset, O. (2012). Project overview coastal highway route e39. *Norwegian Public Roads Administration*.

[8] Golub, G. and Van Loan, C. (1996). *Matrix Computations*. The Johns Hopkins University Press, 3 edition.

[9] Gonzales, M. C. (2007). Influence of bolted items on the results and consistency of modal analysis. Master's thesis, Chalmers University of Technology, Goteborg.

[10] Guillaume, P., Verboven, P., Vanlanduit, S., Van der Auweraer, H., and Peeters, B. (2003). A poly-reference implementation of the least-squares complex frequency-domain estimator. In *Proceedings of the international modal analysis conference*, page 12, Kissimmee, FL.

[11] Katayama, T. (2005). *Subspace methods for system identification*. Springer, 1 edition.

[12] Mastronardi, N., Kressner, D., Sima, V., Van Dooren, P., and Van Huffel, S. (2001). A fast algorithm for subspace state-space system identification via exploitation of the displacment structure. *Journal of Computational and Applied Mathematics*, 132(1):71–81.

[13] Mork, O. K. and Solheim, K. D. (2014). Comparison of predicted and measured dynamic response of the hardanger bridge. Master's thesis, Norwegian University of Science and Technology, Trondheim.

[14] Newland, D. E. (1996). *An Introduction to Random Vibrations, Spectral & Wavelet Analysis*. Prentice Hall, 3 edition.

[15] Peeters, B. (2000). *System Identification and Damage Detection in Civil Engineering*. PhD thesis, Katholieke Universiteit Leuven, Leuven.

[16] Peeters, B. and Van der Auweraer, H. (2005). Polymax: A revolution in operational modal analysis. In *1st international operational modal analysis conference*. Copenhagen.

[17] Peeters, B., Van der Auweraer, H., Guillaume, P., and Leuridan, J. (2004). The polymax frequency-domain method: a new standard for modal parameters estimation. *Shock and Vibration*, 11(3-4):395–409.

[18] Rainieri, C. and Fabbrocino, G. (2014). *Operation Modal Analysis of Civil Engineering Structures*. Springer, 1 edition.

[19] Schwarz, B. and H., R. M. (1999). Experimental modal analysis. In *CSI Reliability Week*. Orlando, FL.

[20] Solheim, A. S. (2013). Structural monitoring of the hardanger bridge. Master's thesis, Norwegian University of Science and Technology, Trondheim.

[21] Van Overscee, P. and De Moor, B. (1996). *Subspace identification for linear systems: theory - implementation - applications*. Kluwer Academic Publishers, 1 edition.

[22] Wall, M. E., Rechtsteiner, A., and Rocha, L. (2003). *A Practical Approach to Microarray Data Analysis*, chapter Singular value decomposition and principal component analysis, pages 91–109. Kluwer Academic Publishers.

[23] Welch, P. D. (1967). The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *Audio and Electroacoustics, IEEE Transactions on*, 15(2):70–73.

[24] Zhang, L., Brincker, R., and Andersen, P. (2005). An overview of operational modal analysis; major development and issues. In *1st international operational modal analysis conference*. Copenhagen.

[25] Zhou, W. and Chelidze, D. (2007). Blind source separation based vibration mode identification. *Mechanical Systems and Signal Processing*, 21(8):3072–3087.

Appendix A

Matlab Code

A.1 Example.m

```
% Example: Script to construct the shear frame case
%
%
% REQUIRED FUNCTIONS
%            shearFrame.m
%            newmark.m
%
%
% Sindre Schanke 2015


clc
clear all
close all


%------------------------------Input-------------------------------------%
nDofs=10; %Number of storys
dt=0.01; %Timestep
nTime=1000000; %Total time
```

```
k=50; %Stiffness for each story
c=0.1; %Damping for each story
m=5; %Mass for each story
%-------------------------------------------------------------------------%


%-------------------------Generating time series--------------------------%
[K,C,M]=shearFrame(nDofs,k,c,m); %K,C and M matrices for the shear frame
P=randn(nDofs,nTime); %White noise force vector
u0=zeros(nDofs,1); %Starting displacement
v0=zeros(nDofs,1); %Starting velocity
a0=zeros(nDofs,1); %Starting acceleration
[u,v,a]=newmark(K,C,M,P,dt,1/6,1/2,u0,v0,a0); %Time series
%-------------------------------------------------------------------------%


%---------------------------Exact solution--------------------------------%
[X,e]=polyeig(K,C,M); %Polynomial eigenvalue problem

f=abs(e); %Natural frequencies
fd=imag(e); %Damped modal frequencies
d=-real(e)./abs(e); %Damping ratio
%-------------------------------------------------------------------------%

clear C K M a0 v0 u0 m c k nDofs nTime P
```

A.2 Examplesolve.m

```
% Examplesolve: Script to test the different methods on the shear frame
%
%
% REQUIRED FUNCTIONS
%           SSICOV.m
```

```
6  %            SSIData.m
7  %            sobi.m
8  %            sobifind.m
9  %            mpsd.m
10 %            PSDplot.m
11 %            PSDplt2.m
12 %            LSCF.m
13 %            pLSCF.m
14 %            FDD.m
15 %
16 %
17 % Sindre Schanke 2015
18
19
20 %------------------------------Time domain--------------------------------&
21 % SSI
22 n=50; %Max system order
23 x=2; %Magnitude of block rows
24 [f,c]=SSICov(u,dt,n,x); %Solving using COV-SSI
25 [f,c]=SSIData(u,dt,n,x); %Solving using DD-SSI
26
27 % Plotting for SSICov and SSIData
28 figure(1)
29 axis([0,7,0,n]);
30
31 % SOBI
32 [A,S] = sobi(u); %Solving using SOBI
33 [f,c] = sobifind(S,dt); %Finding Modal Parameters
34 %-----------------------------------------------------------------------%
35
36
37 %----------------------------Frequency domain----------------------------%
38 % Welchs method
39 NFFT=65536; %Number of samples to use in FFT
40 overlap=0.5 %Amount of overlap
41 [Sy,freq]=mpsd(a,hanning(NFFT),dt,NFFT,overlap); %Calculate PSDs and CPSDs
```

```
42
43 % Peak Picking
44 PSDplot(Sy,freq); %Solving using Peak Picking with PSDs and CPSDs
45 PSDplot2(Sy,freq); %Solving using Peak Picking with PSDs and Coherence
46
47 % LSCF and pLSCF
48 n=300; %Max system order
49 [f,c] = LSCF(Sy,freq,dt,n); %Solving using LSCF
50 [f,c] = pLSCF(Sy,freq,dt,n); %Solving using pLSCF
51
52 % Plotting for LSCF and pLSCF
53 figure(1)
54 axis([0,7,0,n]);
55
56 % FDD
57 [S] = FDD(Sy,freq); %Solving using FDD
58
59 % Plotting for FDD
60 figure(1)
61 semilogy(freq,S(:,1:3)) %Only three first singular values
62 title('Singular value plots')
63 xlabel('\omega (rad/s)')
64 xlim([0 7])
65 %-------------------------------------------------------------------------%
```

A.3 Hardanger.m

```
1 % Hardanger: Script which loads the Hardanger data and arranges it
2 %
3 %
4 % Sindre Schanke 2015
5
6
```

```
clc
clear all
close all


%---------------------Load data and arrange-------------------------------%
load('HB141M-2014-02-03_00-00-56.mat') %Load data

%Data from each measurement channel
H1E=HB.Sensors.H1E.Data';
H1W=HB.Sensors.H1W.Data';
H2W=HB.Sensors.H2W.Data';
H3E=HB.Sensors.H3E.Data';
H3W=HB.Sensors.H3W.Data';
H4E=HB.Sensors.H4E.Data';
H4W=HB.Sensors.H4W.Data';
H5E=HB.Sensors.H5E.Data';
H5W=HB.Sensors.H5W.Data';
H6E=HB.Sensors.H6E.Data';
H6W=HB.Sensors.H6W.Data';
H7E=HB.Sensors.H7E.Data';
H7W=HB.Sensors.H7W.Data';
H8E=HB.Sensors.H8E.Data';
H9E=HB.Sensors.H9E.Data';
H9W=HB.Sensors.H9W.Data';
TVE=HB.Sensors.TVE.Data';
TVW=HB.Sensors.TVW.Data';
TBE=HB.Sensors.TBE.Data';
TBW=HB.Sensors.TBW.Data';

%Gather data from all measurement channels into one matrix
T=[H1E; H1W; H2W; H3E; H3W; H4E; H4W; H5E; H5W; H6E; H6W; H7E; H7W; H8E;
   H9E; H9W; TBE; TBW; TVE; TVW];
T=detrend(T); %Remove linear trend


```

```
T=T(:,1:10:end); %Resample to 20 Hz
dt=0.05; %Time step
%-------------------------------------------------------------------------%

clear HB H1E H1W H2W H3E H3W H4E H4W H5E H5W H6E H6W H7E H7W H8E H9E H9W
clear TVE TVW TBE TBW To Td
```

A.4 Hardangersolve.m

```
% Hardangersolve: Script to test the different methods on the Hardanger
% Bridge
%
%
% REQUIRED FUNCTIONS
%           SSICOV.m
%           SSIData.m
%           sobi.m
%           sobifind.m
%           mpsd.m
%           PSDplot.m
%           PSDplt2.m
%           LSCF.m
%           pLSCF.m
%           FDD.m
%
%
% Sindre Schanke 2015


%------------------------------Time domain--------------------------------&
% SSI
n=400; %Max system order
x=8; %Magnitude of block rows
```

```
25 [f,c]=SSICov(T,dt,n,x); %Solving using COV-SSI
26 [f,c]=SSIData(T,dt,n,x); %Solving using DD-SSI
27
28 % Plotting for SSICov and SSIData
29 figure(1)
30 axis([0,4,0,n]);
31
32 % SOBI
33 [A,S] = sobi(T); %Solving using SOBI
34 [f,c] = sobifind(S,dt); %Finding Modal Parameters
35 %-------------------------------------------------------------------------%
36
37
38 %----------------------------Frequency domain-----------------------------%
39 % Welchs method
40 NFFT=65536; %Number of samples to use in FFT
41 overlap=0.5 %Amount of overlap
42 [Sy,freq]=mpsd(T,hanning(NFFT),dt,NFFT,overlap); %Calculate PSDs and CPSDs
43
44 % Peak Picking
45 PSDplot(Sy,freq); %Solving using Peak Picking with PSDs and CPSDs
46 PSDplot2(Sy,freq); %Solving using Peak Picking with PSDs and Coherence
47
48 % LSCF and pLSCF
49 n=400; %Max system order
50 [f,c] = LSCF(Sy,freq,dt,n); %Solving using LSCF
51 [f,c] = pLSCF(Sy,freq,dt,n); %Solving using pLSCF
52
53 % Plotting for LSCF and pLSCF
54 figure(1)
55 axis([0,4,0,n]);
56
57 % FDD
58 [S] = FDD(Sy,freq); %Solving using FDD
59
60 % Plotting for FDD
```

```
figure(1)
semilogy(freq,S(:,1:3)) %Only three first singular values
title('Singular value plots')
xlabel('\omega (rad/s)')
xlim([0 4])
%-------------------------------------------------------------------------%
```

A.5 SSICov.m

```
function [f,c] = SSICov(Y,dt,n,x)


% SSICov: Covariance-Driven Stochastic Subspace Identification Method
%
% [f,c] = SSICov(Y,dt,n,x)
%
%
% INPUTS    Y: Data matrix from time series
%           dt: Time step
%           n: Max system order
%           x: Magnitude of block rows i
%
% OUTPUTS   f: Natural frequencies of the system
%           c: Damping ratio of the system
%
% REQUIRED FUNCTIONS
%           StabDiag.m
%
%
% Sindre Schanke 2015


%----------------------------Initial steps--------------------------------%
```

```
[l,N]=size(Y); %Measurement channels and number of measurements
i=x*ceil(n/l); %Number of block rows >= n/l 4.207
%-------------------------------------------------------------------------%


%--------------------Toeplitz matrix with SVD---------------------------%
% Computing correlation matrixes from R(1) to R(2i)
for j=1:(2*i)
    R{j}=1/(N-j)*Y(1:l,1:N-j)*transpose(Y(1:l,j+1:N)); %4.205
end

% Gather R into Toeplitz matrix(T1) and one-lag shifted Toeplitz matrix(T2)
for j=1:i
    ln=0;
    for k=i:-1:1
        ln=ln+1;
        T1(j,k)=R(ln+j-1); %4.206
        T2(j,k)=R(ln+j); %4.216
    end
end

% Toeplitz matrixes in array form
Ta1=cell2mat(T1);
Ta2=cell2mat(T2);

% SVD of the block Toeplitz matrix with U, Sum and V
[U,S,V] = svd(Ta1); %4.212
s=diag(S);
clear S Ta1 T1 T2 ln R N Y j k
%-------------------------------------------------------------------------%


%----Calculation of state matrix A and modal parameters for each order----%
f=zeros(n,n); %Define for computer efficiency
c=zeros(n,n); %Define for computer efficiency
for k=1:n
```

```
61      ss=diag(sqrt(s(1:k))); %Define for later use
62      Oi=U(:,1:k)*ss; %Observability matrix Oi. 4.213
63      Ti=ss*V(:,1:k).'; %Reversed controllability matrix Ti 4.214
64
65      %C=Oi(1:l,1:k); %Output influence matrix C 4.209
66      %G=Ti(1:k,l*i-l+1:l*i); %Next state-output covariance matrix G 4.210
67
68      A=inv(ss)*U(:,1:k).'*Ta2*V(:,1:k)*inv(ss);  %State matrix A 4.217
69
70      [phi,M] = eig(A); %Eigenvalue solution of A 4.77
71      my=diag(M); %Eigenvalues from diagonal matrix M
72
73      %mode{k}=[C]*phi; %Mode shapes 4.79
74
75      lambda= log(my)/dt; %From discrete-time to continuous time
76
77      f(1:k,k)=abs(lambda); %Natural frequencies (rad/s) 4.137
78      %fd(1:k,k)=imag(lambda); %Damped modal frequencies (rad/s) 4.138
79      c(1:k,k)=-real(lambda)./abs(lambda);%Damping ratio 4.139
80  end
81  %--------------------------------------------------------------------------%
82
83  StabDiag(f,c,n,1); %Stabilization Diagram
84  end
```

A.6 SSIData.m

```
1   function [f,c] = SSIData(Y,dt,n,x)
2
3
4   % SSIData: Data-Driven Stochastic Subspace Identification Method
5   %
6   % [f,c] = SSIData(Y,dt,n)
```

```
7  %
8  %
9  % INPUTS    Y: Data matrix from time series
10 %           dt: Time step
11 %           n: Max system order
12 %           x: Magnitude of block rows i
13 %
14 % OUTPUTS   f: Natural frequencies of the system
15 %           c: Damping ratio of the system
16 %
17 % REQUIRED FUNCTIONS
18 %           StabDiag.m
19 %
20 %
21 % Sindre Schanke 2015
22
23
24 %-----------------------------Initial steps--------------------------------%
25 [l,N]=size(Y); %Measurement channels and number of measurements
26 i=x*ceil(n/l); %Number of block rows >= n/l 4.207
27 j=N-2*i+1; %Number of columns j in practical applications
28 %--------------------------------------------------------------------------%
29
30
31 %----------------------Hankel matrix with SVD----------------------------%
32 %Hankel Matrix
33 H=zeros(l*2*i,j);
34 for k=1:2*i
35     H((k-1)*l+1:k*l,:)=Y(:,k:k+j-1); %4.229
36 end
37
38 %LQ factorization
39 [Q,L]=qr(H',0); %4.233
40 L=L'; %Lower triangular matrix L 4.234
41 Q=Q'; %Orthonormal matrix Q 4.235
42
```

```
%Values from L and Q matrixes which are used later
L21=L(l*i+1:l*i+l,1:l*i);
L22=L(l*i+1:l*i+l,l*i+1:l*i+l);
L31=L(l*i+l+1:2*l*i,1:l*i);
L32=L(l*i+l+1:2*l*i,l*i+1:l*i+l);
Q1=Q(1:l*i,:);
Q2=Q(l*i+1:l*i+l,:);

%Projection
Pi=[L21;L31]*Q1; %4.236
Pim=[L31 L32]*[Q1;Q2]; %4.237

%Output sequence
Yi=[L21 L22]*[Q1;Q2]; %4.238

%SVD
[U,S]=svd(Pi); %4.240
s=diag(S);

clear L Q L21 L22 L31 L32 Q1 Q2 H S N Y
%-------------------------------------------------------------------------%


%----Calculation of state matrix A and modal parameters for each order----%
f=zeros(n,n); %Define for computer efficiency
c=zeros(n,n); %Define for computer efficiency
for k=1:n
    Uk=U(:,1:k); %Define for later use
    ss=diag(sqrt(s(1:k))); %Define for later use
    Oi=Uk*ss; %Observability matrix Oi 4.241
    Si=pinv(Oi)*Pi; %Kalman filter state sequence 4.242

    Oim=Uk(1:l*(i-1),:)*ss;%Observability matrix Oi with last l row deleted
    Sip=pinv(Oim)*Pim; %Kalman state sequence 4.244

    RHS=[Sip;Yi]*pinv(Si); %4.246
```

```
79      A=RHS(1:k,:); %State matrix A 4.246
80      %C=RHS(k+1:k+l,:); %Output influence matrix C 4.246
81
82      [phi,M] = eig(A); %Eigenvalue solution of A 4.77
83      my=diag(M); %Eigenvalues from diagonal matrix M
84
85      %mode{k}=[C]*phi; %Mode shapes 4.79
86
87      lambda= log(my)/dt; %From discrete-time to continuous time
88
89      f(1:k,k)=abs(lambda); %Natural frequencies (rad/s) 4.137
90      %fd(1:k,k)=imag(lambda); %Damped modal frequency (rad/s) 4.138
91      c(1:k,k)=-real(lambda)./abs(lambda); %Damping ratio 4.139
92  end
93  %--------------------------------------------------------------------------%
94
95  StabDiag(f,c,n,1); %Stabilization Diagram
96  end
```

A.7 StabDiag.m

```
1   function [] = StabDiag(f,c,n,l)
2
3
4   % StabDiag: Plot Stabilization Diagram
5   %
6   % [] = StabDiag(f,c,n,l)
7   %
8   %
9   % INPUTS    f: Natural frequencies of the system
10  %           c: Damping ratio of the system
11  %           n: Max system order
12  %           l: For PLSCF: Number of measurement channels
```

```
%              For other methods: l=1
%
%
% Sindre Schanke 2015


%-------------Plotting the results in a stabilization diagram-------------%
%Sorting frequencies and damping ratio
f(~f) = nan;
[f,ix]=sort(f);
c(~c) = nan;
for p=1:size(c,1)
    if p<n+1
        c(:,p)=c(ix(:,p),p);
    end
end

%Checking stability requirements for frequencies and damping ratio
fu(:,1)=f(:,1); %Unstable frequencies
fs(:,1)=nan; %Stable frequencies
fcu(:,1)=f(:,1); %Unstable damping ratios
fcs(:,1)=nan; %Stable damping ratios
for k=2:n
    for j=1:n*l
        ch=0;
        chd=0;
        for m=1:n*l
            if ch == 0
                sf=abs(f(m,k-1)-f(j,k))/f(m,k-1); %Checking for frequency
                if sf < 0.005 %Limit value for frequency
                    fs(j,k)=f(j,k);
                    fu(j,k)=nan;
                    ch=1;
                else
                    fu(j,k)=f(j,k);
                    fs(j,k)=nan;
```

```
                    ch=0;
                end
            end
            if chd == 0
                sc=abs(c(m,k-1)-c(j,k))/c(m,k-1); %Checking for damping
                if sc < 0.05 %Limit value for damping ratio
                    fcs(j,k)=f(j,k);
                    fcu(j,k)=nan;
                    chd=1;
                else
                    fcu(j,k)=f(j,k);
                    fcs(j,k)=nan;
                    chd=0;
                end
            end
        end
    end
end

%Plotting
figure(1);
for k=1:n
    scatter(fs(:,k),ones(1,n*l)*k,50,'blue');
    scatter(fu(:,k),ones(1,n*l)*k,50,'red');
    scatter(fcs(:,k),ones(1,n*l)*k,50,'blue','+');
    scatter(fcu(:,k),ones(1,n*l)*k,50,'red','+');
    hold on
end
hold off

xlabel('\omega (rad/s)');
ylabel('Order');
title('Stabilization Diagram');
%-------------------------------------------------------------------------%
end
```

A.8 sobi.m

```
function [A,S] = sobi(Y)


% sobi: Second Order Blind Identification
%
% [A,S] = sobi(Y)
%
%
% INPUT     Y: Data matrix from time series
%
% OUTPUTS   A: Mixing Matrix
%           S: Sources
%
% REQUIRED FUNCTIONS
%           JAD.m
%
%
% Sindre Schanke 2015


%----------------------------Initial steps-------------------------------%
[l,N]=size(Y); %Measurement channels and number of measurements
p=min(100,ceil(N/3)); %Number of time lags
Y=Y-kron(mean(Y')',ones(1,N)); %Make data zero mean
%------------------------------------------------------------------------%


%-----------------------Whitening the data-------------------------------%
[~,S,V]=svd(Y',0); %SVD of the observed data
W= pinv(S)*V'; %Whitening matrix
Z=W*Y; %Whitended data 4.265
clear S V Y
%------------------------------------------------------------------------%
```

```
34
35
36 %-----------------------Correlation Matrix-------------------------------%
37  k=1;
38  for j=1:l:p*l
39    k=k+1;
40    Rxp=Z(:,k:N)*Z(:,1:N-k+1)'/(N-k+1)/1;
41    Rz(:,j:j+l-1)=sqrt(sum(diag(Rxp'*Rxp)))*Rxp;
42  end;
43 %-------------------------------------------------------------------------%
44
45
46 %------------------Joint Approximate Diagonalization----------------------%
47 [UA] = JAD(l,N,p,Rz);
48 %-------------------------------------------------------------------------%
49
50 %-----------------------Mixing matrix and Sources-----------------------%
51 A=pinv(W)*UA; %Mixing Matrix 4.277
52 S=UA'*Z; %Sources 4.265 4.276 2.278
53 %-------------------------------------------------------------------------%
54
55 end
```

A.9 JAD.m

```
1 function [UA] = JAD(l,N,p,Rz)
2
3
4 % JAD: Perform Joint Approximation Diagonalization
5 %
6 % [UA] = JAD(l,N,p,Rz)
7 %
8 %
```

```
% INPUTS    l: Measurement channels
%           N: Number of measurements
%           p: Number of time lags
%           Rz: Correlation Matrix
%
% OUTPUT    UA: Unitary matrix
%
%
% Sindre Schanke 2015


%----------------------------Initial steps---------------------------------%
UA=eye(l); %Define for later use
t=1/sqrt(N); %Treshold
%--------------------------------------------------------------------------%


%-----------------Joint Approximate Diagonalization--------------------%
check=1; %Set to start
while check %While under treshold
    check=0;
    for j=1:l-1,
        for k=j+1:l,
            j1=j:l:p*l;
            k1=k:l:p*l;
            G=[Rz(j,j1)-Rz(k,k1);Rz(j,k1)+Rz(k,j1);1i*(Rz(k,j1)-Rz(j,k1))];

            [E,D] = eig(real(G*G'));
            [~,K]=sort(diag(D));

            angle=E(:,K(3));
            if angle(1)<0
                angle=-angle;
            end
            c=sqrt(0.5+angle(1)/2);
            s=0.5*(angle(2)-1i*angle(3))/c;
```

```
45
46              if abs(s)>t
47                  check=1;
48                  index=[j;k];
49                  R=[c -conj(s) ; s c]; %Rotation matrix
50
51                  %Update the correlations matrix
52                  Rz(index,:)=R'*Rz(index,:);
53                  Rz(:,[j1 k1])=[c*Rz(:,j1)+s*Rz(:,k1), ...
54                      -conj(s)*Rz(:,j1)+c*Rz(:,k1)];
55
56                  %Update unitary matrix
57                  UA(:,index)=UA(:,index)*R;
58              end
59          end
60      end
61  end
62  %-------------------------------------------------------------------------%
63
64  end
```

A.10 sobifind.m

```
1   function [f,c] = sobifind(S,dt)
2
3
4   % sobi_find: Find frequencies and damping ratio from sources (sobi)
5   %
6   % [f,c] = sobi_find(S,dt)
7   %
8   %
9   % INPUTS    Y: Sources from SOBI
10  %           dt: Time step
```

```
%
% OUTPUTS    f: Natural frequencies of the system
%            c: Damping ratio of the system
%
%
% Sindre Schanke 2015


%----------------------------Initial steps--------------------------------%
[l,N]=size(S); %Measurement channels and number of measurements
nfft=2048*512; %Number of samples to use in FFT
np=7; %Number of peaks
freq=2*pi*[0:nfft-1].'*1/(dt*nfft); %Frequency in rad/s
%-------------------------------------------------------------------------%


%------Calculations of modal parameters for each measurement channel------%
for k=1:l
    g(:,k)=fft(S(k,:),nfft)/N; %Fast fourier transform

    [~,top]=max(abs(g(:,k))); %Find peak

    ns=floor(np/2); %Number of surrounding values
    if top<(ns+1) %To avoid negative ie
        ie=(1:np);
    else
        ie=(top-ns:top+ns);
    end

    A=[g(ie,k),ones(np,1)];
    B=1i*freq(ie).*g(ie,k);
    x=A\B; %Solve system of linear equations

    lambda=x(1); %Eigenvalues

    f(k,1)=abs(lambda); %Natural frequencies (rad/s) 4.137
```

```
47      %fd(k,1)=imag(lambda); %Damped modal frequency (rad/s) 4.138
48      c(k,1)=-real(lambda)./abs(lambda); %Damping ratio 4.139
49  end
50  %-------------------------------------------------------------------------%
51  end
```

A.11 mpsd.m

```
1   function [Sy,f]=mpsd(Y,win,dt,nfft,noverlap)
2
3
4   % mpsd: Compute Power Spectral density (PSD) and Cross Power Spectral
5   %       density (CPSD) by Welch's averaging method and gather them into an
6   %       array matrix
7   %
8   % [Sy,f]=mpsd(Y,win,fs,nfft,noverlap)
9   %
10  %
11  % INPUTS    Y: Data matrix from time series
12  %           win: Windowing fuction and number of samples for each section
13  %           dt: Time step
14  %           noverlap: Fraction which the sections overlap
15  %           nfft: Number of samples to use in FFT
16  %
17  % OUTPUTS   Sy: PSD and CPSD in array form
18  %           freq: Frequency vector of PSD in rad/s
19  %
20  %
21  % Sindre Schanke 2015
22
23
24  %---------------------------Initial steps-------------------------------%
25  [l,N]=size(Y); %Measurement channels and number of measurements
```

```
fs=1/dt; %Sampling frequency in Hz
f=2*pi*(0:nfft/2)*(fs/nfft); %Frequency vector in rad/s
%-------------------------------------------------------------------------%


%--------------------Calculate PSDs and CPSDs---------------------------%
n=(floor((N-nfft)/(nfft*(1-noverlap))))+1; %Number of windows to be applied
for i=1:l
    for ie=1:l
        S1=zeros(nfft,1);
        index=1:nfft; %Intial index
        for j=1:n
            X1=win.*Y(i,index)';
            if i==ie %Calculate PSD
                S1=S1+fft(X1,nfft).*conj(fft(X1,nfft));
            else %Calculate CPSD
                Y1=win.*Y(ie,index)';
                S1=S1+fft(X1,nfft).*conj(fft(Y1,nfft));
            end
            index=index+floor((nfft*(1-noverlap))); %Update index
        end

        S1=S1/mean(win.^2); %Compensate for windowing
        S1=S1(1:((nfft/2)+1),:); %Remove second half(reflection)
        S1=S1/n; %Average power by number of windows
        S1=S1/(fs*nfft); %Normalize by sampling rate & window length
        S1(2:end-1)=2.*S1(2:end-1); %Account for double-sided nature of FFT
        Sy{i,ie}=S1; %Save in Output matrix
    end
end
%-------------------------------------------------------------------------%

end
```

A.12 PSDplot.m

```
function PSDplot(Sy,freq)


% PSDplot: Plot PSD and CPSD for selected measurement channels
%
% PSDplot(Sy,freq)
%
%
% INPUTS    Sy: PSD and CPSD in array form
%           freq: Frequency vector of PSD in rad/s
%
%
% Sindre Schanke 2015


%----------------------------Initial steps-------------------------------%
l=length(Sy);
%------------------------------------------------------------------------%


%-----------------------Plotting the results-----------------------------%
figure(1)
for i=1:l
    for j=1:l
        subplot(l,l,(i-1)*l+j)
        plot(freq,real(Sy{i,j}),'blue') %Plotting real part
        if i~=j %If CPSD
            hold on
            plot(freq,imag(Sy{i,j}),'red') %Plotting imaginary part
        end
        title(['S',num2str(i),num2str(j)])
        xlabel('\omega (rad/s)')
        xlim([0 7])
```

```
34          hold off
35      end
36  end
37  %-------------------------------------------------------------------------%
38
39  end
```

A.13 PSDplot2.m

```
1   function PSDplot2(Sy,freq)
2
3
4   % PSDplot: Plot PSD and Coherence for selected measurement channels
5   %
6   % PSDplot(Sy,freq)
7   %
8   %
9   % INPUTS    Sy: PSD and CPSD in array form
10  %           freq: Frequency vector of PSD in rad/s
11  %
12  %
13  % Sindre Schanke 2015
14
15
16  %----------------------------Initial steps--------------------------------%
17  l=length(Sy);
18  Nf=length(freq);
19  %-------------------------------------------------------------------------%
20
21
22  %---------------------------Find Coherence--------------------------------%
23  for i=1:l
24      for j=1:l
```

```
        for ie=1:Nf
            if i~=j
                Coh{i,j}(ie)=abs(Sy{i,j}(ie))^2/(Sy{i,i}(ie)*Sy{j,j}(ie));
            end
        end
    end
end
%-------------------------------------------------------------------------%


%-----------------------Plotting the results----------------------------%
figure(1)
for i=1:l
    for j=1:l
        subplot(l,l,(i-1)*l+j)
        if i==j
            plot(freq,real(Sy{i,j})) %Plotting PSD
            title(['S',num2str(i),num2str(j)])
        else
            plot(freq,real(Coh{i,j})) %Plotting Coherence
            title(['Coherence',num2str(i),num2str(j)])
        end
        xlabel('\omega (rad/s)')
        xlim([0 7])
    end
end
%-------------------------------------------------------------------------%

end
```

A.14 FDD.m

```
function [S] = FDD(Sy,freq)
```

```
2
3
4  % FDD: Frequency Domain Decomposition Method
5  %
6  % [S] = FDD(Sy,freq)
7  %
8  %
9  % INPUTS    Sy: PSD and CPSD in array form
10 %           freq: Frequency vector of PSD in rad/s
11 %
12 % OUTPUT   S: Singular values
13 %
14 %
15 % Sindre Schanke 2015
16
17
18 %----------------------------Initial steps-------------------------------%
19 Nf = length(freq);
20 [l,N]=size(Sy);
21 %------------------------------------------------------------------------%
22
23
24 %--------------Rearranging and finding the singular values---------------%
25 for i =1:Nf
26     for j=1:l
27         for k=1:N
28             G(j,k) = Sy{j,k}(i); %Arrange PSD and CPSD by frequency line
29         end
30     end
31     [~,Si] = svd(G); % Take SVD of PSD
32     for j=1:l
33         S(i,j)=Si(j,j); %Gather singular values
34     end
35 end
36 %------------------------------------------------------------------------%
37
```

```
38 end
```

A.15 LSCF.m

```
1  function [f,c] = LSCF(Sy,freq,dt,n)
2
3
4  % LSCF: Least Squares Complex Frequency Method
5  %
6  % [f,c] = LSCF(Sy,freq,dt,n)
7  %
8  %
9  % INPUTS    Sy: PSD and CPSD in array form
10 %           freq: Frequency vector of PSD in rad/s
11 %           dt: Time step
12 %           n: Max system order
13 %
14 % OUTPUTS   f: Natural frequencies of the system
15 %           c: Damping ratio of the system
16 %
17 % REQUIRED FUNCTIONS
18 %           StabDiag.m
19 %
20 %
21 % Sindre Schanke 2015
22
23
24 %----------------------------Initial steps--------------------------------%
25 l=length(Sy); %Measurement channels
26 Nf=length(freq); %Number of frequency lines
27 for i=1:Nf
28     for j=1:l
29         for k=1:l
```

```
            G{i}((j-1)*l+k) = Sy{j,k}(i); %PSD and CPSD by freq line 4.113
        end
    end
end
clear Sy i j k
%--------------------------------------------------------------------------%


%-----Calculation of coefficients and modal parameters for each order-----%
f=zeros(n,n); %Define for computer efficiency
c=zeros(n,n); %Define for computer efficiency
for j=1:n
    Md=zeros(j+1); %Define for later use
    I0=zeros(Nf,j+1); %Define for computer efficiency
    for m=0:j
        I0(:,m+1)=exp(1i*freq*dt*m); %4.126
    end
    I0i=I0'; %Calculate complex transpose for later use
    Rk=real(I0i*I0); %Calculate here to avoid multiple calculations 4.129

    for k=1:l*l
        Yk=zeros(Nf,j+1); %Define for computer efficiency
        for ie=1:Nf
            Yk(ie,:)=-G{ie}(k)*I0(ie,:); %4.127
        end
        Sk=real(I0i*Yk); %4.130
        Tk=real(Yk'*Yk); %4.131
        Md=Md+2*(Tk-Sk'*inv(Rk)*Sk); %4.134
        clear Yk Sk Tk
    end
    clear Rk I0 I0i

    delta=[-inv(Md(1:j,1:j))*Md(1:j,j+1);1]; %4.135
    my=roots(fliplr(delta')); %Solve 4.115 for z

    lambda= log(my)/dt; %From discrete-time to continuous time
```

```
66
67      f(1:j,j)=abs(lambda); %Natural frequencies (rad/s) 4.137
68      %fd(1:k,k)=imag(lambda(1:k,k)); %Damped modal frequency (rad/s) 4.138
69      c(1:j,j)=-real(lambda)./abs(lambda);%Damping ratio initial calc 4.139
70
71      clear lambda my delta Md
72  end
73  %-----------------------------------------------------------------------%
74
75  StabDiag(f,c,n,1); %Stabilization Diagram
76  end
```

A.16 pLSCF.m

```
1   function [f,c] = pLSCF(Sy,freq,dt,n)
2
3
4   % pLSCF: Poly-Reference Least Squares Complex Frequency Method
5   %
6   % [f,c] = pLSCF(Sy,freq,dt,n)
7   %
8   %
9   % INPUTS    Sy: PSD and CPSD in array form
10  %           freq: Frequency vector of PSD in rad/s
11  %           dt: Time step
12  %           n: Max system order
13  %
14  % OUTPUTS   f: Natural frequencies of the system
15  %           c: Damping ratio of the system
16  %
17  % REQUIRED FUNCTIONS
18  %           StabDiag.m
19  %
```

```
%
% Sindre Schanke 2015


%----------------------------Initial steps-------------------------------%
l=length(Sy); %Measurement channels
Nf=length(freq); %Number of frequency lines
for i =1:Nf
    for j=1:l
        for k=1:l
            G{i}(j,k) = Sy{j,k}(i); %PSD and CPSD by freq line 4.149
        end
    end
end
clear Sy i j k
%------------------------------------------------------------------------%


%--Calculation of companion matrix A and modal parameters for each order--%
f=zeros(n*l,n); %Define for computer efficiency
c=zeros(n*l,n); %Define for computer efficiency
for k=1:n
    M=zeros((k+1)*l); %Define for later use
    I0=zeros(Nf,k+1); %Define for computer efficiency
    for m=0:k
        I0(:,m+1)=exp(1i*freq*dt*m); %4.159
    end
    I0i=I0'; %Calculate complex transpose for later use
    R0=real(I0i*I0); %Calculate here to avoid multiple calculations 4.163

    for o=1:l
        Y0=zeros(Nf,(k+1)*l); %Define for computer efficiency
        for j=1:Nf
            Y0(j,:)=kron(-I0(j,:),G{j}(o,:)); %4.160
        end
        S0=real(I0i*Y0); %4.164
```

```
56          T0=real(Y0'*Y0); %4.165
57          M=M+2*(T0-S0'*inv(R0)*S0); %4.167
58      end
59
60      alfa=[-inv(M(1:k*l,1:k*l))*M(1:k*l,k*l+1:(k+1)*l)]; %4.169
61      clear An j m o S0 T0 R0 I0 Y0
62
63      %Companion matrix 4.170
64      A=zeros(k*l,k*l);
65      A(1:(k-1)*l,l+1:k*l)=eye((k-1)*l);
66      for ie=1:k
67          A((k-1)*l+1:k*l,(ie-1)*l+1:ie*l)=-alfa((ie-1)*l+1:ie*l,:)';
68      end
69
70      %Eigenvalueproblem
71      [~,My] = eig(A);
72      my=diag(My);
73
74      lambda=log(my)/dt; %From discrete-time to continuous time 4.136
75
76      f(1:k*l,k)=abs(lambda); %Natural frequencies (rad/s) 4.137
77      %fd(1:k,:)=imag(lambda); %Damped modal frequency (rad/s) 4.138
78      c(1:k*l,k)=-real(lambda)./abs(lambda);%Damping ratio initial calc 4.139
79
80      clear M My An alfa A my M lambda
81  end
82  %-------------------------------------------------------------------------%
83
84  StabDiag(f,c,n,l); %Stabilization Diagram
85  end
```

A.17 ModeShapes.m

```
function [] = ModeShapes(nDofs,geo)


% ModeShapes: Mode Shapes for 2D shear frame
%
% [] = ModeShapes(nDofs,geo)
%
%
% INPUTS    nDofs: Number of storys
%           geo: Height between floors and width of building [h w]
%
%
% Sindre Schanke 2015


%----------------------------Initial steps---------------------------------%
%Construct coordinates
for i=1:nDofs
    coords(i,:)=[0 geo(1)*i]; %Left side of building
    coords2(i,:)=[geo(2) geo(1)*i]; %Right side of building
end

%Ensure only real value of eigenvectors are used
Phi=real(Phi);

%Scale largest deformation to 1
for i=1:length(Phi)
    m=max(abs(Phi(:,i)));
    Phi(:,i)=Phi(:,i)/m;
end
%--------------------------------------------------------------------------%


%----------------------------Plotting--------------------------------------%
figure(1)
for i=1:length(Phi)
```

```
37      %New coordinates
38      nc=[coords(:,1)+Phi(:,i), coords(:,2)];
39      nc2=[coords2(:,1)+Phi(:,i), coords2(:,2)];
40
41      %Plot undeformed
42      subplot(2,5,i)
43      line([0 coords(1,1)], [0 coords(1,2)]);
44      line([5 coords2(1,1)], [0 coords2(1,2)]);
45      for i=1:length(coords)
46          line([coords(i,1) coords2(i,1)],[coords(i,2) coords2(i,2)]);
47      end
48      for i=2:length(coords)
49          line([coords(i-1,1) coords(i,1)],[coords(i-1,2) coords(i,2)])
50          line([coords2(i-1,1) coords2(i,1)],[coords2(i-1,2) coords2(i,2)])
51      end
52
53      %Plot deformed
54      line([0 nc(1,1)], [0 nc(1,2)],'color','red');
55      line([5 nc2(1,1)], [0 nc2(1,2)],'color','red');
56      for i=1:length(nc)
57          line([nc(i,1) nc2(i,1)],[nc(i,2) nc2(i,2)]);
58      end
59      for j=2:length(nc)
60          line([nc(j-1,1) nc(j,1)],[nc(j-1,2) nc(j,2)],'color','red')
61          line([nc2(j-1,1) nc2(j,1)],[nc2(j-1,2) nc2(j,2)],'color','red')
62      end
63  end
64
65  %Title and labels
66  tekst = findobj(figure(1),'Type','Axes');
67  for i=1:length(tekst)
68      ylabel(tekst(i),{'Meter'})
69      xlabel(tekst(i),{'Meter'})
70      title(tekst(i),{'Mode Shapes'})
71  end
72  %--------------------------------------------------------------------------%
```

```
73
74 end
```

A.18 ModeShapesYZ.m

```
1  function [] = ModeShapesYZ()
2
3
4  % ModeShapes: Mode Shapes for 3D plots
5  %
6  % [] = ModeShapes()
7  %
8  %
9  % INPUTS    f:
10 %
11 % REQUIRED FUNCTIONS
12 %           Hardangergeo.m
13 %
14 %
15 % Sindre Schanke 2015
16
17
18 %-----------------------------Initial steps---------------------------------%
19 load('HB141M-2014-02-03_00-00-56.mat') %Load data
20 coord=HB.Prop.SensorPositions(:,1:20); %Load coordinates
21 coord(:,17:20)=[-20 -20 1310 1310;10 -10 10 -10;50 50 50 50]; %Set BC
22
23 %Sort coordinates
24 x=coord(1,:);
25 y=coord(2,:);
26 z=coord(3,:);
27
28 %Ensure only real value of eigenvectors are used
```

```
Phi=real(Phi);
s=size(Phi); %Find mode shapes to be plotted

%Scale largest deformation to a number
for i=1:s(2)
    m=max(abs(Phi(:,i)));
    Phi(:,i)=Phi(:,i)/m;
end

%Set deformation at boundary conditions=0
Phi(49:60,:)=0;
Phi(2:3:20,:)=-Phi(2:3:20,:);
%-------------------------------------------------------------------------%


%-----------------------------Plotting------------------------------------%
%Horizontal plane
for i=1:s(2)
    figure(i)
    subplot(2,1,1)
    scatter(x,y,z) %Show sensors
    hold on

    %Underformed
    Hardangergeo(x,y,1,1) %Plot lines between undeformed points

    %Deformed
    %New coordinates
    nx=x+Phi(1:3:58,i)';
    ny=y+Phi(2:3:59,i)';
    nz=z+Phi(3:3:60,i)';

    scatter(nx,ny,nz,'red') %Show sensors with deformation
    Hardangergeo(nx,ny,0,0) %Plot lines between deformed points
    hold off

```

```
65      %Title
66      title(['Mode',num2str(i)])
67      ylabel('y')
68      set(gca, 'YTickLabelMode', 'manual', 'YTickLabel', []);
69  end
70
71  %Vertical plane
72  for i=1:s(2)
73      figure(i)
74      subplot(2,1,2)
75      scatter(x,z) %Show sensors
76      hold on
77
78      %Underformed
79      Hardangergeo(x,z,1,1) %Plot lines between undeformed points
80
81      %Deformed
82      %New coordinates
83      nx=x+Phi(1:3:58,i)';
84      ny=y+Phi(2:3:59,i)';
85      nz=z+Phi(3:3:60,i)';
86      scatter(nx,nz,'red') %Show sensors with deformation
87      Hardangergeo(nx,nz,0,0) %Plot lines between deformed points
88      hold off
89
90      %Title
91      ylabel('z')
92      xlabel('x')
93      set(gca, 'YTickLabelMode', 'manual', 'YTickLabel', []);
94  end
95  %-------------------------------------------------------------------------%
96
97  end
```

A.19 Hardangergeo.m

```
function [] = Hardangergeo(x,y,c,f)


% Hardangergeo: Draw up the geometry for the Hardanger bridge
%
% [] = Hardangergeo(x,y,c,f)
%
%
% INPUTS    x: x-coordinates
%           y: y/z-coordinates
%           c: 1 for blue and 0 for red
%           f: 1 for drawing lines across bridge
%
%
% REQUIRED FUNCTIONS
%           Hardangergeo.m
%
%
% Sindre Schanke 2015

if c==1
    d='blue';
else
    d='red';
end

line([x(19) x(1)],[y(19) y(1)],'color',d)
line([x(20) x(2)],[y(20) y(2)],'color',d)
line([x(1) x(4)],[y(1) y(4)],'color',d)
line([x(2) x(3)],[y(2) y(3)],'color',d)
line([x(3) x(5)],[y(3) y(5)],'color',d)
line([x(4) x(6)],[y(4) y(6)],'color',d)
line([x(5) x(7)],[y(5) y(7)],'color',d)
```

```
34 line([x(6) x(8)],[y(6) y(8)],'color',d)
35 line([x(7) x(9)],[y(7) y(9)],'color',d)
36 line([x(8) x(10)],[y(8) y(10)],'color',d)
37 line([x(9) x(11)],[y(9) y(11)],'color',d)
38 line([x(10) x(12)],[y(10) y(12)],'color',d)
39 line([x(11) x(13)],[y(11) y(13)],'color',d)
40 line([x(12) x(14)],[y(12) y(14)],'color',d)
41 line([x(13) x(16)],[y(13) y(16)],'color',d)
42 line([x(14) x(15)],[y(14) y(15)],'color',d)
43 line([x(15) x(17)],[y(15) y(17)],'color',d)
44 line([x(16) x(18)],[y(16) y(18)],'color',d)
45
46 if f==1
47 line([x(1) x(2)],[y(1) y(2)],'color',d)
48 line([x(4) x(5)],[y(4) y(5)],'color',d)
49 line([x(6) x(7)],[y(6) y(7)],'color',d)
50 line([x(8) x(9)],[y(8) y(9)],'color',d)
51 line([x(10) x(11)],[y(10) y(11)],'color',d)
52 line([x(12) x(13)],[y(12) y(13)],'color',d)
53 line([x(14) x(15)],[y(14) y(15)],'color',d)
54 line([x(15) x(16)],[y(15) y(16)],'color',d)
55 else
56 end
57
58 end
```

Appendix B

Chapter 3 Additional Figures

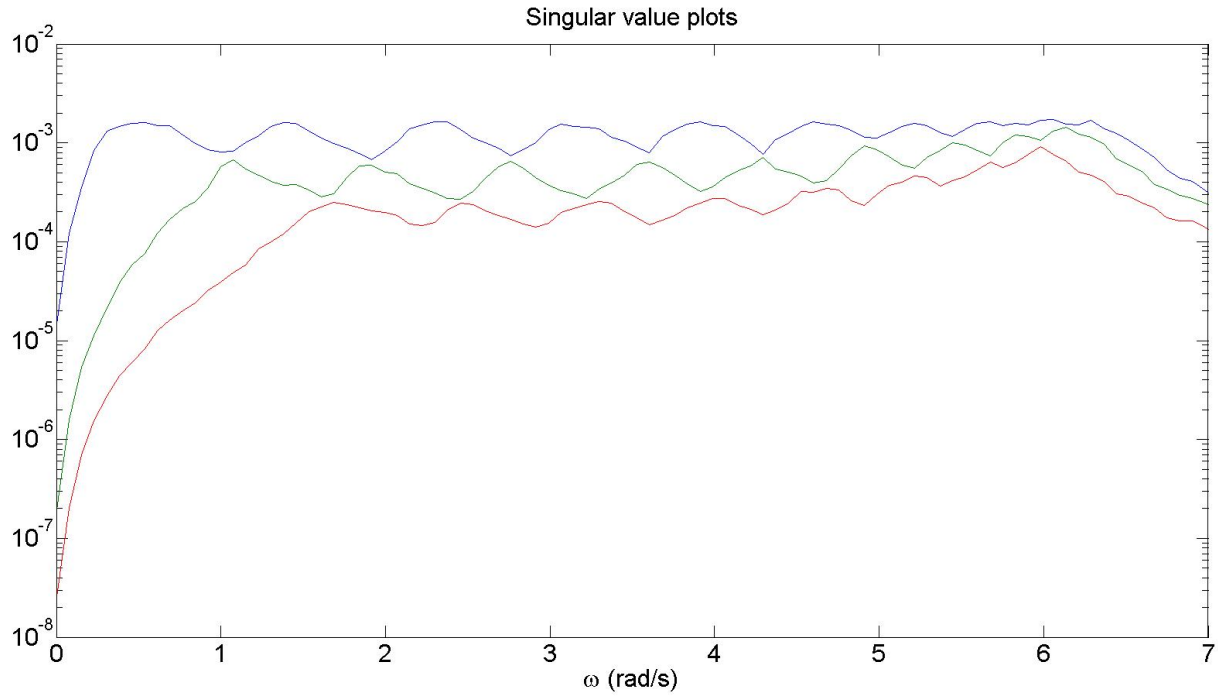

Figure B.1: Singular value plots high damping

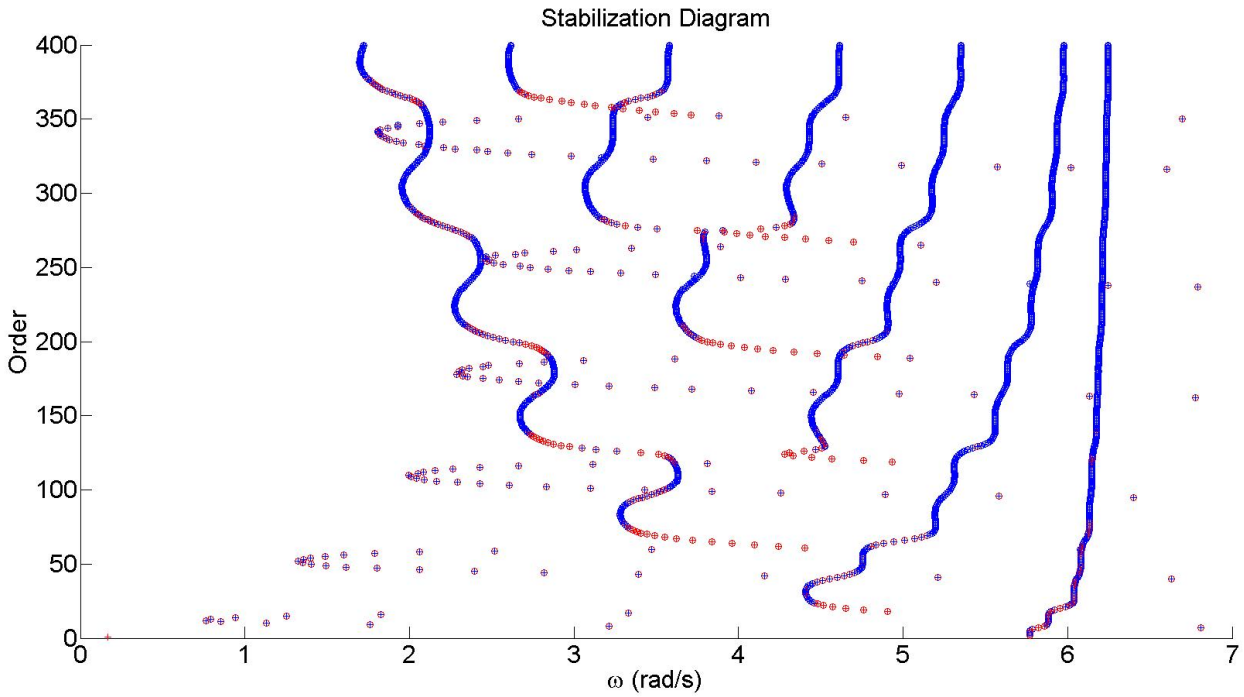

Figure B.2: Stabilization diagram with LSCF low damping $n_{max} = 400$

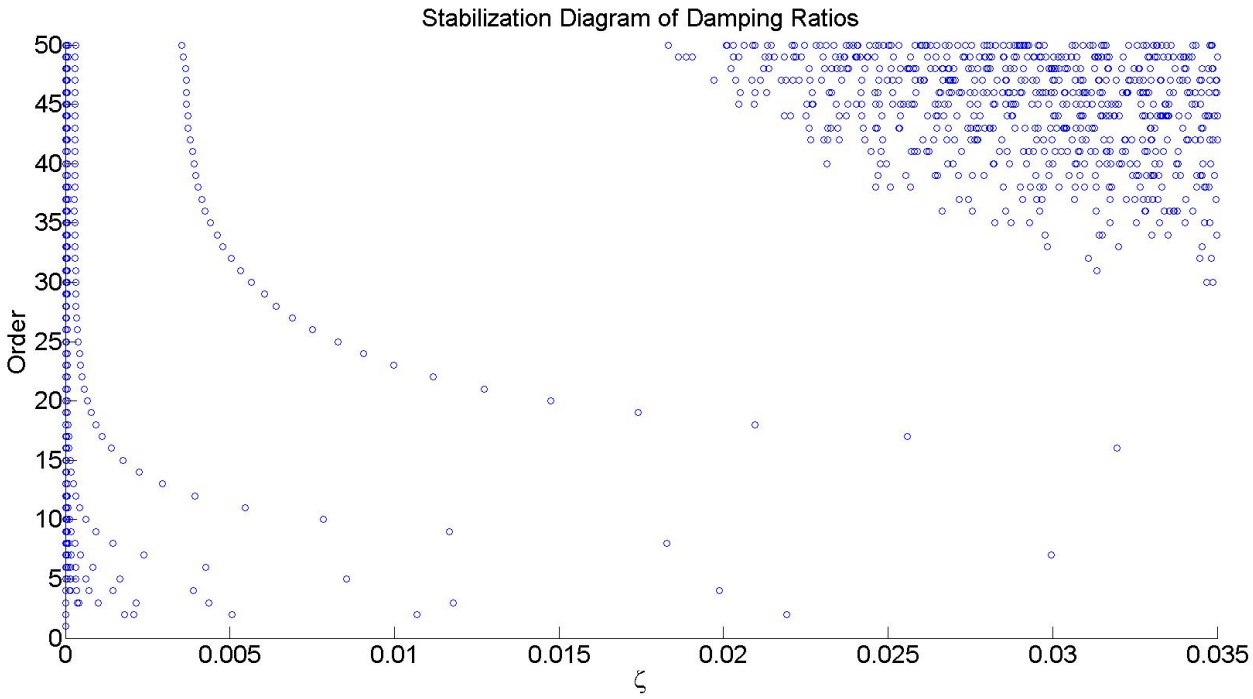

Figure B.3: Damping ratios with pLSCF low damping

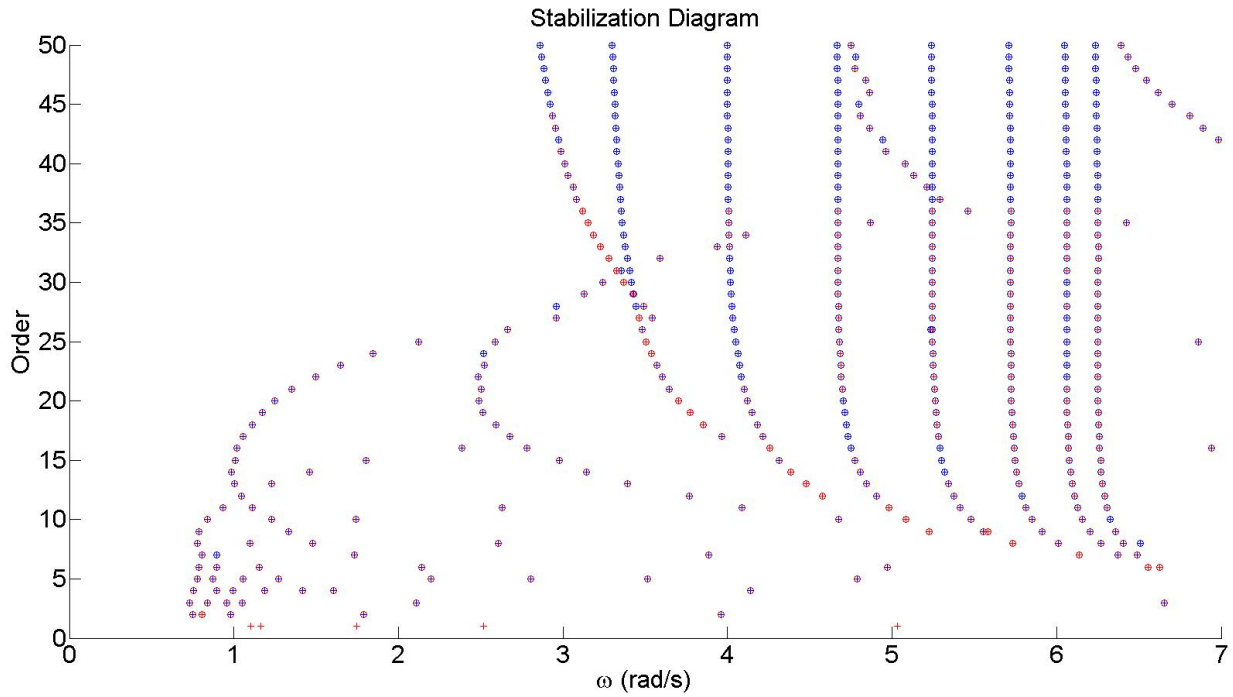

Figure B.4: Stabilization diagram with pLSCF high damping $n_{max} = 50$

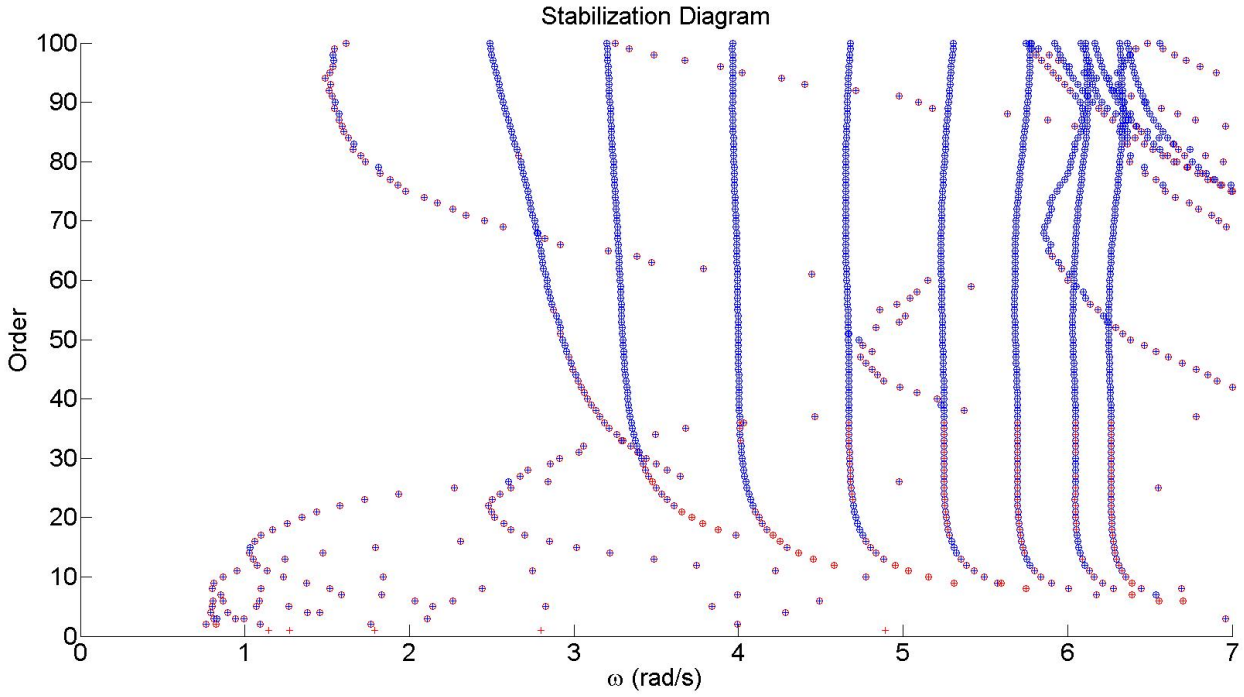

Figure B.5: Stabilization diagram with pLSCF high damping $n_{max} = 100$

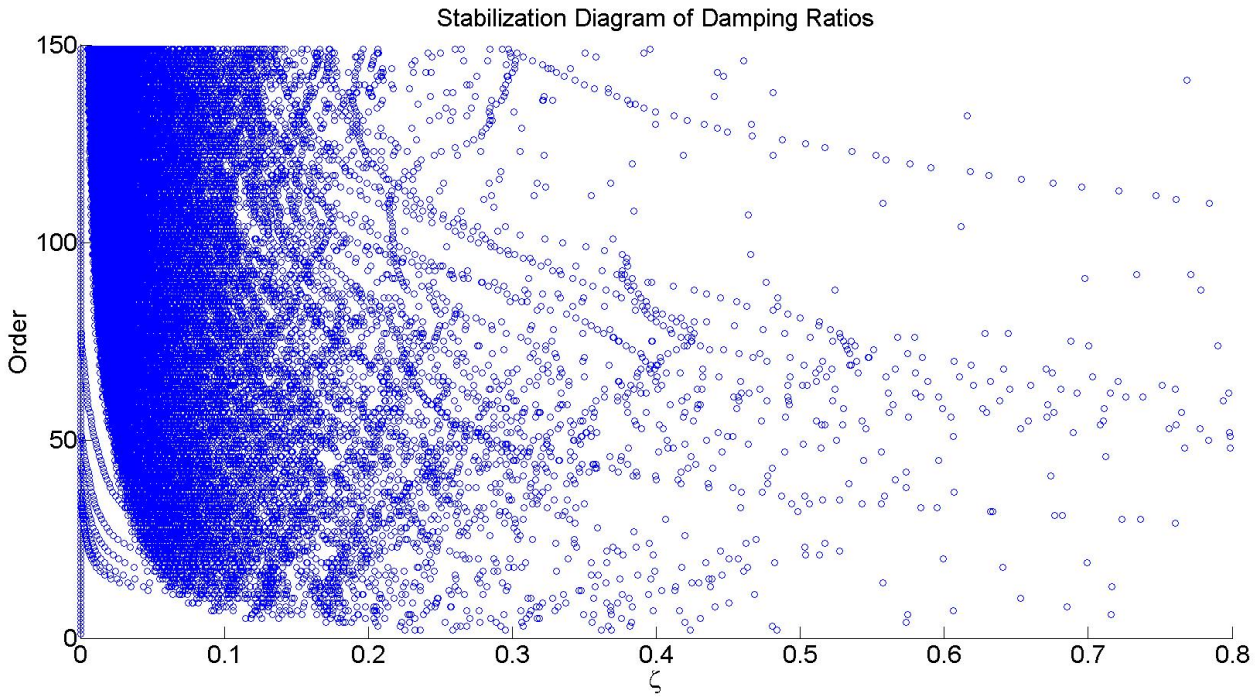

Figure B.6: Damping ratios with pLSCF high damping

Appendix C

Chapter 4 Additional Figures

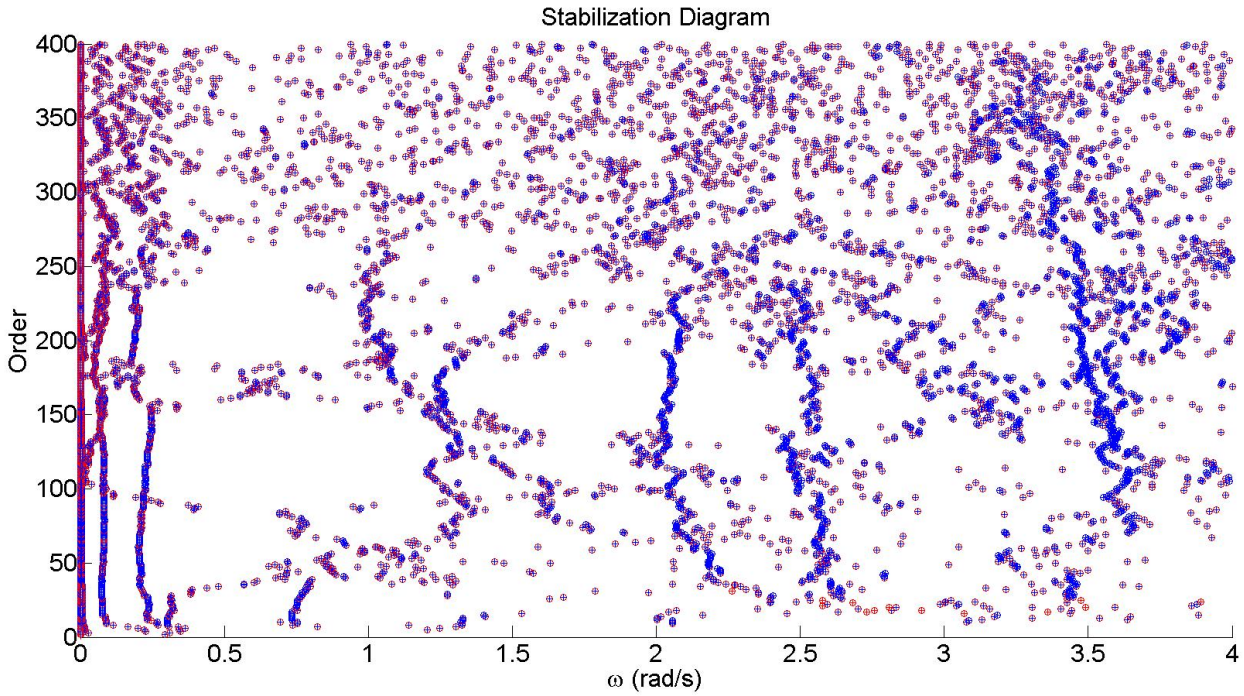

Figure C.1: Stabilization diagram using Cov-SSI with $x = 1$

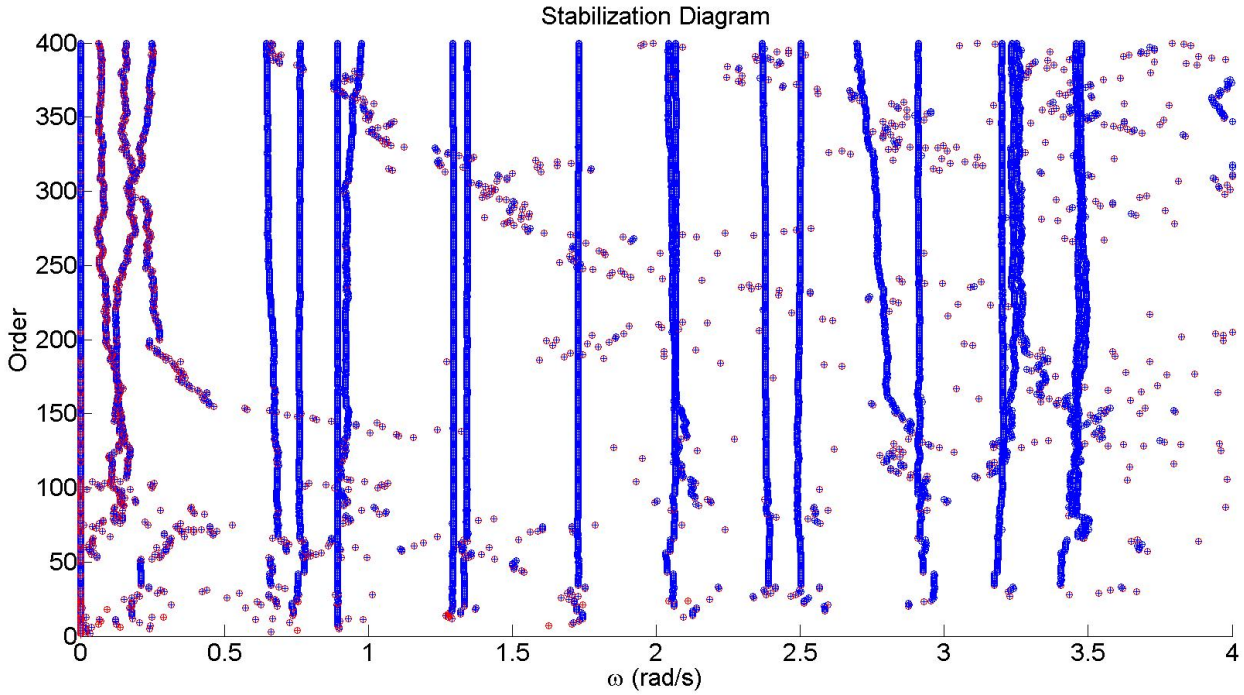

Figure C.2: Stabilization diagram using Cov-SSI with $x = 5$

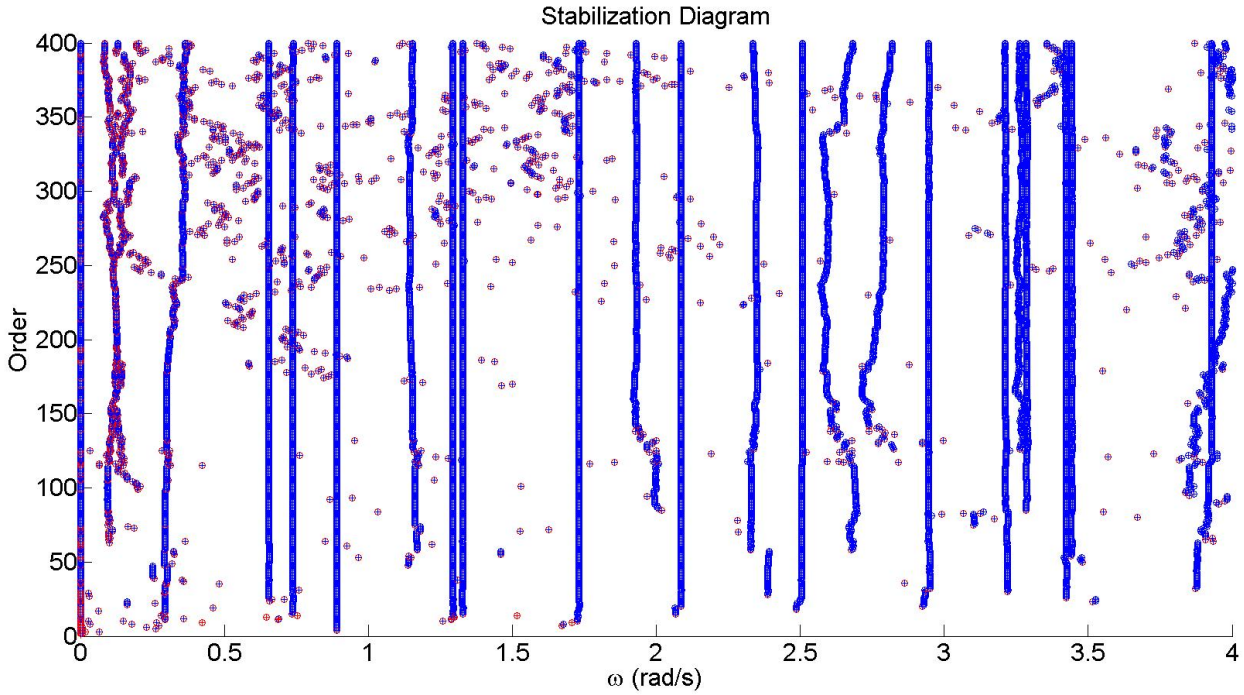

Figure C.3: Stabilization diagram using Cov-SSI with $x = 10$

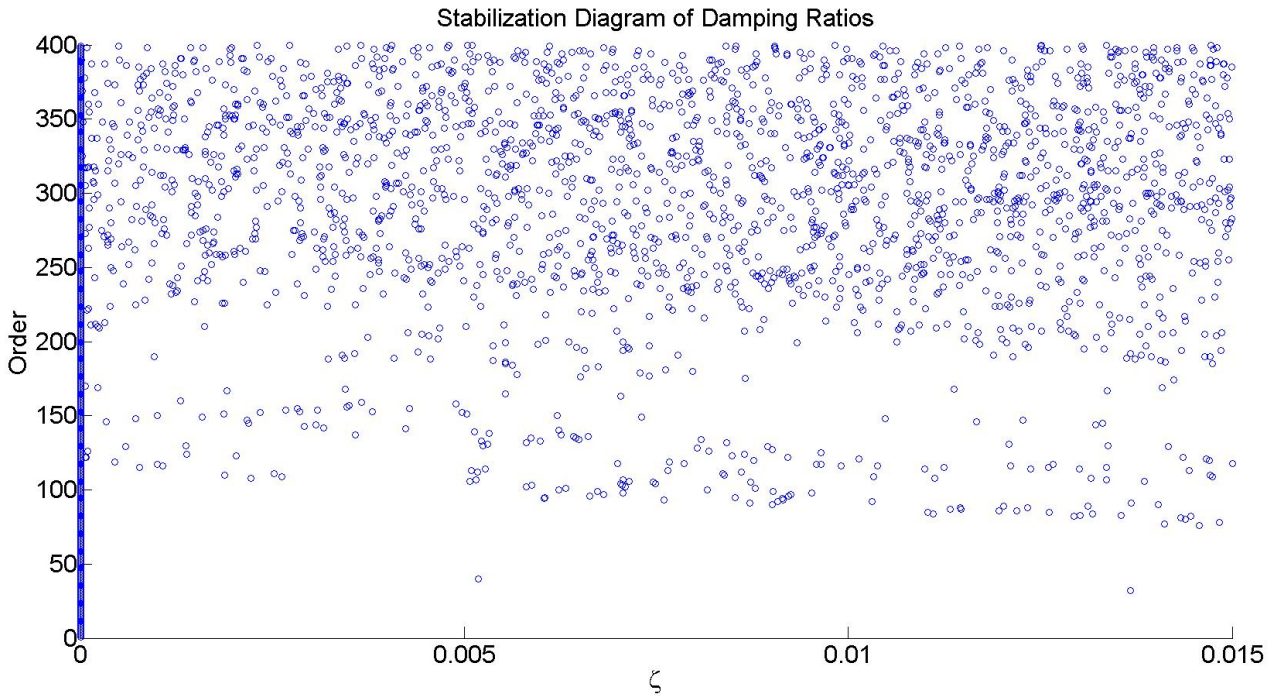

Figure C.4: Damping ratios using Cov-SSI with $x = 1$

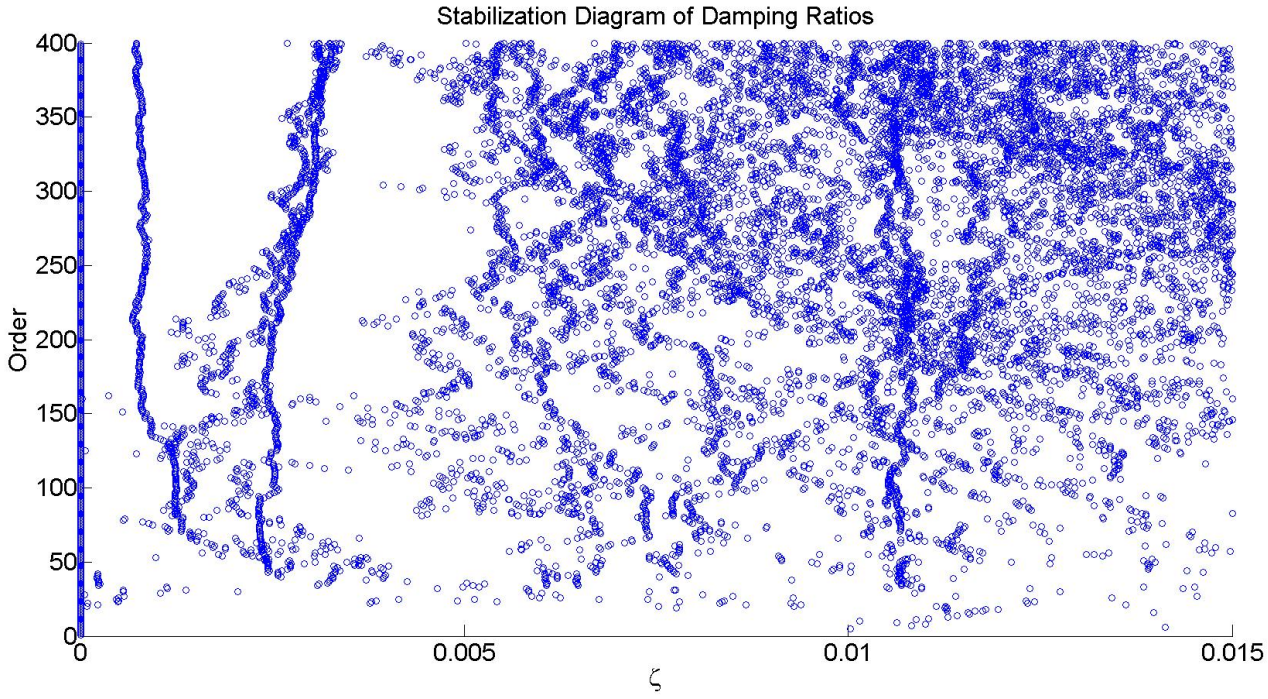

Figure C.5: Damping ratios using Cov-SSI with $x = 5$

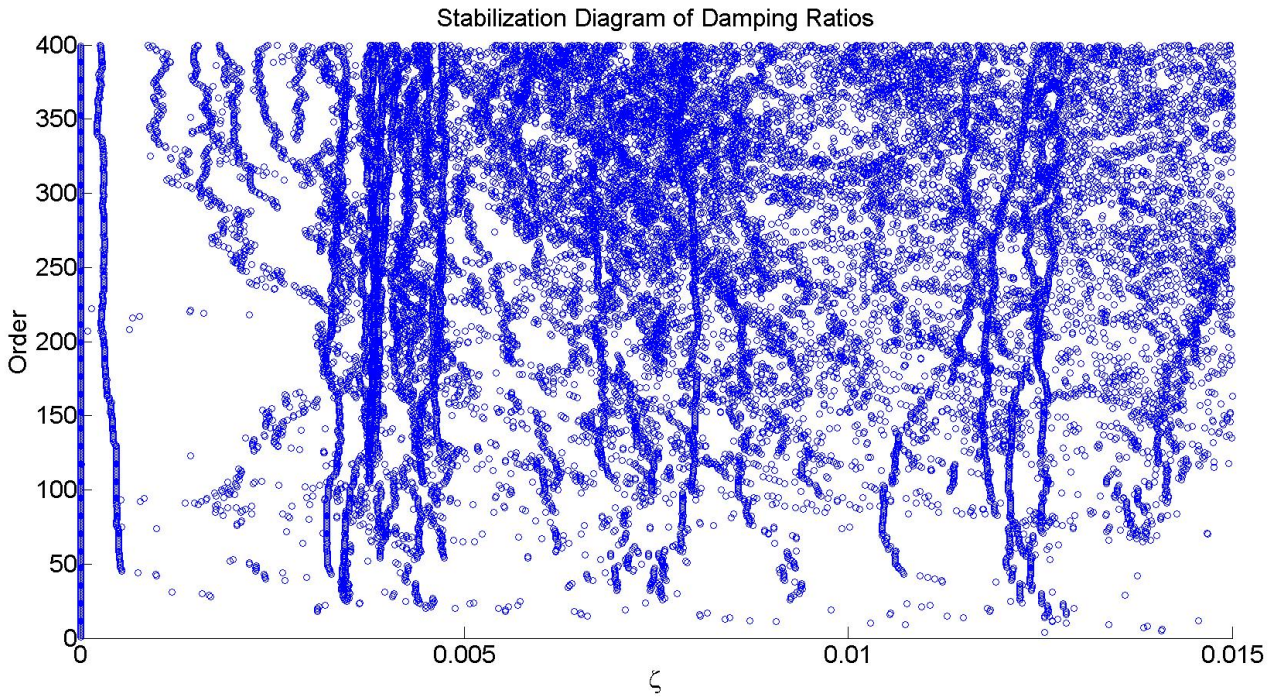

Figure C.6: Damping ratios using Cov-SSI with $x = 10$

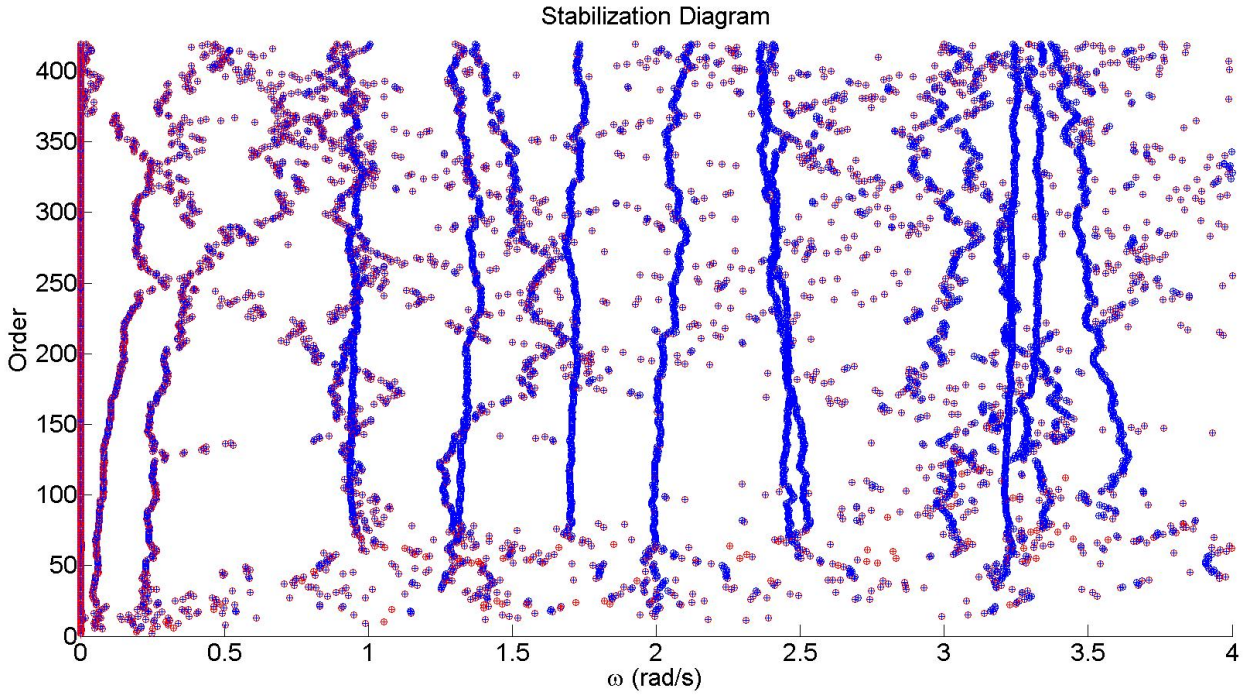

Figure C.7: Stabilization diagram using DD-SSI with $x = 2$

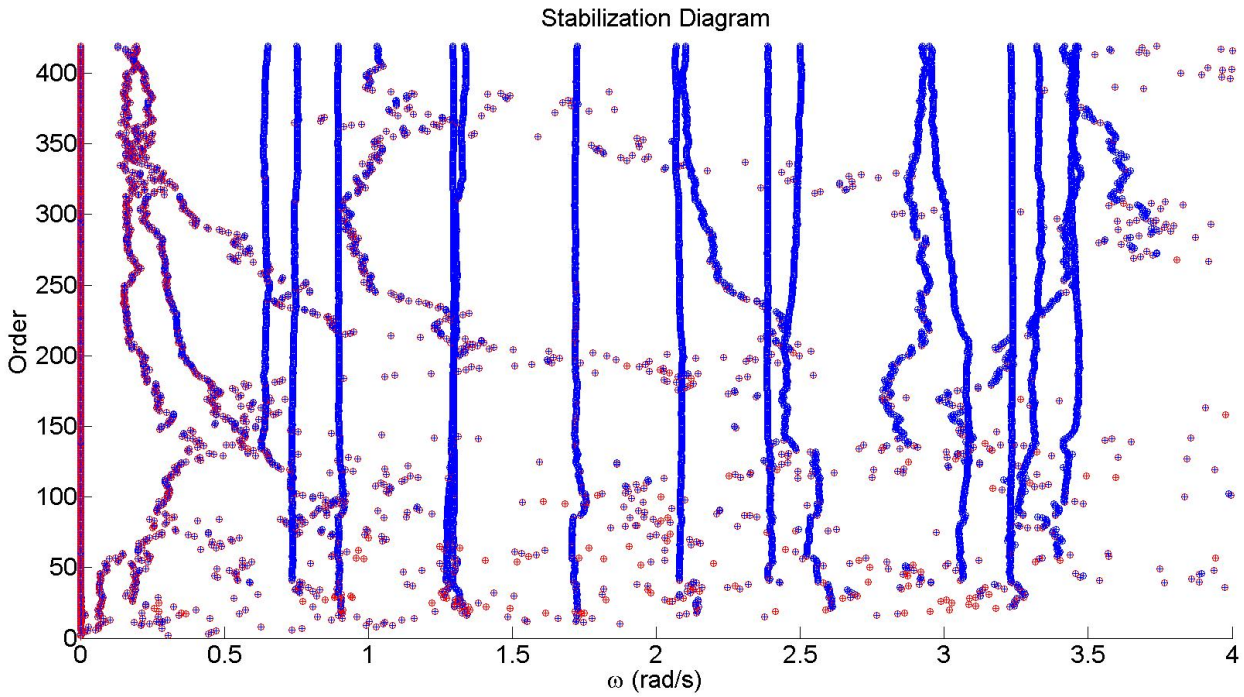

Figure C.8: Stabilization diagram using DD-SSI with $x = 4$

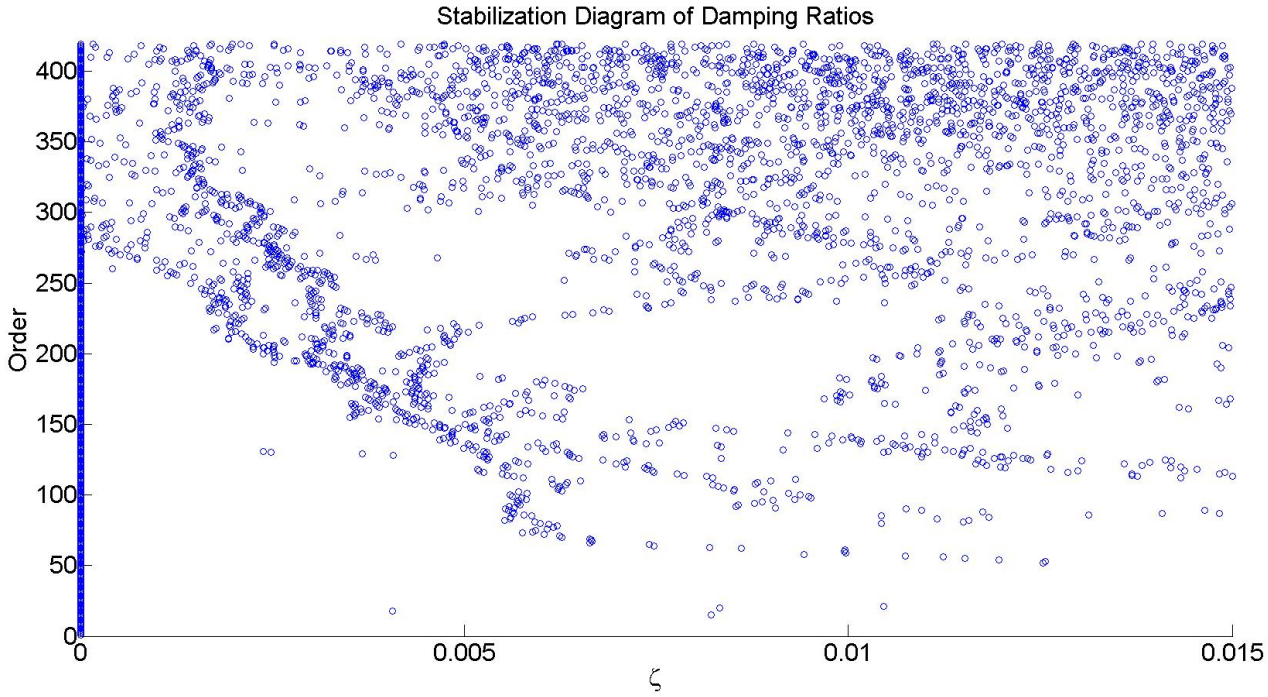

Figure C.9: Damping ratios using DD-SSI with $x = 2$

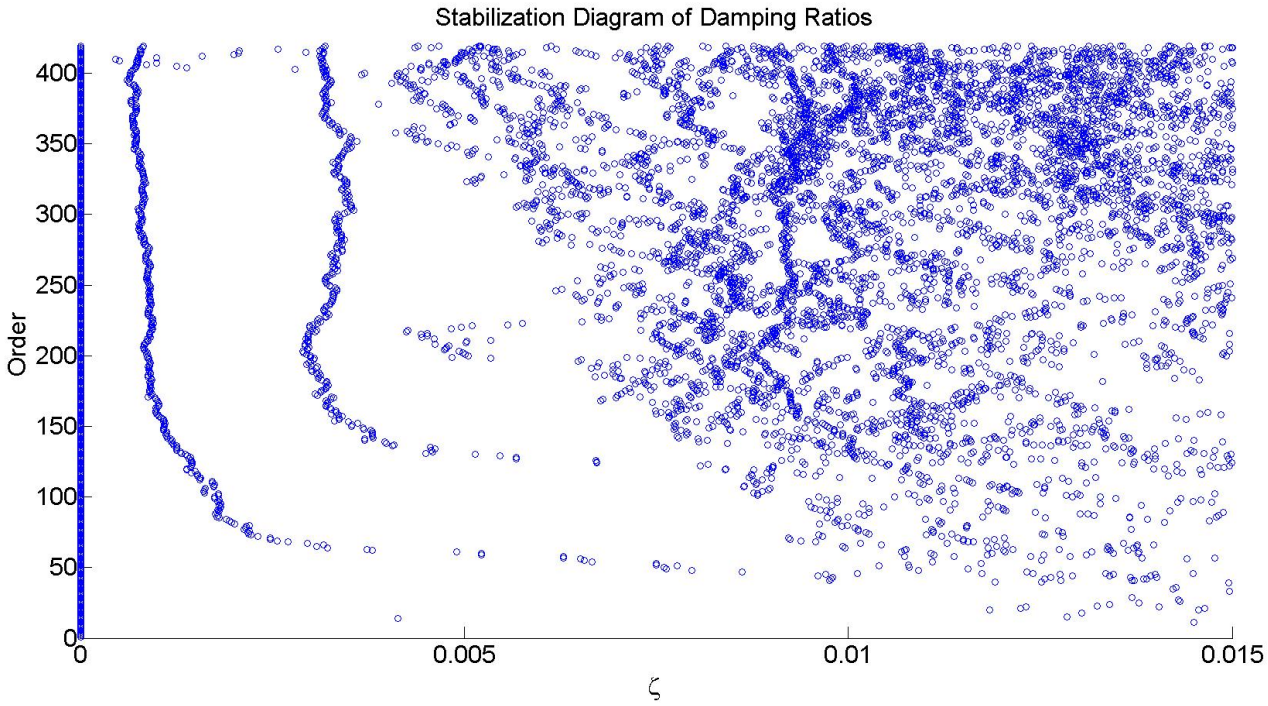

Figure C.10: Damping ratios using DD-SSI with $x = 4$

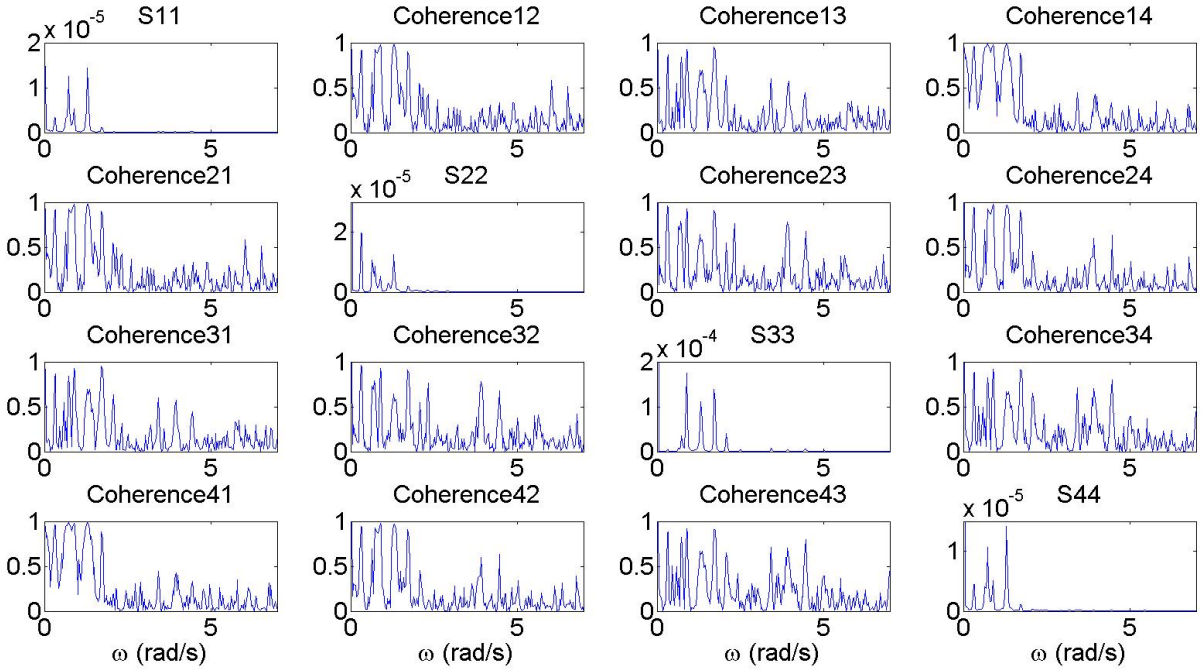

Figure C.11: PSD and Coherence for $NFFT = 4096$

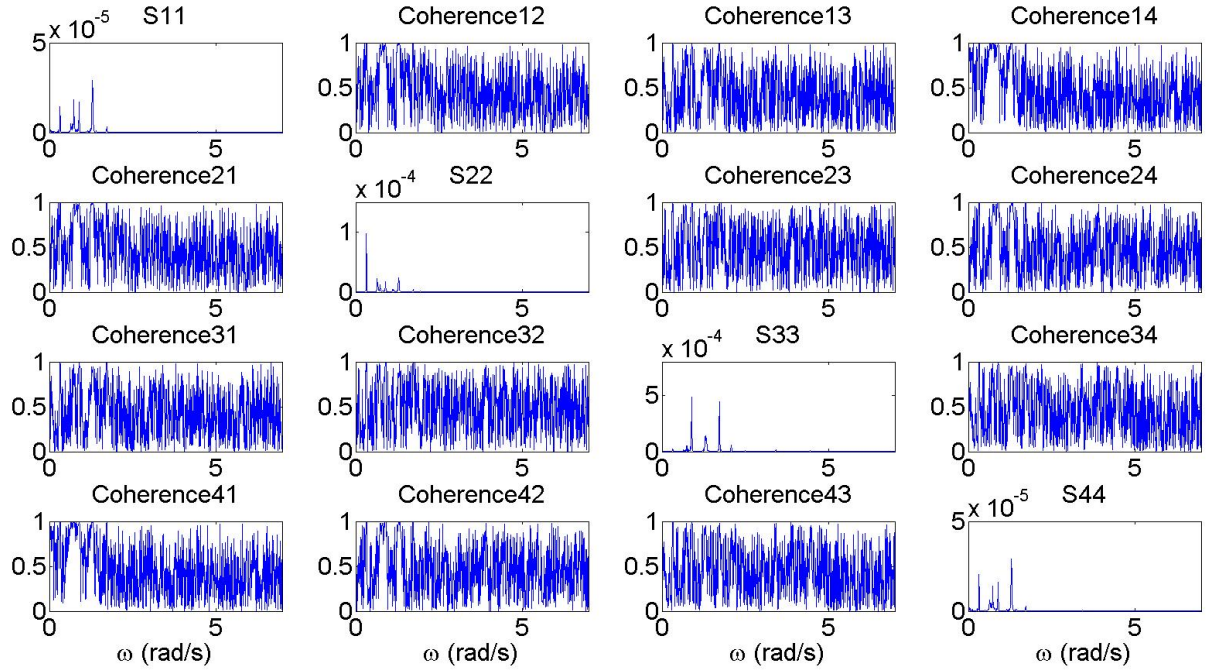

Figure C.12: PSD and Coherence for $NFFT = 16384$

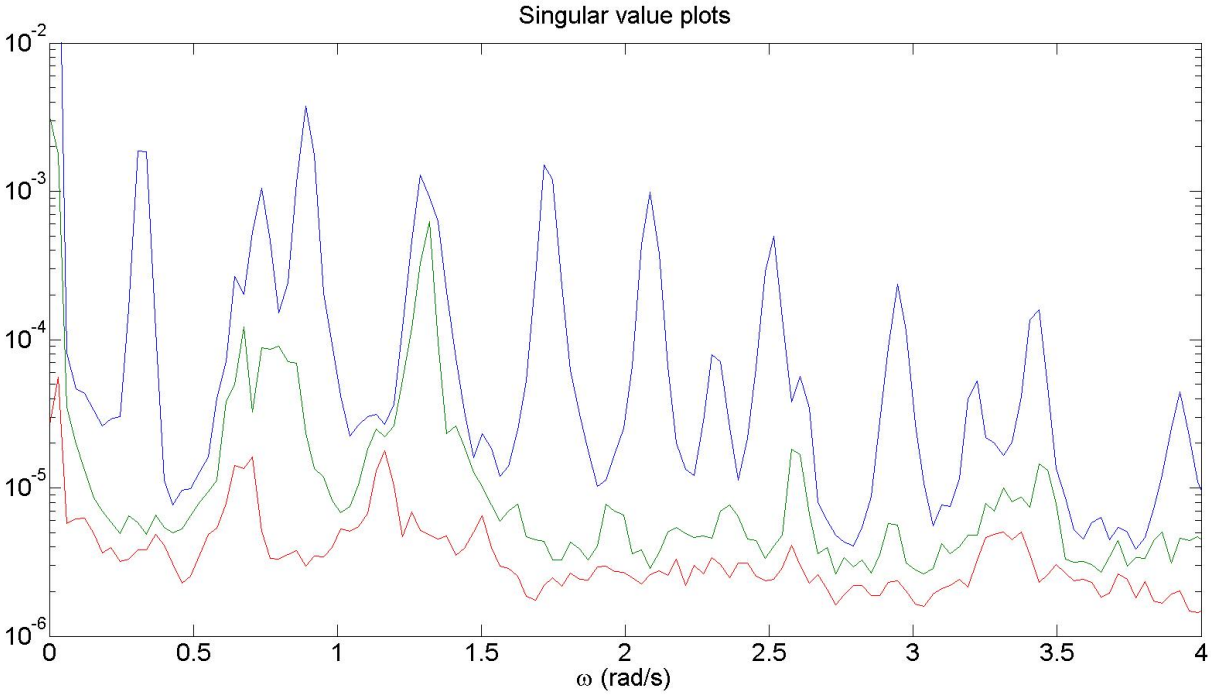

Figure C.13: Singular value plots for $NFFT = 4096$

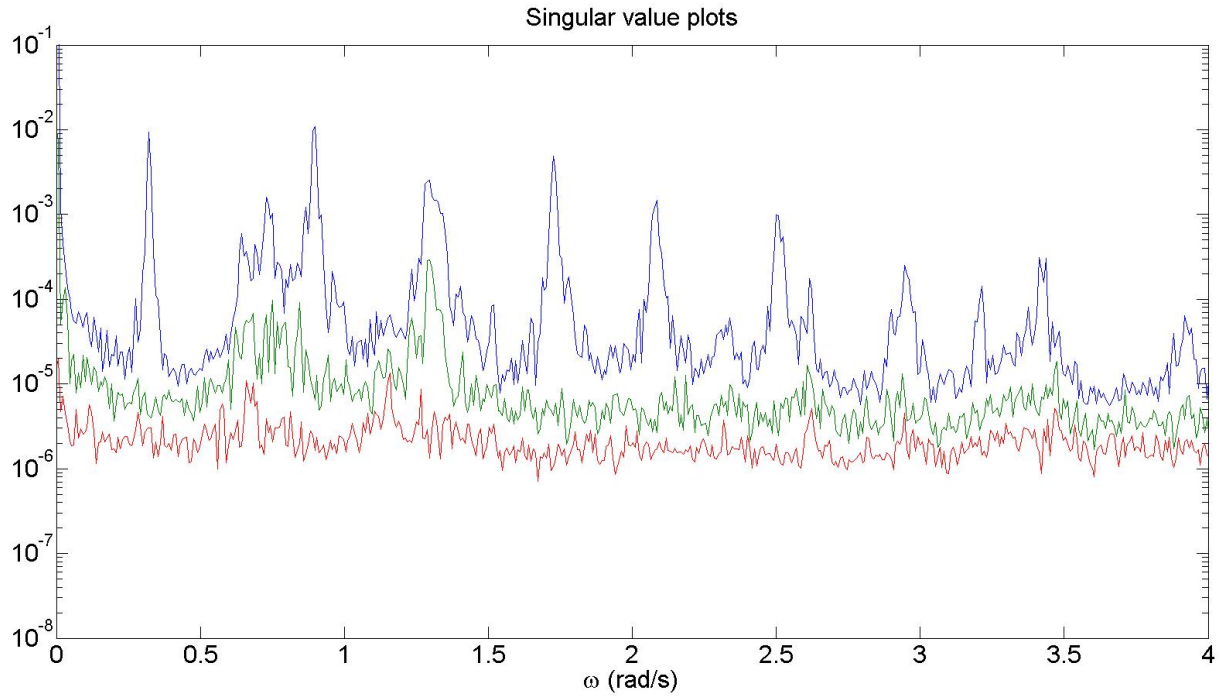

Figure C.14: Singular value plots for $NFFT = 16384$

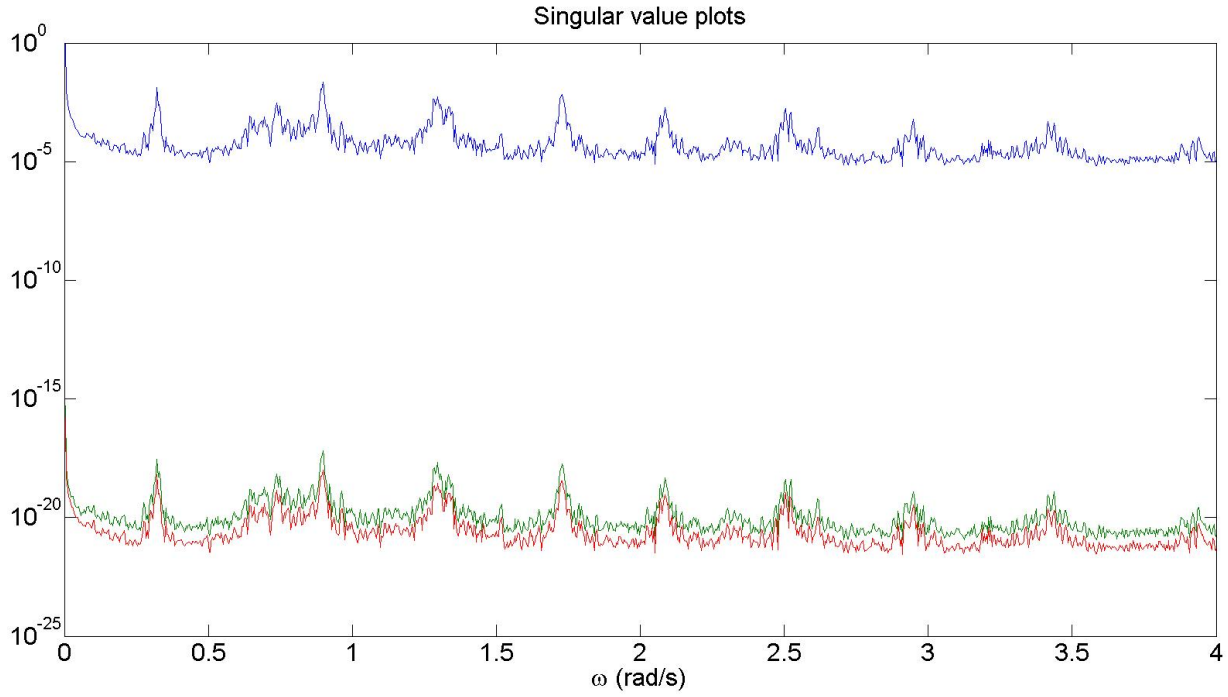

Figure C.15: Singular value plots for $NFFT = 32768$

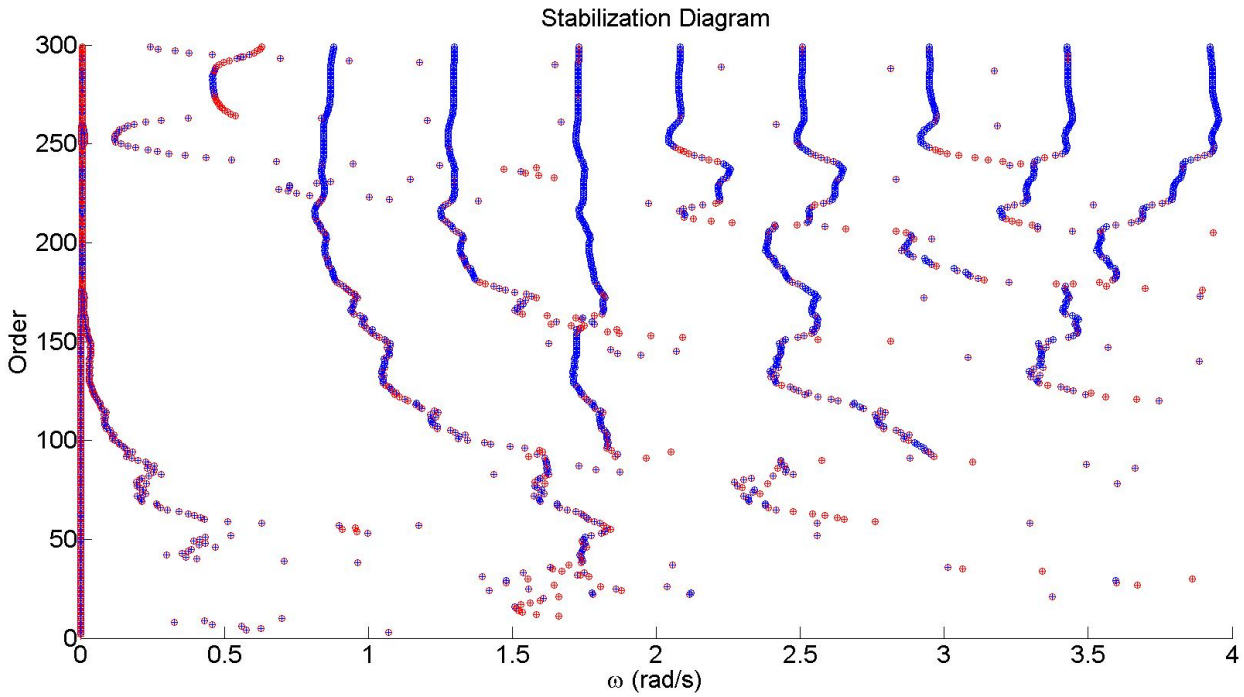

Figure C.16: Stabilization diagram using LSCF $n_{max} = 300$

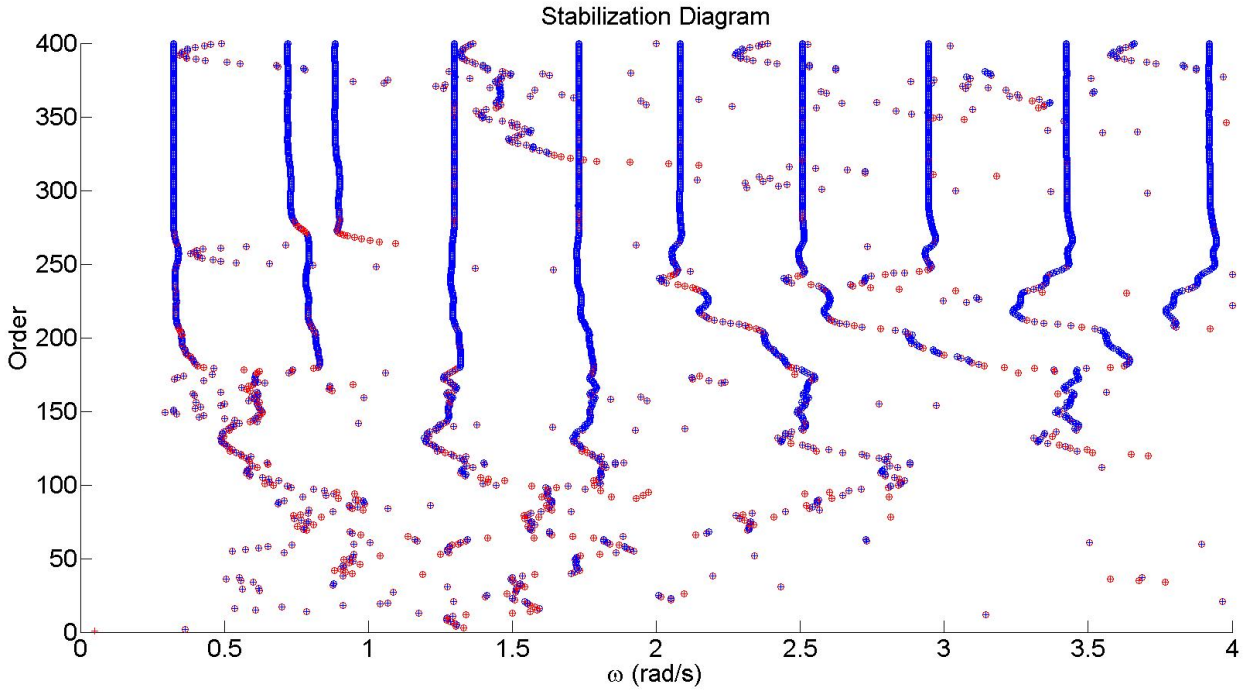

Figure C.17: Stabilization diagram using LSCF $n_{max} = 400$

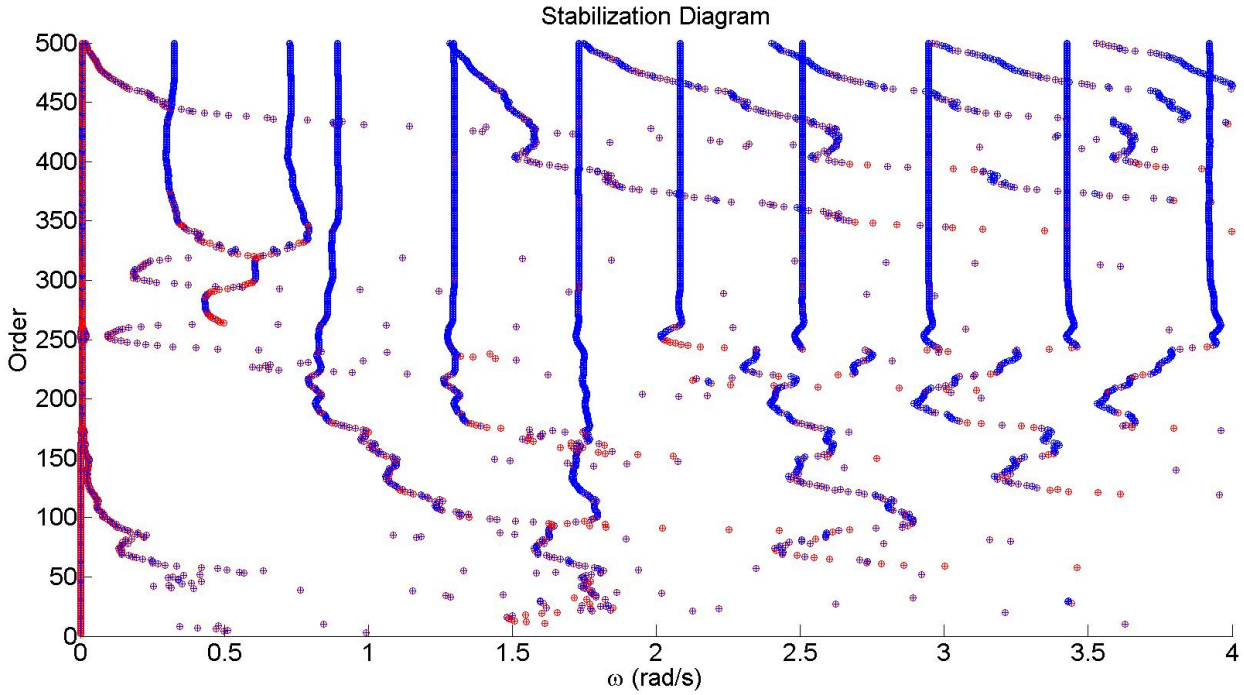

Figure C.18: Stabilization diagram using LSCF $n_{max} = 500$

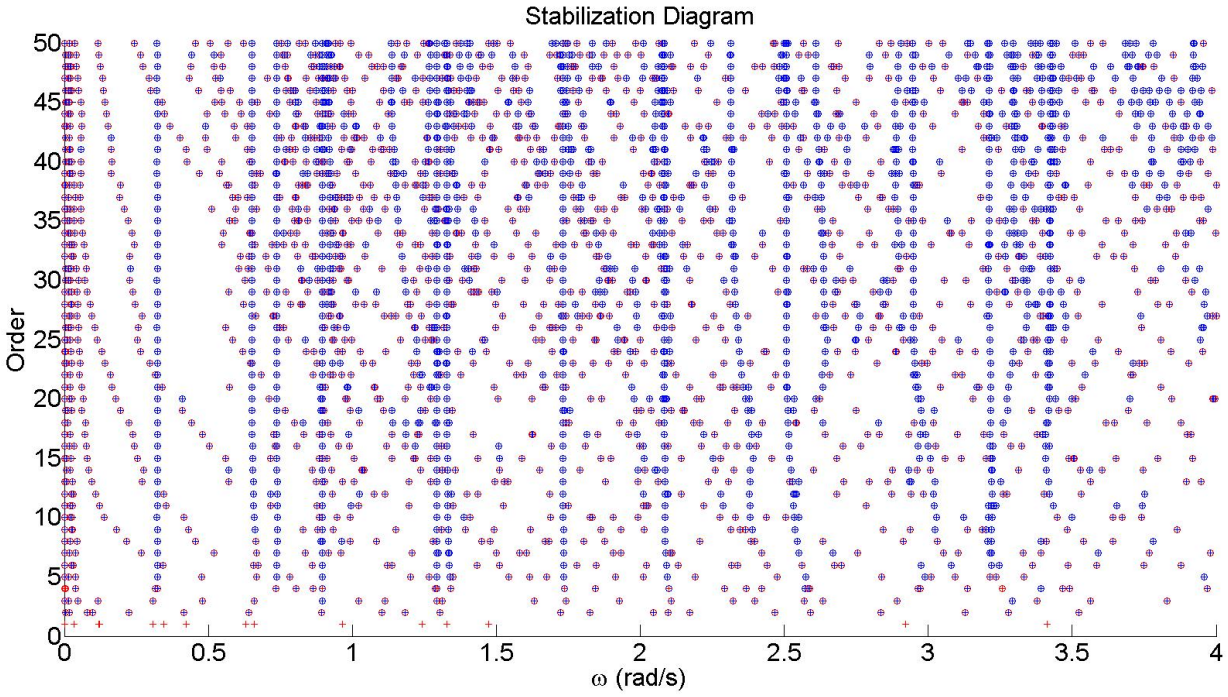

Figure C.19: Stabilization diagram using pLSCF

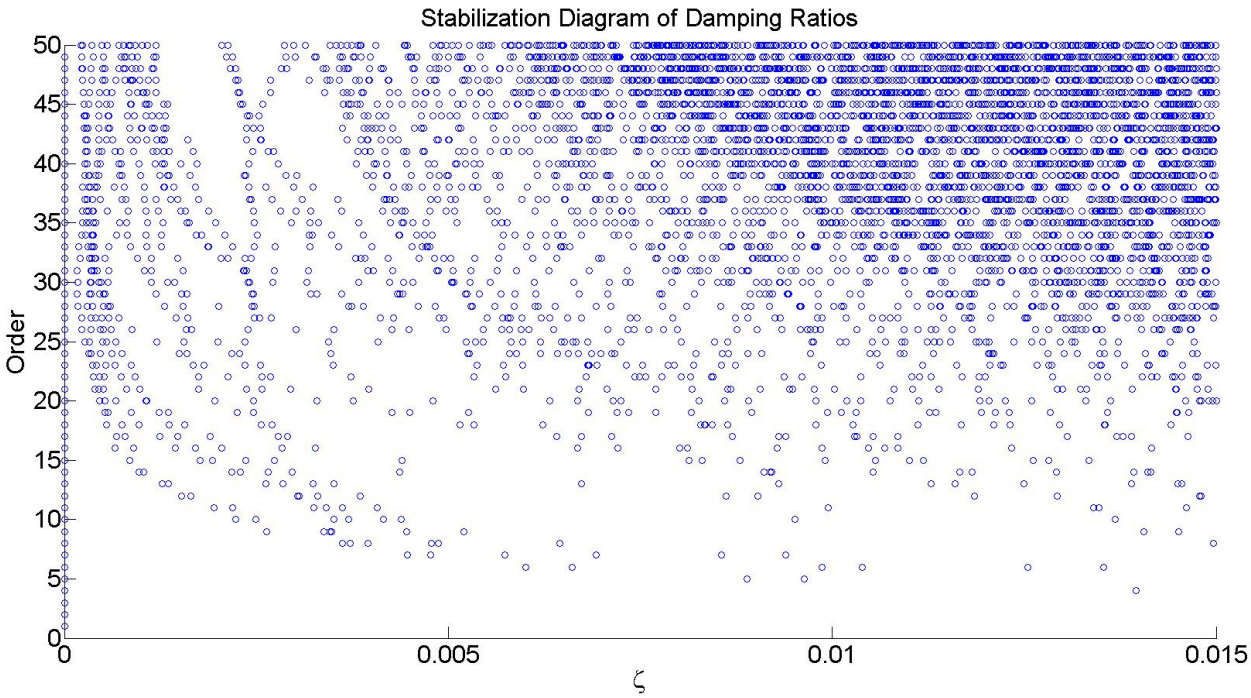

Figure C.20: Damping ratios using pLSCF